HÁSKÓLINN Í REYKJAVÍK

REYKJAVIK UNIVERSITY

Mobile Mechanic

Final Report

Spring 2013

Karl Jóhann Karlsson

Stefán Gunnarsson

School of Computer Science

Instructor: Árni H. Reynisson
Examiner: Hannes Pétursson

T-404-LOKA
School of Computer Science

Table of Contents

INTRODUCTION

Making repairs in the field has changed little over the decades. We believe that Mobile Mechanic will be a useful and easy-to-use tool, for mechanics and supervisors alike, giving both parties a better overview, organization and cutting their paperwork close to zero.

Mobile Mechanic, facilitates the identification of outstanding cases in terms of scope and time, and allows work to be easily planned ahead, cutting out tedious paperwork. Using this tool will lessen mistakes that can occur in documentation and will help keep track of the mechanics' work progress and time spent on each case. Supervisors can more easily track what cases are being worked on, their status, locating the mechanics at all times and have a better overview of cases that have been delayed. Mobile Mechanic will increase the efficiency of the supervisors as it will cut down the manual input necessary to keep the case reports up to date.

Annata's vision is that tablet computers will enter the workplace in greater quantities than before. As a result, a tablet device application like Mobile Mechanic will prove to be a powerful and valuable tool for users of the company's systems. Mobile Mechanic is one of many applications that Annata plans to integrate into their selection of enterprise solutions where users only need access to limited parts of the system. Mobile Mechanic will be the first application of its kind from Annata.

This report introduces Mobile Mechanic, how it was made and how it works. We will also review the team´s development process, successes and failures. Chapter 1 introduces the contractor, Annata. Chapter 2 introduces the Mobile Mechanic solution, reviews the development process and outcome. Chapter 3 discusses tests that were performed during the development phase. Chapter 4 covers requirement analysis, charts, prototypes and other tools that were used to aid in the process. Chapter 5 covers issues relating to project management practices used to make the project management and development process more efficient. Chapter 6 covers details about the project's development progress using Scrum. Chapter 7 includes a progress overview, a product backlog, sprint burn down charts and team members project diaries. Chapter 8 reviews the facilities provided to the team and chapter 9 explores the future of this application, what needs to be done in addition and when it will be used.

1 ANNATA

Annata is a software company, founded in 2001 and specializes in developing add-ons for Microsoft Dynamics AX. Annata is a Microsoft Gold Partner and one of very few associate software partners in developing specialized solutions for the automobile and machine industry. Annata's main product, Annata IDMS, is a specialized software solution, used all over the world. Annata operates a powerful distribution network, located in more than 30 countries.

The company has a staff of 65 people; of which 35 are based in Iceland. The rest is based in offices in the United Kingdom, Denmark, Sweden, Malaysia and Hong Kong. Certified Associates are about 300. Annata has been returned profit every year from its founding.

2 MOBILE MECHANIC

The Mobile Mechanic application was named by Annata. Their vision was to build an application to aid mechanics in workshops and in the field to connect to their enterprise solutions, reduce paperwork, ease their jobs and generate consistency.

This chapter will review analysis of the required features of the Mobile Mechanic application; what the solution has to offer, architecture and design and the software development environment used. Chapter 6 will cover in detail the design of the development features and prototypes.

2.1 ANALYSIS

Mobile Mechanic is based on Annata's vision to create an application that allows users to connect to back-end corporate systems using tablets that operate in the Windows 8 RT operating system environment.

Annata handed in a software project proposal to the Computer Science Department of Reykjavik University, along with an analysis of their requirements. The University accepted the proposed project as a final project of the T-404 LOKA class that was then chosen by a team of two students. Below is the original requirement analysis:

1. Design an application for a repair service person that runs on Windows 8 or Windows 8 RT.
2. More detailed description.
    a. A service desk handles service requests from clients and records the main information in the back-end system.
    b. A Service desk looks at the workload of the service personnel and comes up with a suggestion about the timing of the requests.
    c. A repair service person can look at requests that are assigned to them and can look at other unassigned requests if they are in the same area. A repair service person can select a request and start working on it.
    d. A repair service person arrives at the designated location and performs risk assessment. Such assessments can vary from country to country. Should the location pass the test, the service person can commence the work.

e. While repair is in progress, the repair service person will record the usage of spare parts and description of the work done. The application will automatically record the time spent on each repair. The service person can use the application to take pictures and videos that will belong to the request.
f. At the end of the repair process, the application will display a summary of the work done, and spare parts that a client representative can sign. If the application is offline then it can add more steps in the finalization of the bill.

The application was to be in English and translatable into other languages.

3. The final solution of Mobile Mechanic needs to be able to run in Windows 8. The UI needs to follow Microsoft guidelines on product design and needs to meet the requirements of the Microsoft App Store.
4. The scale of the project is about 500-1200 work hours. It will be possible to break the project into smaller units, and limit its functionality in the first version, according the size of the project group. It is also possible to break the project into parts divided between both the students and employees of Annata. One group might mostly work on UI features and UI functionality and the other group could work on service integration and changes to the back-end systems.
5. Software environment and technology: The project will be done in Visual Studio 2012 on computers that run on Windows 8. Team foundation server will be used for code storage and version control. If the project group is big enough a part of the project can be done on a back-end system where the software environment would be Microsoft Dynamics AX.
6. Facilities for Students: The students will work on computers provided by Annata. The computers will be setup with all the required software tools. The computers will be connected to Annata´s network through Direct Access and therefore it will be possible to both work at school or at home. Students are also welcome to work at the Annata offices and have access to the specialists located there. The contact person that Annata provides the project group will work closely with the students during the whole progress via email, Lync and meetings.
7. Skill requirements: All students that have diligently followed their studies in HR should be able to work on this project. The most important thing is to be able to learn new things.

Initially, the team had a meeting with the Mobile Mechanic product owner at the offices of Annata. The purpose of the meeting was to get feedback on required functionalities and a deeper understanding of the provided feature analysis. A second meeting took place shortly after the first one where mutual decisions were made regarding the features that were to be implemented in the project since it was clear that the team of two would not be able to implement all the features of the provided feature list from Annata.

It was first decided that the employees of Annata would handle all the back-end work needed for the application along and make sure that all the required class entities were provided.

Subsequently it was decided that the students would not implement part a. and b. in feature number 2.

Another decision was that Annata would provide an IDMS connector, to which the application could connect during the development process.

Then it was decided that no risk assessments would be needed for the final release as described in 2.d.

Additional information gathering took place through conversations with employees of Annata. This led the team to write a product backlog. The requirements were classified as A, B and C requirements where A requirements were the most important, then B requirements and C requirements were classified as the least important. During the solution development process, solution changes were logged in the product backlog after consulting with the product owner.

The entire product backlog is provided in the Mobile Mechanic Progress Overview document.

2.2 THE SOLUTION

Main features:

- The user selects and accepts a case from a list of cases assigned to him in the specified region.
- The user can view all the basic information of the selected case
- The user can track the time spent on each job in a case
- Add more job lists (i.e. tasks) to a case
- Add more spare parts needed for a case from spare part lists
- Finalize a summary of the work done in a case, allowing signing through a canvas field and printing out an bill
- Finish a case
- Place a case on hold
- Edit any comments belonging to the current case
- Take photos and videos that are attached to the currently selected case
- Browse through attachment documents that belong to the current case, view photos and videos files.
- Browse through a list of products and services, see their basic information along with price
- Create a new case, along with customer and device information

2.2 DEVELOPMENT ENVIROMENT

The development environment was chosen by Annata and did not change during the course of the project.

