KIEL UNIVERSITY OF APPLIED SCIENCES

MASTER THESIS

Developing Cross-Platform Mobile Application Solution Using Xamarin and Microsoft Azure

---

Author:
Ranjith MURTHY

Supervisor:
Dr. Jens LÜSSEM

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science in Information Technology

in

Department of Computer Science and Electrical Engineering
Kiel University of Applied Sciences

April 6, 2017

Declaration of Authorship

I, Ranjith MURTHY, declare that this thesis titled, "Developing Cross-Platform Mobile Application Solution Using Xamarin and Microsoft Azure" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

Abstract

The global mobile market has experienced a tremendous increase in the number of smart computing device users per household. The mobile device base is strongly divided between different mobile platforms, most importantly Android, iOS and Windows Phone. Applications developed for one platform with traditional development methods only work on that platform, and supporting multiple platforms requires developing the application separately for each of the platforms.

This trend affects small business industry whose user spread across different mobile operating systems. Such companies strive to decrease the development time and with a satisfying solution that is delivered on time. To decrease the time-to-market and thus the cost of the final product, companies seek to develop applications independent of the target mobile operating system by using a cross-platform approach. This approach can eliminate the increased effort that normally comes with developing a separate application for each mobile operating system, providing a more efficient solution.

The aim of this thesis describes How to develop a cost effective multi-target mobile solution strategy for "Customer Feedback Management System for Restaurants". The application targets the leading mobile operating systems iOS Android and Windows. While sharing the business logic through a portable class library. The Xamarin and Microsoft Azure toolchain provides the mechanics for compiling the C# codebase to run on across different Operating platform.

During the research phase focus on finding the suitable data analysis method to analyses the user feedback data. In addition, finding suitable visualization techniques to visualize analyzed feedback data.

Acknowledgements

I would first like to thank my thesis supervisor Dr. Jens LÜSSEM of the Department of Computer Science and Electrical Engineeringat Kiel University of Applied Sciences. The door to Prof.LUSSEM office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to acknowledge Dr. Stephan Schneider of the Department of Computer Science and Electrical Engineeringat Kiel University of Applied Sciences as the second supervisor of this thesis, and I am gratefully indebted to his for his very valuable comments on this thesis.

Finally, I must express my very profound gratitude to my parents and to my brother for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Thank you.
Author
Ranjith MURTHY

Contents

List of Figures

List of Tables

List of Abbreviations

LAH	**L**ist **A**bbreviations **H**ere
WSF	**W**hat (it) **S**tands **F**or

Physical Constants

Speed of Light $c_0 = 2.997\,924\,58 \times 10^8\,\mathrm{m\,s^{-1}}$ (exact)

List of Symbols

a	distance	m
P	power	W ($\mathrm{J\,s^{-1}}$)
ω	angular frequency	rad

For/Dedicated to/To my…

Chapter 1

Introduction

1.1 Background

Welcome to this LaTeX Thesis Template, a beautiful and easy to use template for writing a thesis using the LaTeX typesetting system.

If you are writing a thesis (or will be in the future) and its subject is technical or mathematical (though it doesn't have to be), then creating it in LaTeX is highly recommended as a way to make sure you can just get down to the essential writing without having to worry over formatting or wasting time arguing with your word processor.

LaTeX is easily able to professionally typeset documents that run to hundreds or thousands of pages long. With simple mark-up commands, it automatically sets out the table of contents, margins, page headers and footers and keeps the formatting consistent and beautiful. One of its main strengths is the way it can easily typeset mathematics, even *heavy* mathematics. Even if those equations are the most horribly twisted and most difficult mathematical problems that can only be solved on a super-computer, you can at least count on LaTeX to make them look stunning.

1.2 Problem Statement and Research Questions

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Apple's Pages. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write the `\emph{text}` command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

1.2.1 A (not so short) Introduction to LaTeX

If you are new to LaTeX, there is a very good eBook – freely available online as a PDF file – called, "The Not So Short Introduction to LaTeX". The book's title is typically shortened to just *lshort*. You can download the latest version (as it is occasionally updated) from here: http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

It is also available in several other languages. Find yours from the list on this page: http://www.ctan.org/tex-archive/info/lshort/

It is recommended to take a little time out to learn how to use LaTeX by creating several, small 'test' documents, or having a close look at several templates on: http://www.LaTeXTemplates.com

Making the effort now means you're not stuck learning the system when what you *really* need to be doing is writing your thesis.

1.3 Structure of the Thesis

If you are familiar with LaTeX, then you should explore the directory structure of the template and then proceed to place your own information into the *THESIS INFORMATION* block of the **`main.tex`** file. You can then modify the rest of this file to your unique specifications based on your degree/university. Section **??** on page **??** will help you do this. Make sure you also read section **??** about thesis conventions to get the most out of this template.

If you are new to LaTeX it is recommended that you carry on reading through the rest of the information in this document.

Before you begin using this template you should ensure that its style complies with the thesis style guidelines imposed by your institution. In most cases this template style and layout will be suitable. If it is not, it may only require a small change to bring the template in line with your institution's recommendations. These modifications will need to be done on the **`MastersDoctoralThesis.cls`** file.

Chapter 2

Methodology

2.1 Literature Review

Welcome to this LaTeX Thesis Template, a beautiful and easy to use template for writing a thesis using the LaTeX typesetting system.

If you are writing a thesis (or will be in the future) and its subject is technical or mathematical (though it doesn't have to be), then creating it in LaTeX is highly recommended as a way to make sure you can just get down to the essential writing without having to worry over formatting or wasting time arguing with your word processor.

LaTeX is easily able to professionally typeset documents that run to hundreds or thousands of pages long. With simple mark-up commands, it automatically sets out the table of contents, margins, page headers and footers and keeps the formatting consistent and beautiful. One of its main strengths is the way it can easily typeset mathematics, even *heavy* mathematics. Even if those equations are the most horribly twisted and most difficult mathematical problems that can only be solved on a super-computer, you can at least count on LaTeX to make them look stunning.

2.2 Learning LaTeX

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Apple's Pages. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write the `\emph{text}` command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

2.2.1 A (not so short) Introduction to LaTeX

If you are new to LaTeX, there is a very good eBook – freely available online as a PDF file – called, "The Not So Short Introduction to LaTeX". The book's title is typically shortened to just *lshort*. You can download the latest version (as it is occasionally updated) from here: http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

It is also available in several other languages. Find yours from the list on this page: http://www.ctan.org/tex-archive/info/lshort/

It is recommended to take a little time out to learn how to use LaTeX by creating several, small 'test' documents, or having a close look at several templates on: http://www.LaTeXTemplates.com

Making the effort now means you're not stuck learning the system when what you *really* need to be doing is writing your thesis.

2.2.2 A Short Math Guide for LaTeX

If you are writing a technical or mathematical thesis, then you may want to read the document by the AMS (American Mathematical Society) called, "A Short Math Guide for LaTeX". It can be found online here: `http://www.ams.org/tex/amslatex.html` under the "Additional Documentation" section towards the bottom of the page.

2.2.3 Common LaTeX Math Symbols

There are a multitude of mathematical symbols available for LaTeX and it would take a great effort to learn the commands for them all. The most common ones you are likely to use are shown on this page: `http://www.sunilpatel.co.uk/latex-type/latex-math-symbols/`

You can use this page as a reference or crib sheet, the symbols are rendered as large, high quality images so you can quickly find the LaTeX command for the symbol you need.

2.2.4 LaTeX on a Mac

The LaTeX distribution is available for many systems including Windows, Linux and Mac OS X. The package for OS X is called MacTeX and it contains all the applications you need – bundled together and pre-customized – for a fully working LaTeX environment and work flow.

MacTeX includes a custom dedicated LaTeX editor called TeXShop for writing your '**`.tex`**' files and BibDesk: a program to manage your references and create your bibliography section just as easily as managing songs and creating playlists in iTunes.

2.3 Getting Started with this Template

If you are familiar with LaTeX, then you should explore the directory structure of the template and then proceed to place your own information into the *THESIS INFORMATION* block of the **`main.tex`** file. You can then modify the rest of this file to your unique specifications based on your degree/university. Section **??** on page **??** will help you do this. Make sure you also read section **??** about thesis conventions to get the most out of this template.

If you are new to LaTeX it is recommended that you carry on reading through the rest of the information in this document.

Before you begin using this template you should ensure that its style complies with the thesis style guidelines imposed by your institution. In most cases this template style and layout will be suitable. If it is not, it may only require a small change to bring the template in line with your institution's recommendations. These modifications will need to be done on the **`MastersDoctoralThesis.cls`** file.

