Handpoint – Point of sale

Final Report

Spring 2015

Bjarni K. Árnason

Tómas P. Sævarsson

B.Sc. Computer Science

School of Computer Science

Instructor: Haukur Kristinsson

Examiner: Elín Torfadóttir

T-404-LOKA

School of Computer Science

Table of Contents

Abstract

This report describes the project "Handpoint Point of Sale" (HiPOS) which is an application for the company Handpoint. This project is a B.Sc final project at the Department of Computer Science at Reykjavik University.

The Handpoint Point of Sale (HiPOS) is a Windows desktop application that will make it possible for merchants to communicate with USB and Bluetooth card readers from Handpoint. This connection allows merchants to accept debit/credit cards as payment for their goods. In brief, the merchants enter the amount for the product/service they are selling and send it wirelessly via Bluetooth technology or through a USB cable to the card reader. The customer then inserts his card and enters the PIN number and the transaction takes place. The application creates receipts that can be printed out or sent by email as attachments. Furthermore, there is a function that allows for making refund, sale-reversal and refund-reversal transactions. It is also possible to look up all transactions and print them out again or send them to customers by email afterwards.

The report also includes descriptions of the design and architecture that we used throughout the project, including how the project was planned and organized between the team members.

We utilized the methodology Scrum to manage the project and the artifacts from that, which we used, was user stories, product backlog, sprints and sprint burndown charts to follow the process of the project.

We will discuss how the project progress was throughout the project and how we did in each sprint.

Furthermore we will discuss how we handled the risk of the project, what testing we did and what the future holds for the application.

Introduction

About Handpoint Point of Sale

The project we worked on is called Handpoint Point of Sale, Handpoint POS or HiPOS. In this report we will use HiPOS when referencing the project.

HiPOS is a desktop client application that runs on the Windows 8.1 operating system. We worked on the project in cooperation with Handpoint, which is an Icelandic Mobile POS platform provider. HiPOS is a sales interface for receiving card payments using payment solutions from Handpoint.

HiPOS communicates through Bluetooth and USB technology with card readers from Handpoint and makes it possible for merchants to accept debit/credit cards as payment for their goods. HiPOS uses the Handpoint APIs and SDKs to connect and that gives HiPOS the opportunity to access all the features that the card reader has to offer.

The reason why Handpoint wanted this solution is that they had an old JAVA client which was difficult to maintain. Therefore they had a need for a newer and better one that was not difficult to support and was preferably more user-friendly. The old client also doesn't have the functionality to send email which was one of the highest priorities that Handpoint set for the new client. Furthermore, Handpoint had just released a C# version of their SDK and wanted to know how outside developers would handle that version.

Handpoint

About the company

Handpoint is a Mobile POS platform provider which enables developers to add highly secure EMV payments to their applications. Handpoint currently employs 20 people and has offices in Iceland, Cambridge UK and Palo Alto California. With 13 years experience in the payment market, the founders pioneered major innovations in the mobile payments market, including developing Mobile Point-Of-Sale (MPOS) for the airline industry in 2003 and the world's first Chip and PIN mPOS solution for a handheld device at Manchester United FC's stadium in 2007.

Handpoint today was set up in 2009 as a pure payments company focused on MPOS and has developed a comprehensive platform for developers, enabling them to easily add secure, pre-certified Chip and PIN payments to their app. The Handpoint solution was the world's first Mobile POS application to be PCI-P2PE certified (About, 2015).

The vision, mission, value proposition and competitive positioning of Handpoint is as follows:

Vision

Mobile POS will disrupt Retail and Payments and Handpoint will be a key payments provider enabling that change.

Mission

Handpoint's mission is to be the number one choice for Mobile POS developers in Europe and North America.

Value Proposition

Handpoint provides the most secure Payment service for integrated Mobile POS, which is efficient and easy to deploy.

Competitive Positioning

Handpoint enables developers to be the first to market with the most secure Mobile POS solutions suitable for merchants of all sizes.

Methodologies and tools

This chapter contains information regarding our use of the Scrum methodology, the working environment, team work setup, an agreement between team members, how we planned the project, and development environment information.

Scrum

The team decided to use Scrum to manage the project. Scrum is an iterative and incremental methodology for managing product development and has a few artifacts that we employed.

We used user stories, product backlog, sprints and sprint burndown charts to follow the process of the project. Each story was broken down into tasks and each task estimated separately. We used Scrum Wall to follow each story and task. Each sprint lasted two weeks, from Monday to Sunday. On Sundays we had a team meeting where we planned the week ahead and also talked about retrospective and sprint planning when sprints were finished.

Scrum team

Scrum role	Name
Product owner	Jón Hilmar Gústafsson, software developer, Handpoint
Scrum master/Project manager	Tómas Sævarsson
Development team	Bjarni, Tómas
Stakeholders	Handpoint

We also had meetings on Wednesdays with our instructor and with two weeks interval a status meeting with the product owner. Because the product owner was in-house at Handpoint, he was available for us almost every day if we had any questions. We also contacted other staff members if any help was needed.

Working environment

Team work setup

The team was located at Handpoint headquarters at Hamraborg in Kópavogur. Handpoint set up facilities with computer monitors, a blackboard, a scrum wall and other items that we used. One of the team got a security card with permission to enter the office outside of office hours, which gave us time to work on weekends as well.

We used Google Drive to store all our files and gave our instructor access. Every hour that we worked on the project was registered in a Google sheet file. There we registered the date, when we arrived at Handpoint, when we left the office, how many hours we worked, activities, task numbers and a description of the work we had done.

Agreement between team members

The team decided to divide responsibilities and the division can be seen in the following table. It should be noted that we started this project as a team of three people but unfortunately one team member had to withdraw before the project finished.

Role	Description	Member
Architect	The main objectives for the Architect was to design the environment of the project. This included setting up drawings that described the structure of the system and how everything should work during development time.	Bjarni K. Árnason
Product owner	The main objective for the Product owner was to ensure that the requirements of the project were complied with.	Jón H. Gústafsson
Project manager	The main objective for the Project manager was to ensure that the project was on schedule, to keep track on product backlog, sprints backlogs and to manage meetings and communications with stakeholders	Tómas P. Sævarsson
Programmers	Programmed and designed the look of the project	Bjarni, Tómas
Testers	Tested the project	Bjarni, Tómas

Time planning

The first weeks were all about planning, starting up the project, getting our development environment up and running and writing user stories. In January the team had a meeting with the product owner and wrote all user stories. After that meeting, team members sat down and tasked the user stories. Afterwards we realized that most of the planning hours weren't registered in sprints so there is a gap between sprint-registered hours and actual registered hours (See *Project progress*). Handpoint had estimated that the project would take between 800 and 1000 hours.