- Visual Studio 2012
- ASP.NET 4.5 for Windows Store application for C# and XAML
- Team foundation server 2012
- Microsoft Direct Access
- Microsoft Lync
- Microsoft Blend for Visual Studio 2012

A little work was done on the project using Microsoft Blend but it was decided that Visual Studio provided a good enough preview of the UI.

2.4 CORE DESIGN

For the development of the Mobile Mechanic application the Model-View-ViewModel or MVVM was used. This pattern is mostly used for Microsoft WPF and Windows 8 RT applications. This is pattern is targeted at UI development platforms that support event driven programming. MVVM was designed to make use of data binding functions in WPF to better facilitate the separation of view layer development from the rest of pattern by removing virtually all the GUI code ("Code-Behind"). Unlike traditional web programming which mostly uses CSS, Windows application programming uses XAML or "eXtensive Application Markup Language", which is an XML based language designed by Microsoft.

We go into more detail about the design in the Mobile Mechanic Developer Guide, though an overview image of the system is given below.

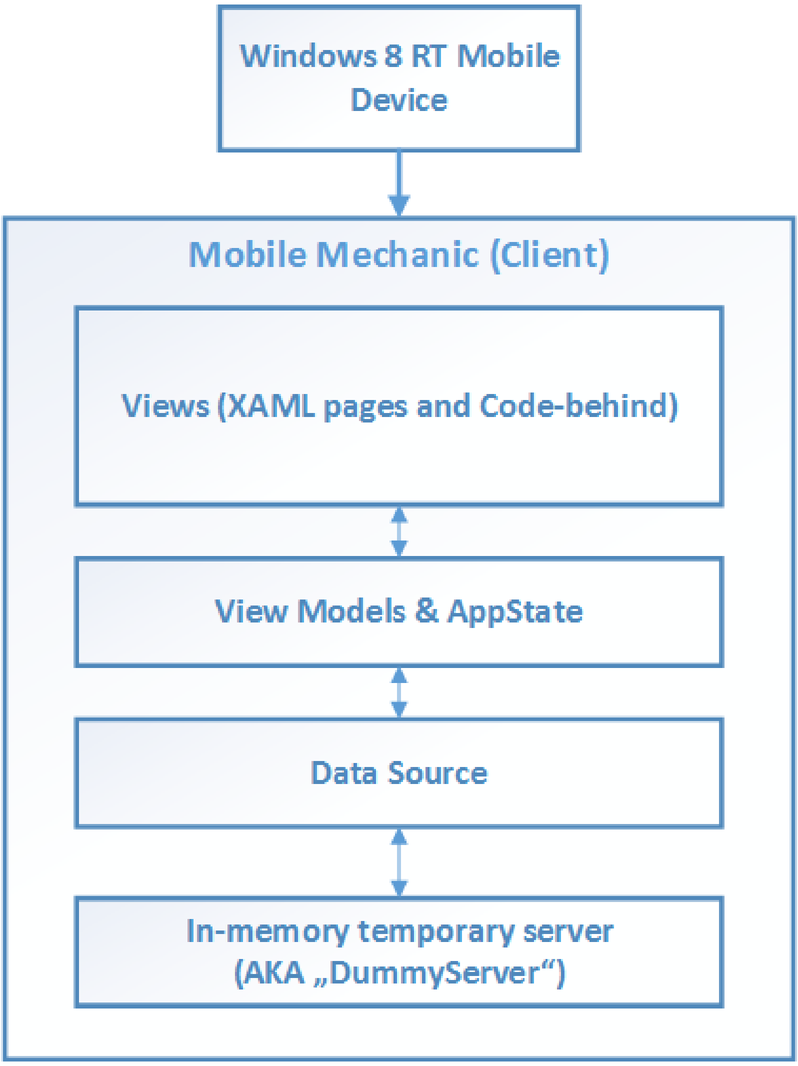

Windows 8 RT Mobile Device
Mobile Mechanic (Client)
Views (XAML pages and Code-behind)
View Models & AppState
Data Source
In-memory temporary server (AKA „DummyServer“)

Figure 1 - Mobile Mechanic System Overview

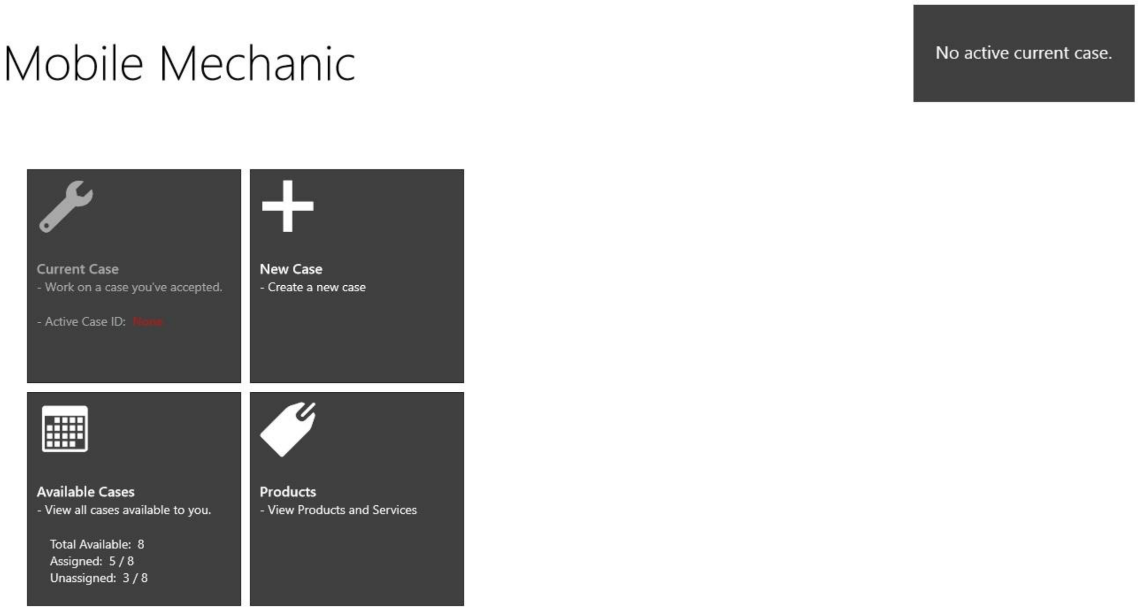
Figure 1 shows the system overview and the inter-dependency of the solution components.

Originally, the Mobile Mechanic application was supposed to connect to a web server named IDMS connector and have its own local database that would be used instead of the web service for when the user doesn't have an internet connection. This, however, did not fit into the time-frame of this project, so it was decided to have the data source communicate with an in-memory, fake server as a temporary solution for this final project. Future versions of the Mobile Mechanic will be connected to the IDMS connector, instead of the data source – this will be discussed in the Future Work section at the end of this report.

2.5 UI DESIGN

After experimenting with different design tools, it was decided to use Microsoft PowerPoint to design the Mobile Mechanic UI. Microsoft PowerPoint has a nice selection of wireframes on which a UI design can easily be based. All the different prototypes created can be found in chapter 4.2. The prototypes did however change a little during the development process, since Visual Studio has a very decent preview tool, where changes and new views or frames can be created without much effort.