Chapter 3

Mobile Application Development

3.1 Mobile Platforms

According to a market share study by IDC (Chau, Govindaraj, and Reith, 2017), the smartphone market is currently clearly dominated by Android, which held over 86.8 percent of the market during the second quarter of 2016. Meanwhile, iOS saw its market share for 2016Q3 grow by 12.7 percent QoQ with 45.5 million shipments. The iPhone 6S followed by its newest model, the iPhone 7 were the best-selling models this quarter. Windows Phone experienced a QoQ decline of 35.2 percent with a total of 974.4 thousand units shipped this quarter. With Microsoft's focus on business users, the decline in the consumer market is expected to continue.

The market shares of the top three mobile platforms and the remaining market during the second quarter of each year between 2015 and 2016 are shown in (e.g. Table 3.1).

3.1.1 Android

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. Beginning with the first commercial Android device in September 2008, the operating system has gone through multiple major releases, with the current version being 7.0 "Nougat", released in August 2016. Android applications ("apps") can be downloaded from the Google Play store, which features over 2.7 million apps as of February 2017. Android's source code is released by Google under an open source license, although most Android devices ultimately ship with a combination of free and open source and proprietary software, including proprietary software required for accessing Google services. Android is popular with technology companies that require a ready-made, low-cost and customizable operating system for high-tech devices.

TABLE 3.1: Smartphone Operating System market shares from years 2015 to 2016.

Period	Android	iOS	Windows Phone	Others
2015Q4	79.6%	18.7%	1.2%	0.5%
2016Q1	83.5%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%
2016Q3	86.8%	12.5%	0.3%	0.4%

Android is built on a modi
ed Linux 2.6 series kernel that provides core system services such as security, memory management, process management, network stack and driver model. The kernel and low level tools are contained in the bottom layer, colored red in the illustration. The basic libraries included in Android are programmed in C and C++, and are accessed through the Android application framework.

The Android runtime contains a set of Java core libraries and the Dalvik virtual machine (VM). The Dalvik VM executes
les in Dalvik Executable (.dex) format, usually transformed from Java byte code to Dalvik byte code.[16 TBD]

Every Android application runs in its own process with its own sandboxed instance of Dalvik VM. abstraction of the underlying hardware for the rest of the software stack.[23 TBD]

The application framework layer gives the developers access to the same framework Application Programming Interfaces (API) used by the core applications. [43] The frameworks are written in Java and provide abstractions the Android libraries and the features of the Dalvik VM.[16]

Applications for Android are developed through the Android Software Development Kit (SDK), usually with the Java programming language. The SDK provides the API libraries and developer tools for building, testing and debugging for Android. Development can be done in any of the current major operating systems and an integrated development environment (IDE) of choice, although Google recommends using Android Studio. Another common option is to use Eclipse IDE with Android Developer Tools (ADT) plugin, provided by Google, which integrates the Android SDK into Eclipse. The ADT allows the developer to test the application with an Android emulator or a connected device, and provides a graphical editor for building the user interface (UI) of the application.[5]

The main distribution channel for Android applications is the Google Play Store, formerly known as Android Market, where developers can publish their applications after registration.

3.1.2 iOS

iOSis a mobile operating system created and developed by Apple Inc. exclusively for its hardware. It is the operating system that presently powers many of the company's mobile devices, including the iPhone, iPad, and iPod Touch. It is the second most popular mobile operating system globally after Android. iPad tablets are also the second most popular, by sales, against Android since 2013.[9]

Originally unveiled in 2007 for the iPhone, iOS has been extended to support other Apple devices such as the iPod Touch (September 2007) and the iPad (January 2010). As of January 2017, Apple's App Store contains more than 2.2 million iOS applications, 1 million of which are native for iPads. These mobile apps have collectively been downloaded more than 130 billion times.

The iOS, there are four abstraction layers: the Core OS, Core Services, Media, and Cocoa Touch layers.

The various core frameworks are written in the Objective-C programming language. The Core OS layer contains the low-level features most of the other technologies are built upon. Applications rarely use these technologies directly, but rather use them through the other frameworks. However, the layer contains frameworks for features such as security, Bluetooth support and communicating with external hardware, that can be used by applications if needed. The layer also contains the kernel environment, drivers and low-level UNIX interfaces of the operating system. Access to the kernel and drivers is restricted to a limited set of system frameworks and applications.[9]

The various system services used by applications are contained in the Core Services layer. This includes technologies to support features like location, iCloud storage, peer-to-peer services and networking. The Media layer above it contains the graphics, audio and video technologies needed to implement multimedia features in applications. Finally in the top layer resides the Cocoa Touch framework, providing the key frameworks for building iOS applications. This includes high-level programming interfaces for making animations, networking and modifying the appearance of the application. Cocoa Touch also handles touch-based inputs and multitasking.[8]

Building iOS applications requires using Apple's Xcode IDE on a Mac computer running OS X 10.8 or later and iOS SDK. Xcode provides the standard tools to code, debug and design the interface for the applications. Generally iOS applications are written in Objective-C language.[8]

Applications for iOS are distributed to consumers exclusively through Apple's App Store. Developers enroll in Apple Developer Program and pay yearly fee to be able to publish applications in the App Store, and applications go through an approval process by Apple before appearing in the store.[8] The approval process causes longer development times, but lowers

the number of low-quality applications in the store.[17] The approval process can pose a challenge for applications developed with various cross-platform methods. For example, in 2010 Apple maintained that apps must be "originally written in Objective-C, C, C++ or JavaScript" to be accepted into the store. The restrictions have been eased since then, but applications still sometimes get rejected for being too slow or not feeling native enough. Apple App Store can also reject apps that download executable code or interpret code not contained within the application archive.[32]

3.1.3 Windows

Windows 10 is a personal computer operating system developed and released by Microsoft as part of the Windows NT family of operating systems. It was officially unveiled in September 2014 following a brief demo at Build 2014. The first version of the operating system entered a public beta testing process in October, leading up to its consumer release on July 29, 2015.[9]

Windows 10 introduces what Microsoft described as "universal apps" expanding on Metro-style apps, The first release of Windows 10 also introduces a virtual desktop system, a window and desktop management feature called Task View, the Microsoft Edge web browser, support for fingerprint and face recognition login, new security features for enterprise environments, and DirectX 12 and WDDM 2.0 to improve the operating system's graphics capabilities for games.

Universal Windows Platform (UWP) apps[1] (formerly Windows Store apps and Metro-style apps)[2] are apps that can be used across all compatible Microsoft Windows devices, including personal computers (PCs), tablets, smartphones, Xbox One, Microsoft HoloLens, and Internet of Things. UWP apps are primarily purchased and downloaded via the Windows Store.[3]

3.1.4 Tizen

Tizen is an open and flexible operating system built from the ground up to address the needs of all stakeholders of the mobile and connected device ecosystem, including device manufacturers, mobile operators, application developers and independent software vendors (ISVs). Tizen is developed by a community of developers, under open source governance, and is open to all members who wish to participate.

The Tizen operating system comes in multiple profiles to serve different industry requirements. The current Tizen profiles are Tizen IVI (in-vehicle infotainment), Tizen Mobile, Tizen TV, and Tizen Wearable. In addition to that, as of Tizen 3.0, all profiles are built on top of a common, shared infrastructure called Tizen Common.

With Tizen, a device manufacturer can begin with one of these profiles and modify it to serve their own needs, or use the Tizen Common base to develop a new profile to meet the memory, processing and power requirements of any device and quickly bring it to market.

Mobile operators can work with device partners to customize the operating system and user experience to meet the needs of specific customer segments or demographics.

For application developers and ISVs, Tizen offers the power of native application development with the flexibility of unparalleled HTML5 support. Tizen also offers the potential for application developers to extend their reach to new “smart devices” running Tizen, including wearables, consumer electronics (TVs, gaming consoles, DVRs, etc.), cars and appliances.

The Tizen project resides within the Linux Foundation and is governed by a Technical Steering Group. The Technical Steering Group is the primary decision-making body for the open source project, with a focus on platform development and delivery, along with the formation of working groups to support device verticals.

The Tizen Association has been formed to guide the industry role of Tizen, including gathering of requirements, identification and facilitation of service models, and overall industry marketing and education.

Tizen provides application development tools based on the JavaScript libraries jQuery and jQuery Mobile. Since version 2.0, a C++ native application framework is also available, based on an Open Services Platform from the Bada platform.

Samsung Releases New Preview of Visual Studio Tools for Tizen

3.2 Mobile Applications

Mobile applications are consist of software/set of program that runs on a mobile device and perform certain tasks for the user.Mobile application is a new and fast developing Segment of the global Information and Communication Technology.