Development environment

In developing the system we used Visual Studio 2013 (.NET v4 since that is the version the HAPI SDK supports) using the C# version of the WPF (Windows Presentation Foundation) framework. A SQLite relational database was used for the local data storage. All code was hosted on a private account on GitHub, but will later on be made publicly accessible through Handpoint's GitHub account. Also, we used Git as a source control system connected to the private repository.

Development rules

We decided to set some development rules before we started.

General rules

1. Code must be in English
2. Comments on code must be in English
3. Comments must be written above relevant code and not at the end of a line or below it
4. Comments must start with a capitalized letter

C#

5. Use four spaces for indentation
6. Use one line space between classes, functions and attributes definitions
7. Use space between keywords and expression for example;

```
if (true) … // Good

if(true) … // Bad

foreach (var item in list) … // Good

foreach(var item in list) … // Bad
```

8. Use PascalCasing for classes, functions and class attributes
9. Use camelCasing for local variables and parameters
10. Open braces in the line below a sentence and close below the last line, for example:

```
public ActionResult Index()
{
    return View();
}
```

11. Use implicit typing for local variables when tag is given by their assignments, for example:

```
var var1 = "This is clearly a string.";
var var2 = 27;
var var3 = Convert.ToInt32(Console.ReadLine());
```

Design and development

In this chapter we will discuss the design of the application. The main objectives of the application were to make it easily maintainable, loosely coupled and testable. The user interface should be user-friendly and follow modern design themes for Windows applications.

Software Architecture

To make our code as maintainable and loosely coupled as possible, we structured our code following the MVVM (Model, View, ViewModel) pattern. Furthermore, for support in using this pattern, we used the MVVM Light Toolkit which is a NuGet package for WPF. The MVVM Light Toolkit writes a lot of boilerplate code to help us bind our application together as well as helping in making data flow between different points in the application. To help us in implementing a modern Windows feel for the application, we used a UI toolkit called MahApps.Metro which gave us access to a lot of styled controls that fitted our idea for the design.

A layering strategy was put in place allowing us to separate our presentation logic from the business logic. In the business logic, a service layer was created. This service layer contains a couple of services that we can use in our presentation logic. Each service has an interface and then an implementation of that interface. We created three services that each take care of a certain task; HapiService wraps the HAPI SDK for easier access and maintainability, MailService contains the mail sending logic and methods, and TransactionsDBContext is a service that implements DbContext using the "Unit of work" pattern which gives us easy access to our database models. We used Entity Framework 6 (ORM) to map our model objects to the database which required us to install a couple of dependencies to support the SQLite database we used. HapiService and MailService use the Singleton pattern with dependency injection so we only have one instance of those services running and inject the needed services into our viewmodels so that they can be used in the presentation layer.

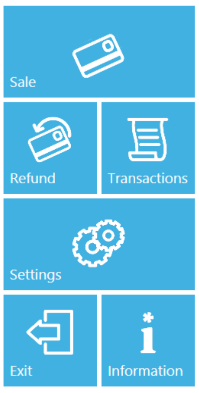
Following the MVVM pattern our viewmodels contain the presentation logic, communicate to the services and create public objects that our views can use to display data. Each view is bound to a specific viewmodel, i.e. SaleView is bound to SaleViewModel and can only access data through that specific viewmodel, and following guidelines for this pattern the viewmodels know nothing about the views. The views then decide how to display the data from the viewmodels and implement the layout of the application window. For people used to web development, a view in our context is similar to a HTML file that contains HTML code as well as CSS code, except our views are written using the XAML language that WPF uses.

We needed a way to make data flow easily through the application and for that we used a messaging system built into the MVVM Light Toolkit. With this system we are able to send messages containing data from our services to viewmodels that register to that specific message and can in turn display the data in our views. The messaging system was both used to send data between classes and to send simple notification messages that would, for example, update the list of transactions or navigate between views if something special happens.

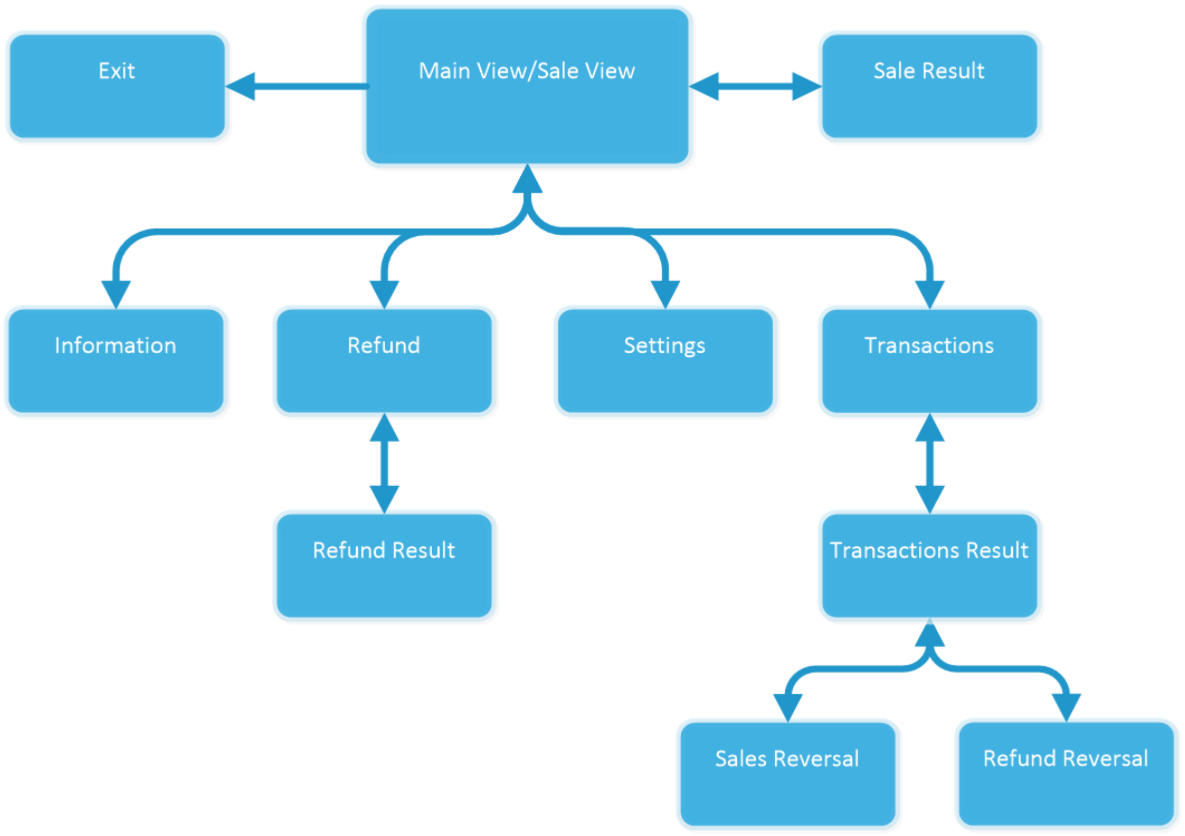
System design

Before we started to program the application, we created a design report which is one of the attachments of this final project. More information can be accessed in that report.

