



Linnæus University

Sweden

Degree project

Cross-Platform Mobile Development:

An Alternative to Native Mobile Development



Author: Suyesh Amatya
Supervisor: Dr. Arianit Kurti

Date: 2013-10-29

Course Code: 5DV00E, 30 credits
Level: Masters

Department of Computer Science

Abstract

Mobile devices and mobile computing have made tremendous advances and become ubiquitous in the last few years. As a result, the landscape has become seriously *fragmented* which brings lots of challenges for the mobile development process. Whilst *native* approach of mobile development still is the predominant way to develop for a particular mobile platform, recently there is shifting towards *cross-platform* mobile development as well.

In this thesis, a survey of the literature has been performed to see the trends in cross-platform mobile development over the last few years. With the result of the survey, it is argued that the *web-based approach* and in particular, *hybrid approach*, of mobile development serves the best for cross-platform development. Using the hybrid approach, a prototype application has also been developed and built into native application for different platforms. This has helped to get a better insight about the domain of cross-platform mobile development and its main advantage of the unification of the development and testing process.

The results of this work indicate that even though cross platform tools are not fully matured they show great potential and reduce the cost associated in developing native mobile applications. Cross-platform mobile development is equally suitable for rapid development of high-fidelity prototypes of the mobile application as well as fairly complex, resource intensive mobile applications on its own right. As the upcoming future trends and the evolution of HTML5 continues to redefine the web, allowing its growth as a software platform, there remains great opportunities for cross-platform mobile development and hence provides an attractive alternative for the native mobile development.

Keywords

Mobile development, literature survey, web-based approach, hybrid approach, cross-platform mobile frameworks, HTML5, jQuery Mobile, PhoneGap, Google Maps, Android, iOS, BlackBerry.

Table of Contents

1. Introduction	1
1.1 Challenges in Mobile Development	1
1.2 Problem Definition	3
1.3 Structure of the Thesis	3
2. State of the Art	5
2.1 Survey of Literature	5
2.2 Planning the Survey (Methods)	5
2.2.1 Choice of Keywords	5
2.2.2 Sources Selection	6
2.2.3 Search Method	6
2.2.4 Inclusion Criteria	7
2.2.5 Timeline of the Literature	7
2.3 Conducting the Survey	8
2.4 Lessons Learned from Survey Results	11
3. Cross-Platform Mobile Development	15
3.1 Cross-Platform Mobile Development Frameworks Comparison on Different Parameters	15
3.2 Decision on Cross-Platform Mobile Development Framework	20
4. Application Concept and Design	21
4.1 Adhering to the Comparison Parameters/Assessment Matrix	21
4.2 Identifying the Requirements and Use Case Modeling	21
4.2.1 Home Page Functional Requirements As Use Case Modeling	22
4.2.2 Home Page Non-Functional Requirements	22
4.2.3 Detail Page Functional Requirements As Use Case Modeling	23
4.2.4 Detail Page Non-Functional Requirements	24
4.3 Application Architecture	24
4.4 Identifying the Appropriate Technological Requirements	25
4.4.1 Google Maps JavaScript API v3	25
4.4.2 HTML5	25
4.4.3 HTML5 Geolocation API	26
4.4.4 jQuery	26
4.4.5 jQuery Mobile	26

4.4.6 CSS	26
4.4.7 Apache Cordova/PhoneGap	27
4.5 Identifying and Preparing the Suitable Data Structure	27
4.5.1 Maintaining a Repository of Bus Stations	28
4.5.2 Pulling Data off HTML files	29
5. Implementing the Application	31
5.1 Developing a Single General Codebase	31
5.1.1 Application Structure	31
5.1.2 Custom CSS	33
5.1.3 Page Initialization	34
5.1.4 Is Geolocation Supported?	34
5.1.5 AJAX Call to the Stations Repository	34
5.1.6 Getting the Current Device Position	35
5.1.7 Calculating and Storing the Distance/Duration to All the Stations From the Current Position	35
5.1.8 Sorting the Nearest Five Stations and Drawing Them on the Map.....	36
5.1.9 Populating the Menu Panel and Registering Click Event to the Items	37
5.1.10 Drawing the Map	37
5.1.11 Constructing the Detail Page	38
5.2 Resulting Web Application	40
5.3 Building into Native Android Application	42
5.4 Building into Native iOS Application	45
5.5 Building into Native BlackBerry Application	48
6. Analysis of the Application and Development Efforts	51
6.1 Reflecting Upon the Development Efforts	51
6.2 Analysis Results	52
7. Conclusion	54
7.1 Answering the Research Questions	54
7.2 Future Directions	56
References	57

Table of Figures

Figure 1.1: Sampling of the mobile platforms and their various programming languages and devices	2
Figure 1.2: Thesis structure	4
Figure 2.1: Search method using the query string	7
Figure 2.2: Timeline of the publications in the conferences	8
Figure 2.3: Different solution approaches and use of cross-platform tools	12
Figure 2.4: The continuum of mobile application development	13
Figure 4.1: Home page use case diagram	22
Figure 4.2: Detail page use case diagram	23
Figure 4.3: Web-based client-server application architecture	24
Figure 5.1: Home screen on the Chrome browser	40
Figure 5.2: Menu panel expanded on Internet Explorer	41
Figure 5.3: Detail page as a normal tabular style presentation on maximized Mozilla Firefox	41
Figure 5.4: Detail page takes stacked presentation style when Mozilla Firefox window is scaled down	42
Figure 5.5: Home screen of the application	44
Figure 5.6: Menu panel	44
Figure 5.7: Manual station selection	45
Figure 5.8: Stacked style detail page	45
Figure 5.9: Normal tabular style detail page	45
Figure 5.10: Home screen on iPhone 6.1 simulator	46
Figure 5.11: Manual station selection	46
Figure 5.12: Menu panel expanded on iPad 6.1 simulator	47
Figure 5.13: Detail page	47
Figure 5.14: Home page and detail page on iPod touch	48
Figure 5.15: Home page on platform WebWorks-TabletOS and device BlackBerry PlayBook	49
Figure 5.16: Detail page on platform BlackBerry 10 WebWorks and device BlackBerry Q10	49
Figure 5.17: On platform BlackBerry 10 WebWorks and device BlackBerry Q10	49
Figure 5.18: On platform WebWorks and device BlackBerry Bold 9700	50

List of Tables

Table 2.1: Survey results summarized	11
Table 3.1: Frameworks comparison results scored	20
Table 6.1: Technical analysis of the application and development efforts across different platforms	52

1. Introduction

“With 5.9 billion mobile-cellular subscriptions, global penetration reaches 87% and 79% in the developing world. Mobile-broadband subscriptions have grown 45% annually over the last four years and today there are twice as many mobile-broadband as fixed-broadband subscriptions.” [1]

With the rapid technological advancements in both hardware and software fronts, coupled with broadband internet and World Wide Web, mobile computing has become ubiquitous. People use different varieties of mobile devices (tablets, smartphones, PDAs, etc) for all sorts of different purposes; want to know when the next bus leaves, watch online movies, learn a recipe for pasta, buy a ticket for the weekend game; you name it. Just the total “*smartphone*” shipment volumes alone reached 712.6 million units in 2012, up a strong 44.1% than in the year 2011 [2]. This prodigious growth in mobile devices is equally complimented by the growth in mobile content or information that these devices consume. According to the research group Gartner Inc., worldwide mobile app store downloads surpassed 45.6 billion in 2012, nearly double the 25 billion downloads in 2011 which by 2016 will reach 310 billion downloads and \$74 billion in revenue [3]. Scott Ellison, vice president, Mobile and Wireless research at IDC says

"Mobile app developers will 'appify' just about every interaction you can think of in your physical and digital worlds. The extension of mobile apps to every aspect of our personal and business lives will be one of the hallmarks of the new decade with enormous opportunities for virtually every business sector." [4]

We have become an “always-connected” society. Hence it will not be an overstatement to say that mobile devices have made inroads into our lives and completely revolutionized the way we live over the last few years.

1.1 Challenges in Mobile Development

Amidst so much of seeming opportunities, there also lie huge challenges in the development of content/ information or applications that these devices will consume or use. Challenges in developing mobile services and applications are multifold. There is a great variety of mobile standards, operating systems on different devices. Often unfortunately, one application can work on one cell phone very well, while it does not work on the other [5]. The two main challenges that mobile landscape presents can be pointed down to *device fragmentation* and *operating system fragmentation*.

Fragmentation is the inability to “*write once and run anywhere*”. More formally, it is the inability to develop an application against a reference *operating context (OC)* and achieve the intended behavior in all *OCs* suitable for the application [6]. Fragmentation affects the whole ecosystem of application users, developers, content providers and distributors, network operators and device manufacturers. As for device fragmentation, we can refer to what J. E. Girón *et al.* [7] have rightly put as

“...in an effort to attract more public, several manufacturers incorporate special features into their models, resulting in a lack of uniformity among them. Thus, devices present different processing, memory, storage,

communication and displaying capabilities. This heterogeneity causes that the application development process becomes not homogeneous for all these devices, increasing not only costs but also the possibility of creating inconsistent versions of each application (one for each device)."

Similarly operating system fragmentation also compounds the problems in mobile development. Different vendors/players in the mobile market have their own platforms running their operating systems. Apple's *iOS*, Google's *Android*, Microsoft's *Windows Phone*, RIM's *BlackBerry OS*, *Symbian*, etc to name a few are the different operating systems that ply on the majority of mobile devices in use. The platform inventors and companies provide their own set of *development environment and tools* in the form of *Software Development Kits (SDKs)* targeted and optimized for their platforms. Choice of a platform relies on how deeply developers want to link the application with the underlying operating system, as capabilities in one operating system may not be available in another. Using an SDK the developer may target a particular operating system and take advantage of its specific capabilities to create an application with those features [8]. Such applications are called *native applications* or *native apps* and they guarantee the best usability, the best features, and the best overall mobile experience. But this *native development approach* by using SDKs has its own drawbacks. These SDKs are tied to the specific platforms and primarily use different programming languages like *Objective-C* for iOS, *Java* for Android, *C#* for Windows Phone, *Java* for BlackBerry OS, *C++* for symbian, etc [9]. The developed *native applications* are not portable to other platforms meaning that one has to almost entirely rewrite the application all over again for any other targeted platform. The figure 1.1 gives a better picture regarding the native development.

PLATFORMS	 Apple iOS	 Android	 RIM (BlackBerry)	 Symbian	 Windows Phone 7	 Palm (webOS)
PROGRAMMING LANGUAGES	OBJECTIVE-C	JAVA	JAVA	SYMBIAN C++	C#	JAVASCRIPT, HTML & CSS
DEVICES (Only a small sample)						

Figure 1.1: Sampling of the mobile platforms and their various programming languages and devices [10].

The figure shows a sampling of six different mobile platforms and their various programming languages and devices. So in the native approach in order to target

different platforms, the developer needs to have different skill sets and familiarity with different platforms. Apart from that, developing an application for each platform individually will escalate the time and cost and top it up with the maintenance cost of all those different versions of applications. So there are challenges in developing mobile applications which are interoperable across different platforms using moreover the same codebase.

1.2 Problem Definition

Even though native development approach comes with all the *bells and whistles*, it has one severe restriction of being tied to a particular platform. This means native approach becomes a very expensive solution especially when looked from the context of fragmentation of the mobile landscape. So there is a need for an approach of mobile development which can address the fragmentation resulting from the different platforms and devices. This obviously and advertently leads to the cross-platform approaches and solutions. The motivation behind this thesis is to explore the cross-platform mobile development as an alternative to native mobile development; how can they be achieved, how can they tackle the aforementioned challenges in mobile development, and what benefits can they bring. So it is envisaged that a literature survey followed by the prototype cross-platform mobile application development will help better understand these questions and the domain in general. To put it succinctly, the work will be aimed at investigating the mobile development approach that leads to *cross-platform* mobile solutions which can alleviate those mentioned challenges and problems. Hence a research question has been formulated as below for which this thesis work tries to find plausible answer.

- *What approaches of mobile development entail to cross-platform mobile applications in the fragmented mobile landscape and what are the potential benefits of such approaches and applications?*

To get a better understanding of the problem definition and answer it effectively the above research question is divided into following two sub-questions.

- a. What kind of practices and technologies exist and how can they address the issues of developing the applications that can run across different platforms and variety of mobile devices?*
- b. What are the potential benefits of adopting such practices and technologies to devise solutions in the mobile development?*

1.3 Structure of the Thesis

The remainder of this thesis is organized as follows. The chapter 2, State of the Art, deals with the work done by other people in the field. A survey of the relevant research papers is conducted to identify the suitable practices and technologies to overcome the problem definition. In chapter 3, Cross-Platform Mobile Development, a cross-platform tool is selected for a prototype application development following the comparison between various tools. This is followed by Application Concept and Design in the 4th chapter. Implementing the Application, the chapter 5, discusses about application implementation issues for various platforms. Analysis of the Application and Development Efforts follows in chapter 6. Then finally the chapter 7, Conclusion,

reflects about the work done, answers the research questions and sheds light on the future directions.

Below is the flowchart representation of the thesis structure. The square box represents the different chapter/stage, and the outcome after each chapter/stage is represented by the label on the arrow. And of course, the arrow direction shows the flow of the thesis report.

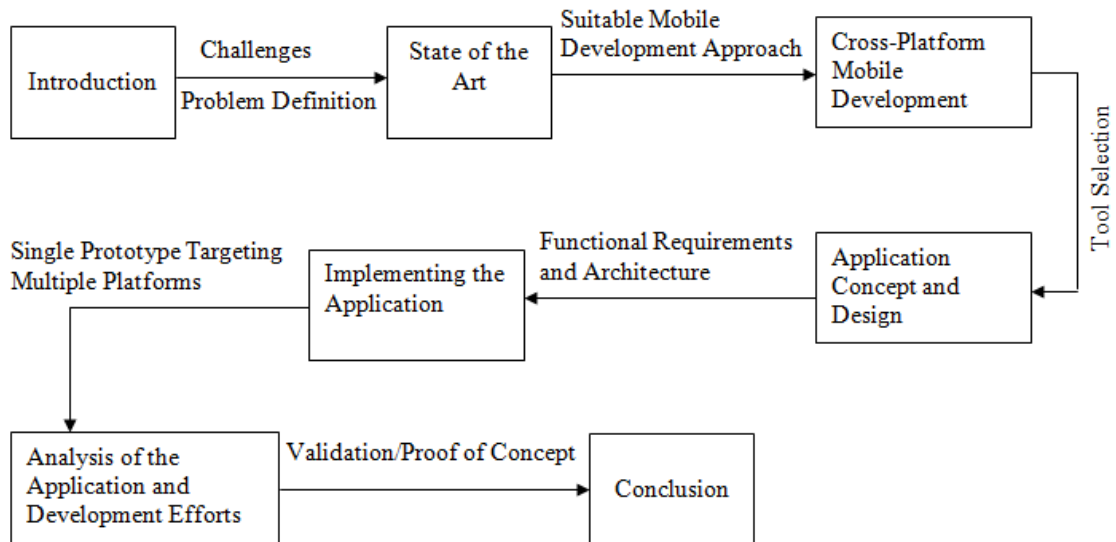


Figure 1.2: Thesis structure.

2. State of the Art

This chapter provides the groundwork towards solving the problem definition that has been raised. Since the thesis is concerned about the operability of applications across multiple platforms in heterogeneous mobile devices, the main focus pertains to issues in the development of cross-platform solutions for these fragmented mobile landscapes. As mentioned for the reasons stated in section 1.1 and 1.2, we are interested in exploring the paradigm of cross-platform mobile development and its possibilities as an alternative to native mobile development. The related work by other people in the field will surely provide a better understanding of the work and other things happening in the domain. For this purpose, a survey of relevant research articles is conducted to gain insight about the prevalent solution approaches and technological choices for the stated problem.

2.1 Survey of Literature

To the best of the author's knowledge, there has not been any systematic literature review (SLR) in the domain of cross-platform mobile development. The attempt here is to conduct a modest survey instead and by NO means should it be considered an equivalent to a SLR which might include several protocol iterations, extensive sources and sample size, detailed data collection, etc. However some cues and ideas are taken from the protocols of conducting SLR in order to make this survey a systematic study of literature. The rest of the chapter describes about how the survey has been conducted.

2.2 Planning the Survey (Methods)

A *protocol*, inspired from Biolchini et al. [11], was developed according to which the literature survey was conducted. We first identified the *keywords* to search for the relevant literature. Then after selecting the digital library sources, we used the keywords in *query string* and performed automated search in these libraries to retrieve the literature. Thereafter, only the papers meeting our *inclusion criteria* were filtered for the survey purpose. More description about the method and the steps involved have been presented hereunder.

2.2.1 Choice of Keywords

Initially we identified the following set of keywords to search for the literature: mobile, interoperability, heterogeneous, and cross-platform. Different combinations of all the four keywords envisioned initially were tried, but it did not yield the expected results. The contexts of the papers were way off target than what was aimed for. And even though some papers cut the list initially, they did not address the problem definition or at the best were describing the theory and architecture rather than providing insights to tangible solutions. The domain is relatively new and there is not so much literature published that can relate to our problem description. The problem was compounded when all the four keywords were used in the search string. And it is not guaranteed that the authors will always use those keywords which we might be looking for, let alone using all the supposed keywords in their papers while trying to address the similar problem description. After several attempts when we did not get the expected results, it was identified that the choice of keywords were not exactly appropriate and especially using them all in different combinations would be futile. Hence the keywords were revised and trimmed down to make it as direct and apt as possible. After many trial iterations, the following keywords

- “mobile”, and
- “cross-platform”

were used to identify the relevant literature. Since we are talking here about the *cross-platform* development approaches and solutions for the *mobile* devices, these two keywords are the most essential ones that can relate to the problem domain and more importantly to the research questions directly. One motivation to keep it basic is to explore the diversified range of development approaches and solutions in the domain.

2.2.2 Sources Selection

To locate the relevant literature, a couple of well renowned digital libraries were identified. The sources lists are:

- IEEE *Xplore* Digital Library, and
- ACM Digital Library

These two digital libraries are one of the biggest, the most prestigious and popular ones where the leading research works in the field of computer science, electrical engineering, electronics, and computing are archived. And of course, using the university student log in, we can access the full-texts of the publications.

2.2.3 Search Method

The two identified keywords (“mobile” and “cross-platform”) were used in the search string to query the digital databases. The databases were searched for the papers published between 2000 and 2012. The purpose for this time frame was to look for the evolvement of different kinds of mobile solutions that were used from the initial days to the modern day latest advancements and solutions in the field.

Following query was performed in the ACM Digital Library:

```
(Title: mobile, AND Title:"cross-platform")
Years 2000-2012
Found 15 within The ACM Guide to Computing Literature
```

Similarly, in IEEE *Xplore* Digital Library:

```
("Document Title":"mobile" AND "Document Title":"cross-
platform")
Search: Full Text & Metadata
Years 2004-2012
Found 14 within The IEEE Xplore Digital Library
```

The method has been depicted in the figure 2.1 below. The query retrieved 15 papers from ACM and 14 papers from IEEE. By reading titles, abstracts and keywords 10 papers were selected further from ACM while 13 papers made through in case of IEEE. Then on eliminating the duplicates from both, total of 19 papers remained. These 19 papers were read fully and thoroughly and in the end 17 papers were selected. The 2 papers were removed as they did not fulfill one or more of the inclusion criteria mentioned below.