Mobile Screens are small, But Mobile apps are big, and life as we know it is on its head again. In a world that's increasingly social and open, mobile apps play a vital role, and Today we have changed the focus from what's on the Web, to the apps on our mobile device. Mobile apps are very imperative. But where do we start? There are many factors that play a part in your mobile strategy, such as your team's development skills, required device functionality, the importance of security, offline capability, interoperability, etc., that must be taken into account. Finally it's not just a question of what mobile application will do, but how we will reach there. get it there.

Mobile applications come in two distinct formats: native apps and web apps. Due to differences in their underlying technology, each approach has inherent advantages and drawbacks.

3.2.1 Native Applications

A native mobile app is built specifically for a particular device and its operating system. Unlike a web app that is accessed over the internet, a native app is downloaded from a web app store and installed on the device. Native apps are written in Java, Objective C, or some other programming language. This is changing with HTML5, but functionality is inconsistent and incomplete.

Native apps have a major advantage over web applications the ability to leverage device-specific hardware and software. This means that native apps can take advantage of the latest technology available on mobile devices and can integrate with on-board apps such as the calendar, contacts, and email. However, this is a double-edged sword: while mobile technology is wildly popular, it is also constantly changing and highly fragmented. This makes the task of keeping up with the pace of emerging technology onerous and costly, especially on multiple platforms.

TABLE 3.2: Native App Development across different platform

	Android	**iOS**	**Windows 10**	**Tizen**
Programming languages	JAVA	Objective-C,Swift	XAML,C#	C,C++
Integrated Development Environment	Android Studio	Xcode	Visual Studio	Tizen Studio
Software Development Kit	Android SDK	iOS SDK	Windows SDK	Tizen SDK
App Distribution	Google play	Appstore	Windows Store	Tizen Store
Community	Very Good	Very Good	Good	Limited

3.2.2 Mobile Web Apps

A mobile web app is a web application formatted for smartphones and tablets, and accessed through the mobile device's web browser. Like a traditional web application, a mobile web app is built with three core technologies: HTML (defines static text and images), CSS (defines style and presentation), and JavaScript (defines interactions and animations).

Since web apps are browser-based, they're intended to be platform and device independent, able to run on any web-enabled smartphone or tablet. A mobile web app is normally downloaded from a central web server each time it is run, although apps built using HTML5 (described below) can also run on the mobile device for offline use.

However, significant limitations, especially for enterprise mobile, are offline storage and security. While you can implement a semblance of offline capability by caching files on the device, it just isn't a very good solution. Although the underlying database might be encrypted, it's not as well segmented as a native keychain encryption that protects each app with a developer certificate. Also, if a web app with authentication is launched from the desktop, it will require users to enter their credentials every time the app it is sent to the background. This is a lousy experience for the user. In general, implementing even trivial security measures on a native platform can be complex tasks for a mobile Web developer. Therefore, if security is of the utmost importance, it can be the deciding factor on which mobile technology you choose.

3.2.3 Hybrid Mobile Applications

Hybrid development combines the best (or worst) of both the native and HTML5 worlds. We define hybrid as a web app, primarily built using HTML5 and JavaScript, that is then wrapped inside a thin native container that provides access to native platform features. PhoneGap is an example of the most popular container for creating hybrid mobile apps.

For the most part, hybrid apps provide the best of both worlds. Existing web developers that have become gurus at optimizing JavaScript, pushing CSS to create

beautiful layouts, and writing compliant HTML code that works on any platform can now create sophisticated mobile applications that don't sacrifice the cool native capabilities. In certain circumstances, native developers can write plugins for tasks like image processing, but in cases like this, the devil is in the details.

On iOS, the embedded web browser or the UIWebView is not identical to the Safari browser. While the differences are minor, they can cause debugging headaches. That's why it pays off to invest in popular frameworks that have addressed all of the limitations.

You know that native apps are installed on the device, while HTML5 apps reside on a Web server, so you might be wondering if hybrid apps store their files on the device or on a server? Yes. In fact there are two ways to implement a hybrid app.

- Local - You can package HTML and JavaScript code inside the mobile application binary, in a manner similar to the structure of a native application. In this scenario you use REST APIs to move data back and forth between the device and the cloud.
- Server - Alternatively you can implement the full web application from the server (with optional caching for better performance), simply using the container as a thin shell over the UIWebview.

3.2.4 Conclusion

Mobile development is a constantly moving target. Every six months, there's a new mobile operating system, with unique features only accessible with native APIs. The containers bring those to hybrid apps soon thereafter, with the web making tremendous leaps every few years. Based on current technology, one of the scenarios examined in this article is bound to suit your needs. Let's sum those up in the following table:

3.3 Cross-Platform Development Tools

3.3.1 PhoneGap

PhoneGap is the open source framework that gets you building amazing mobile apps using web technology.

Developing your mobile app using HTML, CSS and Javascript doesn't mean that you have to give up on native functionality that makes mobile devices so extraordinary. PhoneGap gives you access to all of the native device APIs (like camera, GPS, accelerometer and more), so that the app you build with web tech behaves just like a native app. but still have some limitations.

TABLE 3.3: The Mobile App Comparison Chart: Native vs. Mobile Web vs.Hybrid

	Native	**HTML5**	**Hybrid**
App Features			
Graphics	Native APIs	HTML, Canvas, SVG	HTML, Canvas, SVG
Performance	Fast	Slow	Slow
Native look and feel	Native	Emulated	Emulated
Distribution	Appstore	Web	Appstore
Device Access			
Camera	Yes	No	Yes
Notifications	Yes	No	Yes
Contacts, calendar	Yes	No	Yes
Offline storage	Secure file storage	Shared SQL	Secure file, shared SQL
Geolocation	Yes	Yes	Yes
Gestures			
Swipe	Yes	Yes	Yes
Pinch, spread	Yes	No	Yes
Connectivity			
Connectivity	Online and offline	online	Online and offline
Developer Skills			
language	Objective C, Java	HTML5, CSS, Javascript	HTML5, CSS, Javascript

Since the front end of the application is built in JavaScript, it causes a number of limitations.

- **Data processing:** Native languages are much faster than JavaScript for data processing on the device.
- **Background processing:** A large number of applications rely on background threads to provide a smooth user experience: calculating the GPS positions in the background, for example. PhoneGap APIs are built using JavaScript which is not multi-threaded and hence do not support background processing.
- **Access advanced native functionality:** A number of native APIs are not yet supported by PhoneGap's APIs.
- **Complex Business Logic:** A number of applications such as enterprise applications are quite complex. In this scenario it is simply better to have a certain amount of native code.
- **Advanced Graphics:** Advanced Graphics: Apps that use advanced graphics which can only be accessed using third-party libraries are best done natively.

3.3.2 Titanium

The Titanium SDK helps you to build native cross-platform mobile application using JavaScript and the Titanium API, which abstracts the native APIs of the mobile platforms. Titanium empowers you to create immersive, full-featured applications, featuring over 80% code reuse across mobile apps. Appcelerator licenses Titanium under the Apache 2 license and is free for both personal and commercial use.

Titanium's unique trait among the various available cross-platform mobile solutions is that it creates truly native apps. This is in contrast to web-based solutions that deliver functionality via an enhanced web view. Titanium, not wanting to be limited by the native web view, has engaged in a much deeper integration with the underlying platforms. This gives Titanium developers the ability to leverage native UI components and services, as well as near-native performance.

Since the front end of the application is built in JavaScript, it causes a number of limitations.

- **Increasing complexity:** The development complexities (and costs) rise more than proportionally to application complexity increases. The more complex your applications become, the more often you'll have to deal with, on the one hand technical issues (random crashes, weird behaviors, annoying bugs, etc.), on the other hand a greater effort (code organization, MVC separation, multi-device support, multi-platform support, code readability, etc.)

- **No Freemium:** Appcelerator provides StoreKit, a module to enable In-App Purchase to Apple's App Store, but it's a pain. Buggy, poorly documented and it seems to work only partially. Too unstable for production use. Having to renounce the freemium pricing model (apps that are free to download, but require an in-app purchase to be expanded) is not just a minor inconvenience since 72% of total App Store revenue comes from apps featuring in-app purchases.

- **Toolkit pain:** At first there was only Titanium Developer but since last June there is Titanium Studio, an Eclipse-based IDE built on a modified version of Aptana that allows you to manage projects, test your mobile apps in the simulator or on device and automate app packaging. First of all, I sincerely hate Eclipse, yeah, Eclipse is free and the best open source IDE there is, but offers a very poor IDE experience. Most importantly Titanium Studio can go "crazy", encounter weird glitches, stop printing console messages, but the worst thing is when the build process start to ignore changes. You have to continually clean your project every time you make changes or restart with a brand new project. A productivity tool that is uncomfortable and unstable is not a productivity tool and a development tool that is unproductive is not a development tool.