Below are a few examples of the Mobile Mechanic screens as seen in Figure 2,3,4 and 5.



Mobile Mechanic

No active current case.

Current Case
- Work on a case you've accepted.

- Active Case ID: None

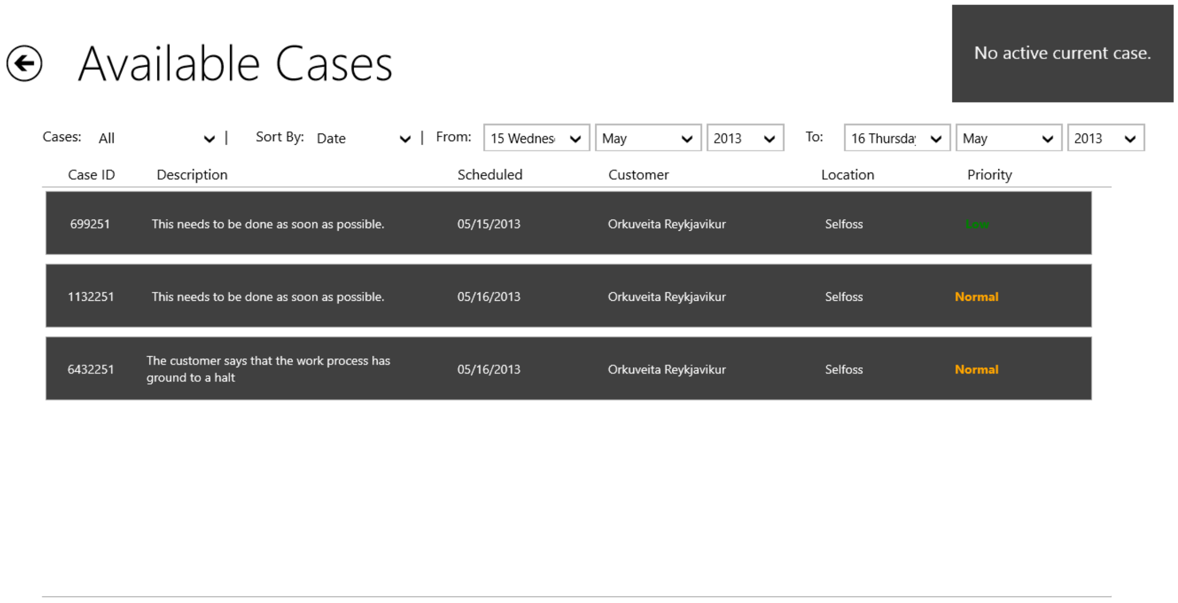
New Case
- Create a new case

Available Cases
- View all cases available to you.

Total Available: 8
Assigned: 5 / 8
Unassigned: 3 / 8

Products
- View Products and Services

Figure 2 -Main Page – The user can access the case list, product list, the "New Case" page or the "Current case" page from this view



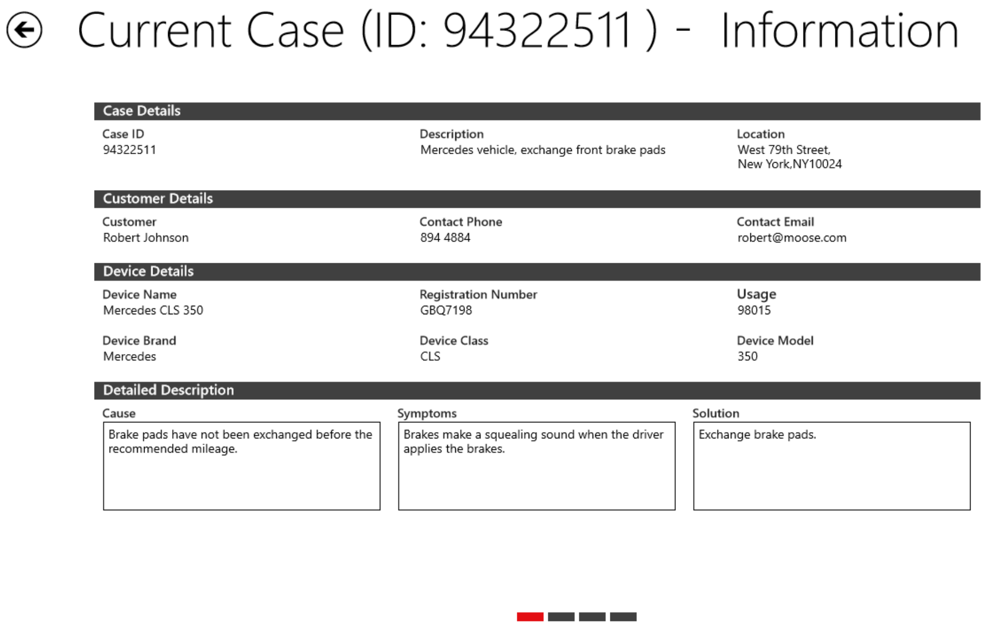
Available Cases

No active current case.

Cases: All | Sort By: Date | From: 15 Wednes May 2013 To: 16 Thursda May 2013

Case ID	Description	Scheduled	Customer	Location	Priority
699251	This needs to be done as soon as possible.	05/15/2013	Orkuveita Reykjavikur	Selfoss	Low
1132251	This needs to be done as soon as possible.	05/16/2013	Orkuveita Reykjavikur	Selfoss	Normal
6432251	The customer says that the work process has ground to a halt	05/16/2013	Orkuveita Reykjavikur	Selfoss	Normal

Figure 3 - Available Case List – The user can select assigned cases here or just create a new one through the app bar.



Current Case (ID: 94322511) - Information

Case Details

Case ID 94322511	Description Mercedes vehicle, exchange front brake pads	Location West 79th Street, New York,NY10024

Customer Details

Customer Robert Johnson	Contact Phone 894 4884	Contact Email robert@moose.com

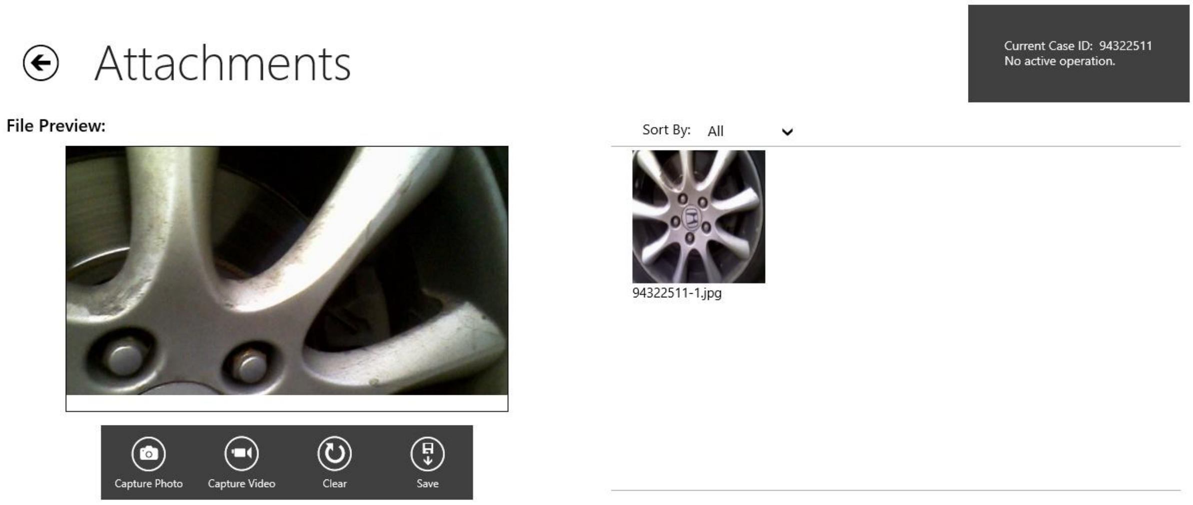
Device Details

Device Name Mercedes CLS 350	Registration Number GBQ7198	Usage 98015
Device Brand Mercedes	Device Class CLS	Device Model 350