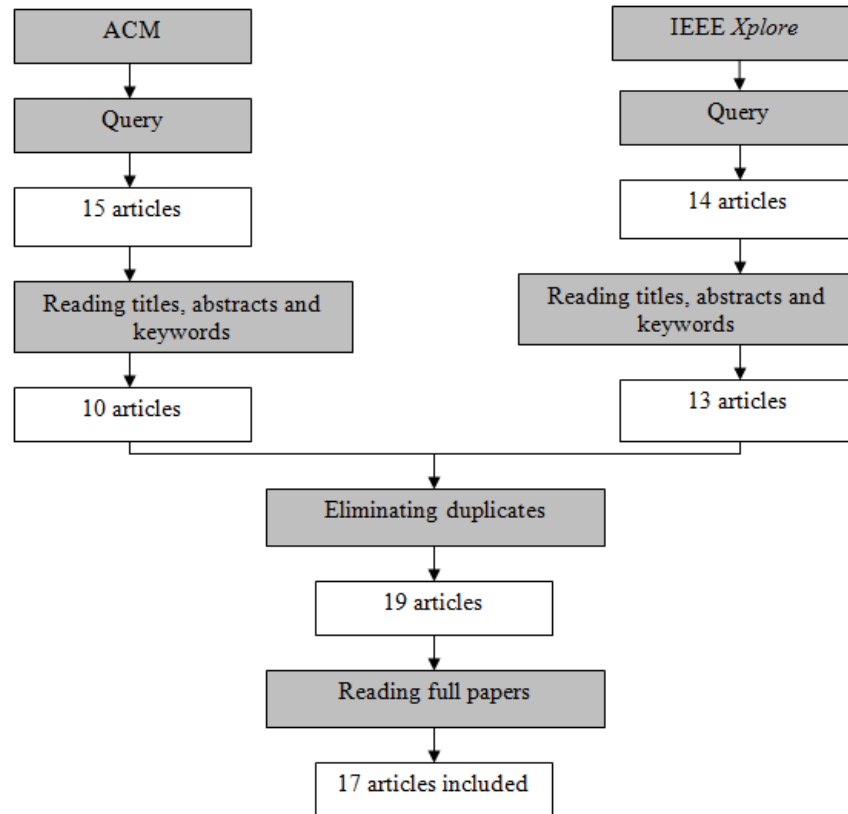


Figure 2.1: Search method using the query string.

2.2.4 Inclusion Criteria

The inclusion criteria are drawn up to make sure we get the papers as highly relevant as possible. Each paper has to fulfill all the criteria in order to be considered for the purpose of our work. The papers that do not fulfill one or more of the criteria will be discarded. Hence the selected papers will help us find answers to our challenges and problem description. Each criterion has been mentioned below.

Criterion 1. Firstly, the papers should be about the mobile devices meaning that the devices should be portable enough to carry around easily, such as mobile/smartphones, PDAs, etc. Albeit not always in the pocket like tablets but not laptops.

Criterion 2. Secondly, the papers should mainly focus on the cross-platform mobile development issues or mostly be relevant to it. Papers raising different kinds of issues related to mobile domain other than the cross-platform are not included.

Criterion 3. Thirdly, the papers should strictly adhere to the problem definition of this thesis. It should be able to address the research question(s) raised above by providing the concrete recommendations and solutions. Papers that do not tackle the research questions, or at the best, propose insubstantial solutions with insufficient amount of information, papers providing vague analytic discussion rather than tangible solutions and recommendations are also excluded.

2.2.5 Timeline of the Literature

After applying the inclusion criteria, we are left with 17 articles. The figure below shows the distribution of these publications over years.

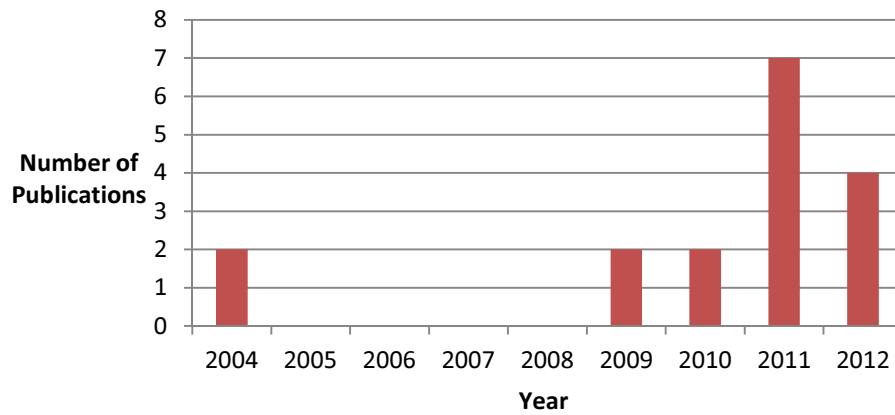


Figure 2.2: Timeline of the publications in the conferences.

In the figure 2.2, the horizontal axis shows the year while the vertical axis shows the number of literature that have been shortlisted for a particular year. Understandably the majority of articles are from the later years 2011 and 2012; however there is also couple of articles each in the year 2009 and 2010. Also included is couple of articles from the year 2004, where they also had the similar problem description albeit in the context of different mobile operating systems then. The survey does not have any articles through the years 2005 to 2008. During those years, the concept of cross-platform mobile applications or even mobile applications was not so much in prominence. Also the hardware aspects of the mobile devices were fairly mediocre and smartphones were hardly heard of then. The emergence of Apple's iOS and Google's Android, both in 2007, changed the whole scenario and initiated the concept of mobile applications. As the platforms and their ecosystems started maturing over the next couple of years, so was the increase in the wide spread acceptance of these systems from the consumers and developers in general. But along the line, the developers community also realized the restrictions brought about by native development and felt for the need to reach the wider audience and consumers based on different platforms. As such, people started looking for alternative means and the cross-platform mobile development gained momentum from 2009 onwards. This is also reflected in the figure 2.2 above, where we see majority of scholarly articles and research relating to cross-platform mobile development getting published from 2009 onwards.

2.3 Conducting the Survey

After the survey plan has been put in place, the survey is conducted on all the included articles. Each article is read fully and thoroughly to gain insights about the problems it has raised and the solutions proposed to tackle those problems. The results of the survey have been summarized in the table below.

References	Problem Description	Proposed Solution	Technologies Used
[12]	Portability and offline execution.	Web-based applications.	Yahoo! Mobile Widget*, Google Gears*, Google Gadget, Apache Shindig.
[13]	Choice of platform and software for cross-platform	Cross-platform web-based	Cabana*(cross-platform web-

	development.	development environment.	based mobile development system), JavaScript.
[14]	Cross-platform mobile development challenges.	A frame of component-based cross-platform mobile web application development. Application development divided into hierarchy of Application Layer, JS Engine Layer, Component Layer and OS Layer.	Not Applicable.
[15]	Secured cross-platform access control for mobile web applications using privacy sensitive JavaScript APIs.	webinos platform, cross-device policy system for web applications on a wide range of web-enabled devices.	XACML (eXtensible Access Control Markup Language).
[16]	Mobile phone game development tools for cross-platform development.	Swerve Studio, X-Forge.	C/C++, Java.
[17]	Cross-platform mobile application with consistent user experience and real-time dynamic content delivery.	Cross-platform mobile development frameworks like Mobile Web Framework (MWF) released by UCLA and other device-agnostic approaches.	Web development technologies, native app wrappers.
[18]	Device fragmentation and consistent user experience.	A hybrid (private/public) model cloud based enterprise mobile application.	Cloud based XML specification.
[19]	Lack of standard for graphics on handheld devices.	GapiDraw platform.	C++.
[20]	Non-uniform standards and the security of payment restricting the development of Mobile-	Mobile payment standard CUPMobile of	Web technologies like HTML, CSS and JavaScript for

	Commerce.	China UnionPay, customized mobile payment middleware CUPFace.	app development.
[21]	Problems for existing mobile instant messaging systems regarding exchange of information across different platforms as they use the private protocols without the inter-connective capability.	XMPP (the Extensible Messaging and Presence Protocol) achieves the unity of various IM protocols across different platforms.	Openfire server based on XMPP; XML, MySQL, Java ME.
[22]	Unreliable network connection, applications adaptive to network conditions, consistent user experience across different platforms, battery life conservation.	A conceptual mobile application architecture which is adaptive, cross-platform, multi-network.	Not Applicable.
[23]	Alternatives to Java ME when writing medical applications for mobile devices across multiple platforms.	An HTML-based medical information application.	HTML, PHP, MySQL.
[7]	Adapting the user interface (UI) across to the actually mobile devices and mobile computing platforms.	A mobile cross-platform client-server architecture, where a set of 4 layers is defined that allows a plastic dynamic deployment of user interfaces.	PhoneGap; Web technologies like JavaScript, XML and HTML5, and native libraries.
[24]	A mobile dictionary app of the English-Czech automatic control terms for the Department of Control Systems and Instrumentation with the view of the fragmented mobile landscape.	The use of HTML5 mobile frameworks.	HTML5, jQuery Mobile.
[25]	Challenges in cross-platform development of mobile widget, and implementation of cross-platform API.	A conceptual MWPD (Mobile Widget Portable Development Library) architecture is a replaceable approach to	Standard web technologies such as HTML, CSS, JavaScript and XML.

		providing cross-platform support for development of mobile widgets.	
[26]	Challenges for IoT based applications regarding orienting different intelligent terminals because of the heterogeneous platforms, which results in a problem of duplicated development.	OpenPlug Studio, based on Flex, is an appropriate cross-platform solution which realizes the conception of once development and multi-deployment.	Flex, ActionScript, MXML, CSS, C++.
[27]	Requirements for the design and development of an end-user programming software system that supports the creation and deployment of cross-platform mobile mashups.	The <i>Mobile Mashup Editor</i> web application consisting of a client representing web browser based GUI editor and a server component representing storage. The <i>Mobile Mashup Viewer</i> based on the cross-platform mobile framework.	Web technologies; Google Web Toolkit, XML, Titanium Mobile Development Platform.

Table 2.1: Survey results summarized. (Note: * Technologies that have been discontinued by their creators.)

In the table 2.1, the results after conducting the survey have been summarized. Each row in the table represents a paper, briefly describing its problem description, proposed solution and the technologies used to devise the solution. A more detailed outcome of the survey mentioning each paper is available here¹. Wide range of problems regarding portability, native platform and device features, consistent user experience, choice of appropriate platform and software, approach of development, non-uniform standards, etc have been raised in the survey papers. Similarly different kinds of cross-platform tools, technologies and approaches have been used to provide solutions for these problems.

2.4 Lessons Learned from Survey Results

It is clear from the survey results that the majority of solutions devised or proposed make use of standard web technology stack of HTML/CSS/JS. The World Wide Web has evolved from a relatively simple document distribution and browsing environment into a general-purpose software platform. Taivalaari and Mikkonen [28] argues that

¹ <http://goo.gl/UbXZD>

emerging technologies such as HTML5 and WebGL enable the use of the Web as a target platform for full-fledged software applications, and the vast majority of end user software will be developed using web technologies. As the software industry continues the paradigm shift towards cloud computing and web-based software, they opine that more and more applications and services are increasingly written for the web instead for specific operating systems or CPUs. So this is also especially true for mobile devices, which are always on the move and nowadays renowned to be always connected, the Web because of its openness has started becoming the technology on which the applications/services are built upon.

Based on the survey outcome, the figure 2.3 shows web-based approaches have been the prevailing cross-platform solutions; however the hybrid approach is also catching up. The figure also shows some *others* solution approaches. This is because some papers from the early days differently tackled the issues and some are mainly concerned with the architecture and model for solutions. Also in the figure, we can see majority of solutions make use of some kind of cross-platform tools, SDKs, frameworks, libraries, platforms, etc.

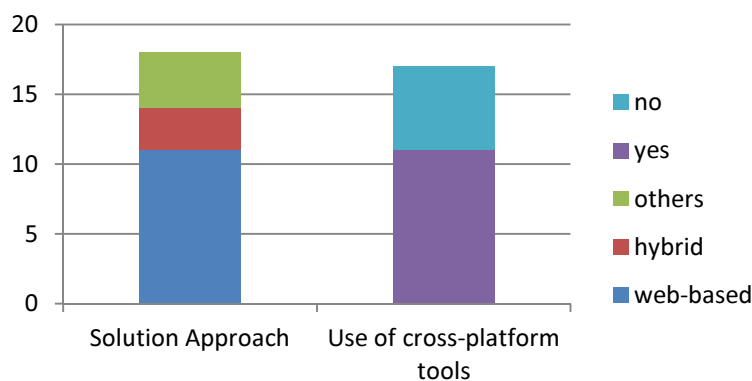


Figure 2.3: Different solution approaches and use of cross-platform tools.

Even in this jungle of different mobile platforms and vendors, one commonality is the default inclusion of standards-compliant web browsers. These browsers from different platform vendors have also started adopting the latest web technologies like HTML5, CSS3, JavaScript mobile frameworks which has even leveraged the web-based solutions. Off late these web-based solutions for mobile devices are often called *HTML5 mobile apps* and are basically a web page or series of web pages or mobile websites designed to work on a tiny screen that try to mimic a native mobile application in looks and feel. They are browser based applications with little or no access to native features on the device and are not subject to app store distribution. Native codes have access to APIs that can access device storage, sensors, camera and data. But this gap is being bridged, too. However strides are being made on the browsers to support different functionalities. Most WebKit browsers (Chrome, Safari) now support hardware accelerated CSS3 animation properties where specific tasks that would otherwise be calculated by the main CPU are offloaded to the graphics processing unit (GPU) resulting in improved render performance and smooth animations [29]. But it is also coming to other browsers and platforms [30]. Most mobile browsers support geolocation today, for example, and iOS recently added Accelerometer and a slew of other HTML5 APIs. Given that the W3C has a Device API Working Group (<http://www.w3.org/2009/dap/>); it is likely we will be seeing many more APIs reach the

browser in the near future. *If a browser does not support a native capability, it's not because it can't or that it won't; it just means it hasn't been done yet* [31]. So it can be safely concluded that the web-based applications result in the solutions that are operable across different platforms and can tackle the issues in the fragmented mobile landscape.

Some articles [17][7][27] in the survey proposed a *hybrid approach* combining the best of both the native and web-based approaches. In this approach the web-based applications are wrapped inside a thin native container that provides access to native platform features, mobile device's hardware such as the camera, accelerometer, and storage. Such applications are called *hybrid applications*. In other words, we basically use the web technologies and not platform dependent programming languages to write an application which then is built into native applications, so they possess the benefits of both. Development of such hybrid applications relies on frameworks such as *PhoneGap* or *Titanium* and gives developers greater control over application design, yet allowing access to device features. Along with enjoying the cross-platform compatibility from a single codebase as well as app store distribution, they also boast of the power, performance and availability on par with the native applications. And these applications are also not confined to browsers. The figure 2.4 shows how developers might use a hybrid solution in this way.

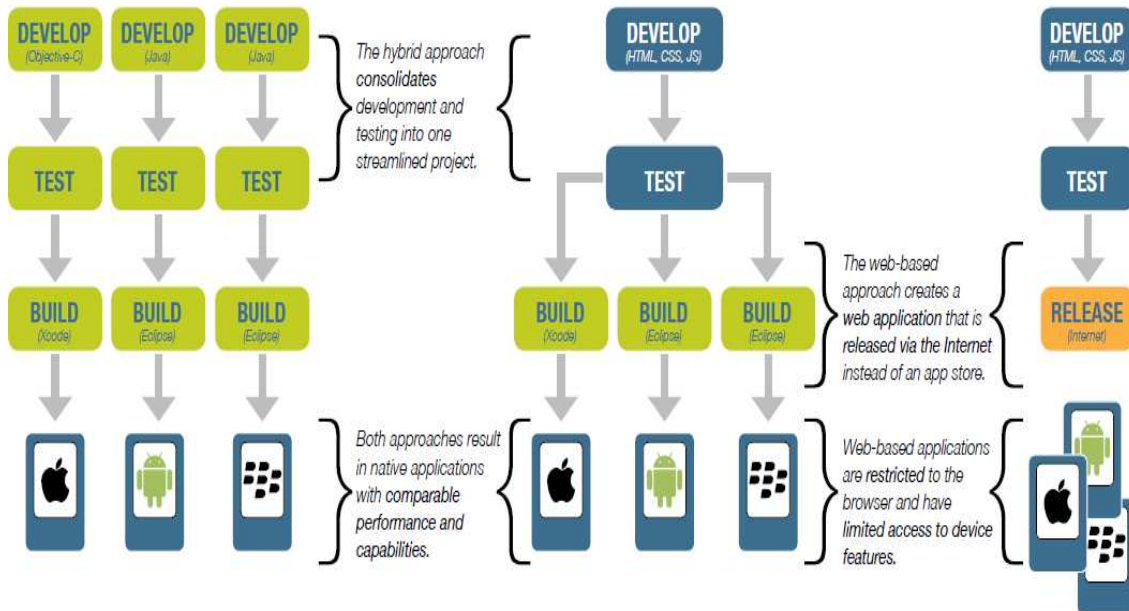


Figure 2.4: The continuum of mobile application development [32].

The figure above shows how hybrid applications are closing the gap between native and web-based application development. The first column depicts the native approach, the second column is the hybrid approach and the last one is the web-based approach. As can be seen from the figure, the hybrid approach unifies the development and testing process into one streamlined project with just the separate build processes required for different targeted platforms. Unlike separate develop, test and build processes of native approach this unified development and testing process is one of the hallmarks of hybrid approach which greatly justifies it as a suitable choice for cross-platform mobile development. Vogel et al. [33] opine that the emerging web and developing frameworks are merging into a unified platform that provides new possibilities, and this pushes the web to act as a software platform. They argue that the latest web standards should not

be tied to one platform, and should offer the support within a broad range of functionalities, so that developers can deploy cross-platform applications.

On the basis of the survey results and analysis, the hybrid approach of development is very suitable because of some of the following reasons.

- Unification of the development and testing process.
For the most part, development and testing are essentially one streamlined process regardless of different target platforms. You build and test it on different browsers, and then port the same code to different platforms to build for those platforms. This greatly saves the time and money.
- Use of the web-based code.
Using standard web technologies guarantee openness and most of the time developers already know these technologies. Hence no need to learn different languages for different platforms.
- Device capabilities and native APIs exposed as JavaScript APIs via frameworks.
Frameworks like PhoneGap, Titanium wrap the web-based code in a thin native container. Then the application is able to access native platform features and device hardware through JavaScript API calls made available by these frameworks. The framework acts like a bridge to communicate these API calls with the native code of different platforms.
- Web apps built into native apps (.apk, .ipa, .bar, etc) using native tool chains and framework and installable on devices.
The build process is perhaps separate for each of the targeted platform. Cross-platforms frameworks are used together with native tool chains (SDKs, IDEs, etc) to build the web apps into native apps. However to do this the framework must support the target platform. These native apps can then be installed on the devices.
- Full screen web view control.
The hybrid approach leverages the device's browser engine and the application is displayed in a full-screen web view control much like a chromeless browser but not in a browser.
- Discovery and distribution via app stores.
Hybrid apps can be submitted to app stores. This means the apps can be discovered, distributed and installed into devices via app stores like the native apps.
- Monetization.
App store distribution means they are subject to licensing and revenue sharing model of the stores. If the developer chooses to sell his app on the store, this is the less complicated model and allows the straight forward ways to monetize.

As such hybrid approach provides the best trade off when it comes to cross-platform mobile development.