- **Flexibility limitations:** All that glitters is not gold. At beginning you'll love the well-defined Titanium API and you will probably love it even more every time you discover a simple property to enable behaviors that would require several lines of code on Xcode. But sooner or later you will face strange bugs and limitations. For example, if you want to apply a cell background gradient to a grouped table (a very common and easy task with Objective-C) you get that the grouped table becomes plain and the gradient color makes the table slow when scrolling, and you will have to use images... So at first you will save a lot of time but as more complex the project grows you'll lose the saved time in fixing and workarounds.

- **Laggy:** Obviously you can have the most smooth, fast and comfortable user experience possible only with apps developed with a native development environment. This is an obvious observation, but which cannot be omitted. Keep in mind that a Titanium application is the result of an automatic conversion

process from web code to native code. Animations are noticeably laggy and apps are not responsive when return from the background. This is particularly evident with Android devices, less evident with iOS devices (especially those with A5 processor).

3.3.3 Xamarin

Launched in 2011, Xamarin is a mono framework used for cross-platform app development. Xamarin brings open source .NET to mobile development, enabling every developer to build truly native apps for any device in C# and F#. We're excited for your contributions in continuing our mission to make it fast, easy, and fun to build great mobile apps.

Xamarin is best for building applications using C# programming language running on .NET Common Language Infrastructure (CLI) (often called Microsoft .NET).

Over 1.5 million developers were using Xamarin's products in more than 120 countries around the world as of May 2015. It is widely used for communicating with the Application Program Interface (API) of common mobile device functions like contacts, camera, and geolocation for android, iOS, and Windows operating systems. It allows developers to use almost 100 % native libraries of both Android and iOS, With a C#-shared codebase, developers can use Xamarin tools to write native Android, iOS, and Windows apps with native user interfaces and share code across multiple platforms, including Windows and macOS.

On February 24, 2016, Microsoft announced it had signed a definitive agreement to acquire Xamarin.

Pros of Using Xamarin for Development

- **One Technology Stack to Code for All Platforms:** Xamarin uses C# complemented with .Net framework to create apps for any mobile platform. Thus, you can reuse up to 96 percent of the source code speeding up the engineering cycle. Xamarin also does not require switching between the development environments as it works with both Xamarin IDE (for Mac) or Visual Studio (for Windows). Although many developers argued about the quality of support provided by both IDEs, Xamarin Visual Studio integration has been largely improved since the company's acquisition by Microsoft. The cross-platform development tools are provided as a built-in part of the IDE at no additional cost.
- **Performance Close to Native:** AUnlike traditional hybrid solutions, based on the web technologies, a cross-platform app built with Xamarin, can still be classified as native. The performance metrics are comparable to those of Java for Android (as explained here) and Objective-C or Swift for native iOS app development. Moreover, the efficiency is constantly being improved to fully match the standards of native development. Xamarin platform offers a complete solution for testing and tracking the app's performance. Its' Xamarin Test Cloud paired with Xamarin Test Recorder tool allow you to run automated UI tests and identify performance issues before the release. However, this service is provided at an additional fee.

- **Native User Experiences:**

  Xamarin allows you to create flawless experiences using platform-specific UI elements. Simple cross-platform apps for iOS, Android or Windows are built using Xamarin.Forms tool, which converts app UI components into the platform-specific interface elements at runtime. As the use of Xamarin.Forms significantly increases the speed of app development, it is a great option for business-oriented projects. Yet, there might be a slight decline in performance due to the extra abstraction layer. For custom app UI and higher performance you can still use Xamarin.iOS and Xamarin.Android separately to ensure excellent results.

- **Full Hardware Support:**

  With Xamarin, your solution gets native-level app functionality. It eliminates all hardware compatibility issues, using plugins and specific APIs, to work with common devices functionality across the platforms. Along with the access to platform-specific APIs, Xamarin supports linking with native libraries. This allows for better customization and native-level functionality with little overhead.

Cons of Using Xamarin for Development

- **Expensive Xamarin License:**

  Business subscription comes at the annual fee of $999 per developer, per device platform, which might seem a little too high if you plan to create only one small app. For example, it will cost you almost $10,000 annually to run a team of five engineers, each building apps for iOS and Android. However, if you are going to build other cross-platform mobile solutions in the future or provide Xamarin app development services, Xamarin license would be a good investment compared to the development cost of native apps.

- **Slightly Delayed Support for the Latest Platform Updates:**

  This depends completely on the Xamarin developer team. It's impossible for third-party tools to provide the immediate support for the latest iOS and Android releases: it takes some time to implement the changes and/or introduce new plugins, etc. Although Xamarin claims to provide same-day support, there still might be some delays.

- **Limited Access to Open Source Libraries:**

  Native development makes extensive use of open source technologies. With Xamarin, you have to use only the components provided by the platform and some .Net open source resources, facing both developers and consumers. While the choice is not quite as rich as it is for Android and iOS mobile app development, the Xamarin Components provide thousands of custom UI controls, various charts, graphs, themes, and other powerful features that can be added to an app in just a few clicks. This includes built-in payment processing (such as Stripe), Beacons and wearables integration, out of the box push notification services, cloud storage solutions, multimedia streaming capabilities and much more.

- **Xamarin Ecosystem Problems:**

TABLE 3.4: The Mobile SDK Xamarin vs: Native vs Hybrid

	Xamarin	**Native**	**Hybrid**
Technology Stack	C# , .net framework, Native libraries	Different Technology stacks for each platform	JavaScript, Html5, CSS
Code Sharing	Yes (Upto 96%)	No	Yes (100%)
UI/UX	Completely Native Ui	Completely Native Ui	Limited Native Ui capabilties
Performance	Good, Close to Native	Excellent	Medium
Hardware Capabilites	Highly supported Xamarin uses platform specific APIs	High Native Tools have completely support	Medium with third party API and plugins .
Time to market	Fast	Time consuming	Faster

Obviously, Xamarin community is significantly smaller than those of iOS or Android. T hus, finding an experienced Xamarin developer could be a challenge. Although the platform is growing its following fueled by the support from Microsoft. Based on the info from different sources, Xamarin community makes 10 percent of the global mobile development society. Despite the fact that the number of Xamarin engineers does not compare to iOS or Android native communities, the platform provides extensive support to its developers. Namely, there is a dedicated educational platform, Xamarin University, that provides resources and practical training for those who are new to this technology. Using this support, the learning curve for an experienced C#.Net engineer is minimal.

3.3.4 Conclusion

When comparing the pros and cons, the listed drawbacks are usually considered to be a collateral damage. Most business owners choose Xamarin mobile app development platform as it decreases the time to market and engineering cost, by sharing the code and using a single technology stack. Yet the purpose of the app and its target audience might be an even more important factor to consider.

Based on our team's experience, the best use-case for Xamarin is enterprise mobile solutions. With standard UI which covers 90 percent of the projects, all the core product logic can be easily shared across the platforms. Hence, platform customization will only take 5-10 percent of the engineering effort.

In case of consumer-facing apps with heavy UI, the amount of shared code decreases drastically. Thus, Xamarin cross-platform development loses its major benefit and might equal in time and cost to native solutions.

However, if you are looking for a Xamarin alternative to build a cross-platform mobile app, you might be disappointed. While the most widely used cross-platform

mobile development tools are PhoneGap/Apache Cordova, Ionic Framework, Appcelerator/Titanium, they rely primarily on web technologies, such as HTM5 or JavaScript. That is why none of these tools can have the same level of performance and native functionality that Xamarin offers

Chapter 4

Introduction to Cross-platform Development with Xamarin

Xamarin offers sophisticated cross-platform support for the three major mobile platforms of iOS, Android, and Windows Phone. Applications can be written to share up to 90% of their code, and our Xamarin.Mobile library offers a unified API to access common resources across all three platforms. This can significantly reduce both development costs and time to market for mobile developers that target the three most popular mobile platforms.

4.1 Introduction to Xamarin

When considering how to build iOS and Android applications, many people think that the native languages, Objective-C, Swift, and Java, are the only choice. However, over the past few years, an entire new ecosystem of platforms for building mobile applications has emerged.

Xamarin is unique in this space by offering a single language – C#, class library, and runtime that works across all three mobile platforms of iOS, Android, and Windows Phone (Windows Phone's native language is already C#), while still compiling native (non-interpreted) applications that are performant enough even for demanding games.

Each of these platforms has a different feature set and each varies in its ability to write native applications – that is, applications that compile down to native code and that interop fluently with the underlying Java subsystem. For example, some platforms only allow apps to be built in HTML and JavaScript, whereas some are very low-level and only allow C/C++ code. Some platforms don't even utilize the native control toolkit.