Detailed Description

Cause	Symptoms	Solution
Brake pads have not been exchanged before the recommended mileage.	Brakes make a squealing sound when the driver applies the brakes.	Exchange brake pads.

Figure 4 - Case Information – Here the user can see all the basic information about his current case.



Attachments

Current Case ID: 94322511
No active operation.

File Preview:

Sort By: All

94322511-1.jpg

Figure 5 - Attachment View – Here, the user can view the case attachments or create new ones in the form of photos or videos

2.6 FINISHED DESIGN DOCUMENTS

2.6.1 Class Diagram

Below is information about the classes used in Mobile Mechanic. Two kinds of classes, provided by Annata, are mainly used. One of those are the entity classes that are special classes from the back end system of Annata IDMS system. Each contains a specific type of information, for example information about a case. The other kind of classes that are most common in this diagram are the ViewModel classes. Those basically hold information needed by each view, along with having functions and event calls to work with this information through the user interface.

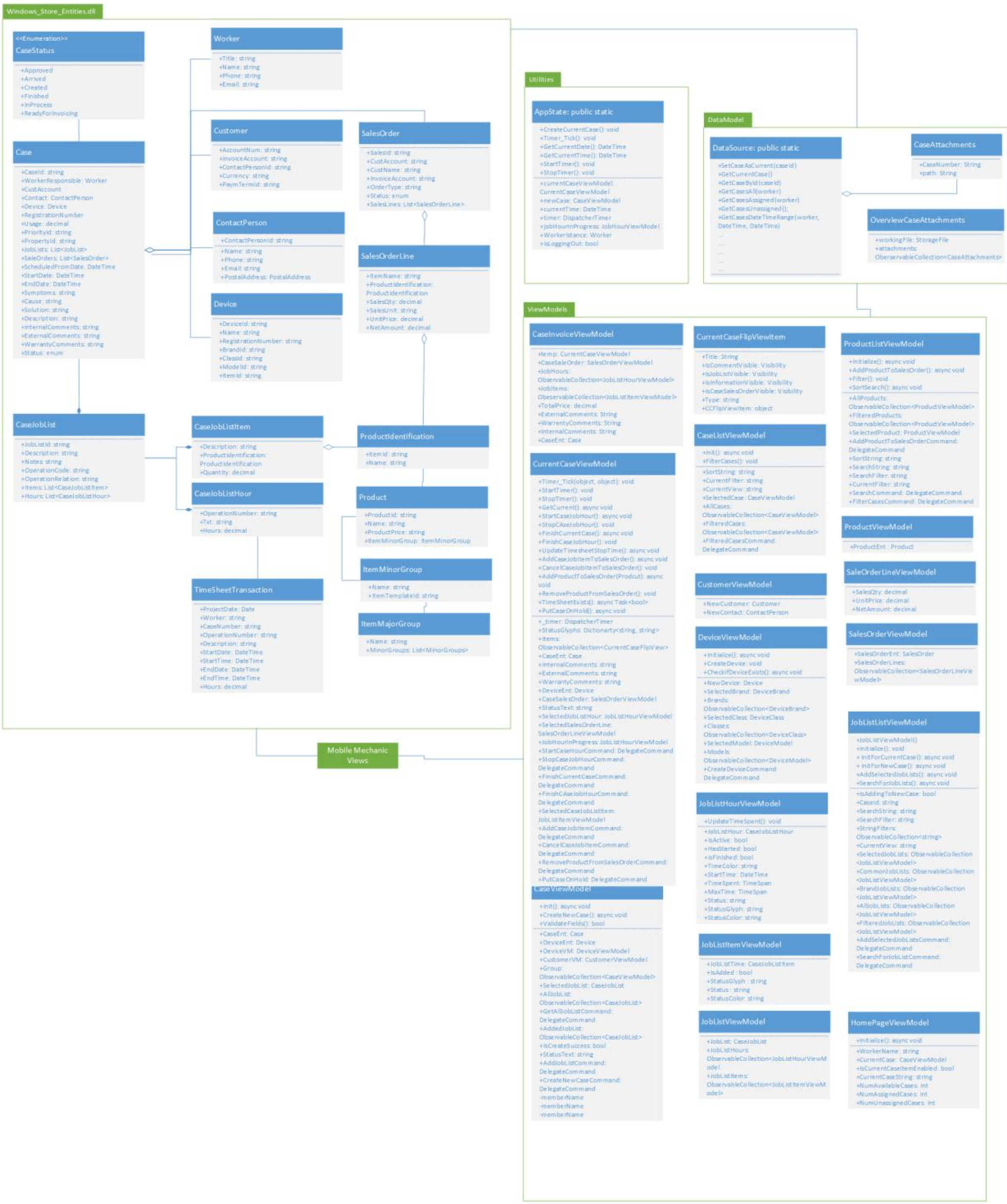

Windows_Store_Entities.dll
<<Enumeration>>
CaseStatus
+Approved
+Arrived
+Created
+Finished
+InProcess
+ReadyForInvoicing
Worker
+Title: string
+Name: string
+Phone: string
+Email: string
Customer
+AccountNum: string
+InvoiceAccount: string
+ContactPersonId: string
+Currency: string
+PaymTermId: string
SalesOrder
+SalesId: string
+CustAccount: string
+CustName: string
+InvoiceAccount: string
+OrderType: string
+Status: enum
+SalesLines: List<SalesOrderLine>
Case
+CaseId: string
+WorkerResponsible: Worker
+CustAccount
+Contact: ContactPerson
+Device: Device
+RegistrationNumber
+Usage: decimal
+PriorityId: string
+PropertyId: string
+JobLists: List<JobList>
+SaleOrders: List<SalesOrder>
+ScheduledFromDate: DateTime
+StartDate: DateTime
+EndDate: DateTime
+Symptoms: string
+Cause: string
+Solution: string
+Description: string
+InternalComments: string
+ExternalComments: string
+WarrantyComments: string
+Status: enum
ContactPerson
+ContactPersonId: string
+Name: string
+Phone: string
+Email: string
+PostalAddress: PostalAddress
SalesOrderLine
+ItemName: string
+ProductIdentification: ProductIdentification
+SalesQty: decimal
+SalesUnit: string
+UnitPrice: decimal
+NetAmount: decimal
Device
+DeviceId: string
+Name: string
+RegistrationNumber: string
+BrandId: string
+ClassId: string
+ModelId: string
+ItemId: string
CaseJobList
+JobListId: string
+Description: string
+Notes: string
+OperationCode: string
+OperationRelation: string
+Items: List<CaseJobListItem>
+Hours: List<CaseJobListHour>
CaseJobListItem
+Description: string
+ProductIdentification: ProductIdentification
+Quantity: decimal
ProductIdentification
+ItemId: string
+Name: string
CaseJobListHour
+OperationNumber: string
+Txt: string
+Hours: decimal
Product
+ProductId: string
+Name: string
+ProductPrice: string
+ItemMinorGroup: ItemMinorGroup
ItemMinorGroup
+Name: string
+ItemTemplateId: string
TimeSheetTransaction
+ProjectDate: Date
+Worker: string
+CaseNumber: string
+OperationNumber: string
+Description: string
+StartDate: DateTime
+StartTime: DateTime
+EndDate: DateTime
+EndTime: DateTime
+Hours: decimal
ItemMajorGroup
+Name: string
+MinorGroups: List<MinorGroups>
Mobile Mechanic Views
Utilities
AppState: public static
+CreateCurrentCase(): void
+Timer_Tick(): void
+GetCurrentDate(): DateTime
+GetCurrentTime(): DateTime
+StartTimer(): void
+StopTimer(): void
+currentCaseViewModel: CurrentCaseViewModel
+newCase: CaseViewModel
+currentTime: DateTime
+timer: DispatcherTimer
+jobHourInProgress: JobHourViewModel
+WorkerIstance: Worker
+isLoggingOut: bool
DataModel
DataSource: public static
+SetCaseAsCurrent(caseId)
+GetCurrentCase()
+GetCaseById(caseId)
+GetCasesAll(worker)
+GetCasesAssigned(worker)
+GetCasesUnassigned();
+GetCasesDateTimeRange(worker, DateTime, DateTime)
CaseAttachments
+CaseNumber: String
+path: String
OverviewCaseAttachments
+workingFile: StorageFile
+attachments: ObervableCollection<CaseAttachments>
ViewModels
CaseInvoiceViewModel
+temp: CurrentCaseViewModel
+CaseSaleOrder: SalesOrderViewModel
+JobHours: ObservableCollection<JobListHourViewModel>
+JobItems: ObeservableCollection<JobListItemViewModel>
+TotalPrice: decimal
+ExternalComments: String
+WarrentyComments: String
+InternalComments: String
+CaseEnt: Case
CurrentCaseFlipViewItem
+Title: String
+IsCommentVisible: Visibility
+IsJobListVisible: Visibility
+IsInformationVisible: Visibility
+IsCaseSalesOrderVisible: Visibility
+Type: string
+CCFlipViewItem: object
ProductListViewModel
+Initialize(): async void