3. Cross-Platform Mobile Development

As learnt from the survey, hybrid approaches have become more prevalent in cross-platform mobile development. Cross-platform mobile development essentially makes use of frameworks allowing developers to create platform independent mobile applications predominantly utilizing already familiar web standards like HTML/JavaScript/CSS. These frameworks act more or less as a *middleware* or *bridge* and provide the platform-specific implementation of API in the native programming language for the language of the framework to communicate with the native code of different platforms [34] [35]. According to Martin Fowler, an author and speaker on the topic of software development [36]

“When you use a cross-platform toolkit you define the user-interface elements in terms of the toolkit. The toolkit then builds native, or native looking elements for each target device. It may do this by generating native code at compile-time, or putting a run-time library on the device and generating the UI elements at run-time.” [37]

3.1 Cross-Platform Mobile Development Frameworks Comparison on Different Parameters

As learned from the section 2.4, hybrid approach provides a better alternative to develop cross-platform mobile solutions. However there are quite a few number of cross-platform mobile development frameworks/tools that employ the hybrid approach of application development. Adobe's *PhoneGap*, Appcelerator's *Titanium*, Motorola Solutions' *RhoMobile Suite*, etc are to name a few popular ones. For the sake of this thesis work, it is felt necessary that there is a need for a review of these frameworks. This section deals with an analytic comparison between these frameworks. Heitkötter et al. [38] list out 14 criteria from the *infrastructure* and *development* perspective to assess cross-platform development approaches. Similarly, D. Gavalas and D. Economou [39], compare different mobile programming platforms in five different areas with respect to various quantitative and qualitative criteria. Taking some inspiration from these criteria and additionally identifying our own few according to the context of this work, a list of comparison parameters has been drawn up. On the basis of these parameters the above mentioned three frameworks will be gauged against. The parameters have been mentioned below with the explanation how each of the three frameworks fares on these parameters.

i. Platform Support

Platform support means the number of different platforms that the framework supports. So the framework that supports more platforms means it does better on this parameter.

PhoneGap: Supports eight different mobile platforms as of March 2013 namely iOS, Android, BlackBerry OS, Windows Phone, webOS, Symbian, Bada and Tizen [40].

Titanium: As of March 2013, Titanium supports iOS, Android and BlackBerry [41].

RhoMobile Suite: As of March 2013, RhoMobile supports iOS, Android, Windows Phone 7 series, BlackBerry along with now discontinued Windows Mobile [42].

ii. Development Environment

The development environment typically means the tool chains associated with the framework, particularly the integrated development environment (IDE) with support of emulator/simulator, debugger and various other features and functionalities.

PhoneGap: Standard tools of web development, such as Firebug, Web Inspector, and any preferred text editor of choice can be used in the development with PhoneGap. There is no requirement of native tool chain. However, to run a PhoneGap application on a native emulator/simulator, developers will generate a project for each of the native platforms they wish to support, configure that project's "web root" directory in Xcode, Eclipse, or whatever native tool chain is needed, and then run the project using that tool. Similar workflow should be followed if the developer wishes to install the native-wrapped PhoneGap application to a device. There is also a cloud based compile service called PhoneGap Build that relieves the developer of the pain of installing and maintaining native SDKs. All one needs to do is upload the web assets- a ZIP file of HTML, CSS and JavaScript, or a single index.html file - to PhoneGap Build, a cloud service that compiles and packages the project and provides the download URLs for all mobile platforms [35] [43].

Titanium: Developers are required to use Appcelerator's IDE Titanium Studio based on Eclipse platform. Since Titanium enables the development of native mobile applications with a vast array of native functionality built-in, the native tool chains need to be installed, for each of the mobile platforms that one wishes to target [44].

RhoMobile Suite: A RhoStudio Eclipse plug-in with a fully-featured simulator that allows the use of single computer to quickly develop, debug and test a single RhoElements application across multiple platforms. No need to install different SDKs or develop, test, debug and maintain separate application versions on different computers. It is a one-tool simplicity for entire solution from application development to application integration with no need to worry about emulators and devices [45].

iii. Access to Native Platform and Advanced Device-Specific Features

Does the framework provide the access to native features and device hardware?

PhoneGap: Supports slew of features like accelerometer, camera, compass, contacts, file, geolocation, media, network, notification (alert, sound, and vibration), storage, etc as of March 2013 [46].

Titanium: Accelerometer, camera, notification, network, database, contacts, filesystem, geolocation, media, map, a wide array of automatically-scaled data storage and web services (such as user logins, photo uploads, checkins, status updates, and push notifications), etc are to name a few. Apart from these, it supports host of many more other features and services as of March of 2013 [47].

RhoMobile Suite: Has support for wide range of features like accelerometer, audio/video capture, barcode, camera, database, file, geolocation, magnetometer, media player, contact, ringtone manager, etc as of March 2013 [48].

Note: Some of the features of the frameworks mentioned above may be available only to some of the devices/platforms that the framework supports.

iv. Approach of Development

This means the approach of development that the developer should take while developing the applications using a particular framework. Is it a simplistic approach or does it involve lot of complex steps during development?

PhoneGap: PhoneGap predominantly makes use of web technologies and basically allows HTML-based web applications to be deployed and installed as native applications. So a PhoneGap application is a “native-wrapped” web application [35]. The user interface for PhoneGap applications is created using HTML, CSS, and JavaScript. Application logic is written in JavaScript by making use of the API provided to access native operating system functionality [49].

Titanium: Titanium is essentially a high level, cross-platform JavaScript runtime and API for mobile development. So writing a Titanium application is writing a native application in JavaScript. In Titanium, only a core of mobile development APIs is normalized across the platforms for code reuse. All the platform-specific APIs, UI conventions, and features are kept intact and should be incorporated for platform-specific development to achieve the native performance and experience. A typical application project directory may consist of a configuration file, localization files, and a directory to contain the images, assets, and JavaScript source as application logic. Normally direct editing of HTML and CSS files are not done unless the application contains both native and HTML-based UI. User interfaces are created with cross-platform AND platform-specific components [35].

RhoMobile Suite: Rhodes applications are written within a Model-View-Controller (MVC) pattern. The views are written with familiar web development toolkit of HTML, CSS, and JavaScript. The controllers are written in Ruby, using Rhodes first mobile implementation of Ruby for each smartphone operating system. Normally there are couple of approaches for developing applications. If high level of reliability or significant business logic processing on the devices itself is required, a native approach is taken. In native approach, applications are written using HTML, CSS and JavaScript, and the Rhodes-style Ruby framework for the application logic that resides on the device itself. Another approach is the hybrid web approach with a more traditional web app model typically written purely with HTML, CSS, and JavaScript or running from any web server application technology (.NET, Java, etc). These hybrid apps have their application logic on the back end web application and are able to perform device capabilities with both HTML tags and JavaScript calls. Since these apps operate in web-connected environment, they can easily be modified and distributed to users each time they are accessed. Hence any changes/updates are directly reflected without the user needing to update from the device [50] [51].

v. Codebase/Code Reuse

How much of code reuse is possible across different platforms when using the particular framework? So if more or less the single codebase can be used across different platforms or the larger amount of code can be reused, the framework is considered

better. The more the amount of changes/tweaks/code rewrite required, the worse the framework is considered on this parameter.

PhoneGap: A single codebase with few tweaks and quirks per platform should be enough for PhoneGap applications.

Titanium: Titanium is not an attempt at “write once, run everywhere”. Moreover it is an attempt to achieve code reuse with a unified JavaScript API, with platform-specific features and native performance to meet user expectations. Hence lots of platform-specific codes exist and ideally a single codebase for multiple platforms is not possible. One codebase for multiple platforms though is a possibility but then one will have to sacrifice those platform-specific native features and performance [35].

RhoMobile Suite: One version of application suffices for any device from any manufacturer it supports. Hence application needs to be written only once and can be run on any mobile device and any operating systems [52].

vi. Architecture and Porting

Architectural complexity, ease of porting across different platforms and footprints left by the frameworks are considered on this parameter.

PhoneGap: PhoneGap is small and simple from architectural view point as only the lowest common denominator of native APIs have been implemented into it [35]. Also it is relatively easy to port to different platforms. From high-level application architecture, PhoneGap applications act as a client for the user to interact with. This client communicates with the application server, typically a web server with server side scripting language, where the business logic is carried out. The result is then passed back to the client to display. At client side, the application architecture is usually a single-page application model. The application logic is inside a single HTML page. Data is retrieved from server using AJAX and displayed at client side by manipulating HTML DOM [49].

Titanium: Abstraction layer and scope of API are large in Titanium. So normally the architecture is bit more complex. Implementing Titanium API for a new platform is a massive undertaking which makes the addition of new platforms difficult [35].

RhoMobile Suite: The suite consists of Rhodes, RhoConnect, RhoElements, and RhoStudio. Rhodes is a framework for building locally executing, device-optimized mobile applications. Rhodes applications are implemented in MVC architecture. RhoConnect is a “mobile app integration” server which helps in creating and managing connections to the backend business systems. RhoElements is a powerful HTML5 development framework. Similarly RhoStudio is an Eclipse installation which makes the cross-platform application development easier and faster. The architecture is somewhat heavy but fairly modularized for different aspects. The porting of applications across different devices is easy [52].

vii. Learning Curve

The learning curve associated with frameworks. What kind of technologies, skills and expertise are required to use a particular framework? Do we need to reinvent the wheel or can we use the already familiar standard technologies?

PhoneGap: Web developers will have easier transition to PhoneGap as they can utilize the already familiar web development skills and standards like HTML, CSS, and JavaScript.

Titanium: Although JavaScript is used for development, however the learning curve is relatively steep for Titanium as it demands lot of framework-specific knowledge. Platform-specific APIs, UI conventions, and features should be incorporated in development which requires experience.

RhoMobile Suite: The development employs an MVC pattern with views written in familiar HTML, CSS, and JavaScript while controllers are written in Ruby. So it will take quite some time for developers who do not know Ruby. Also the different components in the suite like RhoConnect, RhoElements might take some time to sink in for the developers.

viii. Speed and Cost of Development

The speed and cost factor involved in using a particular framework for development.

PhoneGap: As a single codebase can be used for different platforms, PhoneGap speeds up the development and minimizes the cost as well.

Titanium: In case of Titanium, developers cannot simply use a single codebase approach. It requires much of framework-specific knowledge and additional work of native intricacies to be taken care of. This increases the size of code as well as the maintenance cost. Hence the development speed is slow and the cost high.

RhoMobile Suite: Application only needs to be written once and single version of application runs on different devices and platforms. So lot of time and money otherwise spent on creating and managing different versions of each application can be saved [52].

ix. Documentation and Tutorial Available

Extensive and detailed documentation, along with good amount of tutorials, sample examples, codes and active community play crucial role on taking up the framework.

PhoneGap: The documentation available is pretty good with lots of code examples and in most cases a quick example and a full example. Necessary quirks for platforms have also been provided in most cases. The community is fairly robust with PhoneGap Google Group for questions and answers. There is also a PhoneGap IRC channel as well as a PhoneGap Wiki page in github.

Titanium: Fairly structured and extensive documentation but minimalist code samples approach. There is also community questions and answers page.

RhoMobile Suite: The documentation, even though structured and tries to cover lots of aspects, at times provides few code samples and full examples. Discussions page is available for questions and answers. Also there is a Google Group for the same purpose.

3.2 Decision on Cross-Platform Mobile Development Framework

In preceding section 3.1 above, comparison between PhoneGap, Titanium and RhoMobile Suite has been made on various parameters with subsequent elaboration. Now, for each parameter, the frameworks will be gauged on the scale of 1 to 5. A score of 1 means the framework faring worst on that parameter while 5 means an excellent score. It is to be noted that the scores are based mainly on the information taken from the official documentation of each framework and various other online resources on the web. Some inputs have also been taken from couple of papers from the survey and other research publications but no practical testing has been performed for the purpose. As a result, the absolute objectivity of the scores cannot be claimed; however the main purpose here is to provide an overview about a possible assessment matrix for the frameworks. As such although some discrepancies might have remained, the assessment still remains valid and credible for the most part within the context and purpose of this thesis work. Hence on the basis of the resources I pursued and my individual judgement on those, I have scored the frameworks. The result is tabled as below:

Parameters \ Frameworks	PhoneGap	Titanium	RhoMobile Suite
Platform Support	5	2	3
Development Environment	5	3	5
Access to Native Platform and Advanced Device-Specific Features	4	5	3
Approach of Development	5	3	3
Codebase/Code Reuse	4	2	5
Architecture and Porting	5	2	4
Learning Curve	5	3	3
Speed and Cost of Development	4	2	5
Documentation and Tutorial Available	5	3	3
Total	42	25	34

Table 3.1: Frameworks comparison results scored.

The table 3.1 summarizes the score of three different frameworks PhoneGap, Titanium, and RhoMobile Suite compared on nine different parameters. The results suggest that PhoneGap scored the highest total of 42 followed by RhoMobile Suite with score of 34. Titanium scored 25 and came last of the three.

Hence on the basis of this analytical comparison, PhoneGap is deemed as an appropriate choice of framework technology that will be used in the prototype implementation. Implementing a prototype will help better understand how we can use web technologies to develop cross-platforms mobile solutions. How these web-based solutions can be leveraged using framework like PhoneGap? Putting into practice the lessons learnt from the literature survey allows us to gain the practical insight about the survey and the overall process of developing such cross-platform solutions.

4. Application Concept and Design

A *location aware bus app* to show the real time departure schedules of the buses from different bus stations in Växjö is taken as an application concept. This bus application is named *NextBus*. The application is conceptualized as a *Geolocation* and *Google Maps* based web application. This means native feature like Geolocation will be incorporated into the application. Using Google Maps JavaScript API will help understand how the third party APIs/technologies can be effectively integrated and used in the application to create robust cross-platform solutions that meets one's requirements.

4.1 Adhering to the Comparison Parameters/Assessment Matrix

It is also of interest to connect much of the assessment matrix presented above in the section 3.1 with the concept and design of the application. It is planned to be targeted for the most popular platforms namely Android and iOS; and also at least one other platform. Targeting at least three platforms will give us better idea about the variety of platform support of the framework and cross-platform development in general. The application will be developed initially as any normal web application. PhoneGap does not put constraints on the development and test environment. Hence we can use standard web development tools. We can write the code in any of the preferred text editor, debug it using debugger like *FireBug* and test it on the desktop and mobile web browsers. However after we have the code base, to build the application into native application, we must use the native tool chains of the target platform. As a rule of thumb, PhoneGap allows any application that runs on the browser to be packaged, built and ported as native application on the platform it supports. All this makes the overall approach of development a simplistic and straight forward process. We can also get the idea about how much of the code reuse is possible across different platforms and the architectural and porting complexities while building a web app into a native app. The need to learn different programming languages is not required as we will be doing essentially everything with web technologies. And much of the platform specific things will be bridged by the PhoneGap. The learning curve is fairly simple for the people having experience with the web. There is good amount of official documentation, tutorial, examples, community, forums, etc which can help the newcomers and the experienced developers alike. But one of the most important things of all is how much can this approach expedite the development process and bring the cost down. The coming sections describe about the requirements and architecture of the *NextBus* application.

4.2 Identifying the Requirements and Use Case Modeling

The application is location aware, meaning that the device should be geolocation capable, and it will use the device's current location to show the nearest bus stations on the *Google Maps*. The map is displayed on the first page (home page). Interaction with the stations on the map will show the real time departure timings from those stations on the second page (detail page). In case a user wants to select any arbitrary station, instead of the nearest stations displayed, there will be a menu option providing a list of stations from where any station can be selected manually. On selecting a station from the menu, it will be displayed on the map and clicking on it will show its timings on the detail page. From the detail page, the user can use the home button to return to the home page. There will also be a refresh/reload button in the home screen to get the latest data in case the app is idle for a long time. So basically it is a two page/screen application. Exact functionalities on each page are listed below.

4.2.1 Home Page Functional Requirements As Use Case Modeling

Some of the main interactions of user on the home page are termed as functional requirements and have been presented as the following use case diagram.

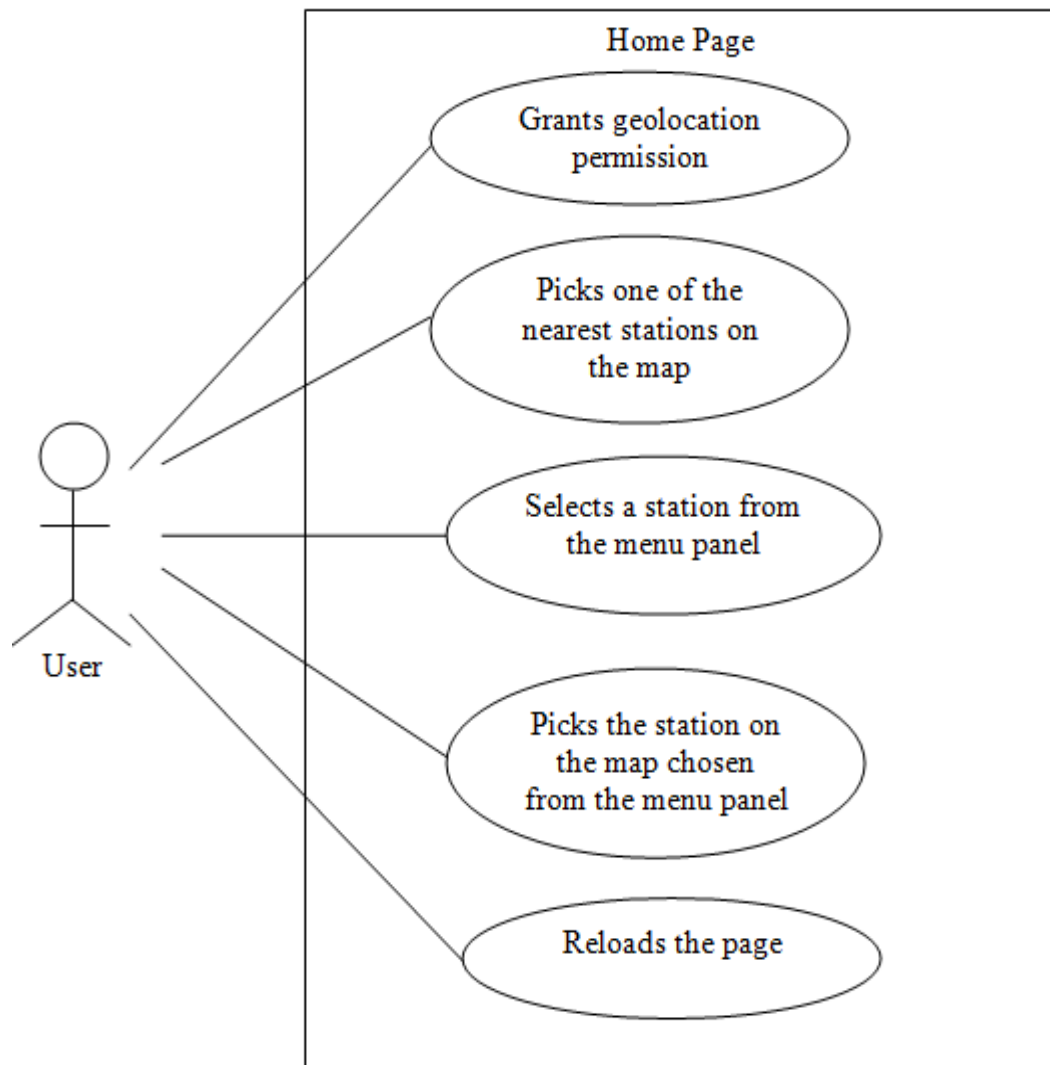


Figure 4.1: Home page use case diagram.

