mPOS for iOS

Final report

Spring 2013

Jón Hilmar Gústafsson

Ragnar Ævar Jóhannsson

B.Sc Computer Sciense

School of Computer Sciense

Instructor: Hlynur Sigurþórsson

Examinar:Torfi Leifsson

T-404-LOKA

School of Computer Sciense

Table of Contents

Table of Figures

Preface

This report describes the final project entitled “mPOS for iOS” which is a B.Sc final project at the Department of computer science at Reykjavík University spring 2013. The mobile Point Of Sale (mPOS) is an application for the iOS mobile operating system[1] developed for a company named Handpoint[2]. mPOS is an application that enables merchants to accept credit/debit card payments using an iPhone and a card reader, the MPED-400 from Datec, that is custom made for Handpoint. In short the merchant inputs the total amount for the product or service that he/she is selling and sends the amount using wireless bluetooth technology to the card reader, the customer then inserts his/her card into the reader and then enters the personal identification number (PIN) and the transaction takes place. A receipt for the transaction is created in the application that the merchant can send to the customer as an email attachment.

[1] http://www.apple.com/ios
[2] http://www.handpoint.com

1. Introduction

With the introduction of smartphones to consumers, starting with the first iPhone in 2007, there have been a steady increase in hardware and software capabilities and today the markets two biggest smartphone manufacturers are Apple with their iPhones and Samsung with a wide variety of different models. There are other hardware manufacturers as well worth mentioning like HTC and Motorola and recently Nokia. Although these smartphones and other mobile devices like tablets all have a lot in common, they do not all share the same operating system, iOS is for Apple devices and Android for a wide range of devices and manufacturers. Windows OS for smartphones and tablets are on the rise but market shares are slim compared to iOS and Android. With the power of these device increasing all the time and more and more people buying them, its no wonder that commerce are looking into how to use mobile devices as a business model, for devices running different OS.

1.1 About mPOS for iOS

The project we worked on is called mPOS and is a mobile application running on the iOS platform and is for mobile devices from Apple Inc, the iPhone, iPad and iPod Touch. These devices run the mPOS application and connect to the MPED-400 card reader using wireless bluetooth technology to enable card payments. We worked on this project in collaboration with Handpoint, an Icelandic payment solution company, and the main reason for this collaboration was that Handpoint had developed solutions for Android mobile devices, Win CE/Mobile and Desktops but due to market demands they needed a solution for the iOS platform as well and that is where our project comes into the picture, create an application that runs on the iOS platform, using the Handpoint Objective-C API framework to perform payments using the card reader. The project has two parts. The first part of the project is to create a prototype, the iPhone client, conducting payments using the framework for mPOS and the second part is to deliver a working iOS prototype to Handpoint, by using other features related to mobile devices, like posting to social media, within the application itself and taking advantage of the all the possibilities that the iPhone smartphone offers

Handpoints mPOS application is for small and medium sized business, who need a mobile and cost efficient solutions to handle sales of their products or services.

mPOS is a payment solutions that is easy to use and has great mobility and allows the users to complete card transactions in a fast and efficient way. Unlike other payment solutions mPOS is cost efficient and offers easy access to sales records and the mobility of the system give the users great freedom to conduct payments, without having to be stuck to a big old cash register.

mPOS gives the user the freedom to sell their product or service whenever wherever, by connecting through a mobile network.

1.2 About Handpoint

Handpoint is an Icelandic company that was founded in 1999 and as of spring 2013 there are about 30 employees. Handpoint now has two offices, one in Hlidarsmari Iceland, and one in the United Kingdom (UK). Handpoint is a growing company focusing on payment solutions for different businesses. Payment solutions that Handpoint develops for their customers is all about accepting “chip&PIN” card payments, using a wide variety of devices such as smartphones, tablets and other handheld devices.

Working with payments solutions, security is a core part of Handpoints operations and the products are PCI-DSS[3] certified and uses different encryption methods for secure transaction that Handpoint handles for their customers. The PCI-DSS certificate was created to prevent security breaches. Handpoint is compliant with the PCI-DSS security standard created by Visa, Mastercard, JCB, Discover, American Express and Discover. The abbreviation stands for Payment Card Industry Data Security Standard.

All PCI-DSS certified companies like Handpoint have to undergo inspection on regular basis to maintain the license to handle, process, store and transmit debit/credit card data.

Handpoint stores card data for their customers and by handling this data, all procedures are more simpler for their customers saving them time, money and effort that they would otherwise have to deploy to undertake PCI inspections for certificates.

[3] https://www.pcisecuritystandards.org/security_standards/

Handpoint also co-brands payment solutions with enterprises and offers application programming interfaces (API) for developers for different devices and platforms like Android, Windows CE and Java for integrating payment solutions for existing enterprise applications or developing solutions from scratch for customers. After our project Handpoint now also have a prototype for the iOS platform. Handpoint are currently evaluating the possibility of exposing their API code for all the different mobile platforms to help developers create custom application for potential customers of the mPOS solution.