Navigation Diagram

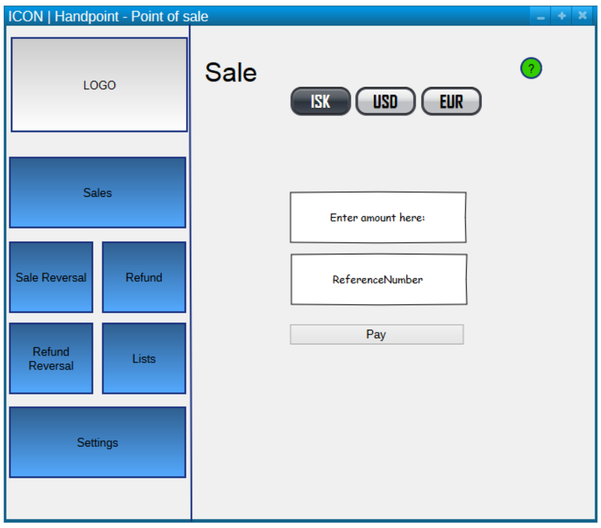


In the picture below, you can see the main connections between views in the program. On the main view, which is also the Sales view, there are icon tiles for each view. From the main view you can click on ‘Refund’, ‘Transactions’, ‘Settings’ and ‘Information’ tiles and from all of them you can go back to the main view. From the ‘Transactions’ list you can double click on a line in the grid view and, depending on the type of transaction, you can either do a sales reversal or a refund reversal. When you click on the ‘Exit’ tile the application shuts down. On the left is a picture of the navigation bar as it is in the HiPOS application.



Gui Mockups

To help us design the program, we used the program Evolus Pencil which is an open-source GUI prototyping tool. With that we created mockups for most of the views we programmed in the project. In the picture on the right is an example of how the mockup of the Sale view was designed.



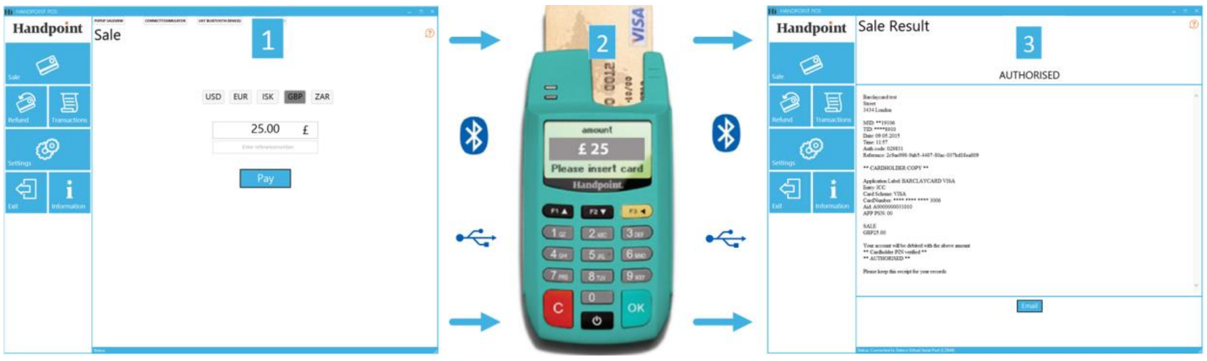
Application Features

The main application features are as follows;

Sale	**Sale transactions** Here users can start a sale
Refund	**Refund transactions** Here users can start a refund
Transactions	**Transactions list** Here users can access all transactions and print and email receipts. It is also possible to make a Sale Reversal and a Refund Reversal
Settings	**Settings** Here users can select different settings. Settings can be changed for printing, email, default currency, devices etc.
Information	**Information** Here users can see information about the application and how to contact Handpoint.
Exit	**Exit** Here users can exit the application

More information regarding each transaction and how to change settings can be found in a User guide that was written along the way.

In the picture below there is an example of a sale transaction.



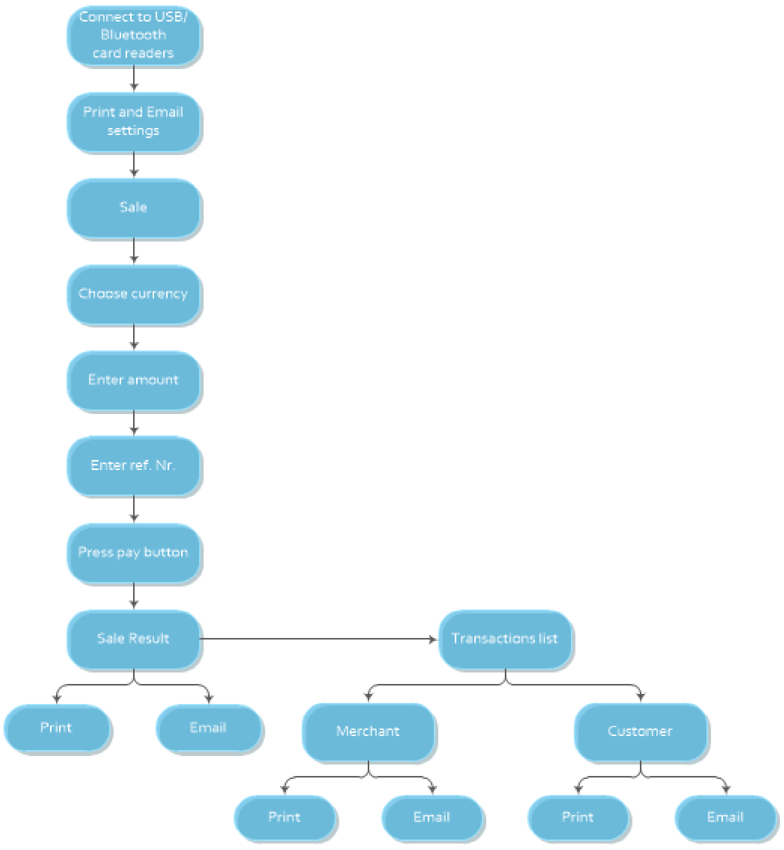
```
Barclaycard test
Street
3434 London

MID: **19106
TID: *****8900
Date: 09.05.2015
Time: 11:57
Auth code: 029831
Reference: 2c9ae998-9ab5-4407-80ac-037bd18ea809

** CARDHOLDER COPY **

Application Label: BARCLAYCARD VISA
Entry: ICC
Card Scheme: VISA
CardNumber: **** **** **** 3006
Aid: A0000000031010
APP PSN: 00

SALE
GBP25.00

Your account will be debited with the above amount
** Cardholder PIN verified **
** AUTHORISED **

Please keep this receipt for your records
```

The first step is to enter the amount in the HiPOS client. Then HiPOS connects through Bluetooth or USB cable to a card reader and asks for card and PIN number. The merchant hands the card reader to the customer, who enters his PIN number. When the customer has successfully entered the code, the card reader, via encrypted messages, communicates with a payment gateway and then a receipt is created in the HiPOS client which the merchant can print out or send via email to the customer.