4.2.2 Home Page Non-Functional Requirements

- i. The app starts and shows the user's current location along with the nearest five bus stations from the user's location on the map.
- ii. There will be a marker (balloon icon) on the user's current location and also markers (bus icons) for the bus stations displayed on the map. Beneath the marker for each bus station there will be a label mentioning the name of the bus station and its distance from the user's current location.
- iii. On clicking any of the bus icons on the map will take to the next page (detail page), where the real time departure schedules for that station will be displayed.
- iv. Alternatively, there will also be a menu on the header from which the user can choose an arbitrary bus station from a list of stations.

- v. On choosing a station manually from the menu, the user's current location along with the chosen station will be displayed on the map with appropriate markers and labels. The label consists of the station name and its distance from the current location. On clicking this bus icon/marker, the detailed departure schedules for that station will be displayed.
- vi. There will also be a refresh/reload button on the header clicking on which will retrieve the fresh data.
- vii. The application should scale/fit the screen, irrespective of the device display size and orientation.
- viii. The map should have maximum zoom level possible whilst bounding all the relevant station(s) and the user's current location in a view.
- ix. And, the usual map interactions like zooming in/out, panning, dragging should also be possible.

4.2.3 Detail Page Functional Requirements As Use Case Modeling

The interactions of user on the detail page are the functional requirements that have been presented as the following use case diagram.

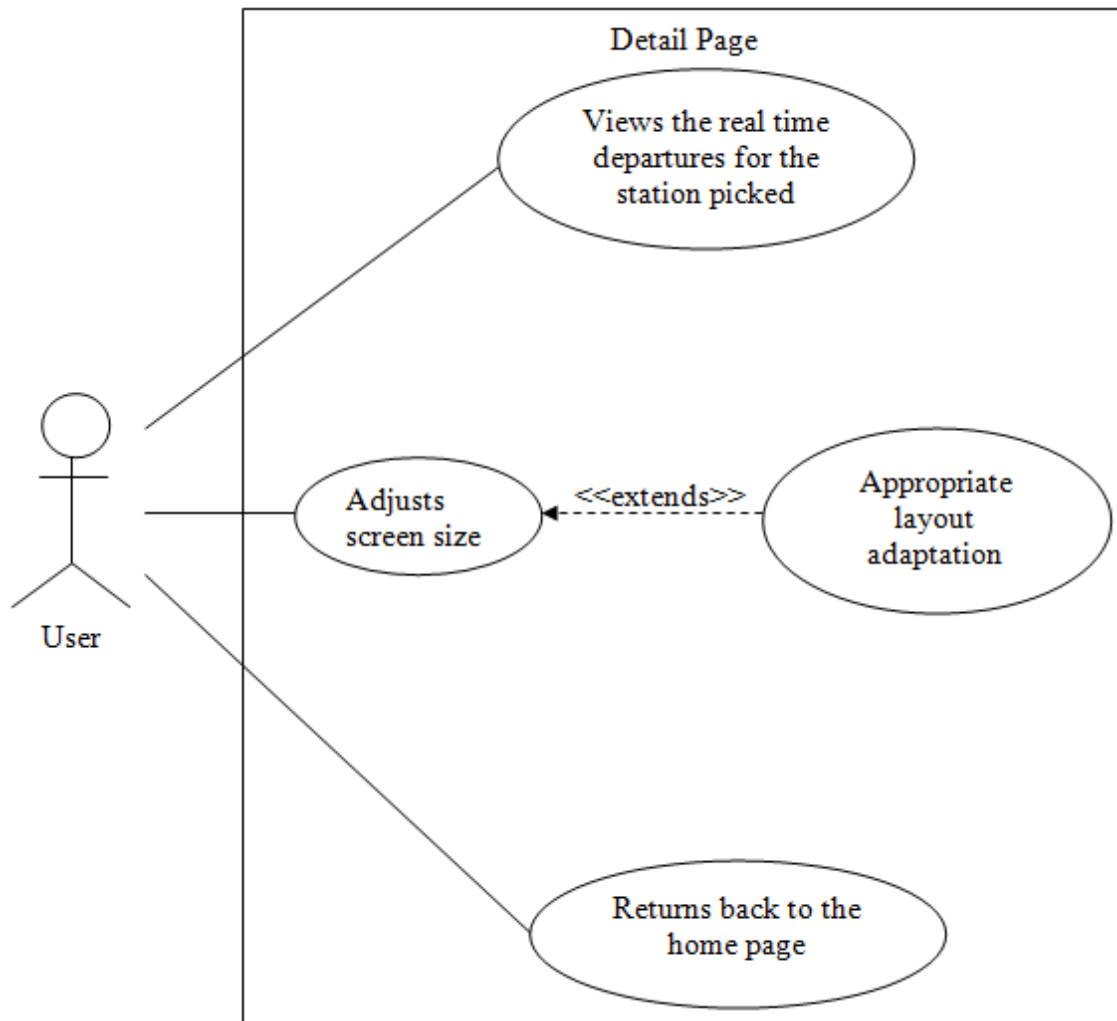


Figure 4.2: Detail page use case diagram.

4.2.4 Detail Page Non-Functional Requirements

- i. Real time detailed departure schedules for the station clicked from the previous page.
- ii. Incorporating *responsive web design (RWD) approach*, to display the departure schedules in a table so that it works optimally across a wide range of device resolutions, screen densities and interaction modes with the same underlying codebase. This means, for smaller screens, the table columns will be collapsed into a stacked presentation that looks like blocks of label/data pairs for each row. While for bigger screens, the design will switch to the normal tabular style presentation.
- iii. A back button on the header to return back to the previous page/view.

4.3 Application Architecture

The application architecture is basically a web-based client-server model. The architecture is depicted in the figure below.

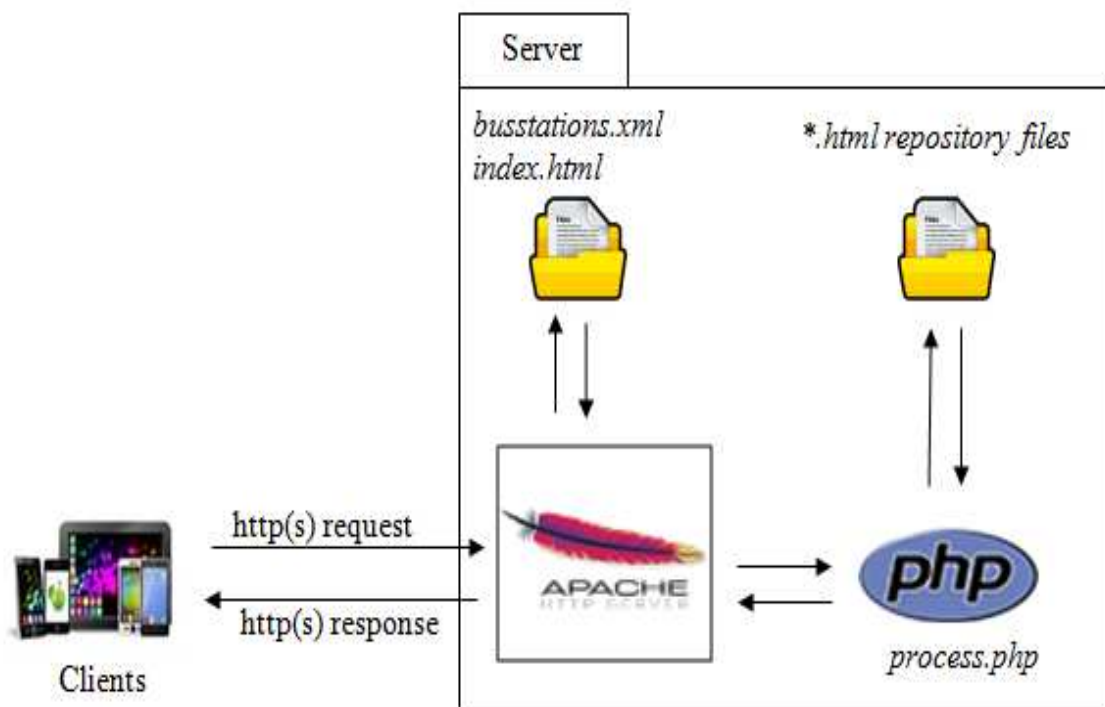


Figure 4.3: Web-based client-server application architecture.

In its simplistic form, the client sends an http request for a page on the server which has a web-server running on it, typically an Apache HTTP server. The web server, on receiving the request, sends back the requested page and associated resources as an http response to the client. This response is then parsed and displayed on the client side by the browsers.

In the context of our application, the file system in the server hosts *html*, *xml* and *php* files of the application. The *index.html* file is an entry point of the application. When the client makes a request, the web server fetches this html file along with the xml file *busstations.xml*. However the *index.html* file also has an Ajax call to the *process.php*.

The web server cannot interpret the PHP code and forwards it to the PHP interpreter. The *process.php* is basically a background PHP process written to extract the data from the html repository files. The PHP interpreter executes this PHP file and sends the results back to the web server as html. The web server then sends back the response *index.html*, which is rendered at the client side. One thing to be noted though is, for the web application this *index.html* resides on the server. However for hybrid/native application, the *index.html* resides on the client side and is rendered on the fly with the response generated from the server.

4.4 Identifying the Appropriate Technological Requirements

As argued in the previous sections, the basic technologies that will be used to develop the app will be *HTML5*, *JavaScript* and *CSS*. And *Apache Cordova/PhoneGap* will be used to package and build the application into native application for different mobile platforms. Additionally, with the identified functional requirements in the preceding section, *Google Maps JavaScript API v3* and *HTML5 Geolocation API* will also be used. Also rather than writing the *plain vanilla JavaScript* code, using special JavaScript libraries like *jQuery* and *jQuery Mobile* will serve better. A brief introduction to all the technologies mentioned is provided in the following subsections.

4.4.1 Google Maps JavaScript API v3

The Google Maps JavaScript API Version 3 is the latest official JavaScript API from Google for its web mapping service application and technology that powers many map-based services. Using the JavaScript programming, the API lets you embed Google Maps in your own web pages. Version 3 of this API is especially designed to be faster and more applicable to mobile devices, as well as traditional desktop browser applications. The API provides a number of utilities for manipulating maps (just like on the <http://maps.google.com> web page) and adding content to the map through a variety of services, allowing you to create robust maps applications on your website. The JavaScript Maps API V3 is a free service, available for any web site that is free to consumers [53]. However there is also *Google Maps API for Business* that provides Enterprise-ready application support with additional features and benefits for a fee.

Since the API is JavaScript based, it can be seamlessly fit into our web-based approach of cross-platform solutions. Furthermore Google Maps provides its own comprehensive list of Google Maps API Web Services like Directions API, Distance Matrix API, Elevation API, Geocoding API, Time Zone API, and Places API off which Geocoding and Distance Matrix API will be used in this application. They also provide thorough documentation and examples on using those APIs to create map based solutions. Other maps services either do not provide support for such comprehensive extent of web services or in case they do, are mainly third party solutions and not their own. Also the documentation and examples are not thorough when it comes to maps services other than Google's.

4.4.2 HTML5

HTML5 is an emerging standard currently being developed by two standards bodies, the Worldwide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG) [54]. It is an evolution from the previous HTML versions and the upcoming standard for content structure and presentation on the World Wide Web and is being designed with the consideration of being able to run on power-constrained devices such as smartphones and tablets [55]. With lots of different features

and capabilities incorporated into HTML5 and increasing support from the browser vendor community for it, this has lowered the barrier between the traditional desktop or native applications and web-based solutions [56]. Today HTML5 is the ubiquitous platform for the web and is making the web platform more powerful in a number of different areas.

4.4.3 HTML5 Geolocation API

The HTML5 Geolocation API is used to get the geographical position of a user. The Geolocation API defines a high-level interface to location information associated only with the device hosting the implementation, such as latitude and longitude. The API itself is agnostic of the underlying location information sources. Location information can be gathered from the user's device using Global Positioning System (GPS) and location inferred from network signals such as IP address, RFID, WiFi and Bluetooth MAC addresses, and GSM/CDMA cell IDs, as well as user input [57]. Using the Geolocation API capable devices, the user can share his/her location with the websites/applications that they trust. The API makes user's *latitude*, *longitude*, *altitude*, *accuracy*, *speed*, etc available to JavaScript which in turn can send it back to the remote web server and do fancy location-aware things like finding local businesses or showing your location on a map [58]. The geolocation API is supported by most browsers on the desktop and mobile devices.

4.4.4 jQuery

jQuery is a free, open-source, fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers [59]. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. Developers can create abstractions for low-level interaction and animation, advanced effects and high-level, theme-able widgets [60]. Used by 55.2% of all the websites, that is a JavaScript library market share of 91.8%, it is by far the most popular JavaScript library in use today [61].

4.4.5 jQuery Mobile

jQuery Mobile is a touch-optimized UI web framework for smartphones and tablets. It is based on jQuery core APIs and jQuery UI allowing the developers to write a unified, HTML5-based user interface system for all popular mobile device platforms. Its lightweight code is built with progressive enhancement, and has a flexible, easily themeable design [62]. It is compatible with the vast majority of all modern desktop, smartphone, tablet, and e-reader platforms and browsers. In addition, feature phones and older browsers are supported because of the progressive enhancement approach. It allows you to write a single website or application that will work on different mobile devices, operating systems, desktop platforms and eliminates the need of rewriting the websites/applications to target different hardware/software platforms. The jQuery Mobile framework can also be used with various other mobile development frameworks like PhoneGap, BlackBerry WebWorks, IBM Worklight, etc.

4.4.6 CSS

Cascading Style Sheets, abbreviated as CSS, is a style sheet language used to describe the presentation of a document written in HTML or XML including different XML variants like SVG, XHTML, XUL. CSS helps to keep the information content of a

document separate from the details of how to display it. The details of how to display the document are known as its *style*. This *style* or *style sheet* consists of a set of *rules* that describes how the structured element must be rendered on screen, on paper, in speech, or on other media. Its primary purpose is to avoid duplication, make maintenance easier, and use the same content with different styles for different purposes [63]. It can also be used to display the web page differently depending upon the screen size, device on which it is being viewed.

CSS3 is the latest evolution of CSS and extends CSS2.1 whilst being completely backwards compatible. CSS3 has been split into modules and each module adds new capabilities or extends the features from previous versions. According to W3Schools [64], some important modules are:

- Selectors
- Box Model
- Backgrounds and Borders
- Text Effects
- 2D/3D Transformations
- Animations
- Multiple Column Layout
- User Interface

As a result of modularization and being an emerging specification, different modules have different stability and statuses [65]. However increasing number of modern web browsers is increasingly supporting many different modules and features.

4.4.7 Apache Cordova/PhoneGap

PhoneGap was donated to the Apache Software Foundation (ASF) under the name Apache Cordova and hence is an open source distribution of Apache Cordova. PhoneGap is an open source framework for quickly building cross-platform mobile apps using HTML5, JavaScript and CSS. Targeting multiple mobile platforms requires knowledge of different languages/frameworks; PhoneGap solves this by allowing the developers to build mobile applications for different mobile platforms using the already familiar standards-based web technologies [66].

Apache Cordova/PhoneGap provides a set of JavaScript APIs which allows the developers to access the native platform and devices features like camera, accelerometer, contacts, etc using JavaScript. Basically there will be uniform platform-specific JavaScript libraries that can be invoked. This will in turn invoke the device-specific native backing code for those JavaScript libraries making possible the use of platform and device specific features. Since the apps are built without any native code and using the standard web technology stack which are consistent across different mobile platforms, the basic codebase remains the same and should be portable to different mobile platforms with minimal to no changes [67]. Of course, these apps are packaged as native apps for each targeted platform using the platform specific SDKS and tool chains and can be made available for installation from each device's app store. UI framework such as jQuery Mobile or Dojo Mobile or Sencha Touch can also be used and combined in the app development.

4.5 Identifying and Preparing the Suitable Data Structure

Initially I emailed *Länstrafiken Kronoberg*, who is responsible for all public transportation within Kronoberg County, to know if they provide the bus schedules data in any appropriate format or as a web service. Since they did not respond, I browsed

their website where I found that they maintain a *Wireless Access Protocol (WAP)* service at the address <http://wap.nastabuss.se/its4wap/> which can be accessed from the desktop/mobile browser with internet connection and on querying it, provides the real time bus schedules in html. Here the user can choose a station from a list of stations using the drop down menu and it will provide the real time timings of bus departures from that station in *html* format. But unfortunately this WAP service went down after a while and is no longer available as of this writing.

However I had saved the html pages, before it went down, for a number of stations and that will be used as a *dummy data* for the purpose of this thesis. So for every bus station there will be a corresponding HTML file from where the data will be extracted for that station. Of course because of this the data will be *static* which means every time I make a *request* for the data of a particular station, I will be pulling off data from an *offline html file* for that station.

4.5.1 Maintaining a Repository of Bus Stations

A repository should be maintained for the bus stations. This repository is an *xml* file with a root node `<nextbus>` that consists of a series of `<station>` nodes. The information about each station is stored inside the `<station></station>` node. Each of this `<station>` node consists of five children nodes namely `<name>`, `<url>`, `<localurl>`, `<latitude>` and `<longitude>`. The `<name>` node contains the *name* of the station, `<url>` node contains the *url* that points to the remote html file from which the data for the bus station was initially supposed to be fetched; but is redundant for our purpose now as the remote data source is down, `<localurl>` node contains the *alternative url* of the offline html file belonging to the station which will be used to extract the data for the station, `<latitude>` node contains the *latitude coordinates* of the station and `<longitude>` node contains the *longitude coordinates* of the station. Also a thing to be noted is, in order to obtain the latitude and longitude coordinates of the stations, *The Google Geocoding API* is used which is a part of the *Google Maps API Web Services*. Further information about this is available at <https://developers.google.com/maps/documentation/geocoding/>.

```
<?xml version="1.0" encoding="utf-8"?>
<nextbus>
  <station>
    <name>Resecentrum</name>
    <url>http://wap.nastabuss.se/its4wap/QueryForm.aspx?hpl=Resecentrum+(V%C3%A4xjö%C3%B6)</url>
    <localurl>http://crossplatform.co.nf/sample%20html/Resecentrum+(V%C3%A4xjö%C3%B6).html</localurl>
    <latitude>56.87677739999999</latitude>
    <longitude>14.80657550</longitude>
  </station>

  <station>
    <name>Lunnabyvägen</name>
    <url>http://wap.nastabuss.se/its4wap/QueryForm.aspx?hpl=Lunnabyvägen+(V%C3%A4xjö%C3%B6)</url>
    <localurl>http://crossplatform.co.nf/sample%20html/Lunnabyvägen+(V%C3%A4xjö%C3%B6).html</localurl>
    <latitude>56.9127830</latitude>
    <longitude>14.74299680</longitude>
  </station>
  ...
</nextbus>
```

...
</nextbus>

Code Extract 4.1: An XML repository containing the information of bus stations.

4.5.2 Pulling Data off HTML files

After maintaining an XML repository of the bus stations, it is now possible to look up the required information of any station. Also using this information we can track the appropriate HTML file for the corresponding bus station and then extract the information off the HTML file as required. However there is again an important security concept for client side languages like JavaScript called the *same-origin policy* [68], due to which JavaScript code running on pages originating from a different site (combination of host/protocol/port number, etc), say for example *http://www.example-social-network.com*, cannot access the methods, properties across pages running on different sites, say for example *http://online-personal-calendar.com*. Though there are some ways to circumvent this and allowing JavaScript to make such access, however I decided to use a *server-side PHP script*, which has no such restriction, to pull off the required data from the html files.