Xamarin is unique in that it combines all of the power of the native platforms and adds a number of powerful features of its own, including:

- **Complete Binding for the underlying SDKs:** Xamarin contains bindings for nearly the entire underlying platform SDKs in both iOS and Android. Additionally, these bindings are strongly-typed, which means that they're easy to navigate and use, and provide robust compile-time type checking and during development. This leads to fewer runtime errors and higher quality applications.

- **Objective-C, Java, C, and C++ Interop:** Xamarin provides facilities for directly invoking Objective-C, Java, C, and C++ libraries, giving you the power to use a wide array of 3rd party code that has already been created. This lets you take advantage of existing iOS and Android libraries written in Objective-C, Java or C/C++. Additionally, Xamarin offers binding projects that allow you to easily bind native Objective-C and Java libraries using a declarative syntax.
- **Modern Language Constructs :** Xamarin applications are written in C#, a modern language that includes significant improvements over Objective-C and Java such as Dynamic Language Features , Functional Constructs such as Lambdas , LINQ , Parallel Programming features, sophisticated Generics , and more.
- **Amazing Base Class Library (BCL):** Xamarin applications use the .NET BCL, a massive collection of classes that have comprehensive and streamlined features such as powerful XML, Database, Serialization, IO, String, and Networking support, just to name a few. Additionally, existing C# code can be compiled for use in an applications, which provides access to thousands upon thousands of libraries that will let you do things that aren't already covered in the BCL.
- **Modern Integrated Development Environment (IDE):** Xamarin uses Xamarin Studio on Mac OS X and Visual Studio on Windows. These are both modern IDE's that include features such as code auto completion, a sophisticated Project and Solution management system, a comprehensive project template library, integrated source control, and many others.
- **Mobile Cross Platform Support:** Xamarin offers sophisticated cross-platform support for the three major mobile platforms of iOS, Android, and Windows Phone. Applications can be written to share up to 90% of their code, and our Xamarin.Mobile library offers a unified API to access common resources across all three platforms. This can significantly reduce both development costs and time to market for mobile developers that target the three most popular mobile platforms.

4.2 How Does Xamarin Work?

Xamarin offers two commercial products: Xamarin.iOS and Xamarin.Android. They're both built on top of Mono, an open-source version of the .NET Framework based on the published .NET ECMA standards. Mono has been around almost as long as the .NET framework itself, and runs on nearly every imaginable platform including Linux, Unix, FreeBSD, and Mac OS X.

On iOS, Xamarin's Ahead-of-Time (AOT) Compiler compiles Xamarin.iOS applications directly to native ARM assembly code. On Android, Xamarin's compiler compiles down to Intermediate Language (IL), which is then Just-in-Time (JIT) compiled to native assembly when the application launches.

In both cases, Xamarin applications utilize a runtime that automatically handles things such as memory allocation, garbage collection, underlying platform interop, etc.

4.3 Understanding the Xamarin Mobile Platform

Welcome to this LaTeX Thesis Template, a beautiful and easy to use template for writing a thesis using the LaTeX typesetting system.

If you are writing a thesis (or will be in the future) and its subject is technical or mathematical (though it doesn't have to be), then creating it in LaTeX is highly recommended as a way to make sure you can just get down to the essential writing without having to worry over formatting or wasting time arguing with your word processor.

LaTeX is easily able to professionally typeset documents that run to hundreds or thousands of pages long. With simple mark-up commands, it automatically sets out the table of contents, margins, page headers and footers and keeps the formatting consistent and beautiful. One of its main strengths is the way it can easily typeset mathematics, even *heavy* mathematics. Even if those equations are the most horribly twisted and most difficult mathematical problems that can only be solved on a super-computer, you can at least count on LaTeX to make them look stunning.

4.4 Architecture

A key tenet of building cross-platform apps is to create an architecture that lends itself to a maximization of code sharing across platforms. Adhering to the following Object Oriented Programming principles helps build a well-architected application:

- **Encapsulation:** Ensuring that classes and even architectural layers only expose a minimal API that performs their required functions, and hides the implementation details. At a class level, this means that objects behave as 'black boxes' and that consuming code does not need to know how they accomplish their tasks. At an architectural level, it means implementing patterns like Façade that encourage a simplified API that orchestrates

- **Complete Binding for the underlying SDKs:** more complex interactions on behalf of the code in more abstract layers. This means that the UI code (for example) should only be responsible for displaying screens and accepting user-input; and never interacting with the database directly. Similarly the data-access code should only read and write to the database, but never interact directly with buttons or labels.

- **Separation of Responsibilities:** Ensure that each component (both at architectural and class level) has a clear and well-defined purpose. Each component should perform only its defined tasks and expose that functionality via an API that is accessible to the other classes that need to use it.

- **Polymorphism :** Programming to an interface (or abstract class) that supports multiple implementations means that core code can be written and shared across platforms, while still interacting with platform-specific features.

The natural outcome is an application modeled after real world or abstract entities with separate logical layers. Separating code into layers make applications easier to understand, test and maintain. It is recommended that the code in each layer be physically separate (either in directories or even separate projects for very large applications) as well as logically separate (using namespaces).

4.4.1 Typical Application Layers

Throughout this document and the case studies we refer to the following six application layers:

- **Data Layer:** – Non-volatile data persistence, likely to be a SQLite database but could be implemented with XML files or any other suitable mechanism.
- **Data Access Layer :** – Wrapper around the Data Layer that provides Create, Read, Update, Delete (CRUD) access to the data without exposing implementation details to the caller. For example, the DAL may contain SQL statements to query or update the data but the referencing code would not need to know this.
- **Business Layer:** – (sometimes called the Business Logic Layer or BLL) contains business entity definitions (the Model) and business logic. Candidate for Business Façade pattern.
- **Service Access Layer:** – Used to access services in the cloud: from complex web services (REST, JSON, WCF) to simple retrieval of data and images from remote servers. Encapsulates the networking behavior and provides a simple API to be consumed by the Application and UI layers.
- **Application Layer:** – Code that's typically platform specific (not generally shared across platforms) or code that is specific to the application (not generally reusable). A good test of whether to place code in the Application Layer versus the UI Layer is (a) to determine whether the class has any actual display controls or (b) whether it could be shared between multiple screens or devices (eg. iPhone and iPad).
- **User Interface (UI) Layer:** – The user-facing layer, contains screens, widgets and the controllers that manage them.

An application may not necessarily contain all layers – for example the Service Access Layer would not exist in an application that does not access network resources. A very simple application might merge the Data Layer and Data Access Layer because the operations are extremely basic.

4.4.2 Common Mobile Software Patterns

Patterns are an established way to capture recurring solutions to common problems. There are a few key patterns that are useful to understand in building maintainable/understandable mobile applications.

- **Model, View, ViewModel (MVVM):** The Model-View-ViewModel pattern is popular with frameworks that support data-binding, such as Xamarin.Forms. It was popularized by XAML-enabled SDKs like Windows Presentation Foundation (WPF) and Silverlight; where the ViewModel acts as a go-between between the data (Model) and user interface (View) via data binding and commands.
- **Model, View, Controller (MVC):** A common and often misunderstood pattern, MVC is most often used when building User Interfaces and provides for a separation between the actual definition of a UI Screen (View), the engine

behind it that handles interaction (Controller), and the data that populates it (Model). The model is actually a completely optional piece and therefore, the core of understanding this pattern lies in the View and Controller. MVC is a popular approach for iOS applications.

- **Business Façade:** AKA Manager Pattern, provides a simplified point of entry for complex work. For example, in a Task Tracking application, you might have a TaskManager class with methods such as GetAllTasks() , GetTask(taskID) , SaveTask (task) , etc. The TaskManager class provides a Façade to the inner workings of actually saving/retrieving of tasks objects.

- **Singleton:** The Singleton pattern provides for a way in which only a single instance of a particular object can ever exist. For example, when using SQLite in mobile applications, you only ever want one instance of the database. Using the Singleton pattern is a simple way to ensure this.

- **Provider:** A pattern coined by Microsoft (arguably similar to Strategy, or basic Dependency Injection) to encourage code re-use across Silverlight, WPF and WinForms applications. Shared code can be written against an interface or abstract class, and platform-specific concrete implementations are written and passed in when the code is used.

- **Async:** Not to be confused with the Async keyword, the Async pattern is used when long-running work needs to be executed without holding up the UI or current processing. In its simplest form, the Async pattern simply describes that long-running tasks should be kicked off in another thread (or similar thread abstraction such as a Task) while the current thread continues to process and listens for a response from the background process, and then updates the UI when data and or state is returned.

Each of the patterns will be examined in more detail as their practical use is illustrated in the case studies. Wikipedia has more detailed descriptions of the MVVM, MVC, Facade, Singleton, Strategy and Provider patterns (and of Design Patterns generally).