2. An overlook at the mPOS project

To explain how the project works and what task we needed to solve while creating the mPOS prototype for Handpoint, let’s look at how the system works as a whole and take a closer look at the Objective-C API framework, the Heft Library that are both parts of Handpoints Software Development Kit (SDK). The pictures in the following sections will give you the overall parts of the system and what parts interact with each other.

2.1 mPOS from the users perspective

1 Type in the amount on your mobile app that connects to the card reader via bluetooth

2 Hand the card reader to your customer. He/she inserts the card and verifies the transaction with a PIN.

3 Now you can email the receipt to your customer. All card details are encrypted.

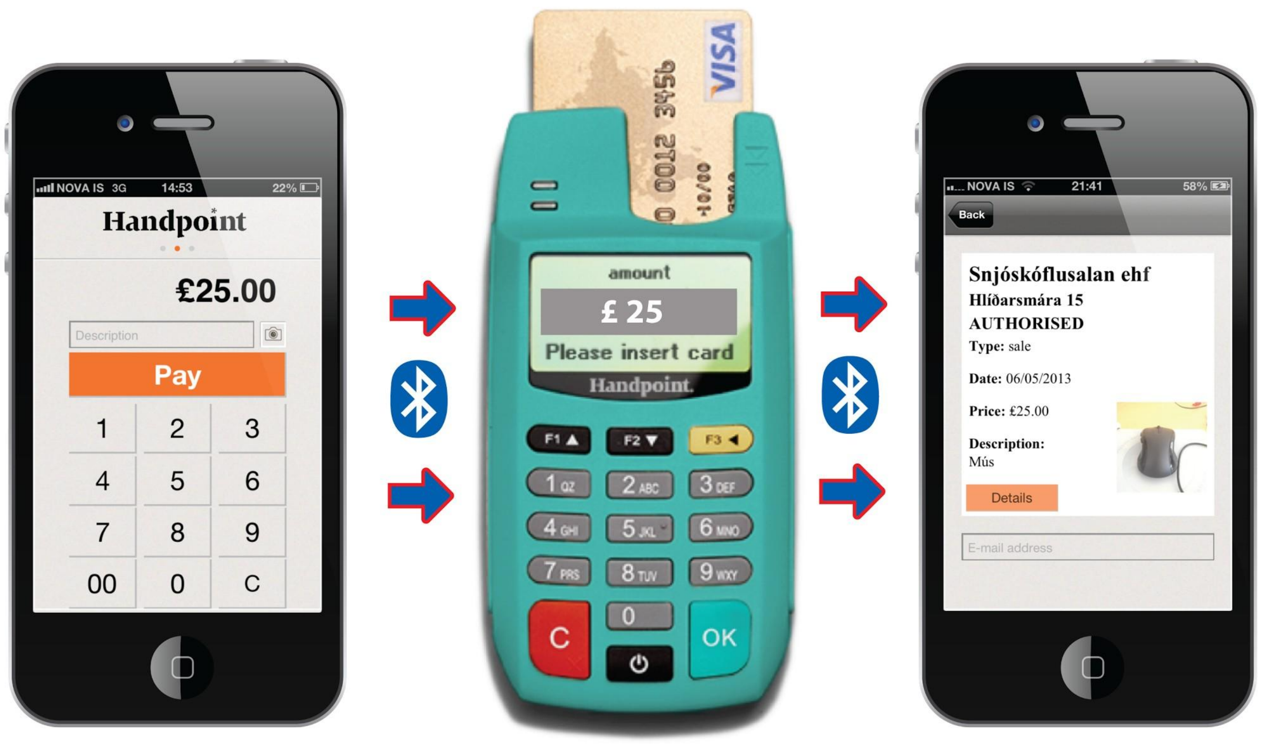

Figure 1 - Application conducts card transaction

2.2 mPOS from the developers perspective

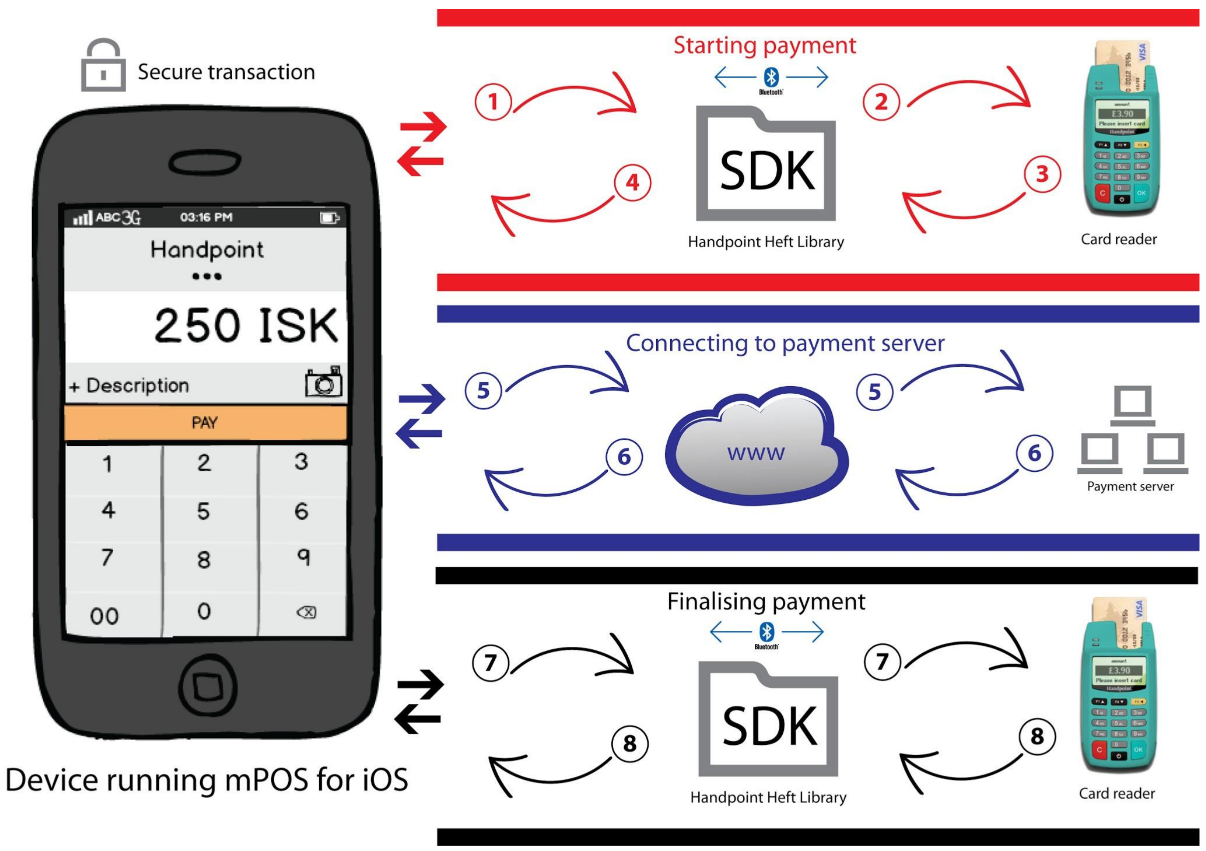

Figure 2 - Overview of the different steps of the Handpoint service

The three steps:

Starting payment:

1 The user initiates a transaction by pressing the Pay button in the application. The application calls a function in the Heft library with the amount and currency as parameters.

2 The Heft library sends these information forward to the card reader, which then displays the amount and asks for the customers card. When the card has been inserted, the card reader asks the customer to enter his PIN number and press OK. The card reader then verifies that the PIN number matches the card.

3 If the PIN number matches the card, the card reader encrypts all the necessary data, card and transaction information, and prepares to send the encrypted package to the payment servers at Handpoint using the Heft library.

4 The readers sends the encrypted package back to the Heft library.

Connecting to payment server:

5 The Heft library uses the internet connection on the device it is running on to send the encrypted package from the card reader to the payment