Using PHP's *cURL*, *DOMDocument*, *DOMXPath* the html file is traversed and the required data is extracted from inside the relevant tags. The data is then encoded into *JSON*. From JavaScript, an *AJAX* request can be made to this server-side script to fetch the required data.

```
<?php
header ('Content-type: text/html; charset=utf-8');

if(!empty($_GET['url']))
    $url = $_GET['url'];
else
    $url = "http://crossplatform.co.nf/sample%20html/
           Resecentrum+(V%C3%A4xj%C3%B6).html";

$curl = curl_init($url);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE);
$output = curl_exec($curl);
curl_close($curl);

$DOM = new DOMDocument;
$DOM->loadHTML($output);

$xpath = new DOMXPath($DOM);
$tags = $xpath->query('//span[@id="LabelForecasts"]');
$tag_data = array();
foreach ($tags as $tag) {
    if(trim($tag->nodeValue) != "")
        $tag_data[] = trim($tag->nodeValue);
}

$tags = $xpath->query('//span[@id="LabelTimeandStop"]');
foreach ($tags as $tag) {
    if(trim($tag->nodeValue) != "")
        $tag_data[] = trim($tag->nodeValue);
}

$tags = $xpath->query('//table[@id="GridViewForecasts"]
                    /tr[@class="darkblue_pane"]/th');
```

```

foreach ($tags as $tag) {
    if(trim($tag->nodeValue) != ""){
        $str = trim($tag->nodeValue);
        $tag_data[] = str_replace("&nbsp;", "", $str);
    }
}

$tags = $xpath->query('//table[@id="GridViewForecasts"]
                      /tr[@class="white_pane"]/td |
                      //table[@id="GridViewForecasts"]
                      /tr[@class="lightblue_pane"]/td');
foreach ($tags as $tag) {
    if(trim($tag->nodeValue) != "")
        $tag_data[] = trim($tag->nodeValue);
}

echo json_encode($tag_data);

?>

```

Code Extract 4.2: A PHP script to extract data from HTML files.

5. Implementing the Application

In the beginning the application is developed like any normal web application using a web server with PHP support and testing it in the browser. The first target is to build a single general *codebase* for the *web application* which will then be built into *native applications* for different platforms. I registered for a free web hosting with PHP support under the domain <http://crossplatform.co.nf> which is used for the live production as well as testing mode. *Adobe Dreamweaver CS4* is used as the text editor, *FileZilla 3.7.1* is used as the FTP client. The testing is carried out on browsers *Google Chrome Version 29.0.1547.57 m*, *Mozilla Firefox 23.0.1*, and *Internet Explorer 10 Version 10.0.9200.16660* and also in android mobile browser and safari browser in iPhone.

5.1 Developing a Single General Codebase

The following subsections will describe how the application is realized in terms of coding while providing brief description about different code parts.

5.1.1 Application Structure

The entire application is coded into a single page *index.html*. The other pages *busstations.xml* and *process.php* have been mentioned in the design section. However this is the main page where the application logic resides and is the entry point for the application. The page is structured as below:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>NextBus Application</title>

    <meta name="viewport"
      content="width=device-width, initial-scale=1">

    <link rel="stylesheet" href="http://code.jquery.com/
      mobile/1.3.1/jquery.mobile-1.3.1.min.css" />

    <script src="//ajax.googleapis.com/ajax/
      libs/jquery/1.9.1/jquery.min.js"></script>
    <script src="http://code.jquery.com/mobile/
      1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type="text/javascript"
      src="https://maps.googleapis.com/
      maps/api/js?key=AIzaSyDJWtBGtXPE9BeyZyEc8lFvi3I0fs_-
      7mY&sensor=false"></script>
    <script type="text/javascript"
      src="markerwithlabel_packed.js"></script>

    <style type="text/css">
      <!-- Custom CSS -->
    </style>

    <script>
      /** JavaScript Application Logic**/
    </script>
  </head>

  <body>
```

```

<div data-role="page">

    <div data-role="header">
    </div><!-- /header -->

    <div data-role="content">
    </div><!-- /content -->

    <div data-role="footer">
    </div><!-- /footer -->

</div><!-- /page -->
</body>
</html>

```

Code Extract 5.1: Skeleton of page *index.html*.

All the necessary imports, local or *Content Delivery Network (CDN)*, of *css* and *js* files go inside the `<head>` section. Some custom *css* and the entire JavaScript *application logic* are also inside the `<head>`. The `<body>` contains a jQuery Mobile page `<div data-role="page">` which in turn contains *header*, *content* and *footer*. Here just for demonstration I have used a single page template inside the `<body>` tag. However in the real application I have used a multi-page structure, by stacking multiple *divs* with a *data-role* of "page" inside the `<body>` tag and have also used a *panel* with `<div data-role="panel">`. The *html* of the multi-page structure with *panel*, *headers*, *contents*, *footers* in the application is like below:

```

<body>

<div data-role="page" id="map-page" data-url="map-page">

    <div data-role="panel" id="mypanel" data-theme="a">
        <ul data-role="listview" data-theme="a" id="mylist">
            <li data-icon="delete">
                <a href="#map-page" data-rel="close">Close menu</a>
            </li>
            <li data-role="list-divider">Stations</li>
            <!-- Remaining panel content will be
                added dynamically here -->
        </ul>
    </div><!-- /panel -->

    <div data-role="header">
        <a href="#mypanel" data-role="button"
            data-icon="bars" data-iconpos="notext">Choose Station</a>
        <h1>NextBus App</h1>
        <a href="index.html" data-ajax="false" data-role="button"
            data-icon="refresh" data-iconpos="notext">Refresh</a>
    </div><!-- /header -->

    <!--Div for map display.-->
    <div data-role="content" id="map-canvas">
    </div><!-- /content -->

    <div data-role="footer" data-id="myfooter"
        class="ui-bar" data-position="fixed">
        <h4>© 2013 Suyesh Amatya</h4>
    </div><!-- /footer -->

```

```

</div><!-- /page -->

<div data-role="page" id="detail-page">

    <div data-role="header" data-position="fixed">
        <a href="#map-page" data-icon="home"
            data-iconpos="notext" data-rel="back">Home</a>
        <h1>Timings for station: </h1>
    </div><!-- /header -->

    <div data-role="content" id="contbl">
        <!-- tabular display -->
    </div><!-- /content -->

    <div data-role="footer" data-position="fixed">
        <h1>© 2013 Suyesh Amatya</h1>
    </div><!-- /footer -->

</div><!-- /page -->

</body>

```

Code Extract 5.2: jQuery Mobile multi-page structure of the application.

5.1.2 Custom CSS

The CSS is mainly applied dynamically by the jQuery Mobile framework. However some custom CSS has also been written to mainly layout the application.

```

<style type="text/css">

    html { height: 100%; }

    body { height: 100%; margin: 0; padding: 0; }

    #map-page{ height: 100%; }

    #map-canvas {
        padding: 0;
        position : absolute !important;
        top : 40px !important;
        right : 0;
        bottom : 0px !important;
        left : 0 !important;
    }

    #mypanel .ui-panel-inner{height: 40px !important;}

    .labels {
        color: red;
        background-color: white;
        font-family: "Lucida Grande", "Arial", sans-serif;
        font-size: 10px;
        font-weight: bold;
        text-align: center;
        border: 2px solid black;
        white-space: nowrap;
    }

</style>

```

Code Extract 5.3: Custom CSS.

5.1.3 Page Initialization

All the JavaScript code is bound to the event “pageinit” which is triggered on the page being initialized, after initialization occurs. The code is executed after this event is fired. The *page id* “#map-page” is also passed upon which the *function* will work.

```
$(document).on("pageinit", "#map-page", function(){  
    ...  
    ...  
    ...  
})
```

Code Extract 5.4: jQuery Mobile page initialization.

5.1.4 Is Geolocation Supported?

After the page initialization, we check for the browser support of the Geolocation API. If it does not support, we are out of luck and just draw a default map.

```
if(navigator.geolocation){  
    ...  
    ...  
    ...  
}  
else{  
    alert("Browser doesn't support geolocation!");  
    drawDefaultMap();  
}  
  
function drawDefaultMap(){  
    var map = new google.maps.Map(  
        document.getElementById("map-canvas"),  
        {  
            zoom: 10,  
            center: defaultLatLng,  
            mapTypeId: google.maps.MapTypeId.ROADMAP  
        }  
    );  
  
    var marker = new google.maps.Marker({  
        position: defaultLatLng,  
        map: map,  
        title: "Somewhere in Växjö!"  
    });  
}
```

Code Extract 5.5: Checking for geolocation support.

5.1.5 AJAX Call to the Stations Repository

If the browser supports the Geolocation API, then an AJAX request is made to the xml stations repository. On success, the response is handled in the function `parseXml`.

```
function parseXml(xml) {  
    ...  
    ...  
    ...  
}  
  
$.ajax({  
    type: "GET",
```

```

url: "http://crossplatform.co.nf/busstations.xml",
dataType: "xml",
success: parseXml
});

```

Code Extract 5.6: An ajax request to the xml repository.

5.1.6 Getting the Current Device Position

We call a method `getCurrentPosition` on the `navigator.geolocation` object which is used to get the current position of the device. The method takes one, two or three arguments. The first mandatory argument is the *success callback function* that is invoked with a `Position` object in case of a successful attempt. The second optional argument is the *error callback function* which can be invoked with a `PositionError` object in case of a failure. The third optional argument is a `PositionOptions` object.

```

navigator.geolocation.getCurrentPosition(success, fail,
    {maximumAge: 500000, enableHighAccuracy:true, timeout: 6000});

function success(pos) {
    ...
    ...
    ...
}

function fail(error) {
    if(error.code == 1) alert("The user denied the permission!");
    else if(error.code == 2) alert("Network down
                                or position unavailable");
    else if(error.code == 3) alert("Taking too long! Timed Out!");
    drawDefaultMap();
}

```

Code Extract 5.7: Getting current device position, handling success and failure.

5.1.7 Calculating and Storing the Distance/Duration to All the Stations From the Current Position

For each station in the xml repository, we calculate the *distance* to that station from the current position and the *duration* required to reach there by walk. The result is stored in an array. *Google's Distance Matrix Service* is used for this purpose and is accessed within the code via the `google.maps.DistanceMatrixService` object.

```

allStations.each(function(){

    /*
    ** The Client-side Google Distance Matrix Service JavaScript
    ** API v3 Method
    */

    var name = $(this).find("name").text();
    var localurl = $(this).find("localurl").text();
    var latitude = $(this).find("latitude").text();
    var longitude = $(this).find("longitude").text();
    var destination = new google.maps.LatLng(latitude, longitude);

    var service = new google.maps.DistanceMatrixService();
    service.getDistanceMatrix(

```

```

        {
            origins: [origin],
            destinations: [destination],
            travelMode: google.maps.TravelMode.WALKING
        }, function(response, status) {
            calculatedDistanceResponse(response, status,
                cbAfterAllStationsPopulated)
        }
    );

    function calculatedDistanceResponse(response, status, callback){
        if (status == google.maps.DistanceMatrixStatus.OK) {
            var origins = response.originAddresses;

            for (var i = 0; i < origins.length; i++) {
                var results = response.rows[i].elements;
                for (var j = 0; j < results.length; j++) {
                    var element = results[j];
                    var distance = element.distance.text;
                    var duration = element.duration.text;

                    stations.push({ "name": name, "localurl": localurl,
                        "latitude": latitude, "longitude": longitude,
                        "distance": distance, "duration": duration});

                    counter++;
                    if(counter === stationsLength)
                        callback();
                }
            }
        }
    }

});

```

Code Extract 5.8: Calculate/Store distance/duration to all the stations.

5.1.8 Sorting the Nearest Five Stations and Drawing Them on the Map

The array in which the result is stored is sorted by distance and we get the nearest five stations from the current position. These five stations are drawn on the map.

```

var stationsSorted = stations.sort(function(a,b){
    return parseFloat(a.distance) - parseFloat(b.distance)
});

var nearestFiveStations = [];
//Appropriate zoom level to fit in all the markers in the map.
var latlngbounds = new google.maps.LatLngBounds();

for(var i = 0; i < 5; i++){
    nearestFiveStations.push(stationsSorted[i]);
    latlngbounds.extend(new google.maps.LatLng(
        nearestFiveStations[i].latitude,
        nearestFiveStations[i].longitude));
}

drawMap(origin, nearestFiveStations, latlngbounds);

```

Code Extract 5.9: Drawing the nearest five stations on the map.

5.1.9 Populating the Menu Panel and Registering Click Event to the Items

The menu panel is populated with all the stations and related data. Then we register a *click* event to each item in the panel, clicking upon which will draw the relevant station on the map.

```
$.each(stationsAlphabetically, function(key, val){
    var htmlStr = '<li><a href="#" id="'+val.name+'" data-
    localurl="'+val.localurl+'"
    data-latitude="'+val.latitude+'" data-
    longitude="'+val.longitude+'"
    data-distance="'+val.distance+'" data-
    duration="'+val.duration+' ">'+val.name+'</a></li>';
    $("#mypanel ul").append(htmlStr);

});
$("#mylist").listview("refresh");

$('#mylist').children().each(function(){
    var anchor = $(this).find('a');
    if(anchor.attr('id') != null){
        anchor.click(function(){
            var latlngboundsForPanelLocation =
                new google.maps.LatLngBounds();
            $("#mypanel").panel("close");
            var panelStation = [];

            panelStation.push({
                "name": anchor.attr('id'),
                "localurl": $(this).data('localurl'),
                "latitude": $(this).data('latitude'),
                "longitude": $(this).data('longitude'),
                "distance": $(this).data('distance'),
                "duration": $(this).data('duration')
            });

            latlngboundsForPanelLocation.extend(
                new google.maps.LatLng(
                    panelStation[0].latitude,
                    panelStation[0].longitude
                )
            );
            latlngboundsForPanelLocation.extend(origin);

            drawMap(origin, panelStation,
                latlngboundsForPanelLocation);

        });
    }
});
```

Code Extract 5.10: Filling menu panel with stations and registering click event.

5.1.10 Drawing the Map

The map is drawn inside the function `drawMap`. The function is invoked using the appropriate arguments which are used to draw the map with appropriate stations and the markers on it. The markers are attached with the event listener.

```

function drawMap(currentLatLng, otherLocations, bounds) {
    var myMapOptions = {
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var map = new google.maps.Map(document.getElementById(
        "map-canvas"), myMapOptions);

    // Add a current lat/lng geolocation overlay to the map center
    var markerCurrentLocation = new MarkerWithLabel({
        position: currentLatLng,
        map: map,
        title: "Current Location!",
        labelContent: "Current Location!",
        labelAnchor: new google.maps.Point(22, 0),
        labelClass: "labels",
        labelStyle: {opacity: 0.60}
    });

    var iconImage = 'images/bus.png';
    // Add overlays to other locations
    $.each(otherLocations, function(index, value){
        var markerOtherLocations = new MarkerWithLabel({
            position: new google.maps.LatLng(value.latitude,
                value.longitude),
            map: map,
            title: value.name+" "+value.distance,
            icon: iconImage,
            labelContent: value.name+" "+value.distance,
            labelAnchor: new google.maps.Point(50, 0),
            labelClass: "labels", // the CSS class for the label
            labelStyle: {opacity: 0.60}
        });

        google.maps.event.addListener(markerOtherLocations, 'click',
            function() {
                $.when(callAjax()).done(function(a){
                    // Redirect with AJAX
                    $.mobile.changePage("#detail-page",
                        { transition: "flip" });
                });
            });

        ...
        ...
        ...
    });

    map.fitBounds(bounds);
}

```

Code Extract 5.11: The drawMap function.

5.1.11 Constructing the Detail Page

Every time a user *clicks* on one of the markers on the map, it will construct the *detail page* on the fly. The detail page consists of the departure schedules from the *clicked* bus station. Data is fetched from the PHP file *process.php* via *AJAX* request. Using the *JSON* response, the page is constructed and displayed to the user.

```

function callAjax(){
    $.ajax({
        url: 'http://crossplatform.co.nf/process.php?url='
            +encodeURIComponent(value.localurl),
        dataType: 'json',
        success: function(data){
            $("#detail-page h1").html(data[1]);
            $("#contbl").empty();

            var thdata1 = "<th>" + data[2] + "</th>";
            var thdata2 = "<th>" + data[3] + "</th>";
            var thdata3 = "<th>" + data[4] + "</th>";
            var thdata4 = "<th>" + data[6] + "</th>";
            var tblrow = $("<tr></tr>");
            var thead = $("<thead></thead>");
            var table = $("<table data-role='table' id='time-
                table' '+' data-mode='reflow'
                class='ui-responsive table-stroke'>");

            tblrow.append(thdata1);
            tblrow.append(thdata2);
            tblrow.append(thdata3);
            tblrow.append(thdata4);

            thead.append(tblrow);
            table.append(thead)

            var row = $("<tr>");
            var flag = 0;
            var tbody = $('<tbody>');
            $.each(data, function(key, val){
                if(key >= 8){
                    if(key%4 != 3){
                        if(flag == 0)
                            row.append("<th>" + val + "</th>");
                        else
                            row.append("<td>" + val + "</td>");
                        flag++;
                    }
                    else if(key%4 == 3){
                        row.append("<td>" + val + "</td>");
                        tbody.append(row);
                        row = $("<tr>");
                        flag = 0;
                    }
                }
            })