4.5 Setting Up A Xamarin Cross Platform Solution

Regardless of what platforms are being used, Xamarin projects all use the same solution file format (the Visual Studio .sln file format). Solutions can be shared across development environments, even when individual projects cannot be loaded (such as a Windows project in Xamarin Studio).

When creating a new cross platform application, the first step is to create a blank solution. This section what happens next: setting up the projects for building cross platform mobile apps

4.5.1 Code Sharing Options

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Apple's Pages. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write

the `\emph{text}` command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

4.5.2 Portable Class Libraries

Historically a .NET project file (and the resulting assembly) has been targeted to a specific framework version. This prevents the project or the assembly being shared by different frameworks.

A Portable Class Library (PCL) is a special type of project that can be used across disparate CLI platforms such as Xamarin.iOS and Xamarin.Android, as well as WPF, Universal Windows Platform, and Xbox. The library can only utilize a subset of the complete .NET framework, limited by the platforms being targeted.

4.5.3 Shared Projects

The simplest approach to sharing code files is use a Shared Project.

This method allows you to share the same code across different platform projects, and use compiler directives to include different, platform-specific code paths.

Shared Projects (also sometimes called Shared Asset Projects) let you write code that is shared between multiple target projects including Xamarin applications.

They support compiler directives so that you can conditionally include platform-specific code to be compiled into a subset of the projects that are referencing the Shared Project. There is also IDE support to help manage the compiler directives and visualize how the code will look in each application.

If you have used file-linking in the past to share code between projects, Shared Projects works in a similar way but with much improved IDE support.

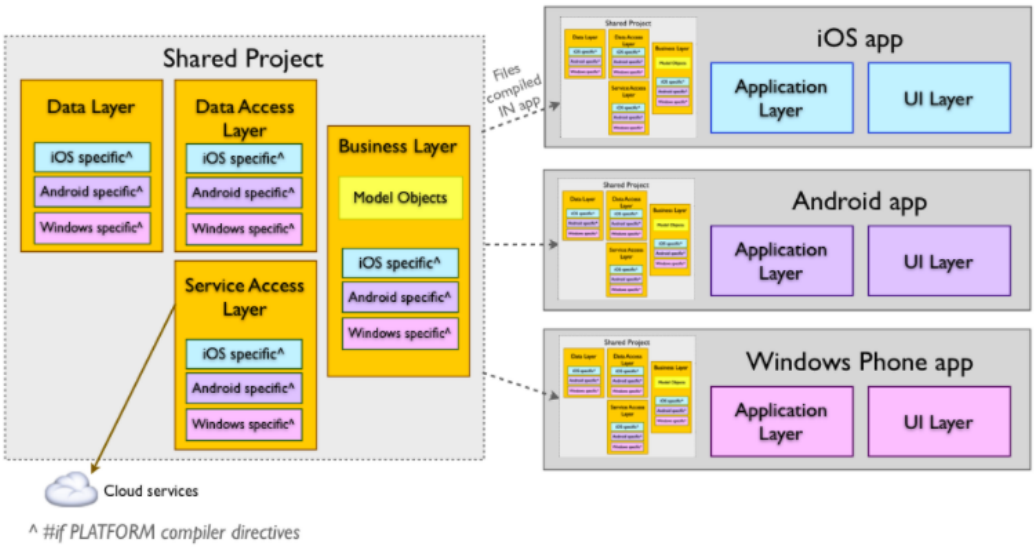

FIGURE 4.1: Shared Projects application architecture.

Shared Project Example

4.5.4 .NET Standard

Introduced in 2016, .NET Standard projects provide an easy way to share code across platforms, producing assemblies that can be used across Windows, Xamarin platforms (iOS, Android, Mac), and Linux.

.NET Standard libraries can be created and used like PCLs, except that the APIs available in each version (from 1.0 to 1.6) are more easily discovered and each version is backwards-compatible with lower version numbers.

4.5.5 Core Project

Shared code projects should only reference assemblies that are available across all platforms – ie. the common framework namespaces like System, System.Core and System.Xml.

Shared projects should implement as much non-UI functionality as is possible, which could include the following layers:

- **Data Layer :** Code that takes care of physical data storage eg. SQLite-NET, an alternative database like Realm.io or even XML files. The data layer classes are normally only used by the data access layer.
- **Data Access Layer:** Defines an API that supports the required data operations for the application's functionality, such as methods to access lists of data, individual data items and also
- **Service Access Layer :** An optional layer to provide cloud services to the application. Contains code that accesses remote network resources (web services, image downloads, etc) and possibly caching of the results.
- **Business Layer:** Definition of the Model classes and the Façade or Manager classes that expose functionality to the platform-specific applications.

4.5.6 Platform-Specific Application Projects

Platform-specific projects must reference the assemblies required to bind to each platform's SDK (Xamarin.iOS, Xamarin.Android, Xamarin.Mac, or Windows) as well as the Core shared code project.

The platform-specific projects should implement:

- **Application Layer:** Platform specific functionality and binding/conversion between the Business Layer objects and the user interface.
- **User Interface Layer:** Screens, custom user-interface controls, presentation of validation logic.

4.6 Dealing with Multiple Platforms

If you are familiar with LaTeX, then you should explore the directory structure of the template and then proceed to place your own information into the *THESIS INFORMATION* block of the **`main.tex`** file. You can then modify the rest of this file to your unique specifications based on your degree/university. Section **??** on page **??** will help you do this. Make sure you also read section **??** about thesis conventions to get the most out of this template.

If you are new to LaTeX it is recommended that you carry on reading through the rest of the information in this document.

Before you begin using this template you should ensure that its style complies with the thesis style guidelines imposed by your institution. In most cases this template style and layout will be suitable. If it is not, it may only require a small change

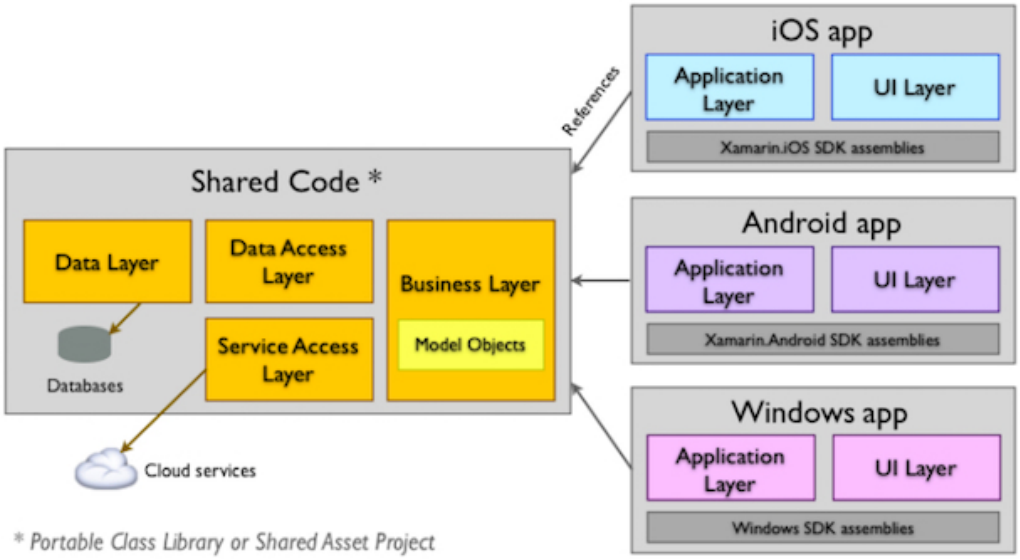

FIGURE 4.2: application architecture.

to bring the template in line with your institution's recommendations. These modifications will need to be done on the **`MastersDoctoralThesis.cls`** file.

4.7 Practical Code Sharing Strategies

If you are familiar with LaTeX, then you should explore the directory structure of the template and then proceed to place your own information into the *THESIS INFORMATION* block of the **`main.tex`** file. You can then modify the rest of this file to your unique specifications based on your degree/university. Section **??** on page **??** will help you do this. Make sure you also read section **??** about thesis conventions to get the most out of this template.

If you are new to LaTeX it is recommended that you carry on reading through the rest of the information in this document.

Before you begin using this template you should ensure that its style complies with the thesis style guidelines imposed by your institution. In most cases this template style and layout will be suitable. If it is not, it may only require a small change to bring the template in line with your institution's recommendations. These modifications will need to be done on the **`MastersDoctoralThesis.cls`** file.