servers. The application on the device using the Heft library has no access to the information in these encrypted packages. When the payment servers receive the encrypted package, they can decrypt it and read the relevant information from it. The payment servers then use these information to check for coverage at the relevant card company. The payment servers than take the response from those checks and encrypt them also in an secure package.

6 The payment servers then send the encrypted package back to the device through the internet.

Finalising payment:

7 The Heft library then passes it along to the card reader. At that point the card reader can decrypt the response package from the payment server and access the transactions results. Then the card reader displays the response, whether the transaction was accepted, declined or one of numerous other responses available.

8 Simultaneously to displaying that information on the card reader screen he sends the relevant information back to the device via the Heft library so the application can display the result, create a receipt, store the transaction data or many of other possible actions the application could perform.

To be clear, no sensitive data like card information, are stored in the application or the card reader. All sensitive data is encrypted and decrypted at the card reader and payment servers.

3. Our goals for the mPOS application

Like stated in the section 1.2 about Handpoint our main goal was to create a mPOS prototype for iOS that uses the newly developed Objective-C API, a part of Handpoints SDK. The application should be able to connect to the card reader using bluetooth wireless technology and conduct a transaction from start to finish. The application will both be a working test application for the new API and a demo client to show how the Handpoint solution can be used with iOS devices. The mPOS application should be able to performs the following actions:

- Input the total amount of the sale, using a keyboard layout, that is quite similar to the basic keyboard of a calculator with a added pay button that starts the transaction phase.
- The iPhone needs to connect to the card reader using bluetooth, and it can only connect to the designated card reader, since on release the iPhone and the card reader are paired together with a special encrypted hex key, that is obtained when a contract is signed by the merchant using the mPOS application and the payment service, that Handpoint offers their customers.
- As shown in section 2.2 the transaction procedure, the main goal is to complete a transaction from start to finish, sending a data packet from the iPhone to the card reader. The card reader then encrypts the data packet, sends it back to the iPhone. The iPhone and the card reader communicates using bluetooth. When the encrypted data packet has arrived to the iPhone, the packet is sent through to the card vendor server, who checks if there is coverage for the transaction, and then sends the packet back to the iPhone. If there was coverage for the transaction the sales is complete. If the customer did not have coverage the transaction fails.
- Create a receipt, reading the XML from the packet and using some of the information like date and amount and merge it into a portable document file (PDF).
- Send the PDF receipt to the customer, as an email message attachment, by typing in the customers email address.
- Other features that the application could have and that we will work on if time lets us are e.g. an calculator to do regular arithmetic operation like addition and subtraction, posting information from the merchant on Facebook and Twitter from within the application, printing receipt to name a few of the ideas we have for the mPOS application.

3.1 The deliverable for the project

Our goal, in collaboration with Handpoint, is to make a prototype version of the mPOS application running on the iOS platform, that conducts a transaction in a secure way. When planning how to make the mPOS application possible with the Handpoint staff we created a requirement list for the project. At the beginning of the project there were 23 A requirements 3 B and 11 C but as we started working on the project, when needed to make some changes to the requirements, by splitting some of them up into smaller parts and adding new ones. We ended up with 34 A requirements, up from 23. The requirements from Handpoint were not changed during the projects but the changes mentioned above were changes by our team to better understand the overall project. The visual layout for the project like icons, colour theme, logo etc were given to us from the beginning of the project and all changes that took place were done in collaboration with Handpoint.

3.2 Our work environment

Since Handpoint is dealing with money transactions security and certified standards are very tight. During our project at Handpoint we have never actually seen what goes on behind the scenes, the process of conducting a transaction on the mPOS application, running on Handpoints backends. We have only had the Handpoint SDK too work with while developing the application. Handpoints SDK was updated during our work there and that forced us to make some minor changes to some features that we had up an running against the old SDK.

Handpoint created a workspace for us in their office and set up an office like environment for us to get started with all the basic stuff, computer screens, tables, chairs, an iPhone and the Apple Development Licence, that enables developers to run applications on devices in our case the iPhone. We also had access to the staff at Handpoint, incase we needed something. It´s been a real joy for us as a team working at Handpoint. Since we were also taking classes at RU during the project we also conducted a lot of work on the project at RU and at our respective homes.

4. User Groups

The mPOS application main user group are small and medium sized business, e.g. cab drivers, hairdressers, traveling salespersons or just about every other person or small groups of people that are selling products and/or services that need a mobile solution to accept debit/credit cards and are not bound to a specific store or location, which is usually the case with older card reading solutions like the gsm based card reader. Since the potential users of the application is so wide the mPOS solution has to be simple and easy to use and fit the 19 year old who is selling old stuff at a flea market but also make sense for the local charity foundation, usually consisting of older and not that technical people who are raising money by selling e.g. cakes and hand knitted products for some local charity.

4.1 Scenarios

When working on the mPOS application our team created some potential scenarios on how the application might be used and in what kind of environment, to get a better understanding on what our focus should be during the development phase.

4.1.1 The Cab Driver

In some countries like the UK not all cab drivers accept debit/credit cards. Since being a cabby means that your work is all about mobility, the mPOS solution offers them an easy, practical and simple solution to accept card payments from their customers. When the trip has been concluded the cabby puts the total amount for the trip on his iPhone, the customer puts his card in the card reader and enters the pin and then the transaction completes. In the long run it could also minimize the need for the cabby to carry a lot of cash in his car.

4.1.2 The Hairdresser

Most of us need a haircut every now and in many saloons where hairdressers perform their work, they actually rent a seat, from the owner of the saloon, and have separate bookkeeping between them. Sometimes the desks are crowded with card readers taking up a lot of space. It would be a whole lot more convenient for the hairdresser to put in the amount on a phone and conduct the transaction using the mPOS solution, it would save space on the desk, he or she could finish the payment procedure from any place with in the saloon. Another benefit of mPOS for hairdressers would be that the application enables them to take house calls, e.g. old and disable people could get a haircut at their place, and the hairdresser would use the mobile mPOS application to receive payments on the spot. The mPOS application would also give the hairdresser a chance to become a truly mobile hairdresser, visiting e.g. a company on a regular basis, setting up a small saloon at some designated place within the companies premises and cut some of it's staff and conducting transaction on the spot.