+AddProductToSalesOrder(): async void
+Filter(): void
+SortSearch(): async void
+AllProducts: ObservableCollection<ProductViewModel>
+FilteredProducts: ObservableCollection<ProductViewModel>
+SelectedProduct: ProductViewModel
+AddProductToSalesOrderCommand: DelegateCommand
+SortString: string
+SearchString: string
+SearchFilter: string
+CurrentFilter: string
+SearchCommand: DelegateCommand
+FilterCasesCommand: DelegateCommand
CaseListViewModel
+Init(): async void
+FilterCases(): void
+SortString: string
+CurrentFilter: string
+CurrentView: string
+SelectedCase: CaseViewModel
+AllCases: ObservableCollection<CaseViewModel>
+FilteredCases: ObservableCollection<CaseViewModel>
+FilteredCasesCommand: DelegateCommand
CurrentCaseViewModel
+Timer_Tick(object, object): void
+StartTimer(): void
+StopTimer(): void
+GetCurrent(): async void
+StartCaseJobHour(): async void
+StopCAseJobHour(): void
+FinishCurrentCase(): async void
+FinishCaseJobHour(): void
+UpdateTimesheetStopTime(): async void
+AddCaseJobitemToSalesOrder(): async void
+CancelCaseJobitemToSalesOrder(): void
+AddProductToSalesOrder(Prodcut): async void
+RemoveProductFromSalesOrder(): void
+TimeSheetExists(): async Task<bool>
+PutCaseOnHold(): async void
+_timer: DispatcherTimer
+StatusGlyphs: Dictionarty<string, string>
+Items: ObservableCollection<CurrentCaseFlipView>
+CaseEnt: Case
+InternalComments: string
+ExternalComments: string
+WarrantyComments: string
+DeviceEnt: Device
+CaseSalesOrder: SalesOrderViewModel
+StatusText: string
+SelectedJobListHour: JobListHourViewModel
+SelectedSalesOrderLine: SalesOrderLineViewModel
+JobHourInProgress: JobListHourViewModel
+StartCaseHourCommand: DelegateCommand
+StopCaseJobHourCommand: DelegateCommand
+FinishCurrentCaseCommand: DelegateCommand
+FinishCAseJobHourCommand: DelegateCommand
+SelectedCaseJobListItem: JobListItemViewModel
+AddCaseJobItemCommand: DelegateCommand
+CancelCaseJobItemCommand: DelegateCommand
+RemoveProductFromSalesOrderCommand: DelegateCommand
+PutCaseOnHold: DelegateCommand
ProductViewModel
+ProductEnt : Product
SaleOrderLineViewModel
+SalesQty: decimal
+UnitPrice: decimal
+NetAmount: decimal
CustomerViewModel
+NewCustomer: Customer
+NewContact: ContactPerson
DeviceViewModel
+Initialize(): async void
+CreateDevice: void
+CheckIfDeviceExists(): async void
+NewDevice: Device
+SelectedBrand: DeviceBrand
+Brands: ObservableCollection<DeviceBrand>
+SelectedClass: DeviceClass
+Classes: ObservableCollection<DeviceClass>
+SelectedModel: DeviceModel
+Models: ObservableCollection<DeviceModel>
+CreateDeviceCommand: DelegateCommand
SalesOrderViewModel
+SalesOrderEnt: SalesOrder
+SalesOrderLines: ObservableCollection<SalesOrderLineViewModel>
JobListListViewModel
+JobListViewModel()
+initialize(): void
+ InitForCurrentCase(): async void
+ InitForNewCase(): async void
+AddSelectedJobLists(): async void
+SearchForJobLists(): async void
+IsAddingToNewCase: bool
+CaseId: string
+SearchString: string
+SearchFilter: string
+StringFilters: ObservableCollection<string>
+CurrentView: string
+SelectedJobLists: ObservableCollection <JobListViewModel>
+CommonJobLists: ObservableCollection <JobListViewModel>
+BrandJobLists: ObservableCollection <JobListViewModel>
+AllJobLists: ObservableCollection <JobListViewModel>
+FilteredJobLists: ObservableCollection <JobListViewModel>
+AddSelectedJobListsCommand: DelegateCommand
+SearchForJobListCommand: DelegateCommand
JobListHourViewModel
+UpdateTimeSpent(): void
+JobListHour: CaseJobListHour
+IsActive: bool
+HasStarted: bool
+IsFinished: bool
+TimeColor: string
+StartTime: DateTime
+TimeSpent: TimeSpan
+MaxTime: TimeSpan
+Status: string
+StatusGlyph: string
+StatusColor: string
CaseViewModel
+Init(): async void
+CreateNewCase(): async void
+ValidateFields(): bool
+CaseEnt: Case
+DeviceEnt: Device
+DeviceVM: DeviceViewModel
+CustomerVM: CustomerViewModel
+Group: ObservableCollection<CaseViewModel>
+SelectedJobList: CaseJobList
+AllJobList: ObservableCollection <CaseJobList>
+GetAllJobListCommand: DelegateCommand
+AddedJobList: ObservableCollection <CaseJobList>
+IsCreateSuccess: bool
+StatusText: string
+AddJobListCommand: DelegateCommand
+CreateNewCaseCommand: DelegateCommand
-memberName
-memberName
-memberName
JobListItemViewModel
+JobListTime: CaseJobListItem
+IsAdded : bool
+StatusGlyph : string
+Status : string
+StatusColor: string
JobListViewModel
+JobList: CaseJobList
+JobListHours: ObservableCollection<JobListHourViewModel
+JobListItems: ObservableCollection<JobListItemViewModel>
HomePageViewModel
+Initialize(): async void
+WorkerName: string
+CurrentCase: CaseViewModel
+IsCurrentCaseItemEnabled: bool
+CurrentCaseString: string
+NumAvailableCases: int
+NumAssignedCases: int
+NumUnassignedCases: int