            table.append(tbody);
            table.appendTo("#contbl");
            $('#detail-page').trigger('pagecreate');
        }
    });
}

```

Code Extract 5.12: The callAjax function.

5.2 Resulting Web Application

The resulting application is also available at <http://crossplatform.co.nf>. However since it is a location aware local bus application, the site functionalities outside of Växjö will not work. But one can hardcode the latitude/longitude coordinates of region around Växjö as the *current position* to check the functions of the application. Until now, the codebase of the application has been developed. The application has also been tested against the *functional requirements* listed out in the design section and works pretty fine. Below are some screenshots of the application on different browsers.

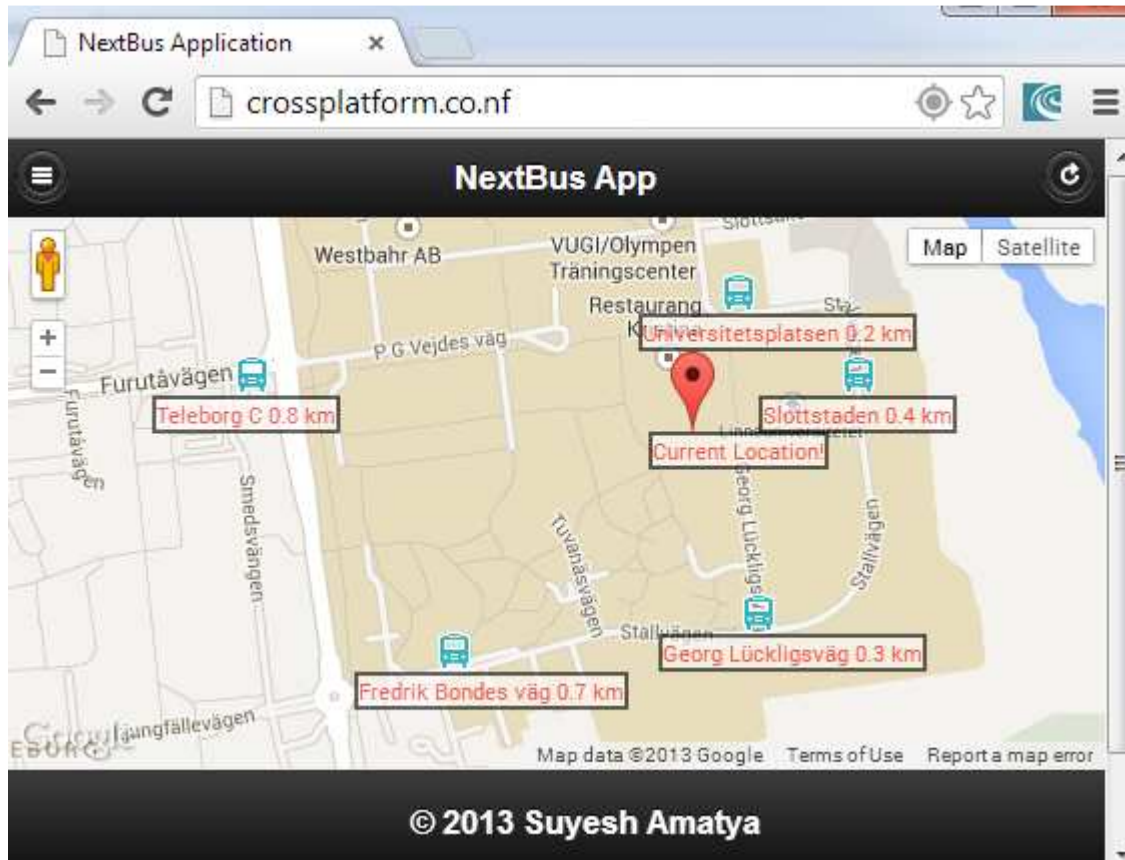


Figure 5.1: Home screen on the Chrome browser.

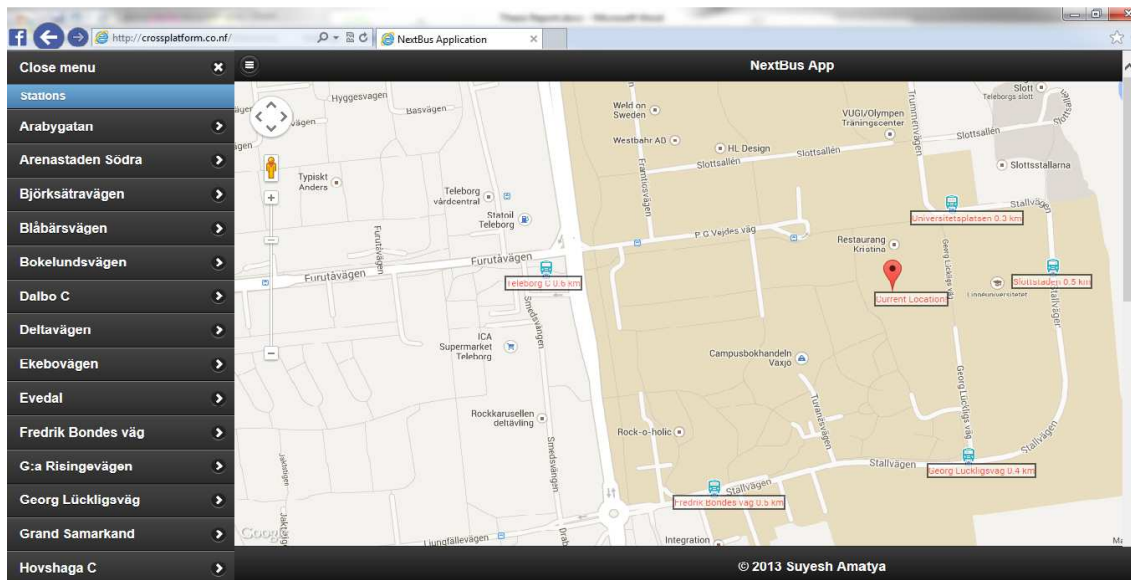


Figure 5.2: Menu panel expanded on Internet Explorer.

Firefox

Timings for station:

crossplatform.co.nf/#detail-page

Teleborg C (Växjö) kl. 16:31

Linje	Destination	Nästa tur (min)	Därefter
1	Hovshaga Kurortsv.	36	--
1	Hovshaga via Resecentrum	6	18
1	Teleborg	6	17
5	Sandsbro	26	56
5	Teleborg	17	47
241	Jät	ca 44	--
241	Växjö	ca 33	--
358	Växjö Teleborg	ca 55	--

Teleborg C (Växjö) kl. 16:31

Figure 5.3: Detail page as a normal tabular style presentation on maximized Mozilla Firefox.

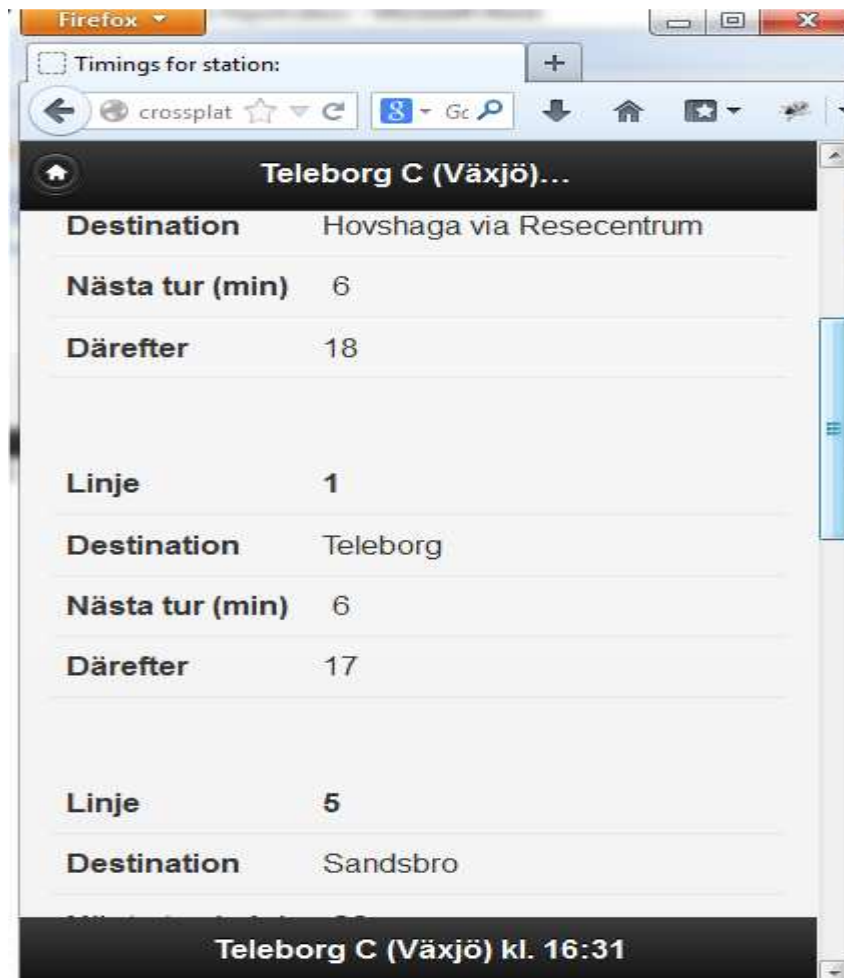


Figure 5.4: Detail page takes stacked presentation style when Mozilla Firefox window is scaled down.

Now we have a web application, we will build it into native applications for different platforms. With the same codebase, and may be few tweaks or changes per platform, the application will be packaged and built into native application using PhoneGap and platform specific native SDKs and tool chains for each targeted platform.

5.3 Building into Native Android Application

PhoneGap 2.9.0 was used for this purpose. Using *The Cordova Command-line Interface (CLI)* as mentioned here², one can create new projects, build them on different platforms, and run them within an emulator. However not all the things as mentioned there worked for android platform when I tried on my *Windows 7* machine. I was able to run the `create` command to initialize and create the project structure. But I could not run commands like `platform add`, `build`, or `emulate`, etc. So I just used the CLI tool to initialize the project code and structure, after which I used the platform IDE to develop them further.

A very little change in the codebase was done for it to be able to properly run in an android device. But that was mainly to account for the way how a PhoneGap-Android

² <http://goo.gl/5Ujy83>

project is structured and the correct way of using jQuery Mobile and PhoneGap together. No change whatsoever was required in the application logic. Appropriate platform-specific configuration and permissions were however added to *config.xml* and *AndroidManifest.xml* to be able to access the Geolocation feature.

```
<link rel="stylesheet" href="js/jquery.mobile-1.3.1.min.css" />

<script type="text/javascript" src="cordova.js"></script>
<script type="text/javascript" src="js/index.js"></script>

<script src="js/jquery-1.9.1.min.js"></script>

<script src="js/jquery.mobile-1.3.1.min.js"></script>

<script>
    var deviceReadyDeferred = $.Deferred();
    var jqmReadyDeferred = $.Deferred();

    document.addEventListener("deviceReady", deviceReady, false);

    function deviceReady() {
        deviceReadyDeferred.resolve();
    }

    $(document).on("pageinit", "#map-page", function () {
        jqmReadyDeferred.resolve();
        $.support.cors = true;
        $.mobile.allowCrossDomainPages = true;
    });

    $.when(deviceReadyDeferred, jqmReadyDeferred).
        then(doWhenBothFrameworksLoaded);