4.1.3 The Contractor

We all need some professional assistance every now and then, be it a plumber, painter or some other professionals to help us solve problems that we might have at home or wherever we have a situation requiring us to get help from professionals. Let look at some example where the mPOS application, would benefit these contractors. You have hired a painter to paint some walls at your place and most of the time, the painter finishes the job, writes some receipt for the job, makes the bookkeeper at the firm set up a bill and the bill is sent out to you, with all the extra work and added cost of this double if not triple work of getting the bill to you. With the mPOS application the painter would just let you know that the job is about to be completed, you get back to your place, check out the work and if your are happy as a puppy with the job, the painter can bring out the card reader, insert your card, you enter the pin and the transaction is conducted and if you would like a receipt, the painter just types in your email and sends it to you. mPOS could also be used for getting help with flat tires or when you run out of gas, just call some service provider, they show up, fix your tyre or fill up your tank, you pay them and of you go. The mPOS is usable for all kinds of contractors, it really just about imagining a scenario that could happen, how it is solved and how the payment is conducted. It would simplify the payment process a whole lot and get rid of the middleman, usually banks, that charge contractors for using billing services.

4.1.4 The Artist

The creative artist, although many of them don´t like to admit it, need to sell stuff to have money to pay rent, buy red wine and a Mac computer. Artists could use the mPOS application for quick and easy payments when showing off their artwork, whether it is in an art gallery, on a market, in a park or just at their home or studio. A lot of these sales are conducted on impulse from the buyer’s perspective and being able to accept card payments could make or break the sale. For example Icelanders do carry very little cash, but are very frequent users of debit and credit cards. Accepting card payment on the go, should help out artists, especially for impulse buys, and like in the others scenarios state in earlier sections, eliminating the need for the user of the mPOS application to carry cash and change will just make sales more simpler, easier and more safe to conduct.

5. Application Development

In this chapter you find in detail what tools the team used to make this project happen, the overall setup of the project and the risk that were at hand and how the were solved.

5.1 Coding Environment

When developing the mPOS application our team used Xcode 4.6.2 development environment that supports Objective-C, the de facto language for the iOS platform. Objective-C is a general-purpose, high-level, object oriented language and is used by Apple for the iOS operating system and their respective Application Programming Interface (API) as well as Cocoa Touch, which is a User Interface (UI) framework for building applications that run on the iPhone, iPad and iPod Touch from Apple Inc.

The mPOS application is developed using the Model View Controller (MVC) pattern and by using this pattern we have made future developments of the application simpler, because it is easy to add functions and view with in the current application, without rewriting much of the code.

5.2 Source Control

In the beginning we decided to use git as source control storage and all our code is stored at BitBucket[4] and we choose them since they offer free tools that are private from the public and offers really easy sharing with in the specific group we created for this project.

5.3 Other Tools

We decided to use some online tools to simply sharing within the team and with Handpoint, enabling everybody with interest in the project to have full access through the Internet. Google docs[5] took care of the files, DropBox[6] took care of bigger files and provided storage for icons, pictures and logos, for quick and easy access and finally we used Trello[7] as a virtual scrum board, taking care of the to-do's and other related files.

5.4 Developing methodology

After some discussion we decided to use scrum to keep the project on track. User stories, backlog, sprint planning, planning poker and sprint burndown where the tools we used. Since our team is a two man team, we really didn't need a whole lot more since keeping two people up to date is really not that hard. When writing this final

[4] http:www.bitbucket.org
[5] http://docs.google.com
[6] http://www.dropbox.com
[7] http://www.trello.com

report, hindsight tells us that maybe we should have taken a better look at some other methodology and if we would do this again, we both agree that Kanban might have been more appropriate for a two person team, focusing on helping out the team when problem arouse.

5.5 Scrum team

Product owner: Freyr Ólafsson, Chief Technology Officer, (CTO), Handpoint
Development team: Jón Hilmar og Ragnar Ævar, computer science students at RU.
Stakeholders: Handpoint
Other people of intresset: Erla Ósk Ásgeirsdóttir, Marketing Director,(MD) , Handpoint

5.6 Project setup

As we mentioned in section 5.4 in this report, our team used scrum to analyze and set up the project. We decided to use week long sprints, starting them on Monday and finishing them on Sundays, giving us time to complete the retrospective and edit the burndown list, the backlog and other modules making up the project, so that before the next sprint started, we had a clear idea how the project was unwinding, and all the tools ready to plan the upcoming sprint. This setup with week long sprints was by our opinion good and kept our group focused and gave us a great overview of where we heading and what we completed.

5.7 Project meetings

On Sundays we held the retrospective and then we planned the next sprint. Our team held meetings with the product owner and other staff at Handpoint every Monday, to let them know how the project was progressing and showed of the features that we had implemented to get feedback from stakeholder.

5.8 Risk analysis

We have to admit that our first risk analysis was far from perfect and was lacking some aspects. But thanks to the first brief meeting with the supervisors at RU, this was pointed out to us in a professional way and that made us go back to this part of the project and reconsider and rewrite the analysis.