Figure 6 – Class diagram

A larger image of this class diagram as seen in Figure 6 will accompany this report in the final hand in.

3 TESTING

Testing has been a large part of this project in order for the end result to be of the highest possible. For this project, significant amounts of time have been allocated to prepare and research tests. It was decided to implement two types of tests; a Usability test and to a small extent Unit tests. By performing these tests we were able to get feedback on the Mobile Mechanic solution, how it performs with a user and if it performs correctly as a Windows 8 application.

3.1 USABILITY TESTING

In order to perform usability testing, users with different backgrounds and experience were selected. Usability testing took place at Annata's offices at Mörkin 4, Reykjavik. Tasks were handed out for the chosen users to perform on the Mobile Mechanic application. Afterwards, the user's experience was documented, based on an interview.

3.1.1 Users background

Three users were selected, one female and two males. All with good computer know-how, being computer users on a daily basis.

User A is a geologist, studying computer science at Reykjavik University. She is in the age group of 30-35. She has worked as a geologist for several years and currently works at a bank.

User B is a male, system administrator, currently is studying computer science at Reykjavik University. He is in the age group of 20-25. He has years of experience in managing computer systems.

User C is a male student in the age group of 20-25, well versed in using computer software.

3.1.2 The Experiment

The users were given tasks to preform and were supposed to perform these task without complications. The users are marked by A, B and C in the table below. A grade scale is used to determine how quickly the task went for each user as seen in the table.

Scale:

1. Fast - 0 to 30 seconds
2. Medium - 30 to 60 seconds
3. Slow - 60 or more seconds
4. Not able to complete

Task:		A	B	C
1	Log into the Mobile Mechanic and accept any Case	2	1	1
2	Go into the current Case and Change the warranty comment	1	1	1
3	Add a new job list item to the current Case	3	1	2
4	Start a job Operation in the current Case	1	1	1
5	Select attachments in Current Case and take a photo	1	1	1
6	Save the Photo and View it in a bigger screen	1	1	1
7	Add a new spare part(Item) to a job	1	1	1
8	Finish your current case	1	2	1
9	Sign your name, print out an invoice and submit the case	1	1	1
10	Create a new case	1	1	2
11	Add a new device to the case		1	1
12	Add a new contact to the case	1	1	1
13	Add a job list item, fill in any details necessary and submit the case	1	2	1
14	Log out of the system	2	1	1

Table 1

3.1.3 Result

Mobile Mechanic is relatively easy to use, and users with some computer knowledge are quick adapting to it, even with little to know knowledge of how the system works. Our grade scale is 1-4 and no tasks ware completed on the grade 4, so all tasks were completed. All the users had different small delays in their tasks but all were able to complete their tasks. The average of completion was between 1 and 2 that is under average since the average show to be 2 as seen in table 1. That shows us as a results that our system is easy for an average to a professional user to use.

After the test we did an interview to get feedback on what the users thought was missing or might be better in Mobile Mechanic.

Generally our users felt that Mobile Mechanic is a nice and smooth solution. They liked the simplicity of the system and how everything was thought out in a simple way. They did have a few pointers on how we could make our solution better:

- They found that the first screen that had the web service address was a bit confusing, as two users began by typing in the user name in the web service text box when asked to log in.
- Finding out where they could create a new case took them some time, so they suggested placing a fourth button on the main screen to take the user to the "New Case" page. We fixed that problem promptly and did as they suggested.
- Placing the "log out" button in the Windows charm Settings area was confusing – though that is generally what seems to be the norm with Windows 8 apps, as odd as it may be.

3.1.4 Unit Testing

Two simple unit test were done in the beginning of the development of Mobile Mechanic, these were performed through Visual Studio through a separate Unit test project. They were as follows:

- Assert when accepting a Case that the user does not have another case selected, if true then that the case he selected is not null.
- Assert that the authentication returns true by the username typed in when logging into the system and that the password matches with that username, else returns the appropriate error message.

Both these unit tests were done in the beginning of the development process. It was decided that as most of the input from the user was in the form of fetching and displaying data, that extensive unit testing was not needed. When implementing features that got data from the data source, the results expected forehand were known as the team placed created the data as it was needed. If there were any problems, they were primarily UI and binding related.

3.2 UI TESTING

When implementing each UI feature and/or change, it was thoroughly tested with the Microsoft Surface tablet simulator provided by Visual Studio before adding another UI feature to the application. In a sense this worked out well, as designing application for Windows 8 RT application can give instant feedback on the UI and what they will look like in conjunction with other features.

3.3 CONCLUSION

The usability tests gave us good insights was wrong with the application, as Mobile Mechanic is first and foremost an UI application, relaying information from the user to the backend system, the team concluded that this was the most important test that could be done at this stage of the application.

At the end the project, especially during the bug fixing phase the team regretted not having more unit test, as when fixing one bug created another bug in more than one occasion. More unit tests could have made bug fixing easier or even nearly unnecessary. It was a lesson learned and will help the team members in future projects. If anything, we’ve been taught to that Test-Driven Development should be respected – even though it might take a while to get used to and even slow you down, there is a very good reason why it is so highly recommended.

There are a number of other tests that need to be done, relating to the IDMS connector. Lacking the required connection, such tests are impossible. In the future, latency and connection based security tests would need to be performed. Such tests will naturally have to be performed in future iterations in the development of Mobile Mechanic.

4 REQUIREMENT ANALYSIS

4.1 VISION STATEMENT

The process of making repairs in the field has changed little over the decades. For that purpose, Mobile Mechanic will be a useful and easy to use tool both for the mechanics and their supervisors, giving both parties a better overview, organization and cutting their paperwork close to zero.

Using Mobile Mechanic, enables mechanics to easily know what work needs to be done, when it needs to be done and enables easier planning of the work ahead. Additionally, cutting out tedious paperwork from the mechanics, this tool will decrease potential mistakes that can occur in documentation. The tool will furthermore help keep track of mechanics work progress and time spent on each case.

The supervisors can now more easily track what cases are being worked on, their status, locate the mechanics at all times and have a better overview of cases that have been delayed. The Mobile Mechanic will increase the efficiency of the supervisors as it will cut down the manual input necessary to keep the case reports up to date.