Project progress

As we mentioned before, we used Scrum to plan and have an overview of the progress of the project. To begin with, the goal of the project was to implement Sale, Refund, Sale Reversal and Refund Reversal with features like printing, email, transactions list etc. However, at a status meeting in March, Handpoint decided to change that.

The process on the right shows the point of success from Handpoint's point of view.



The definition of success for Handpoint was that it would be possible to make a sale transaction and to follow that through the whole process. That means, in short, that the application could connect to the card reader, start a sale, communicate with the card reader, print a receipt and send a receipt as an attachment in an email.

Product backlog

In the product backlog we wrote all the user stories regarding the project. We used priority to decide which stories we had to implement first and then we used importance on stories with priority A to decide in which order we would work on the tasks.

We created 57 stories which we divided into A, B, C and D priority. The stories were prioritized as follows;

Priority	Count
A	32
B	15
C	6
D	4

User stories with priority A were the target for the project at first but like stated above Handpoint changed that during the project.

In the beginning it was estimated that the project would take around 800-1000 hours to complete and when this report is written we have registered 863 hours as can be seen in the table below.

	Activities					
			Time registered			
Nr.	Activity	Status	Tómas	xxxxx	Bjarni	SUM
1	Planning	Finished	41	8	8	56
2	Work setup/Work plan	Finished	111	41	46	198
3	Software installation	Finished	4	3	8	15
4	Meetings	Finished	8	6	6	19
5	Programming	Finished	81	56	254	391
6	Testing	Finished	4	0	0	4
7	Final report	Finished	116	0	12	128
8	Presentation	In process	0	0	0	0
9	Research	Finished	18	12	23	52
	SUM		382	126	355	
	TOTAL		863			

The workload each month throughout the semester can been seen in the table on the right.

Month	Total hours	% of total
Jan	120	13,9%
Feb	181	21,0%
Mar	194	22,5%
Apr	115	13,3%
May	253	29,3%
Sum	863	100%
Avg.	173	

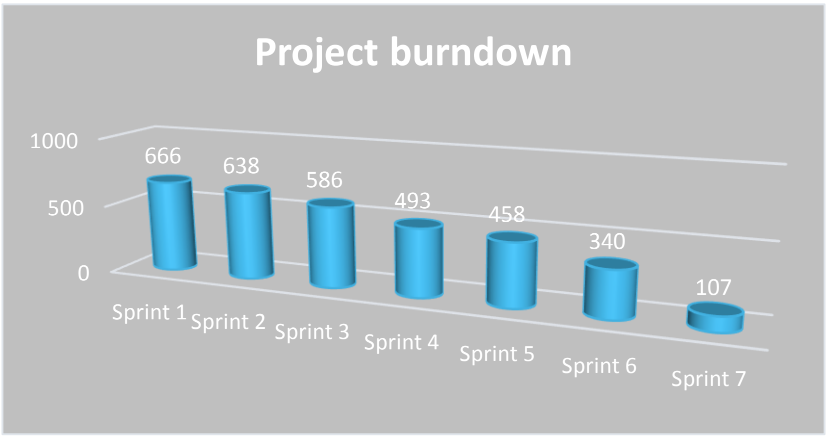
Tasks

We created 140 tasks and estimated they would require around 750 hours. In the latter stages of the project we estimated that with testing, fixing, final report and presentation the project should take about 1.080 hours.

After final exams in April we sat down and went over all user stories and all tasks just to make sure that we were not forgetting anything and also to clarify how many hours we needed to work before deadline. After that meeting we were able to close some user stories because tasks within them had already been concluded while working on other things. We were also able to lower our estimates on many tasks because when we planned them in the beginning, we really did not know what we were getting into. For example, we had one user story with seven tasks and we had estimated that would require around 44 hours but because we were more familiar with the development environment and the tools we were working with we could lower estimates there to seven hours.

Sprints

We planned seven sprints with 750 hours estimated in total. Like stated before the reason behind the gap between registered hours in sprints (643 hours) and actual hours (863 hours) is all the planning time and the reports written at the beginning. None of that was put into sprints which afterwards proved to be a big mistake. When we figured that out, however, we always had tasks delegated for report writing and planning.



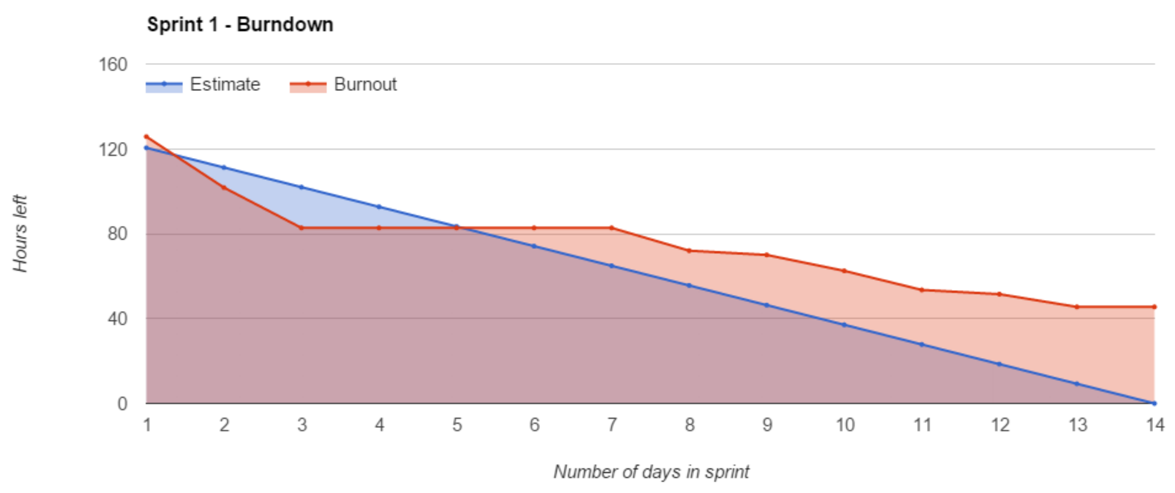
In the table below are overview of all the sprints.

Sprint	Started	Days	Estimated hours	Registered hours	Difference
Sprint 1	26.1.2015	14	130	84	46
Sprint 2	9.2.2015	14	115	28	87
Sprint 3	23.2.2015	14	68	52	16
Sprint 4	9.3.2015	14	112	93	19
Sprint 5	23.3.2015	14	61	35	26
Sprint 6	20.4.2015	14	118	118	0
Sprint 7	4.5.2015	14	146	233	-87
Sum		**98**	**750**	**643**	**107**

Sprint 1

GOAL: To start HIPOS project ✓

For Sprint 1 we estimated 130 hours but registered only 84. The difference there is due to planning which was not registered in the sprint.