4.8 Dealing with Multiple Platforms

If you are familiar with LaTeX, then you should explore the directory structure of the template and then proceed to place your own information into the *THESIS INFORMATION* block of the **`main.tex`** file. You can then modify the rest of this file to your unique specifications based on your degree/university. Section **??** on page **??** will help you do this. Make sure you also read section **??** about thesis conventions to get the most out of this template.

If you are new to LaTeX it is recommended that you carry on reading through the rest of the information in this document.

Before you begin using this template you should ensure that its style complies with the thesis style guidelines imposed by your institution. In most cases this template style and layout will be suitable. If it is not, it may only require a small change to bring the template in line with your institution's recommendations. These modifications will need to be done on the **`MastersDoctoralThesis.cls`** file.

4.9 Cross-Platform User Interfaces with Xamarin.Forms

If you are familiar with LaTeX, then you should explore the directory structure of the template and then proceed to place your own information into the *THESIS INFORMATION* block of the **`main.tex`** file. You can then modify the rest of this file to your unique specifications based on your degree/university. Section **??** on page **??** will help you do this. Make sure you also read section **??** about thesis conventions to get the most out of this template.

4.9.1 eXtensible Application Markup Language (XAML)

If you are new to LaTeX it is recommended that you carry on reading through the rest of the information in this document.

Before you begin using this template you should ensure that its style complies with the thesis style guidelines imposed by your institution. In most cases this template style and layout will be suitable. If it is not, it may only require a small change to bring the template in line with your institution's recommendations. These modifications will need to be done on the **`MastersDoctoralThesis.cls`** file.

4.10 Testing and App Store Approvals

Many apps (even Android apps, on some stores) will have to pass an approval process before they are published; so testing is critical to ensure your app reaches the market (let alone succeeds with your customers). Testing can take many forms, from developer-level unit testing to managing beta testing across a wide variety of hardware.

4.10.1 Test on All Platforms

TThere are slight differences between what .NET supports on Windows phone, tablet, and desktop devices, as well as limitations on iOS that prevent dynamic code to be generated on the fly. Either plan on testing the code on multiple platforms as you develop it, or schedule time to refactor and update the model part of your application at the end of the project.

It is always good practice to use the simulator/emulator to test multiple versions of the operating system and also different device capabilities/configurations.

You should also test on as many different physical hardware devices as you can.

Devices in cloud The mobile phone and tablet ecosystem is growing all the time, making it impossible to test on the ever-increasing number of devices available. To solve this problem a number of services offer the ability to remotely control many different devices so that applications can be installed and tested without needing to directly invest in lots of hardware.

Xamarin Test Cloud offers an easy way to test iOS and Android appliations on hundreds of different devices.

4.10.2 Test Management

When testing applications within your organization or managing a beta program with external users, there are two challenges:

Distribution – Managing the provisioning process (especially for iOS devices) and getting updated versions of software to the testers. Feedback – Collecting information about application usage, and detailed information on any errors that may occur. There are a number of services help to address these issues, by providing infrastructure that is built into your application to collect and report on usage and errors, and also streamlining the provisioning process to help sign-up and manage testers and their devices.

The Xamarin Insights Preview offers a solution to the second part of this issue, providing crash reporting and sophisticated application usage information.

4.10.3 Test Automation

Xamarin UITest can be used to create automated user interface test scripts that can be run locally or uploaded to Test Cloud.

4.10.4 Unit Testing

Touch.Unit Xamarin.iOS includes a unit-testing framework called Touch.Unit which follows the JUnit/NUnit style writing tests.

Refer to our Unit Testing with Xamarin.iOS documentation for details on writing tests and running Touch.Unit.

Andr.Unit There is an open-source equivalent of Touch.Unit for Android called Andr.Unit. You can download it from github and read about the tool on @spouliot's blog.

Windows Phone Here are some links to help setup unit testing for Windows Phone:

Chapter 5

Introduction to Microsoft azure

5.1 Connected Services in Xamarin Studio

Welcome to this LaTeX Thesis Template, a beautiful and easy to use template for writing a thesis using the LaTeX typesetting system. If you are writing a thesis (or will be in the future) and its subject is technical or mathematical (though it doesn't have to be), then creating it in LaTeX is highly recommended as a way to make sure you can just get down to the essential writing without having to worry over formatting or wasting time arguing with your word processor.

LaTeX is easily able to professionally typeset documents that run to hundreds or thousands of pages long. With simple mark-up commands, it automatically sets out the table of contents, margins, page headers and footers and keeps the formatting consistent and beautiful. One of its main strengths is the way it can easily typeset mathematics, even *heavy* mathematics. Even if those equations are the most horribly twisted and most difficult mathematical problems that can only be solved on a super-computer, you can at least count on LaTeX to make them look stunning.

5.2 Azure App Services

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Apple's Pages. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write the `\emph{text}` command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

5.2.1 A (not so short) Introduction to LaTeX

If you are new to LaTeX, there is a very good eBook – freely available online as a PDF file – called, "The Not So Short Introduction to LaTeX". The book's title is typically shortened to just *lshort*. You can download the latest version (as it is occasionally updated) from here: http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

It is also available in several other languages. Find yours from the list on this page: http://www.ctan.org/tex-archive/info/lshort/

It is recommended to take a little time out to learn how to use LaTeX by creating several, small 'test' documents, or having a close look at several templates on: http://www.LaTeXTemplates.com
Making the effort now means you're not stuck learning the system when what you *really* need to be doing is writing your thesis.

5.2.2 A Short Math Guide for LaTeX

If you are writing a technical or mathematical thesis, then you may want to read the document by the AMS (American Mathematical Society) called, "A Short Math Guide for LaTeX". It can be found online here: http://www.ams.org/tex/amslatex.html under the "Additional Documentation" section towards the bottom of the page.

5.2.3 Common LaTeX Math Symbols

There are a multitude of mathematical symbols available for LaTeX and it would take a great effort to learn the commands for them all. The most common ones you are likely to use are shown on this page: http://www.sunilpatel.co.uk/latex-type/latex-math-symbols/

You can use this page as a reference or crib sheet, the symbols are rendered as large, high quality images so you can quickly find the LaTeX command for the symbol you need.

5.2.4 LaTeX on a Mac

The LaTeX distribution is available for many systems including Windows, Linux and Mac OS X. The package for OS X is called MacTeX and it contains all the applications you need – bundled together and pre-customized – for a fully working LaTeX environment and work flow.

MacTeX includes a custom dedicated LaTeX editor called TeXShop for writing your '**.tex**' files and BibDesk: a program to manage your references and create your bibliography section just as easily as managing songs and creating playlists in iTunes.

5.3 Active Directory Authentication

If you are familiar with LaTeX, then you should explore the directory structure of the template and then proceed to place your own information into the *THESIS INFORMATION* block of the **main.tex** file. You can then modify the rest of this file to your unique specifications based on your degree/university. Section **??** on page **??** will help you do this. Make sure you also read section **??** about thesis conventions to get the most out of this template.

If you are new to LaTeX it is recommended that you carry on reading through the rest of the information in this document.

Before you begin using this template you should ensure that its style complies with the thesis style guidelines imposed by your institution. In most cases this template style and layout will be suitable. If it is not, it may only require a small change to bring the template in line with your institution's recommendations. These modifications will need to be done on the **MastersDoctoralThesis.cls** file.

5.3.1 WebAPI

This LaTeX Thesis Template is originally based and created around a LaTeX style file created by Steve R. Gunn from the University of Southampton (UK), department of Electronics and Computer Science. You can find his original thesis style

file at his site, here: http://www.ecs.soton.ac.uk/~srg/softwaretools/document/templates/

Steve's **ecsthesis.cls** was then taken by Sunil Patel who modified it by creating a skeleton framework and folder structure to place the thesis files in. The resulting template can be found on Sunil's site here: http://www.sunilpatel.co.uk/thesis-template

Sunil's template was made available through http://www.LaTeXTemplates.com where it was modified many times based on user requests and questions. Version 2.0 and onwards of this template represents a major modification to Sunil's template and is, in fact, hardly recognisable. The work to make version 2.0 possible was carried out by Vel and Johannes Böttcher.

Chapter 6

Chapter Title Here

6.1 Welcome and Thank You

Welcome to this LaTeX Thesis Template, a beautiful and easy to use template for writing a thesis using the LaTeX typesetting system.

If you are writing a thesis (or will be in the future) and its subject is technical or mathematical (though it doesn't have to be), then creating it in LaTeX is highly recommended as a way to make sure you can just get down to the essential writing without having to worry over formatting or wasting time arguing with your word processor.

LaTeX is easily able to professionally typeset documents that run to hundreds or thousands of pages long. With simple mark-up commands, it automatically sets out the table of contents, margins, page headers and footers and keeps the formatting consistent and beautiful. One of its main strengths is the way it can easily typeset mathematics, even *heavy* mathematics. Even if those equations are the most horribly twisted and most difficult mathematical problems that can only be solved on a super-computer, you can at least count on LaTeX to make them look stunning.