Risk level is on scale 1-5 (1 being the most unlikely to happen, and 5 most likely)
Risk effect is on scale 1-5 (1 having the least effect, and 5 most effect)

5.8.1 Risk analysis list

Risk: Failure in receiving an Apple developer account to test on devices.
Risk Level: 1
Risk Effect: 4

Handling: Check if Handpoint can take care of that, which they did.
Solution Date: 04.02.2013

Risk: Handpoint's iOS SDK not functioning properly.
Risk Level: 3
Risk Effect: 4
Handling: Reserve time to analyse the problem. Ask Hanpoint to address the problem. Worst case: Implement a testing stub.
Solution Date: Handpoint solved this during the projected, but with minor changes in code keep working just fine.

Risk: Testing devices not available at the time when needed.
Risk Level: 3
Risk Effect: 2
Handling: Repeat the importance of access to testing devices to Handpoint. Focus on other problems while waiting, if that is an option. Worst case: Find a way to simulate the device with a stub.
Solution Date: When starting to deploy our applications there were some sharing going on but Handpoint handed out devices for us late in April, dedicated to the project.

Risk: Some parts of our code do not satisfy Handpoint's security standards
Risk Level: 4
Risk Effect: 5
Handling: Get some directions from Handpoint staff on how to improve these problems.
Solution Date: Ongoing process, but it turned out the SDK solved this and we have never had access to the SDK source code and could not make any potential hazards to the backend that handles all the critical transaction data

Risk: Access to the Handpoint workspace will be limited outside office hours.
Risk Level: 2
Risk Effect: 1
Handling: Schedule other work sessions at other location, like RU or at home.
Solution Date: 11.02.2012

Risk: Team member gets sick for some number of days.
Risk Level: 3
Risk Effect: 1
Handling: Other member updates sick team member about conclusions of meetings in his absence. Team evaluates the need to reorganize the current sprint if necessary
Solution Date: Our team stayed healthy since we are both brought up on fish and spent a lot of time outdoors as kids

Risk: Failure in the source control storage servers of BitBucket
Risk Level: 1
Risk Effect: 1
Handling: Local copies of code will always be stored on 2-3 developing computers
Solution Date: We keep local copies at all times during the project

Risk: Failure in data storage (hard drive) in a team member computer.
Risk Level: 3
Risk Effect: 1
Handling: Copies are available at source control server and other development computers.
Solution Date: Source Control worked like a charm. Thanks to BitBucket for providing this.

Risk: Complete failure of a team members development computer.
Risk Level: 2
Risk Effect: 1
Handling: Extra development computer located at Handoint's office can be used until team members has a working computer.
Solution Date: Everything still working as of today and kept working until we finished work on the project 17.05.2013

Risk: Bluetooth connection between phone and card reader does not work.
Risk Level: 2
Risk Effect: 2
Handling: Getting help with the config, from Handpoint card reader developer.
Solution Date: 08.04.2013

Risk: Providers of online tools shut down or have other technical difficulties.
Risk Level: 1
Risk Effect: 2
Handling: Keep local copies of our work and take regular backup of important files.
Solution Date: Google is doing just fine as well as GitHub.

Risk: Other courses in HR and their workload is not complete now in advance making time planning difficult.
Risk Level: 3
Risk Effect: 3
Handling: Might have to alternate sprints or other work related to this project
Solution Date: This did not materialize, the team refactored time when needed sleeping a little less.

Risk: Hardware used for testing fails

Risk Level: 3
Risk Effect: 2
Handling: If this happens there will be hours lost to the project, how many hours depends on how quick a replacement hardware is provided.
Solution Date: This is ongoing until the final presentation, but as of today 16.05.2013 everything has been working just fine

Risk: Team members knowledge of objective-c insufficient.
Risk Level: 3
Risk Effect: 3
Handling: Assist each other with problems that might come up and in some cases get outside assistance if we can’t find a solution.
Solution Date: Over knowledge and skills in Objective-C and iOS got better during the project and we solved problems together.

5.9 How we handled the project risks

The main risks that we were faced with when working on the project could be divided into two parts the library that was fresh out of development at Handpoint and the hardware that runs the application using the library. Our teams main concerns about the library was that it would be difficult to understand and that our knowledge of Objective-C would be insufficient to make it work with the application and the other concern was what if there are flaws in the library that makes it impossible for us finish the application. Our team decided that if the library would be in our way, we would have to use the service stub pattern and create some dummy data to keep the project on track.

It turned out that our concerns about the library where not that much of a problem, it worked like a charm and although Handpoint updated the library during our development, we only had to make minor changes to the applications to make it work with this new version. The hardware risk that we faced, using bluetooth to connect to the card reader itself and a iPhone to test the application on a device all worked out just fine, of course we had some minor concerns that took our team some time to figure out, but after we figured out the symbiosis of the iPhone, the library and the card reader we were in good shape as well as our project as a whole.