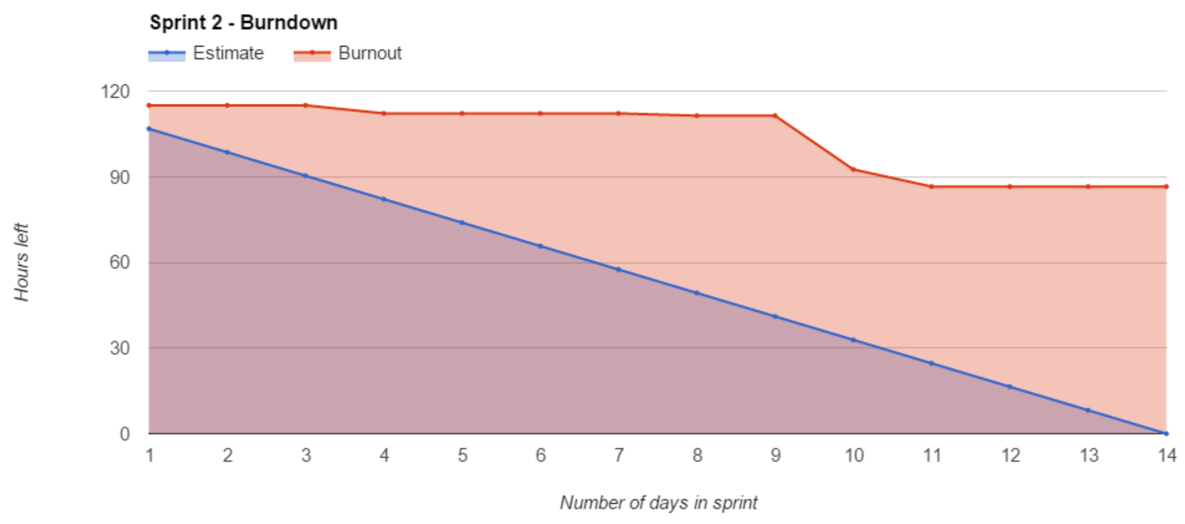


In this sprint we worked on setting up the source control and structure of the project, created Gui mockups of main views and set up the initial appearance of the application.

Sprint 2

GOAL: To implement Sale transaction

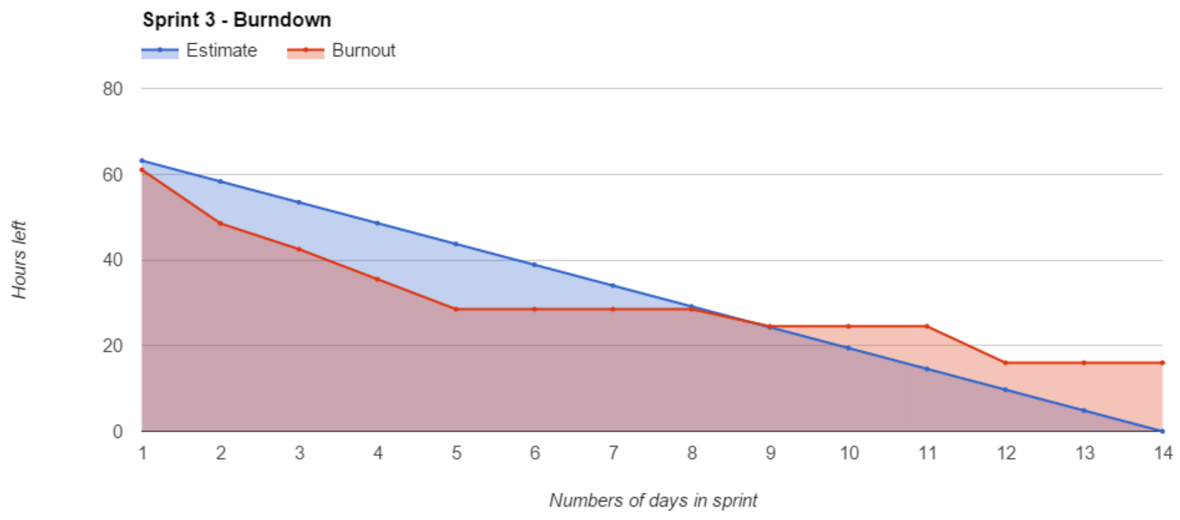
Our estimation for Sprint 2 was 115 hours but we registered only 28 hours. This was because of all the planning, reporting and meetings we did at that time. Also, at the time we had some other school projects that required our attention.



Sprint 3

GOAL: To implement Sale transaction✓

For Sprint 3 we estimated 68 hours of work and registered 52 hours. We reached our goal which was to implement a sale transaction.

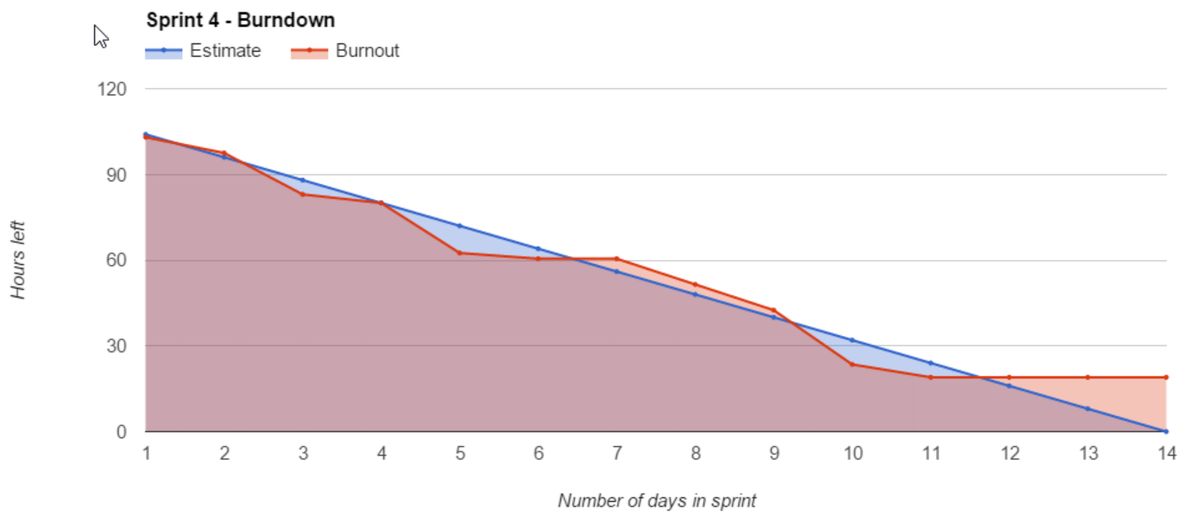


This was the best sprint so far and we reached some milestones. This was the sprint where the development environment clicked and after that everything went more smoothly .