    function doWhenBothFrameworksLoaded() {
        ...
        ...
        ...
    }
</script>
```

Code Extract 5.13: Some changes made in the codebase.

As seen from the code extract 5.13, instead of using the *CDN* imports of *js* can *css* files, they have been used locally in the project with additional imports of *cordova.js* and *index.js*. Sometimes the *CDN* imports have been reported of not working, and same happened with me, so this is rather a precaution. To ensure that both the jQuery Mobile and PhoneGap are loaded before they can be used, jQuery *Deferred* object has been used. This is also more of a good practice than anything essential. So nothing much has been changed in the codebase.

The web application was built into a native android application using android SDK and tool chains in an Eclipse environment. The information about how to do this is also available on PhoneGap Documentation here³. *Eclipse Helios version 3.6.2* was used as a

³ <http://goo.gl/iKbCt>

platform IDE with *Android Development Tools (ADT) version 22.0.5* plugin. The application was built against *Android Platform 4.0 API Level 14* and was successfully tested on *Sony Xperia U (ST25i)* running *android version 4.0.4*. The screen captures of the application running on the device are provided below.



Figure 5.5: Home screen of the application.

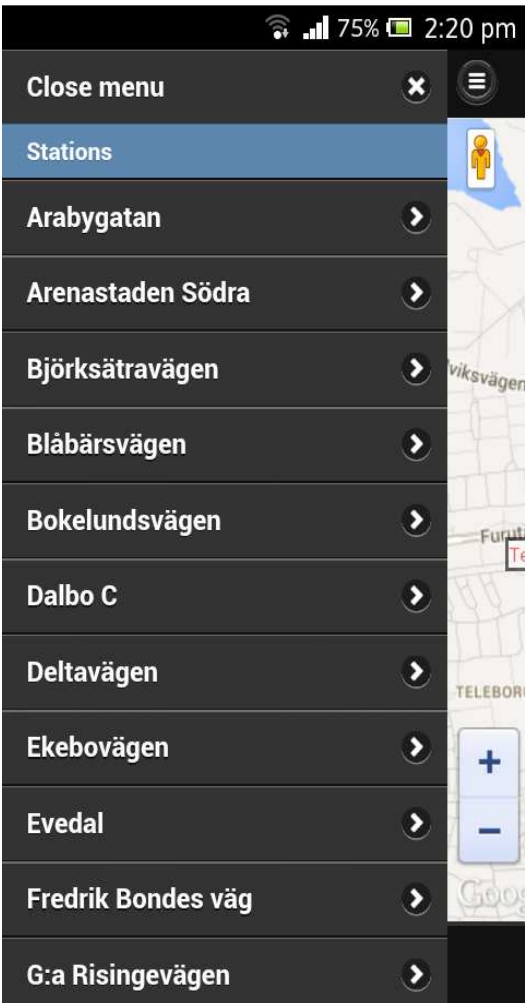


Figure 5.6: Menu panel.

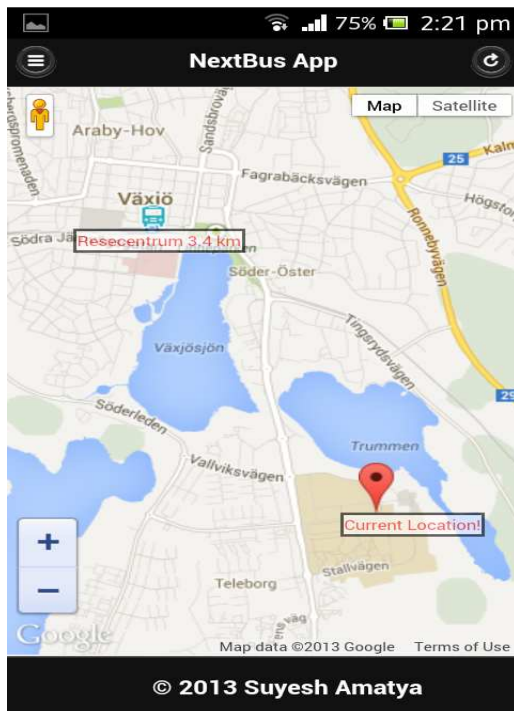


Figure 5.7: Manual station selection.



Figure 5.8: Stacked style detail page.



Figure 5.9: Normal tabular style detail page.

5.4 Building into Native iOS Application

To build iOS applications, one needs to have the Apple's required tool chains running on OS X operating system on Intel-based Macs. Hence the Apple's ecosystem is a closed and stricter one in terms of both hardware and software requirements when it comes to application development for iOS. For my purpose I used, a *MacBook* running *Mac OS X Lion version 10.7.5*. Using the *cordova-ios*, from *cordova-3.0.0* distribution,

and following the instructions here⁴, I created the project structure for the application. Then I used the native tool chains *Xcode version 4.6.3* and *iOS SDK 6.1* to further build the web application into a native iOS application.

A very little change in the main codebase and for the same reason, as was previously done for the android application, was performed. In fact after the changes, the *index.html* for android and iOS looked exactly identical. Appropriate platform-specific configuration was added to *config.xml* to be able to access the Geolocation feature. Some of the screenshots of the application running on *iPhone 6.1* and *iPad 6.1* simulators are provided below.

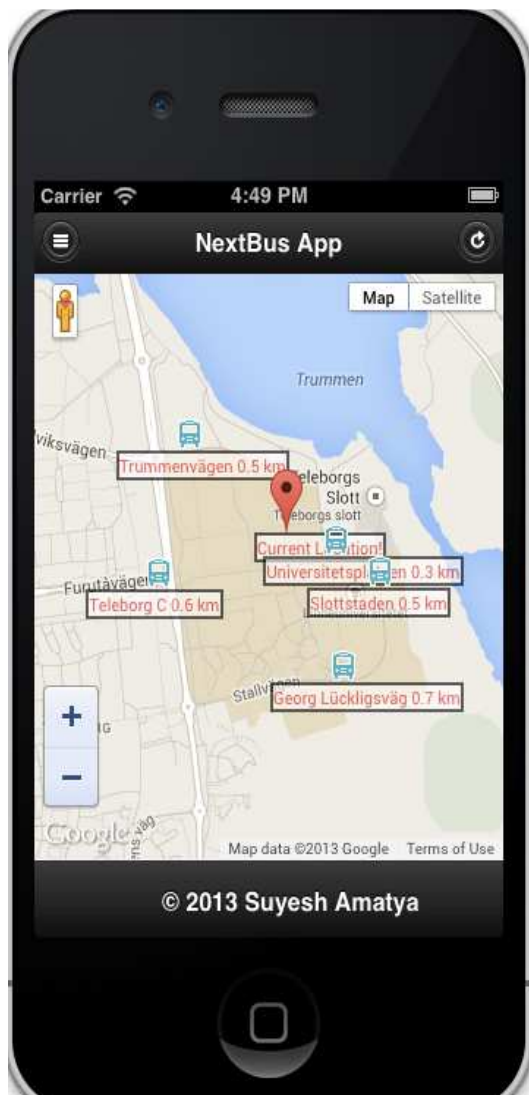


Figure 5.10: Home screen on iPhone 6.1 simulator.

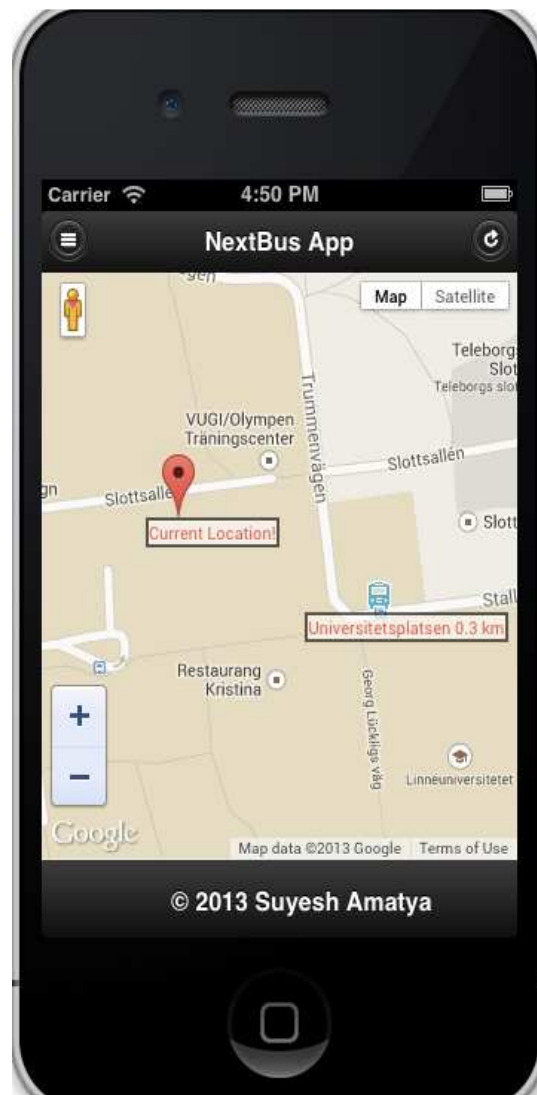


Figure 5.11: Manual station selection.

⁴ <http://goo.gl/LhwHu7>

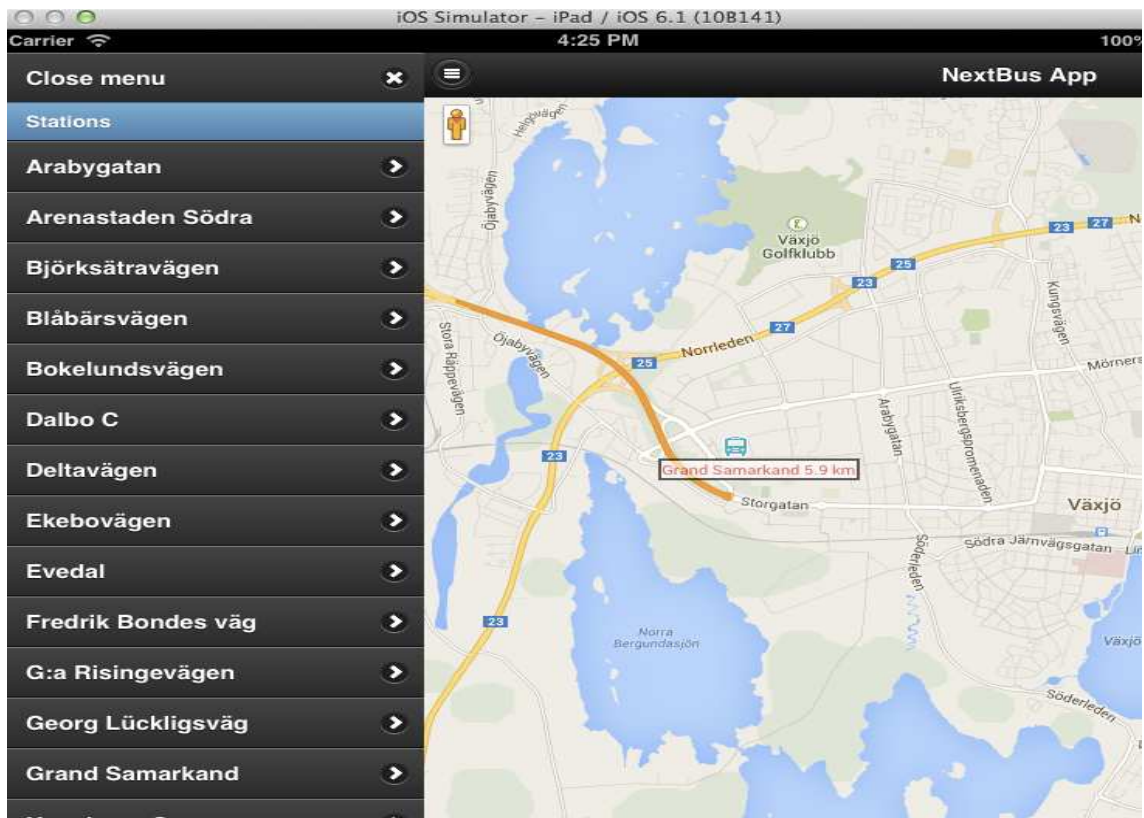


Figure 5.12: Menu panel expanded on iPad 6.1 simulator.

The screenshot shows the detail page for Teleborg C (Växjö) at 16:31. The page displays a table with bus line numbers, destinations, and arrival times.

Linje	Destination	Nästa tur (min)	Därefter
1	Hovshaga Kurortsv.	36	--
1	Hovshaga via Resecentrum	6	18
1	Teleborg	6	17
5	Sandsbro	26	56
5	Teleborg	17	47
241	Jät	ca 44	--
241	Växjö	ca 33	--
358	Växjö Teleborg	ca 55	--

Figure 5.13: Detail page.

The application was also successfully tested on a real device *iPod touch* (4th generation) running software version 5.1.1.

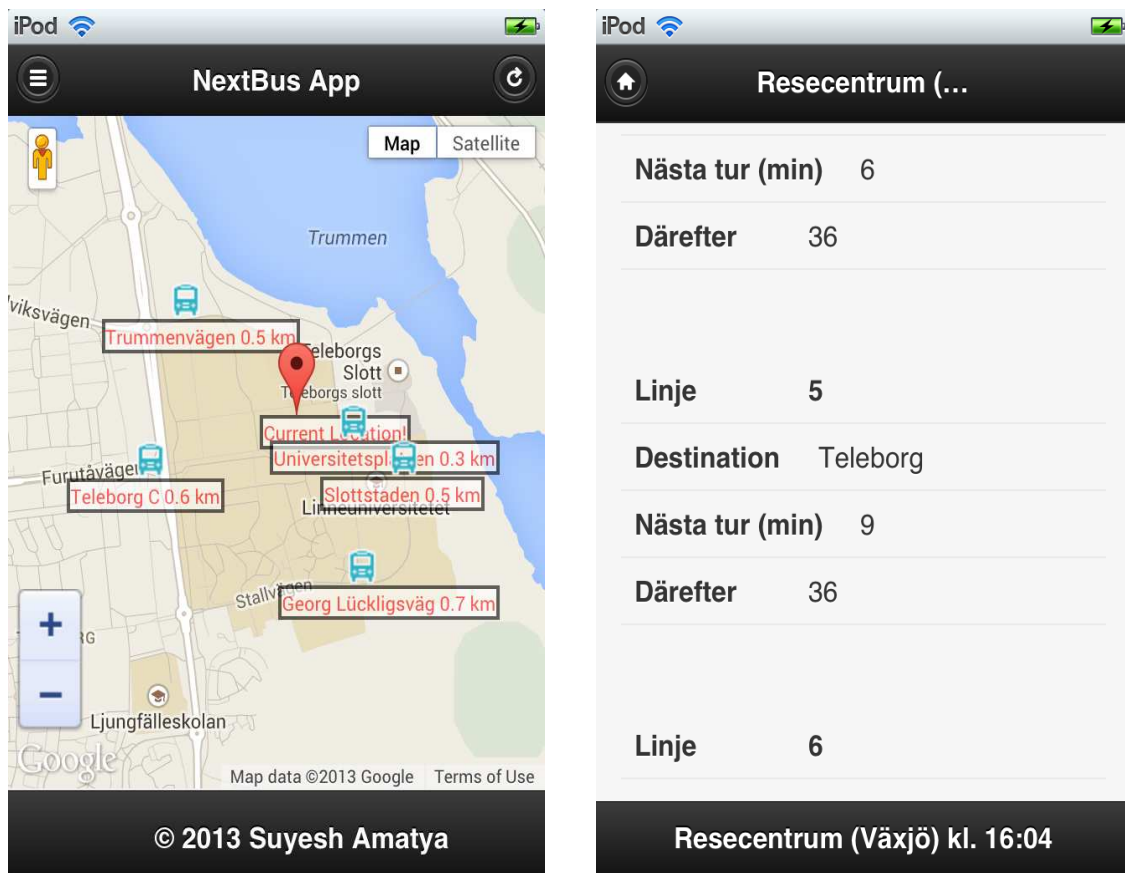


Figure 5.14: Home page and detail page on iPod touch.

5.5 Building into Native BlackBerry Application

Cordova for BlackBerry makes use of the *BlackBerry WebWorks* framework. BlackBerry WebWorks SDK are available for *BlackBerry 10*, *BlackBerry PlayBook*, *BlackBerry 7* and *earlier*. Using any of these SDKs, the web application can be built into BlackBerry WebWorks application which is a native BlackBerry application that runs on a BlackBerry smartphone or BlackBerry PlayBook tablet. Instead of using the command-line tool of SDK, I used a UI-based tool called *Ripple emulator* that is extension to the *Google Chrome* browser. The Ripple emulator uses the BlackBerry WebWorks SDK behind the scenes to build and package the application. However one can quickly test the application in a browser-based environment for different devices and platforms in real-time even without installing SDKs. So there is no hassle of installing multiple SDKs, simulators, redeploying the application or restarting the simulator, etc.

With that said it is always wise to test the applications in simulators and real devices. One can install appropriate WebWorks SDK and use it with Ripple emulator to build and package the application. Using Ripple emulator and SDK, one can generate the *.cod* or *.bar* file which can be installed and run on simulators and devices. I tested the application on the Ripple emulator and it worked fine. Some screen shots of application running on Ripple emulator emulating different platforms and devices are provided below.

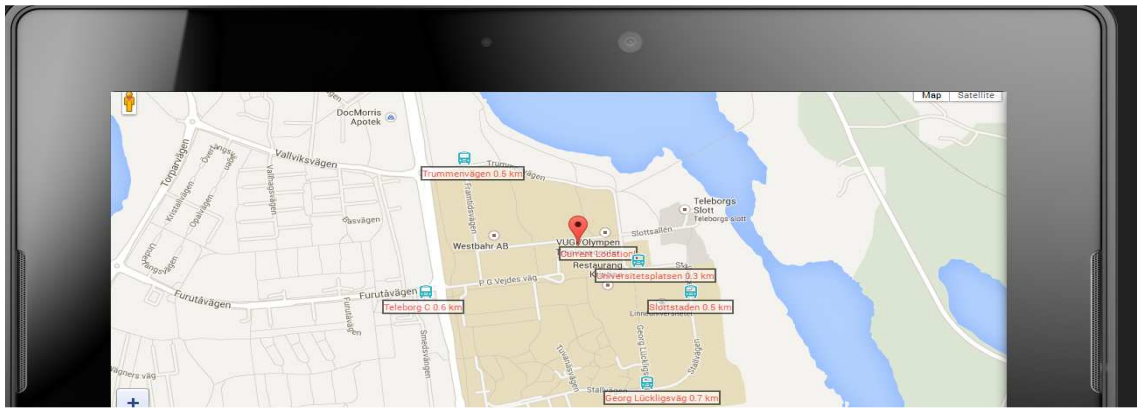


Figure 5.15: Home page on platform WebWorks-TabletOS and device BlackBerry PlayBook.



Figure 5.16: Detail page on platform BlackBerry 10 WebWorks and device BlackBerry Q10.

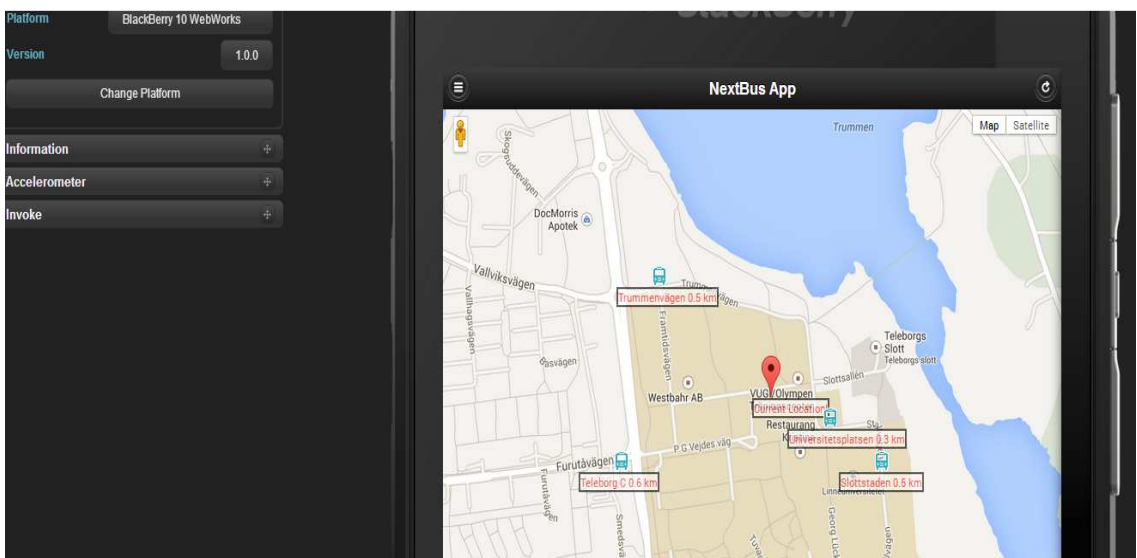


Figure 5.17: On platform BlackBerry 10 WebWorks and device BlackBerry Q10.



Figure 5.18: On platform WebWorks and device BlackBerry Bold 9700.

6. Analysis of the Application and Development Efforts

The preceding section described in length about the implementation issues. Here we reflect upon the application and the development efforts to validate the ideas presented in favor of the cross-platform mobile development.

6.1 Reflecting Upon the Development Efforts

Initially I started out with developing the codebase for the web application. At this point, there is no PhoneGap involved in the process whatsoever. The development, debugging and testing processes were very much like the normal web development. But of course the difference being that you have to take into account the small display and limited processing power of the mobile phones. I used jQuery Mobile framework to build the UI of the application. It provides lots of readymade UI widgets, components with predefined themes that work perfectly across mobile and desktop viewports and of course one can also define their own theme and CSS. Since it is also supported by PhoneGap, I did not need to reinvent the wheel and used it to make the pages, headers, footers, different buttons, panels, etc of the application. I also achieved the smooth page transitions, animation and Ajax navigation using jQuery Mobile.

Any preferred text editor can be used to write the code and I tested the application on desktop as well as mobile browsers. One can also mimic the mobile browser display on the desktop browser by resizing the desktop browser viewport to the size of the mobile browser. However a better result can be achieved by testing the application on Ripple emulator. The Ripple emulator is a multi-platform mobile environment emulator that is custom-tailored to mobile HTML5 application development and testing. It acts as an extension to the Google Chrome browser that allows you to quickly see how your application looks and functions on multiple mobile devices and platforms. JavaScript debugging, HTML DOM inspection, automated testing, and multiple device and screen resolution emulation in real-time can be done without redeploying the application or restarting the emulator. I also used Firebug to test and debug the application on the Mozilla Firefox browser.

After the web application was ready, the same codebase was ported to Android, iOS and BlackBerry platforms and built into respective native applications. At this point I used PhoneGap along with the native tool chains of the platforms. However for BlackBerry, I used the Ripple emulator to run the application instead of native tool chains. A very few changes in the original codebase was required and some platform-specific configurations and permissions were added. I successfully tested the respective native applications on real android and iOS devices. The BlackBerry version was also successfully tested on the Ripple emulator on the browser. Using Ripple along with the SDK, I was also able to generate the actual deployment application for BlackBerry as well but I did not test it on real BlackBerry device.

The main benefit of this hybrid approach of cross-platform mobile development was obviously the saved time and resources. With a single codebase I was able to deploy three native applications for the different platforms. Single development process was enough for the three target platforms. So it can be argued that just alone in the development process I was able to save twice the amount of development hours, provided if I know the different programming languages required by three different target platforms. Even though Android and BlackBerry OS both use Java as their programming language for native development, the flavors vary greatly while using for each of these platforms. So it is not always a straight forward transition even if you happen to know the language. Even for Java SE/ME developers, transitioning to android

take some time getting used to Harmony flavored Java implementation for Dalvik virtual machine. The testing process for individual platforms also took far lesser time, since majority of the testing had been performed during the web development part. And it is also well known issue that restarting emulators, redeploying the application during native development especially on android platform is very slow. Whilst the build process is separate for each platform, but fortunately it is not the most time consuming process while developing an application.

6.2 Analysis Results

The table 6.1 provides the results of analysis by shedding more light on these issues discussed and presenting them in more quantifiable terms.

Technical Issues	Android	iOS	BlackBerry
Developing the single common codebase	~200 hours		
Lines of code in the codebase	~454		
Tools used to build web application into native application	PhoneGap, Native tool chains	PhoneGap, Native tool chains	Ripple emulator, Native SDK
Lines of code added/changed to port as native application	~20	~20	No changes
Additional imports	cordova.js, index.js	cordova.js, index.js	No imports
Platform-specific configuration and permission for CORS and Geolocation	config.xml, AndroidManifest.xml	config.xml	config.xml
Restrictions from developer's perspective	No	Apple's hardware, OS required. Registration as Apple developer required even to deploy locally.	No
Build and deployment speed	Slow	Relatively fast	Quick (Browser based Ripple emulator used)
Native look and feel	OK	OK	Good
Performance and speed	Good	Good	Good

Table 6.1: Technical analysis of the application and development efforts across different platforms.

It took around 200 hours to develop the general codebase of the application. The application's `index.html` consists of around 454 lines of code. However there are also PHP background service process.php of around 54 lines of code and XML repository `busstations.xml` of around 262 lines of code written as part of the application. Since they are platform agnostic, they have not been taken into consideration in this analysis. To build this web application into native Android application, I used PhoneGap along with Android SDK and Eclipse IDE. For iOS, I used PhoneGap together with Xcode running on a Mac machine. One can also create new projects, build them on different platforms, and run them within an emulator or device using PhoneGap's command-line interface (CLI). But even if one wishes to use CLI for the whole application lifecycle, there should be appropriate SDKs installed for each target platform. Alternatively, I used CLI to initialize the project code and application skeleton, after which I used Android and iOS platforms' respective SDKs and IDEs to develop them further. For BlackBerry, most of the time I just used the Ripple emulator to test the application on the Chrome browser. I also generated the deployment application, using Ripple with the BlackBerry SDKs from the browser environment. PhoneGap and IDEs were not used for this purpose.

Around 20 lines of codes were added for Android and iOS platform in the original codebase. But these additions were not an absolute requirement for the application to be able to run on the devices; instead it was done mainly to follow the conventions of loading jQuery Mobile and PhoneGap when used together. While testing for BlackBerry, the web app directly ran on the Ripple emulator without any changes on the codebase whatsoever. However while testing on emulators; all the platforms require inputting the Geolocation values in one way or other to the emulators. Additional imports of PhoneGap files `cordova.js` and `index.js` were required on Android and iOS platforms. To allow Cross-Origin Resource Sharing (CORS) and Geolocation access, I also had to whitelist the domains and provide proper permissions respectively on each of the platform's `config.xml` file. For Android, additionally proper Geolocation permissions also need to be granted on `AndroidManifest.xml`. Android and BlackBerry did not provide any restrictions to deploy and test the application locally. iOS was a very closed ecosystem since you can only develop on Apple's hardware and operating systems. That means you have to use compatible tool chains of Xcode, Mac OS, Mac machines. And even to test the application locally on your iOS device, you have to get registered on Apple Developer Program which requires subscription fee. Build and deployment process was relatively slow with Android platform, and especially while working with emulators. For iOS it was faster than Android, and since the application was tested mainly in the browser environment with Ripple for BlackBerry, it was very quick. But trying to generate the deployment application with Ripple along with SDK was also understandably slow. Native look and feel wise, it was kind of satisfactory with both Android and iOS platforms. But since it was a prototype application I did not invest more time on the design aspects and mostly used the jQuery Mobile predefined themes, it was understandable. Spending more time on design and CSS can definitely make it look and feel like native applications. On the Ripple emulator it looked better. Even though there was fair bit of computation and complexity on the application logic, the performance and speed of the application was good across all the platforms.

7. Conclusion

Though native approach of mobile development allows the developers to exploit the full potential of the device hardware and platform features, it is not always the best approach when it comes to mobile development. For fairly high hardware and resource intensive applications, native approach might make a better case but the majority of applications being developed and available on application stores do not require such resources most of the time. Taking a native approach advertently/inadvertently means taking an expensive approach. This cost factor can sometimes be the single most important factor that dictates which approach might best suit one's needs and business requirements.

7.1 Answering the Research Questions

In this thesis work, I have explored the mobile development approaches that will allow developing cross-platform mobile solutions. We shall see why cross-platform solution is an ideal way to minimize the cost associated in developing mobile applications and targeting different platforms. The research questions formulated in the problem definition section will be answered below.

- a. *What kind of practices and technologies exist and how can they address the issues of developing the applications that can run across different platforms and variety of mobile devices?*

From the survey carried out, we can see that predominantly the already existing standard web technologies stack of HTML/JS/CSS are used to devise the cross-platform solutions. Standards-compliant mobile web browsers are included in every mobile operating system which makes these web-based solutions a really viable one. The web-based practices have been prevalent even from the initial days. However this has really gained momentum off late mainly due to the recent developments and added functionalities of HTML5 [69]. HTML5, today, supports rich set of growing features like *Semantics*, *Offline & Storage*, *Realtime/Communication*, *File/Hardware Access*, *Graphics/Multimedia*, *CSS3*, etc. The browser vendor community has strongly embraced HTML5 as most of the latest browsers from different vendors are increasingly supporting many different class of these features. Furthermore, JavaScript frameworks like *jQuery Mobile*, *jQT* (formerly *jQTouch*), etc leverage the HTML5 web applications to a great deal. They eliminate the cross-browser implementation incompatibilities and help build touch-optimized HTML5 based UI system accessible on wide variety of smartphone, tablet and desktop devices. Using these frameworks, one can easily incorporate *progressive enhancement* and *responsive web design* principles in the apps development. These frameworks also provide a host of readymade widgets which are flexible enough to be extended and easily themed to meet one's design needs. These web apps, for the most part, are mobile-optimized websites that run on the browser and are accessible through a URL. Deep linking with the underlying operating system and full exploitation of device capabilities are not possible with the web-based approach. Hybrid-approach can bridge this gap and move the web apps closer towards the native apps. Cross-platform frameworks like PhoneGap, Titanium, Sencha Touch wrap these web apps in a thin native container and act as a middleware exposing the native platform and device features through JavaScript APIs. The frameworks together with native tool chains are used to package and build these web apps into native apps and enable them for app store distribution and installation on devices. The resulting hybrid

apps are essentially native apps that run on full screen web view control leveraging the device's browser engine.

The prototype web app in this thesis was built using jQuery Mobile. Web apps or hybrid apps are powered by the device's browser engine. And jQuery Mobile supports full enhanced experience with Ajax-based animated page transitions for the vast majority of all modern browsers running on different devices and platforms. That being said, the feature phones and older browsers are also supported but might lack Ajax navigation features and others might support basic HTML experience that is still functional. So using framework like jQuery Mobile to tackle this fragmented landscape of mobile development makes a perfect sense. The web app worked perfectly across different mobile and desktop browsers of different platforms. The web app provided the general codebase, and then I used PhoneGap to build it into hybrid app for each of the targeted platforms. PhoneGap wraps the hybrid app in a thin native container and exposes the native APIs and device features through JavaScript APIs. This hybrid app can then be installed and run on the respective device/platform for which it was targeted against like any other native apps.