Objective-C that we both knew only as another programming language and is used for Apple iOS development, was like talking to aliens at first, but being an object-oriented language (OOP) we very on the roll after doing some basic tutorials, like “Hello World”. Another thing about iOS development is that there are a lot of frameworks that helps out the developer implementing different features in applications, but we had little idea about them when we started the project.

6. System Design

Our iOS solution is all about the MVC pattern, focusing on code reusability and separation of concerns. Since our project is the first iOS prototype for Handpoints mPOS solutions, using MVC makes future development of the application easier for developers since adding new views, data or adding other ideas is simpler using this pattern, eliminating the time is takes to rewrite large section of the application.

Storing data is all done locally on the iPhone and since this is application is a prototype our team decided to use two different frameworks for storing data, SQLite[8] and CoreData[9]. SQLite stores the transaction data and CoreData stores the data the user can edit. The reason our team decided to use two different frameworks for storing data is that since we had never developed anything for the iOS platform before we wanted to get experience of both frameworks and also to give future developers an idea of how the two frameworks are set up, how they work within the application and what both are capable of achieving.

Our solutions uses other frameworks as well, that are related to other features in the application and or not directly part of the core function of conducting a payment transaction, like e.g. Social Framework for posting data to Facebook and Twitter.

[8] http://http://www.sqlite.org/

[9] http://developer.apple.com/library/mac/#documentation/cocoa/Conceptual/CoreData/cdProgrammingGuide.html

7. Progress

To help us plan the project and how to develop the mPOS application, we used as mentioned in section 5.4, scrum. Using scrum we had an overview of the progress we were making during the development phase. As mentioned in section 5.6 the project was divided into sprints that were a week long and there were 15 sprints all in all. To measure the progress we used burndown charts to see how we were progressing and how many story points our team was burning during each sprint. The velocity in the first sprint whereas we expected low in the beginning but as our understanding and skills in Objective-C and the Xcode environment evolved, the velocity increased and we made significant progress during the last sprints of the project, as the picture below shows.

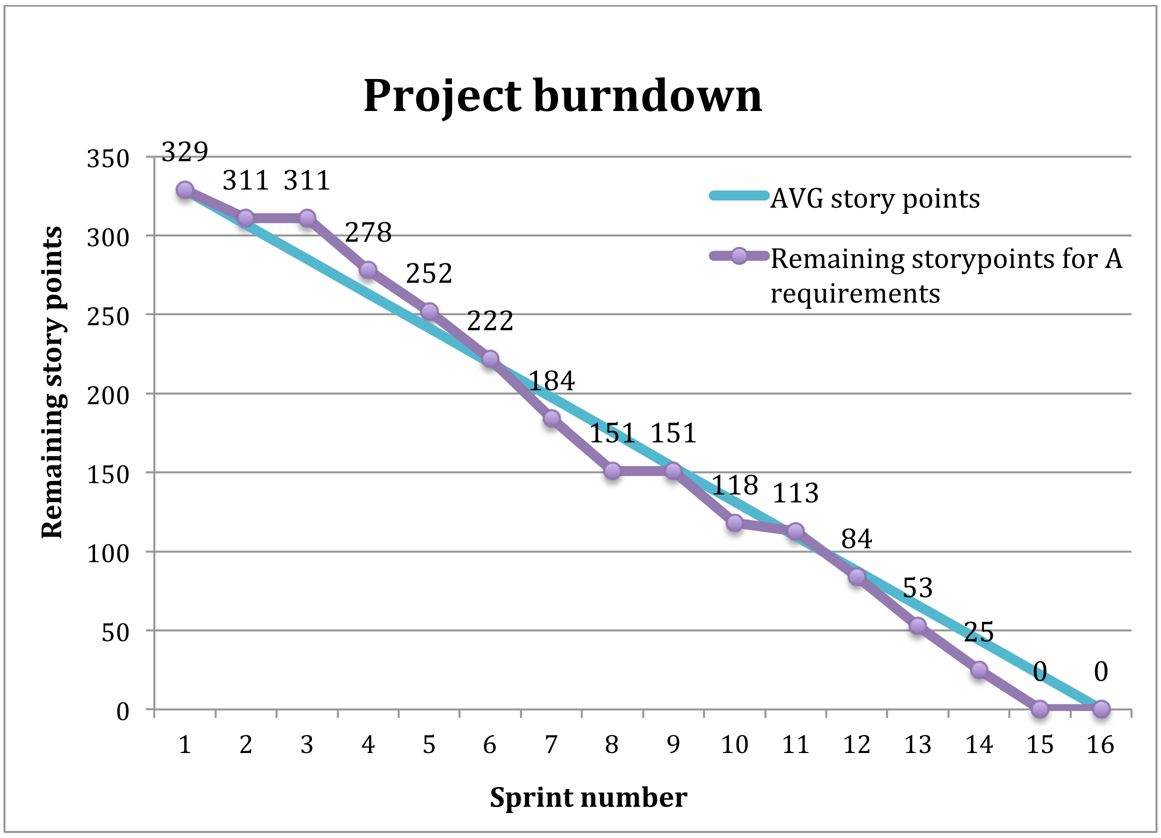

Figure 3 - Project burndown chart

In the above mentioned picture you also notice that during sprint 6 and 8 our team velocity was almost flat and the reason for this is that during this sprint we rewrote some of the code from earlier sprints, using our new knowledge about the language and environment to improve our applications, making the code more readable, clearer and easier to understand from the future developers perspective.