Sprint 4

GOAL: To implement transaction list✓

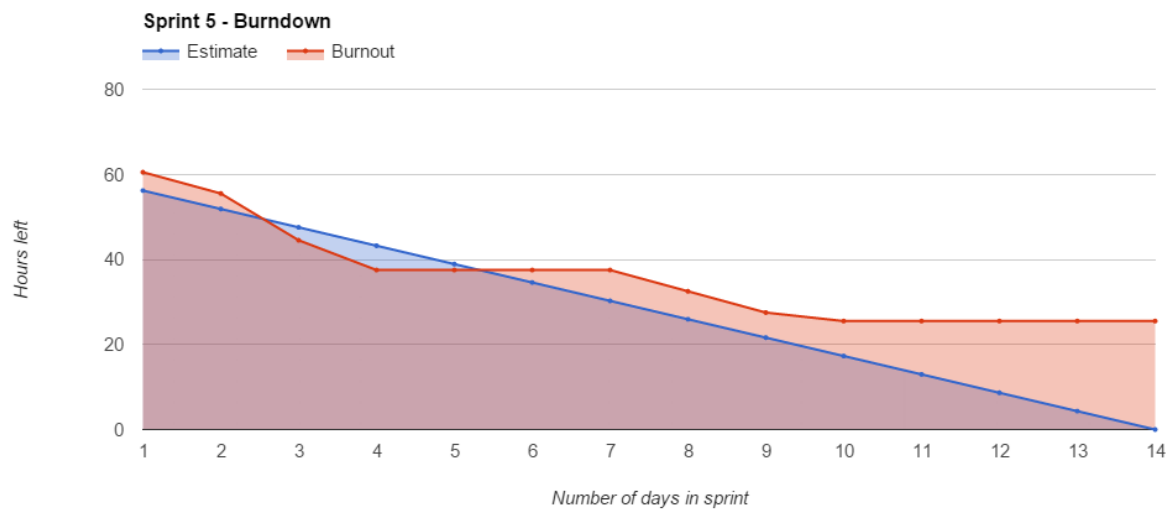
Our estimation for this sprint was 112 hours and in the end we had registered 93 hours. This was one of the best sprints since the project got off the ground and was very productive. Many nice features were finished during this sprint.



Sprint 5

GOAL: To implement cancel function and remove popups and use views instead✓

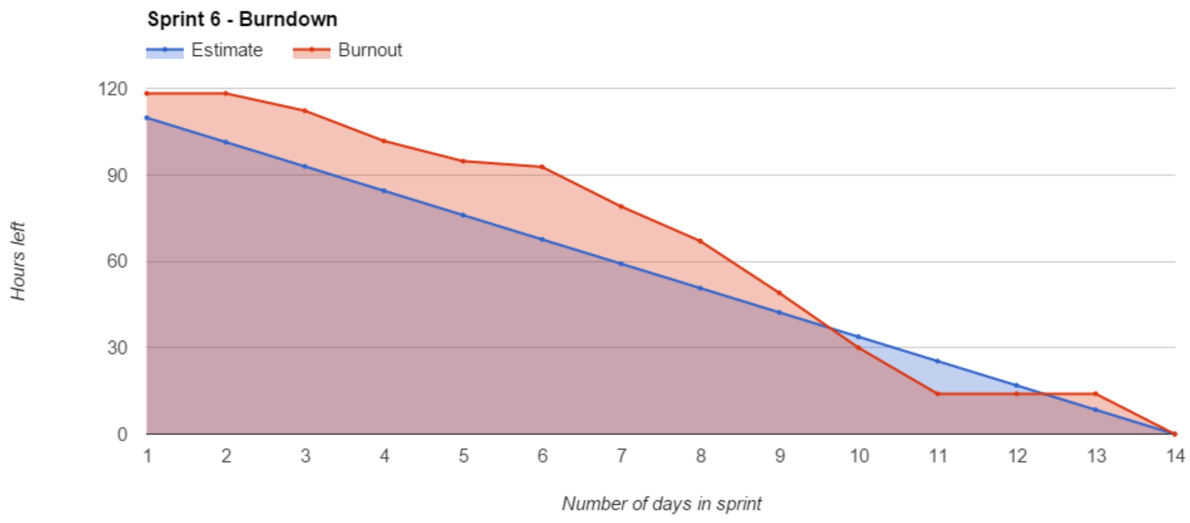
For this sprint we estimated 61 hours of work but registered only 35. The reasons for the few hours registered here were final exams and other school projects.



Sprint 6

GOAL: To implement printing, email and settings view✓

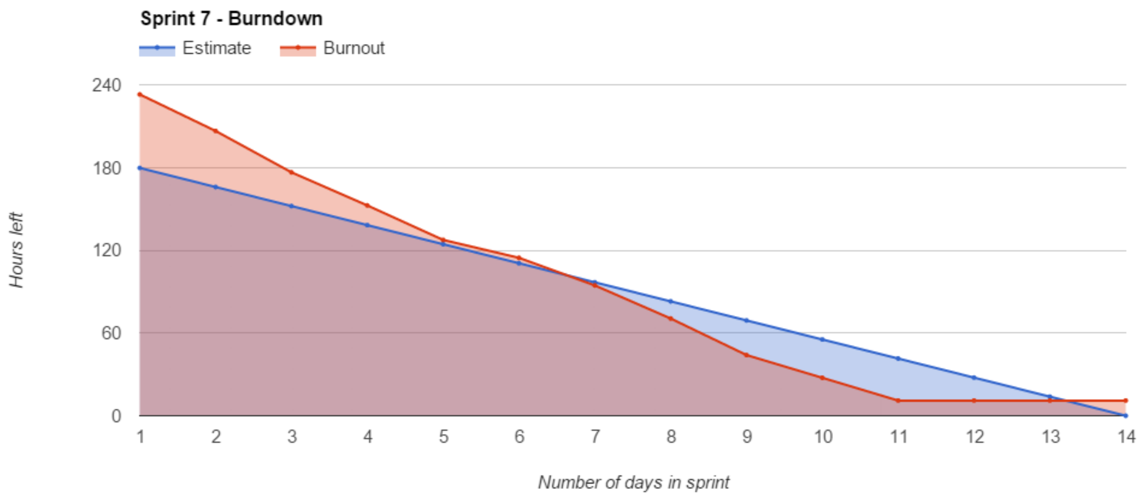
Because of final exams we did not start this sprint until 20 April. We estimated to work for 118 hours and we registered exactly that amount of time. In this sprint we implemented printing and email features and finished the settings window.



Sprint 7

GOAL: To finish HiPOS project✔

This was the final sprint. We foresaw a workload of 148 hours and when this report is written we have registered 233 hours. This sprint was all about finishing the features Handpoint wanted, fixing minor things, writing the final report, user guide and operation manual, and preparing the presentation.