4.2 PROTOTYPES

Here on the next page are a two examples of prototypes at the beginning of the project as seen in Figure 7 and 8.

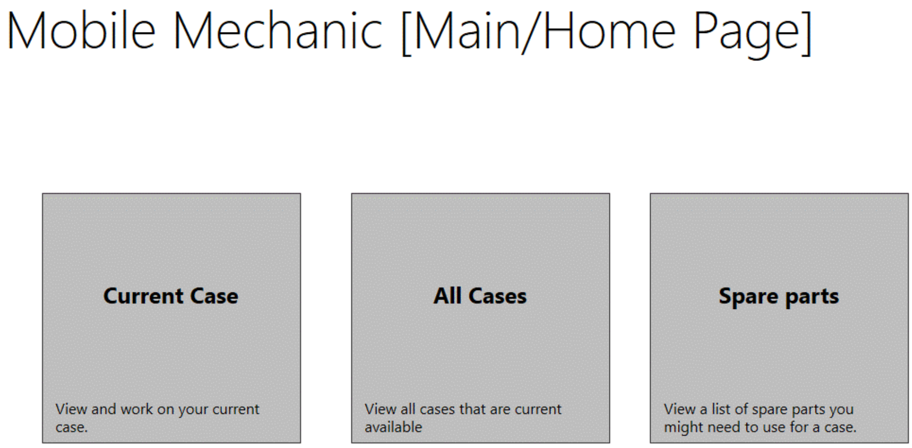
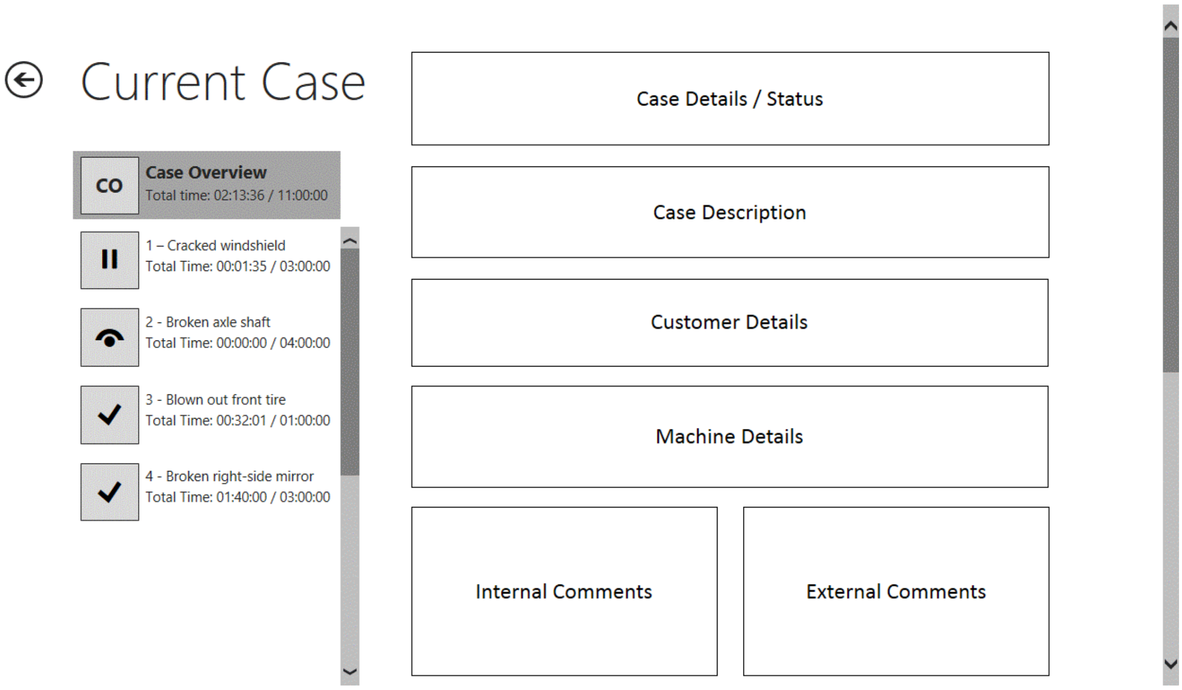

Mobile Mechanic [Main/Home Page]
Current Case
View and work on your current case.
All Cases
View all cases that are current available
Spare parts
View a list of spare parts you might need to use for a case.

Figure 7 – Prototype – Main page



Current Case

CO **Case Overview**
Total time: 02:13:36 / 11:00:00

1 – Cracked windshield
Total Time: 00:01:35 / 03:00:00

2 - Broken axle shaft
Total Time: 00:00:00 / 04:00:00

3 - Blown out front tire
Total Time: 00:32:01 / 01:00:00

4 - Broken right-side mirror
Total Time: 01:40:00 / 03:00:00

Case Details / Status

Case Description

Customer Details

Machine Details

Internal Comments

External Comments

Figure 8 - Prototype - Current Case

4.3 FLOW CHARTS

Two types of flow charts were created: A general flow chart of the Mobile Mechanic and another one for the processes. Here is the first version of the general flow chart as seen in figure 9.

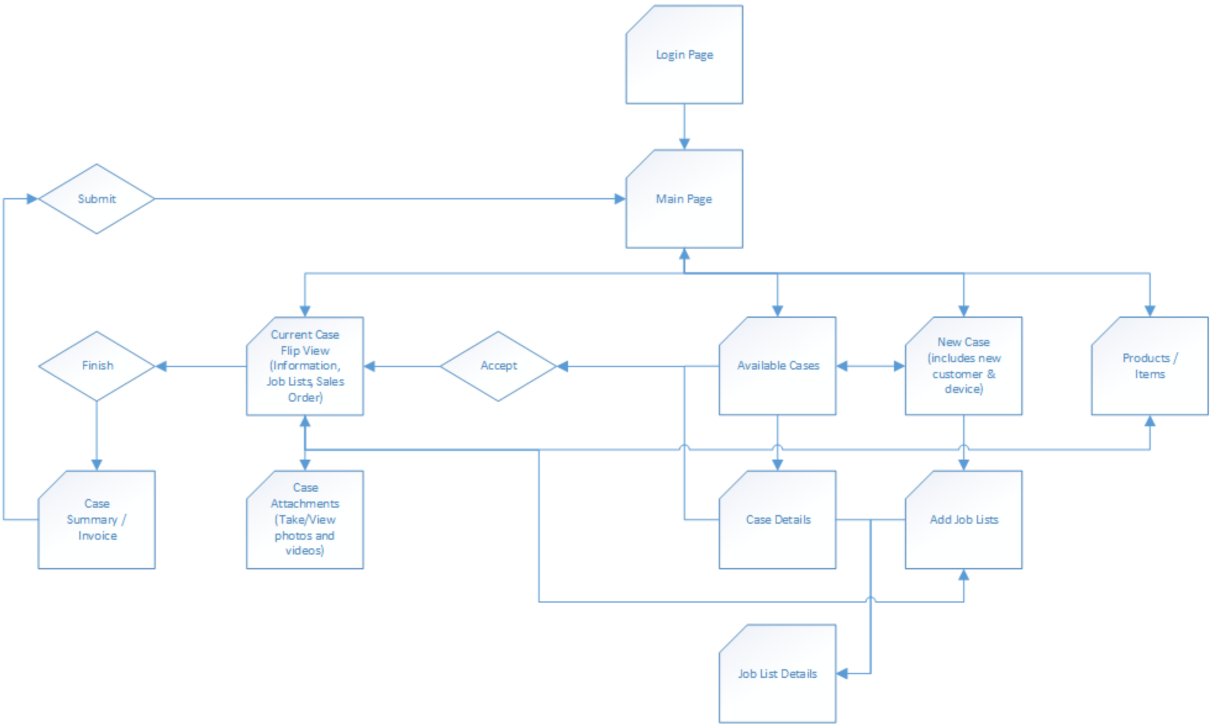

Login Page
Main Page
Submit
Finish
Current Case Flip View (Information, Job Lists, Sales Order)
Accept
Available Cases
New Case (includes new customer & device)
Products / Items
Case Summary / Invoice
Case Attachments (Take/View photos and videos)
Case Details
Add Job Lists
Job List Details