6.2 Learning LaTeX

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Apple's Pages. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write the `\emph{text}` command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

6.2.1 A (not so short) Introduction to LaTeX

If you are new to LaTeX, there is a very good eBook – freely available online as a PDF file – called, "The Not So Short Introduction to LaTeX". The book's title is typically shortened to just *lshort*. You can download the latest version (as it is occasionally updated) from here: http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

It is also available in several other languages. Find yours from the list on this page: http://www.ctan.org/tex-archive/info/lshort/

It is recommended to take a little time out to learn how to use LaTeX by creating several, small 'test' documents, or having a close look at several templates on: http://www.LaTeXTemplates.com

Making the effort now means you're not stuck learning the system when what you *really* need to be doing is writing your thesis.

Chapter 7

Chapter Title Here

7.1 Welcome and Thank You

Welcome to this LaTeX Thesis Template, a beautiful and easy to use template for writing a thesis using the LaTeX typesetting system.

If you are writing a thesis (or will be in the future) and its subject is technical or mathematical (though it doesn't have to be), then creating it in LaTeX is highly recommended as a way to make sure you can just get down to the essential writing without having to worry over formatting or wasting time arguing with your word processor.

LaTeX is easily able to professionally typeset documents that run to hundreds or thousands of pages long. With simple mark-up commands, it automatically sets out the table of contents, margins, page headers and footers and keeps the formatting consistent and beautiful. One of its main strengths is the way it can easily typeset mathematics, even *heavy* mathematics. Even if those equations are the most horribly twisted and most difficult mathematical problems that can only be solved on a super-computer, you can at least count on LaTeX to make them look stunning.

7.2 Learning LaTeX

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Apple's Pages. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write the `\emph{text}` command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

7.2.1 A (not so short) Introduction to LaTeX

If you are new to LaTeX, there is a very good eBook – freely available online as a PDF file – called, "The Not So Short Introduction to LaTeX". The book's title is typically shortened to just *lshort*. You can download the latest version (as it is occasionally updated) from here: `http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf`

It is also available in several other languages. Find yours from the list on this page: `http://www.ctan.org/tex-archive/info/lshort/`

It is recommended to take a little time out to learn how to use LaTeX by creating several, small 'test' documents, or having a close look at several templates on: `http://www.LaTeXTemplates.com`

Making the effort now means you're not stuck learning the system when what you *really* need to be doing is writing your thesis.

Chapter 8

Chapter Title Here

8.1 Welcome and Thank You

Welcome to this LaTeX Thesis Template, a beautiful and easy to use template for writing a thesis using the LaTeX typesetting system.

If you are writing a thesis (or will be in the future) and its subject is technical or mathematical (though it doesn't have to be), then creating it in LaTeX is highly recommended as a way to make sure you can just get down to the essential writing without having to worry over formatting or wasting time arguing with your word processor.

LaTeX is easily able to professionally typeset documents that run to hundreds or thousands of pages long. With simple mark-up commands, it automatically sets out the table of contents, margins, page headers and footers and keeps the formatting consistent and beautiful. One of its main strengths is the way it can easily typeset mathematics, even *heavy* mathematics. Even if those equations are the most horribly twisted and most difficult mathematical problems that can only be solved on a super-computer, you can at least count on LaTeX to make them look stunning.

8.2 Learning LaTeX

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Apple's Pages. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write the `\emph{text}` command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

8.2.1 A (not so short) Introduction to LaTeX

If you are new to LaTeX, there is a very good eBook – freely available online as a PDF file – called, "The Not So Short Introduction to LaTeX". The book's title is typically shortened to just *lshort*. You can download the latest version (as it is occasionally updated) from here: `http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf`

It is also available in several other languages. Find yours from the list on this page: `http://www.ctan.org/tex-archive/info/lshort/`

It is recommended to take a little time out to learn how to use LaTeX by creating several, small 'test' documents, or having a close look at several templates on: `http://www.LaTeXTemplates.com`

Making the effort now means you're not stuck learning the system when what you *really* need to be doing is writing your thesis.

8.2.2 A Short Math Guide for LaTeX

If you are writing a technical or mathematical thesis, then you may want to read the document by the AMS (American Mathematical Society) called, "A Short Math Guide for LaTeX". It can be found online here: `http://www.ams.org/tex/amslatex.html` under the "Additional Documentation" section towards the bottom of the page.

8.2.3 Common LaTeX Math Symbols

There are a multitude of mathematical symbols available for LaTeX and it would take a great effort to learn the commands for them all. The most common ones you are likely to use are shown on this page: `http://www.sunilpatel.co.uk/latex-type/latex-math-symbols/`

You can use this page as a reference or crib sheet, the symbols are rendered as large, high quality images so you can quickly find the LaTeX command for the symbol you need.

8.2.4 LaTeX on a Mac

The LaTeX distribution is available for many systems including Windows, Linux and Mac OS X. The package for OS X is called MacTeX and it contains all the applications you need – bundled together and pre-customized – for a fully working LaTeX environment and work flow.

MacTeX includes a custom dedicated LaTeX editor called TeXShop for writing your '`.tex`' files and BibDesk: a program to manage your references and create your bibliography section just as easily as managing songs and creating playlists in iTunes.

Chapter 9

Chapter Title Here

9.1 Welcome and Thank You

Welcome to this LaTeX Thesis Template, a beautiful and easy to use template for writing a thesis using the LaTeX typesetting system.

If you are writing a thesis (or will be in the future) and its subject is technical or mathematical (though it doesn't have to be), then creating it in LaTeX is highly recommended as a way to make sure you can just get down to the essential writing without having to worry over formatting or wasting time arguing with your word processor.

LaTeX is easily able to professionally typeset documents that run to hundreds or thousands of pages long. With simple mark-up commands, it automatically sets out the table of contents, margins, page headers and footers and keeps the formatting consistent and beautiful. One of its main strengths is the way it can easily typeset mathematics, even *heavy* mathematics. Even if those equations are the most horribly twisted and most difficult mathematical problems that can only be solved on a super-computer, you can at least count on LaTeX to make them look stunning.

9.2 Learning LaTeX

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Apple's Pages. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write the `\emph{text}` command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

9.2.1 A (not so short) Introduction to LaTeX

If you are new to LaTeX, there is a very good eBook – freely available online as a PDF file – called, "The Not So Short Introduction to LaTeX". The book's title is typically shortened to just *lshort*. You can download the latest version (as it is occasionally updated) from here: http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

It is also available in several other languages. Find yours from the list on this page: http://www.ctan.org/tex-archive/info/lshort/

It is recommended to take a little time out to learn how to use LaTeX by creating several, small 'test' documents, or having a close look at several templates on: http://www.LaTeXTemplates.com

Making the effort now means you're not stuck learning the system when what you *really* need to be doing is writing your thesis.

Chapter 10

Conclusion

10.1 Goal Fulfilment

Welcome to this LaTeX Thesis Template, a beautiful and easy to use template for writing a thesis using the LaTeX typesetting system.

If you are writing a thesis (or will be in the future) and its subject is technical or mathematical (though it doesn't have to be), then creating it in LaTeX is highly recommended as a way to make sure you can just get down to the essential writing without having to worry over formatting or wasting time arguing with your word processor.

LaTeX is easily able to professionally typeset documents that run to hundreds or thousands of pages long. With simple mark-up commands, it automatically sets out the table of contents, margins, page headers and footers and keeps the formatting consistent and beautiful. One of its main strengths is the way it can easily typeset mathematics, even *heavy* mathematics. Even if those equations are the most horribly twisted and most difficult mathematical problems that can only be solved on a super-computer, you can at least count on LaTeX to make them look stunning.

10.2 Future Work

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Apple's Pages. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write the `\emph{text}` command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

Appendix A

Frequently Asked Questions

A.1 How do I change the colors of links?

The color of links can be changed to your liking using:

`\hypersetup{urlcolor=red}`, or

`\hypersetup{citecolor=green}`, or

`\hypersetup{allcolor=blue}`.

If you want to completely hide the links, you can use:

`\hypersetup{allcolors=.}`, or even better:

`\hypersetup{hidelinks}`.

If you want to have obvious links in the PDF but not the printed text, use:

`\hypersetup{colorlinks=false}`.

Bibliography

Chau, Melissa, Navina Govindaraj, and Ryan Reith (2017). "Smartphone OS Market Share, 2016 Q3." In: *International Data Corporation Smartphone OS Market Share, 2016 Q3* 72.1, pp. 1–1. URL: http://www.idc.com/promo/smartphone-market-share/os.