Risks

When we started this project, one of the first tasks was to set up a risk analysis. A risk analysis should address what could possibly go wrong, what the likelihood is of it happening, how it will impact the project and what can be done about it.

We used a scale from 1-5 to set our risk likelihood, 1 being most unlikely to happen and 5 most likely. We used the same scale for the impact, where 1 predicted the least impact and 5 the most impact. Then, to calculate the risk, we multiplied these numbers together. More information regarding risk factors can be found in the risk analysis table.

Risk Matrix

We designed a risk matrix to make the risk visual and in the table on the right you can see the matrix. Six risk factors were low-risk, five medium-risk and one high-risk.

🟩 Low 🟨 Med 🟥 Hi		Impact				
		1	2	3	4	5
Likelihood	1	🟩	🟩	🟩	🟨	🟨
	2	🟩 5,10	🟩 3,4	🟨	🟨	🟨
	3	🟩 9,11	🟨 6,7,8	🟨 1,2	🟨	🟥 12
	4	🟨	🟨	🟨	🟥	🟥
	5	🟨	🟨	🟥	🟥	🟥

Risk Analysis Table

Risk factor nr.	Risk factor	Like-lihood	Impact	Risk	Risk mgmt approach	Early warning signs	Solution date
1	Team members knowledge of WPF is not good enough resulting in delay in programming	3	3	9	Team member will try to assist each other with problems and if that is not enough we will seek outside help	When working on something we haven't worked on before	24.4.2015
2	Other courses and their workload is not known by now causing difficulty in planning resulting in fewer burn down hours	3	3	9	Possible that sprints has to be changed	When projects in other courses are handed out we can adjust our plans	24.4.2015
3	Connection between card reader and computer through Bluetooth not working	2	2	4	Get Handpoint to help on a solution	No early signs	25.2.2015
4	Connection between card reader and computer through USB not working	2	2	4	Get Handpoint to help on a solution	No early signs	22.3.2015
5	If Handpoint decides to limit our visits outside office hours	2	1	2	Plan to use HR, home or other facilities	We will know this next week and can adjust our plans according to their decision	4.3.2015
6	Testing devices like tablets not available when needed	3	2	6	If that happens, test other equipment instead but ask Handpoint to get to us the devices we need	When starting to program features that you need to test with Handpoint devices	30.4.2015
7	Testing devices like printers not available when needed	3	2	6	If that happens, test other equipment instead but ask Handpoint to get to us the devices we need	When starting to program features that you need to test with Handpoint devices	14.3.2015
8	Testing devices like card readers not available when needed	3	2	6	If that happens, test other equipment instead but ask Handpoint to get to us the devices we need	When starting to program features that you need to test with Handpoint devices	20.2.2015
9	Sickness in the team	3	1	3	It is possible that sprints have to be adjusted due to this. Team evaluates the need if this comes up.	When team member is feeling ill	4.5.2015
10	Team member computer crashes	2	1	2	Handpoint will have backup computers available.	No early signs	4.5.2015
11	Team member hard drive crashes	3	1	3	Since we are using source control our code should always be accessible	Blue screen flashes	4.5.2015
12	SDK from Handpoint not supporting Win 8.1	3	1	3	Ask Handpoint to fix it	When starting to implement features	

In the end, we were able to eliminate every risk factor except risk factor 12. When starting to implement the feature Start/Stop monitoring connection against the USB card driver, we began to notice high CPU problems. When researching this problem, we found out that the Start/MonitoringConnections service from Handpoint did not support the Windows 8.1 operating system so this feature was put on hold.

We should have had a risk factor from the start which would assess the problem if some of the team members were to quit before we finished, because in the end this had quite an impact on our project.

Testing

In this chapter we will discuss the tests that we carried out. We had planned to carry out four different types of testing, Think-aloud, UI, Unit and system testing, but the loss of one team member changed that. In the end we could only manage to do the system testing but an employee of Handpoint will start testing the application as soon as we hand in our code.

While testing HiPOS, we used MPED-400 card readers and two different POS printers, Epson TM-T88V and Star TSP 100. HiPOS has successfully completed multiple test transactions with special test cards and test payment servers.

System testing

The following table shows the test description that we carried out after the program was finished. This test was performed to verify that all transactions would process like Handpoint had described. The system was tested 12 and 13 May 2015 and one error was found relating to the testing of 5.1. We fixed that and when testing the system again no error was found.

Nr.	Test description	Correct conclusion	Error/Comment
1.1	Open the program and connect to card reader. If connection method not chosen, go to 'Settings', choose 'USB', and press 'Search'. When the application finds device chose one from list and press 'Connect'.	Program is connected with the card reader with a status at the bottom confirming connection	ok
2.1	Make a sale transaction by choosing currency, enter amount and press 'Pay' button	Message dialog with information regarding the transaction opens. When transaction is finished, sale result view with customer receipt opens and you see an email button. Both merchant and customer receipt should have been printed	ok
2.2	Go to settings view, change customer default printing to 'Off' and do a 'Sale' transaction (2.1). When you get a question whether you want to print receipt, press 'Yes'.	Message dialog with information regarding the transaction opens. When transactions is finished, message dialog opens with question if you want to print receipt. After pressing 'Yes', view with customer receipt open and you see an email	ok

		button. Merchant and customer receipt should have been printed.	
2.3	Go to settings view, change merchant default printing to ‘Off’ and do a ‘Sale’ transaction (2.1). When you get a question whether you want to print receipt or not, press ‘Yes’.	Message dialog with information regarding the transaction opens. When transaction has finished, message dialog opens with question if you want to print receipt. After pressing ‘Yes’, sale result view with customer receipt open and you see an email button. Customer receipt should have been printed.	ok
3.1	Make a ’Refund’ transaction by choosing currency, enter amount and press refund button. When you get a question whether you want to print receipt or not, press ‘Yes’.	Message dialog with information regarding the transaction opens. When transaction has finished, message dialog opens with question if you want to print receipt. After pressing ‘Yes’, view with customer receipt open and you see an email button. Customer receipt should have been printed.	ok
3.2	Go to settings view, change customer and merchant default printing to ‘On’ and do a refund transaction (3.1).	Message dialog with information regarding the transaction opens. When transaction has finished, refund result view with customer receipt opens and you see an email button. Merchant and customer receipts should have been printed.	ok
4.1	Go to transactions and find the transaction that you did in 2.1. Double click on the line.	Transaction Result view with merchant and customer receipts opens. Under each receipt there are print and email buttons. At the bottom is a sale reversal button.	ok
4.2	While in the same state that 4.1 ended in, choose ‘Print’ button under ‘Merchant receipt’.	Merchant receipts should have been printed.	ok
4.3	While in the same state that 4.2 ended in, choose ‘Print’ button under ‘Customer receipt’	Customer receipts should have been printed.	ok
4.4	While in same state that 4.3 ended in, choose ‘Email’ button under ‘Merchant receipt’. Enter email address and press ‘OK’.	Message dialog opens with a box to enter email address. After pressing ‘OK’, new message dialog opens with information that email is being sent. Merchant receipt should have been sent via email.	ok
4.5	While in same state that 4.4 ended in, choose ‘Email’ button under ‘Customer receipt’. Enter email address and press ‘OK’.	Message dialog opens with a box to enter email address. After pressing ‘OK’, new message dialog opens with information that email is being sent. Customer receipt should have been sent via email.	ok
4.6	While in same state that 4.5 ended in, press ‘Sale Reversal’ button. Message dialog opens with a question if you want to make a reversal transaction. Press 'Yes' Button.	Message dialog with a question opens. Press ' Yes' button and a message dialog with information regarding the transaction opens. When transaction has finished, result view with customer receipt opens and you see an email button.	ok
4.7	Go to ‘Transactions’ and find the transaction that you did in 3.1. Double click on the line.	Transaction result view with merchant and customer receipts opens. Under each receipt there are	ok