7.1 Time frame

In the beginning of the project we estimated 800 hours to complete the tasks and have a running application. To keep track of team hours we used a Google document, where team members put in the hours for working on the project. When we started working on the project our team spent a lot of time reading and experimenting, by doing for example the classical Hello World tutorial and we are aware that some hours spent on the project are unaccounted for since our team had application development for iOS as our main interest during this time, trying out various things.

7.2 Project total time

On time of writing this final report, our team have spent 860 hours in total on the project. As mentioned there are other hours that are unaccounted for, but we enjoyed our experience and work on the mPOS application so much, that we are just going to let them fly in the early spring wind. The 860 hours are divided between team member as follows; Jón Hilmar 450 hours and Ragnar 410 hours. As you notice there is a small gap between members, but we are pleased with each others contributions to the project.

8. The deliverable of the mPOS application

The artifacts of the work described in this report is twofold. First we have an iOS prototype that uses the SDK developed by Handpoint to complete a payment transaction from start to finish on the iOS environment. On completing our prototype Handpoint will have a iPhone client, that they know work with their iOS SDK. If Handpoint chooses to develop the mPOS by their own or co-branding with other developers, they can be sure that crucial feature of conducting payments with the card reader works.

Second, our application has some features to show the potential of the application. Smartphones of today have the power and hardware onboard to really accomplish great thing and our application just barely scratches the surface in that regard. The possibilities are really endless, but of course when dealing with money transaction there can be laws and regulations that need to be fulfilled, regulating how transaction are handled and what must be in place before the mPOS solution can be regarded as a complete bookkeeping tool for businesses.

For more detailed information about the mPOS application please refer to the attached development and user guides for the mPOS application. They can also be found on the included CD.

9. Testing

The mPOS application has been tested with the MPED-400 card reader. The applications has successfully completed multiple test transactions with special test cards and test payment servers. In addition to numerous functionality tests we ran some live tests and diagnostics using the Instruments tools[10] in Xcode. The Instruments tools check for memory leaks, memory and CPU usage and other application activities. The output from those tests and diagnostics were surprisingly positive.

10

http://developer.apple.com/library/mac/#documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/Introduction/Introduction.html

10. Future work

Handpoint now have a mPOS prototype that runs on the iOS platform, fulfilling Handpoints needs and a complement to their Android solution, the two biggest smartphone platforms as of today, according to a recent report from Gartner Inc[11]. There is a great demand from potential users of the mPOS solution from Handpoint running on iOS, according to staff at Handpoint.

The fact that the solution is build around the iPhone also means that it runs on other iOS devices like the iPad and the iPod touch. We tested our application on these devices and it worked fine, but mPOS needs some refinements for iPad due to screen size, and making use of the screen to have access to more features in one view, so there are all kinds of other possibilities that developers can work with while designing an mPOS solution for the iPad. In the future application specialized for some particular work sector is also possible with features for that particular job, the possibilities are endless.

Also future development of the mPOS application would by our opinion, have some kind of integration with iCloud, the service that helps users synchronize data across devices as well as storing data and possible integration with other web services that assist potential users e.g. dedicated databases for inventory or other web based services that a company might benefit in using with the mPOS solution, instead of other market solutions.

[11] http://www.gartner.com/newsroom/id/2482816

11. Conclusion

Our team is really proud of how this project unveiled. We had a lot of fun during the project, that help us keep spirits up and learned a great deal about application development for the iOS platform. The two of us really started out with nothing more than a project description, but thought will, determination and good organization we managed to create a working prototype that we are happy with, although we sometimes felt like lost tourist on a glacier, but together we managed to find our way back to civilization, without calling in the cavalry.

We would like thank all the people at Handpoint that we have encountered during our work with special acknowledge to our product owner Freyr and Erlu Ósk in marketing at Handpoint. The fact that our iOS mPOS solution will be used for demonstration purposes and selling point for potential Handpoint customers is just great and shows out team that our work as real value for Handpoint which was the main purpose of the project, creating a application that uses the library and runs on the iOS platform.

12. Review from Handpoint

We've had the pleasure of having two fellows from RU here with us this year. They proved to be not only good looking but also smart and good to work with. They have been very independent in their work, asking the required questions to the correct people. Otherwise more or less self managed. Output is very promising.

- Freyr Ólafsson, CTO(Chief technology officer)

Reykjavík 17. maí 2013

Jón Hilmar Gústafsson
171183-2949

Ragnar Ævar Jóhannsson
210175-3349