b. What are the potential benefits of adopting such practices and technologies to devise solutions in the mobile development?

Using jQuery Mobile eased a lot of task while developing the application. It provides a lot of widgets with predefined themes which one can easily use, extend and customize in their applications. Like in this application, I have used a jQuery Mobile HTML5 page widget with header, content and footer. Also used were buttons, panel, listview, responsive tables with one of the predefined themes and CSS being applied on the fly. Implementing Ajax navigation, animated page transitions were also easily achieved using the framework. One can also build the highly customized themes with highly polished visuals, CSS3 properties and gradients very easily using the *ThemeRoller*. Here one can drag and drop colors and mix and match for unlimited possibilities and the custom theme can be downloaded as a CSS file. All this means, the developers can do more in less time and build the application quickly without reinventing the wheel. Furthermore the framework takes care of the fragmentation in the ecosystem and allows developers to concentrate on application development. Using PhoneGap, with little or no code changes the same application was also built into native applications for different platforms and installed and run on different devices.

Apart from this cross-platform development provides the developers the different options to devise the mobile solutions. One can build the solution as a web app which is a series of web pages running on the server and does not need to be installed on the devices. Hence any updates to these mobile optimized websites will be automatically available without the users having to perform any updates thus guaranteeing the availability of the latest version for everyone. In case one prefers the app distribution through stores, local installation on devices and monetization possibility, hybrid approach can be undertaken. So cross-platform mobile development practices and technologies bring lots of benefits especially when considered from the context of the fragmented mobile landscape. Some of the main benefits can be summarized as:

- Relying on already familiar standard web technologies for development purposes.
- For the most part: code once, deploy anywhere.
- Shorter development time, thus reduced cost of development.

- Suitable for, but not limited to, rapid prototyping purposes and fast high fidelity mockups.

The main hallmark of the cross-platform mobile development is the unification of the whole or most of the application development life cycle. As can be seen in the figure 2.4 from section 2, hybrid approach has unified development and test process with separate build process for different target platforms. The web-based approach has entirely one stream lined development, testing and release process. This essentially means saving time and cost.

7.2 Future Directions

Cross-platform development promises a lot of potential and offers great advantages in the field of mobile development. Nevertheless it is still not fully matured and is in the initial phases of development as compared to native mobile development. HTML5 itself is an emerging standard and the real opportunities of cross-platform development will be more attractive once the HTML5 standard is fully developed and includes the functions for access to native features of the phone. Though some features have already been well deployed, few are in experimental phases and limitedly implemented, and a lot more features still need to be incorporated. However works are underway to bring as many features as possible into the web [69].

The world of browser market is also fairly fragmented with different vendors and implementations. As such the usual problems in web development might also persist in the cross-platform development. So the mantra “*code once, deploy anywhere*” might most likely become “*code once, debug everywhere*”. Also native code always comes out as a winner against JavaScript when performance is the factor. But proper design architecture, well-written and optimized JavaScript code can greatly offset this issue. Nowadays we can see growing number of web-based cross-platform solutions for high traffic, complex and performance intensive applications. There is larger pool of developers who are already familiar with the web development paradigm and this has contributed to the rising popularity and increasing adoption of cross-platform development approach. All these emerging trends in cross-platform development can also be deemed exciting with the new development in the mobile OS market where we have Firefox OS and Ubuntu Phone that use HTML5 as a main development language.

At the end of the day, choosing the development approach depends on the app objectives and the business realities it tries to address. For some specific needs, native approach might be a better choice but for the most part the web-based cross-platform solutions provide a very good and attractive alternative. Like mentioned previously, there is a continuous shifting towards cloud computing and web-based software with more and more applications and services coming to the Web. This has allowed the World Wide Web to evolve into a general-purpose software platform. As the prominence of the Web ecosystem grows more and its stakes as a software platform continues to increase, there is certainly exciting times and opportunities ahead for the cross-platform mobile development.

References

- [1] International Telecommunication Union, "The World in 2011 — ICT Facts and Figures," ITU, 2011. [Online]. Available: <http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf>. [Accessed 14 12 2012].
- [2] International Data Corporation, "Strong Demand for Smartphones and Heated Vendor Competition Characterize the Worldwide Mobile Phone Market at the End of 2012, IDC Says," IDC, 24 01 2013. [Online]. Available: <https://www.idc.com/getdoc.jsp?containerId=prUS23916413>. [Accessed 18 02 2013].
- [3] Gartner, Inc., "Gartner Says Free Apps Will Account for Nearly 90 Percent of Total Mobile App Store Downloads in 2012," Gartner, Inc., 11 09 2012. [Online]. Available: <http://www.gartner.com/newsroom/id/2153215>
<http://www.gartner.com/DisplayDocument?ref=clientFriendlyUrl&id=2126015>. [Accessed 18 02 2013].
- [4] International Data Corporation, "IDC Forecasts Worldwide Mobile Applications Revenues to Experience More Than 60% Compound Annual Growth Through 2014," IDC, 13 12 2010. [Online]. Available: <http://www.idc.com/about/viewpressrelease.jsp?containerId=prUS22617910§ionId=null&elementId=null&pageType=SYNOPSIS#.USIVyaVmj0e>. [Accessed 18 02 2013].
- [5] Y. C. et al., "Virtual Campfire - Cross-Platform Services for Mobile Social Software," in *Mobile Data Management: Systems, Services and Middleware, 2009. MDM '09. Tenth International Conference on*, 2009.
- [6] D. C. Rajapakse, "FRAGMENTATION OF MOBILE APPLICATIONS," National University of Singapore, 28 04 2008. [Online]. Available: <http://www.comp.nus.edu.sg/~damithch/df/device-fragmentation.htm>. [Accessed 18 12 2012].
- [7] J. E. G. et al., "Mechanism for dynamic deployment of plastic mobile cross-platform user interfaces," in *Electrical Engineering Computing Science and Automatic Control (CCE), 2011 8th International Conference on*, 2011.
- [8] M. C. Srinivas Ramadath, "Mobile Application Development: Challenges and Best Practices," Accenture, 2012. [Online]. Available: <http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-Mobile-Application-Development-Challenges-Best-Practices.pdf>. [Accessed 20 12 2012].
- [9] S. Kaltofen, "Design and implementation of an end-user programming software system to create and deploy cross-platform mobile mashups," 2010.
- [10] Adam M. Christ, "Bridging the Mobile App Gap," *Sigma Journal: Inside the Digital Ecosystem*, vol. 11, no. 1, p. 28, October 2011.
- [11] J. Biolchini, P. G. Mian, A. C. C. Natali and G. H. Travassos, "Systematic Review in Software Engineering," Rio de Janeiro, 2005.
- [12] Y.-W. Kao and C.-F. e. a. Lin, "A Cross-Platform Runtime Environment for Mobile Widget-based Application," in *2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Beijing, 2011.
- [13] P. E. Dickson, "Cabana: A Cross-platform Mobile Development System," in *ACM Special Interest Group on Computer Science Education*, Raleigh, North

Carolina, 2012.

- [14] B. Pan and K. e. a. Xiao, "Component-based mobile web application of cross-platform," in *10th IEEE International Conference on Computer and Information Technology (CIT 2010)*, Bradford, 2010.
- [15] J. Lyle and S. e. a. Monteleone, "Cross-platform access control for mobile web applications," in *IEEE International Symposium on Policies for Distributed Systems and Networks*, Chapel Hill, NC, 2012.
- [16] C. Xin, "Cross-Platform Mobile Phone Game Development Environment," in *International Conference on Industrial and Information Systems*, Haikou, 2009.
- [17] B. Siegfried, "Enhanced Student Technology Support with Cross-Platform Mobile Apps," in *39th annual ACM Special Interest Group on University and College Computing Services*, San Diego, California, 2011.
- [18] T. Lakshman and X. Thuijs, "Enhancing Enterprise Field Productivity via Cross Platform Mobile Cloud Apps," in *MCS '11 Proceedings of the second international workshop on Mobile cloud computing and services*, Bethesda, Maryland, 2011.
- [19] J. Sanneblad and L. E. Holmquist, "The GapiDraw Platform: High-Performance Cross-Platform Graphics on Mobile Devices," in *MUM '04 Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, College Park, Maryland, 2004.
- [20] Z. Qiu and L. e. a. Luo, "A Cross-platform Mobile Payment Solution Based on Web Technology," in *2012 Spring Congress on Engineering and Technology (S-CET)*, Xian, 2012.
- [21] X. Sun and Z. e. a. Du, "A Secure Cross-platform Mobile IM System for Enterprise Applications," in *International Conference on Uncertainty Reasoning and Knowledge Engineering*, Bali, 2011.
- [22] W.-H. Wang and C. e. a. Vick, "Cross-Platform Multi-Network Mobile Application Architecture," in *Proceedings of the IEEE 6th Circuits and Systems Symposium on (Volume:1) Emerging Technologies: Frontiers of Mobile and Wireless Communication, 2004*, Shanghai, 2004.
- [23] M. D. Bloice and F. e. a. Wotawa, "Java's Alternatives and the Limitations of Java when Writing Cross-Platform Applications for Mobile Devices in the Medical Domain," in *Proceedings of the ITI 2009 31st International Conference on Information Technology Interfaces, 2009. ITI '09*, Dubrovnik, 2009.
- [24] P. Smutny, "Mobile development tools and cross-platform solutions," in *2012 13th International Carpathian Control Conference (ICCC)*, High Tatras, 2012.
- [25] B. Zhang and W. e. a. Wang, "Research and Implementation of Cross-platform Development of Mobile Widget," in *2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*, Xi'an, 2011.
- [26] N. Yu, C. Liu and J. Chen, "The Development and Application of Cross-Platform Coal Mine Mobile Information System," in *2011 International Conference on Computer Science and Network Technology (ICCSNT)*, Harbin, 2011.
- [27] S. Kaltofen, M. Milrad and A. Kurti, "A Cross-Platform Software System to Create and Deploy Mobile Mashups," in *ICWE'10 Proceedings of the 10th international conference on Web engineering*, Vienna, 2010.
- [28] A. Taivalsaari and T. Mikkonen, "Objects in the Cloud May Be Closer Than They

- Appear Towards a Taxonomy of Web-Based Software," in *2011 13th IEEE International Symposium on Web Systems Evolution (WSE)*, Williamsburg, VA, 2011.
- [29] M. Ubl, "Improving the Performance of your HTML5 App," 14 February 2011. [Online]. Available: <http://www.html5rocks.com/en/tutorials/speed/html5/>. [Accessed 17 October 2013].
 - [30] "Can I use... Support tables for HTML5, CSS3, etc," [Online]. Available: <http://caniuse.com/#feat=transforms3d>. [Accessed 17 October 2013].
 - [31] Andre Charland, Brian LeRoux, "Mobile Application Development: Web vs. Native," *Queue - Data*, vol. 9, no. 4, p. 4, April 2011.
 - [32] Adam M. Christ, "Bridging the Mobile App Gap," *Sigma Journal: Inside the Digital Ecosystem*, vol. 11, no. 1, p. 29, October 2011.
 - [33] B. e. a. Vogel, "Architectural Concepts: Evolution of a Software System Across Design and Implementation Stages in Dynamically Changing Environments," in *2012 IEEE 36th International Conference on Computer Software and Applications Workshops (COMPSACW)*, Izmir, 2012.
 - [34] Andre Charland and Brian LeRoux, "Mobile Application Development: Web vs. Native," *Queue - Data*, vol. 9, no. 4, p. 2, April 2011.
 - [35] Kevin Whinnery, "Comparing Titanium and PhoneGap," Appcelerator Inc., 12 May 2012. [Online]. Available: <http://developer.appcelerator.com/blog/2012/05/comparing-titanium-and-phonegap.html>. [Accessed 18 March 2013].
 - [36] C. S. a. T. Martin Fowler, "About Martin Fowler," [Online]. Available: <http://martinfowler.com/aboutMe.html>. [Accessed 19 March 2013].
 - [37] C. S. a. T. Martin Fowler, "Developing Software for Multiple Mobile Devices," ThoughtWorks, Inc., 19 June 2012. [Online]. Available: <http://martinfowler.com/articles/multiMobile/#ui-translate>. [Accessed 19 March 2013].
 - [38] H. Heitkötter, S. Hanschke and T. A. Majchrzak, "Comparing cross-platform development approaches for mobile applications," in *WEBIST*, Porto, 2012.
 - [39] D. Gavalas and D. Economou, "Development Platforms for Mobile Applications: Status and Trends," *Software, IEEE*, vol. 28, no. 1, pp. 77-86, 2010.
 - [40] "Getting Started Guides," Adobe Systems Inc, [Online]. Available: http://docs.phonegap.com/en/2.5.0/guide_getting-started_index.md.html#Getting%20Started%20Guides. [Accessed 20 March 2013].
 - [41] "Titanium Mobile Development Platform," Appcelerator Inc., [Online]. Available: <http://www.appcelerator.com/platform/titanium-platform/>. [Accessed 20 March 2013].
 - [42] "RhoMobile Suite," Motorola Solutions, [Online]. Available: <http://www.motorola.com/Business/US-EN/Business+Product+and+Services/Software+and+Applications/RhoMobile+Suite>. [Accessed 20 March 2013].
 - [43] "Adobe PhoneGap Build FAQ," Adobe Systems Incorporated, [Online]. Available: <https://build.phonegap.com/faq>. [Accessed 21 March 2013].
 - [44] "Quick Start," Appcelerator Inc., [Online]. Available:

- http://docs.appcelerator.com/titanium/latest/#!/guide/Quick_Start. [Accessed 21 March 2013].
- [45] "RhoStudio," Motorola Solutions, Inc., [Online]. Available: <http://www.motorola.com/Business/US-EN/Business+Product+and+Services/Software+and+Applications/RhoMobile+Suite/RhoStudio>. [Accessed 21 March 2013].
 - [46] "Supported Features," Adobe Systems Inc., [Online]. Available: <http://phonegap.com/about/feature/>. [Accessed 22 March 2013].
 - [47] "API Documentation," Appcelerator Inc., [Online]. Available: <http://docs.appcelerator.com/titanium/latest/#!/api>. [Accessed 22 March 2013].
 - [48] "Mobile API Compatibility," Motorola Solutions, Inc., [Online]. Available: <http://docs.rhobile.com/rhoelements/apicompatibility>. [Accessed 22 March 2013].
 - [49] Andrew Trice, "PhoneGap Blog," Adobe Systems Inc., 02 May 2012. [Online]. Available: <http://phonegap.com/2012/05/02/phonegap-explained-visually/>. [Accessed 23 March 2013].
 - [50] "RhoHub Tutorial," Motorola Solutions, Inc., [Online]. Available: <http://docs.rhobile.com/rhohub/tutorial>. [Accessed 24 March 2013].
 - [51] "RhoElements Application Models," Motorola Solutions, Inc., [Online]. Available: <http://docs.rhobile.com/rhoelements/rhoelements-app-models>. [Accessed 24 March 2013].
 - [52] "RhoMobile Suite Brochure," Motorola Solutions, Inc., June 2012. [Online]. Available: http://www.motorola.com/web/Business/Products/Software%20and%20Applications/RhoMobile_Suite/_Documents/_StaticFiles/RhoMobile_Brochure_Final.pdf. [Accessed 24 March 2013].
 - [53] "Google Maps JavaScript API v3," Google Inc., 24 June 2013. [Online]. Available: <https://developers.google.com/maps/documentation/javascript/>. [Accessed 20 August 2013].
 - [54] D. Hjältström, "Utilizing web standards for cross platform mobile development," Växjö, 2012.
 - [55] K. Andersson and J. Dan, "Mobile e-Services Using HTML5," in *2012 IEEE 37th Conference on Local Computer Networks Workshops (LCN Workshops)*, Clearwater, FL, 2012.
 - [56] A. Juntunen, E. Jalonen and S. Luukkainen, "HTML 5 in Mobile Devices – Drivers and Restraints," in *46th Hawaii International Conference on System Sciences*, Hawaii, 2013.
 - [57] "Geolocation API Specification," W3C, [Online]. Available: <http://www.w3.org/TR/geolocation-API/#introduction>. [Accessed 21 August 2013].
 - [58] P. Mark, "Geolocation - Dive Into HTML5," [Online]. Available: <http://diveintohtml5.info/geolocation.html>. [Accessed 21 August 2013].
 - [59] "jQuery Home Page," The jQuery Foundation, [Online]. Available: <http://jquery.com/>. [Accessed 21 August 2013].
 - [60] "jQuery Wiki Page," [Online]. Available: <http://en.wikipedia.org/wiki/JQuery>. [Accessed 21 August 2013].

- [61] "Usage Statistics and Market Share of JavaScript Libraries for Websites, August 2013," W3Techs, [Online]. Available: http://w3techs.com/technologies/overview/javascript_library/all. [Accessed 21 August 2013].
- [62] "jQuery Mobile Home Page," The jQuery Foundation, [Online]. Available: <http://jquerymobile.com/>. [Accessed 21 August 2013].
- [63] "Why use CSS?," Mozilla Developer Network, 2 August 2013. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_Started/Why_use_CSS. [Accessed 21 August 2013].
- [64] "CSS3 Introduction," W3Schools, [Online]. Available: http://www.w3schools.com/css3/css3_intro.asp. [Accessed 21 August 2013].
- [65] "CSS Current Work," W3C, 1 August 2013. [Online]. Available: <http://www.w3.org/Style/CSS/current-work>. [Accessed 21 August 2013].
- [66] "PhoneGap | About," Adobe Systems Inc., 2013. [Online]. Available: <http://phonegap.com/about/>. [Accessed 25 August 2013].
- [67] "About Apache Cordova," The Apache Software Foundation, 2013. [Online]. Available: <http://cordova.apache.org/>. [Accessed 25 August 2013].
- [68] "Same-origin Policy," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Same_origin_policy. [Accessed 27 August 2013].
- [69] S. Amatya and A. Kurti, "Cross-Platform Mobile Development: Challenges and Opportunities," in *ICT Innovations 2013*, vol. 231, Springer International Publishing, 2014, pp. 227-228.



Linnæus University

Sweden

Faculty of Technology
SE-391 82 Kalmar | SE-351 95 Växjö
Phone +46 (0)772-28 80 00
teknik@lnu.se
[Lnu.se/faculty-of-technology?l=en](https://lnu.se/faculty-of-technology?l=en)