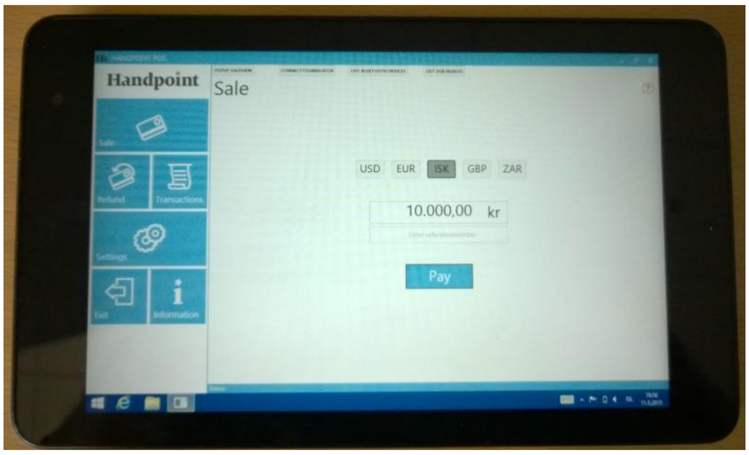
		print and email buttons. At the bottom is a refund reversal button.	
4.8	While in same state that 4.7 ended in, press ‘Refund Reversal’ button. Message dialog opens with a question if you want to make a reversal transaction. Press 'Yes' Button.	Message dialog with a question opens. Press ' Yes' button and a message dialog with information regarding the transaction opens. When transaction has finished, result view with customer receipt opens and you see an email button. Merchant and customer receipts should have been printed.	ok
4.9	Go to transactions and find the Sale reversal you did in 4.6. Press 'Not Available' button. Press 'Ok' button.	New line in the transactions list, with the type ‘Sale void should be there. You see an 'Not Available' button. After double click a message dialog opens stating that reversal is not possible on this transaction.	ok
4.10	Go to transactions and find the Refund reversal you did in 4.6. Press 'Not Available' button. Press 'Ok' button.	New line in the transactions list, with the type Refund void should be there. You see an 'Not Available' button. After double click a message dialog opens stating that reversal is not possible on this transaction.	ok
5.1	Go to settings. Change ‘Shared secret key’. Make a sale transaction (2.1)	On card reader the message “Not processed” is shown; in application nothing happens	ok
5.2	Go to settings. Change default currency to ISK.	Default currency in Sale and Refund view should be set to ‘ISK’ button.	ok
5.3	Go to settings. Change default currency to USD	Default currency in Sale and Refund view should be set to ‘USD’ button.	ok
5.4	Go to settings view. Set email server to smtp.google. Go to transactions and double click on a transaction. Press ‘Email’ button, enter email address and press ‘OK’.	Message dialog with information that mail was not sent should be shown.	ok
5.5	Go to settings. Change email server to smtp.gmail.com. Set email subject to “demo” and email body to “This is a demo body”. Go to transactions and double click on a transaction. Press ‘Email’ button, enter email address and press ‘OK. Check your email.	Your inbox should have an email with “demo” and the date in the subject, the receipt as an attachment and the body should say “This is a demo body”.	ok
6.1	Go to information. Press ‘www.handpoint.com’	Handpoint website Should open in browser	ok
6.2	Go to information. Press ‘support@handpoint.com’	Default email program should open with support@handpoint.com as an email address.	ok
6.4	Press ’Exit’ tile. Press ‘Quit’.	Message dialog opens with a question if you want to quit. The application shuts down.	ok

Future work

Handpoint now have a great application to serve their customers that are using Windows 8.1.

It is our opinion that they can, in the future, extend it with features such as a tablet version of the project, a multi-user platform version and a mobile version as well as implementing more device management features.

Even though it wasn't a top priority, we tried the program on a Dell Venue 8 pro, Windows Tablet with Windows 8.1 operating system and the program works but needs a tweak here and there to make it more tablet-friendly.



HiPOS is also an ideal tool for Handpoint's salespeople to show future customers how connections, payments and other functions work in this market.

Deliverables

The deliverables, to Handpoint, which we are describing here in this report are as follows:

- All source code is in a single solution that is stored in our private repository at GitHub. This repository will be transferred over to Handpoint's repository. Handpoint is the rightful owner of all source code written for this product.
- User guide which outlines every possible functionality the program offers
- Development Guide which outlines how the project is set up and a guide for developers from Handpoint on how to continue programming the project.

Conclusion / Summary

To participate in such a big project gives us experience that we can continue to use in our future jobs. We are really proud of what we have achieved and we learned a lot about the development of an application for Windows OS. This all started as an idea from Handpoint but will in the end be used by Handpoint's staff and customers.

We would like to thank all the people at Handpoint for trusting us with this project and extend special thanks to the product owner, Jón Hilmar, who helped us continuously in getting the job done.

We would also like to thank our families for their support. Without them this wouldn’t be possible. Furthermore, we thank our instructor, Haukur, for always believing in us.

Review from Handpoint

It's safe to say that we at Handpoint could not have been luckier with the team that chose our Windows POS client project. They have produced an excellent application that will prove to be very useful both for our internal development and our clients.

The team have shown themselves to be extremely organised and approached all aspects of the project in a structured and consistent way. In their work they have been very independent and on the few occasions they needed assistance they had good questions and also comments that we will definitely use to improve our online product documentation.

We are also very happy with how well they managed to adapt our ideas to the layout and themes of Windows 8 despite not having any prior experience with that, much like ourselves.

Overall we are more than satisfied with the results of this final project and wish the team all the best in their future endeavours from all the staff at Handpoint.

____

Jón Hilmar Gústafsson

Integration support manager

Reykjavík 15. maí 2015

Bjarni K. Árnason

040986-2789

Tómas P. Sævarsson

181270-3499

Bibliography

About. (n.d). Handpoint mobile POS. Retrieved 5. May 2015 from
https://www.handpoint.com/about-handpoint/