Figure 9 - Flow Chart

5 PRACTICES

This chapter will review how the teamwork was planned and structured.

5.1 ROLES AND RESPONSIBILITIES

Both team members got responsibilities to follow throughout the project. Responsibilities were given due to the time and effort each person was able to provide.

Source control maintenance:	Karl J. Karlsson & Stefán Gunnarsson
Scrum master:	Karl J. Karlsson
Final Report:	Stefán Gunnarsson
User Manual:	Karl J. Karlsson
Testing:	Karl J. Karlsson & Stefán Gunnarsson

Source control was handled by both Karl and Stefán. Since this was only a two man team, it seemed fitting that both made sure that this was in order.

The job of scrum master was to be the communicant between the company and the team along with enforcing that the team adhered to the scrum methodology and kept performance at its highest.

Both Stefán and Karl contributed to the final report but it was the responsibility of Stefán to make sure that it met the standards provided by our instructor and that the final version would be ready on its final hand-in date.

The User Manual was the responsibility of Karl. Both Stefán and Karl contributed to the work creating the manual but Karl had to make sure it was ready for the final hand in.

Both Stefán and Karl handled the testing of the Mobile Mechanic system together. The team did not feel the need to assign this responsibility specifically to either one.

6 SCRUM

After some discussion, the Scrum methodology was chosen for this project since it is decently suited for a project of this size and timeframe. Scrum is an agile based work structure, at start you break down the work into stories and each story into tasks and as such seems quite intuitive. The only concern was that the team only consisted of two people and it would therefore be counterproductive to follow the methodology. Using Scrum did however turn out well. The team did at least gain experience in using it and the two members were becoming accustomed to it towards the end of the project.

During the analysis phase, a product backlog was created that contained the requirements for the Mobile Mechanic project. The project requirements were then split into stories and Planning Poker was used to estimate them. The stories were then split into tasks.

In the beginning of the project a decision was made to have the Mobile Mechanic product owner review the previous sprint at the start of each sprint, both to and then the product owner decided which stories the team should take during the upcoming sprint. The product owner answered any questions related to the stories selected, which in first the sprint was Kristinn Jóhannsson but due to time constraints on his side he had to turn that position over to Ósk Ólafsdóttir which remained the product owner for the remaining sprints. Further details concerning each sprint can be found in the Progress report that will following in the final hand in.

6.1 STORY POINTS

Since neither of the team members had much experience using scrum and were new to the Windows 8 application development the decision was taken to use a tool named planning poker, where each team member would vote on points for each of the stories, this process was then repeated if the difference was too high until both member had settled on a similar number.

The story-point scale was: 1-2-3-5-8-13-21-34-55, or the Fibonacci numbers.

Even though the planning went well there were a few cases that the team members disagreed on the difficulty. In the end both parties agreed on a number without any hard feelings.

6.2 SPRINTS

The decision was taken to have of the first six sprints last a period of two weeks each and the last two sprints last a period of one week each. The idea behind the two week sprints is that both members had other obligation at school and therefore could not work on Mobile Mechanic on a daily basis. This would also suit the team at start since neither of the team members had any experience with developing applications like Mobile Mechanic. Therefore, two week intervals would be short enough time for a sprint in case there were changes made to the project requirements. For the last three sprints, the teams schedule had opened up as the exams were over, so it was decided to leave one of the remaining sprints with standard two-week length, and cut the durations of the last two sprints to one week each, and work every day until the project reached its hand in deadline.

6.2.1 Daily meetings

Each workday the team meet for around 5 minutes, Each member of the team explained what he would be doing during that day, what he had done during the previous workday and if there are any problems that were stopping or delaying his work. Most meeting only took 5 minutes or less but at a few occasions that number reached 10 minutes.

6.2.2 Sprint review

After each sprint the team meet for a review session, we went over what was accomplished during the sprint, showed each feature implementation and how they interacted with previous features. These were informal meetings with no slides or other work prepared for them. At all the meetings, the product owner was in attendance.

6.2.3 Sprint Retrospective

At the end of each sprint we reviewed what went well and what might have gone better, since this was the first real scrum project this team had done, these meetings were very helpful during the first few sprints. All decisions on changes to the task list were made at these meetings, these meetings helped the team improve its processes. At all the meetings, the product owner was in attendance and gave input when needed.

6.3 SPRINT CAPACITY

As show in table 2, this was to be a typical two week sprint where each member would return about 50 hours each and was subjected to changes during each sprint depending on the team members other duties.

	Mon	Tue	Wed	Thur	Fri	Sat	Sun	Mon	Tue	Wed	Thur	Fri	Sat	Sun	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Total
Karl	10	0	10	0	0	0	5	10	0	10	0	0	0	5	50
Stefán	10	0	10	0	0	0	5	10	0	10	0	0	0	5	50

Table 2

The team calculated that it could take 6 two week sprints and it was uncertain how the last two sprints would turnout.

It was estimated that the team would get around 70% percent efficiency out of each hour spent on the project, this would be explained by bathroom breaks, lunch and simply learning the languages used to create the application.

It was the team hope that at the end of the project they would at least reach a 500-600 hours on full velocity.

6.4 SPRINT VELOCITY

To determine the initial velocity, it was decided to break down a couple of stories that the team was decently confident about. The total amount of points assigned to those stories were 13 points and that it would take about 40 hours to implement those stories, based on the hours we estimated it would take to implement each task within those two stories. This came down to little over 3 hours per point, later this number was fined tuned down to an even three as the team gained more experience. The plan

That left the team with the estimated velocity per sprint of: (100*0.7) / 3 = 23 story points per sprint.

At the end of the project, this number had changed very little, we will take a better look at each sprint in the progress overview and note some conclusions there.

6.5 WORK HOURS

The scrum master was responsible for logging hours of each team member.

7 PROGRESS OVERVIEW

In this chapter we will refer to the product backlog and take a closer look at the release burndown chart as well as the story points completed during each sprint, showing how the development progressed. Finally there will be a brief mention of the overall hours spent on this project.

7.1 PRODUCT BACKLOG

The product backlog consist of all the requirement of the Mobile Mechanic project scope, these requirements are then made into our user stories, then each divided into tasks and priority.

We refer to the "Mobile Mechanic – Progress Overview" document for a more detailed look at the Product Backlog.

7.2 RELEASE BURN-DOWN

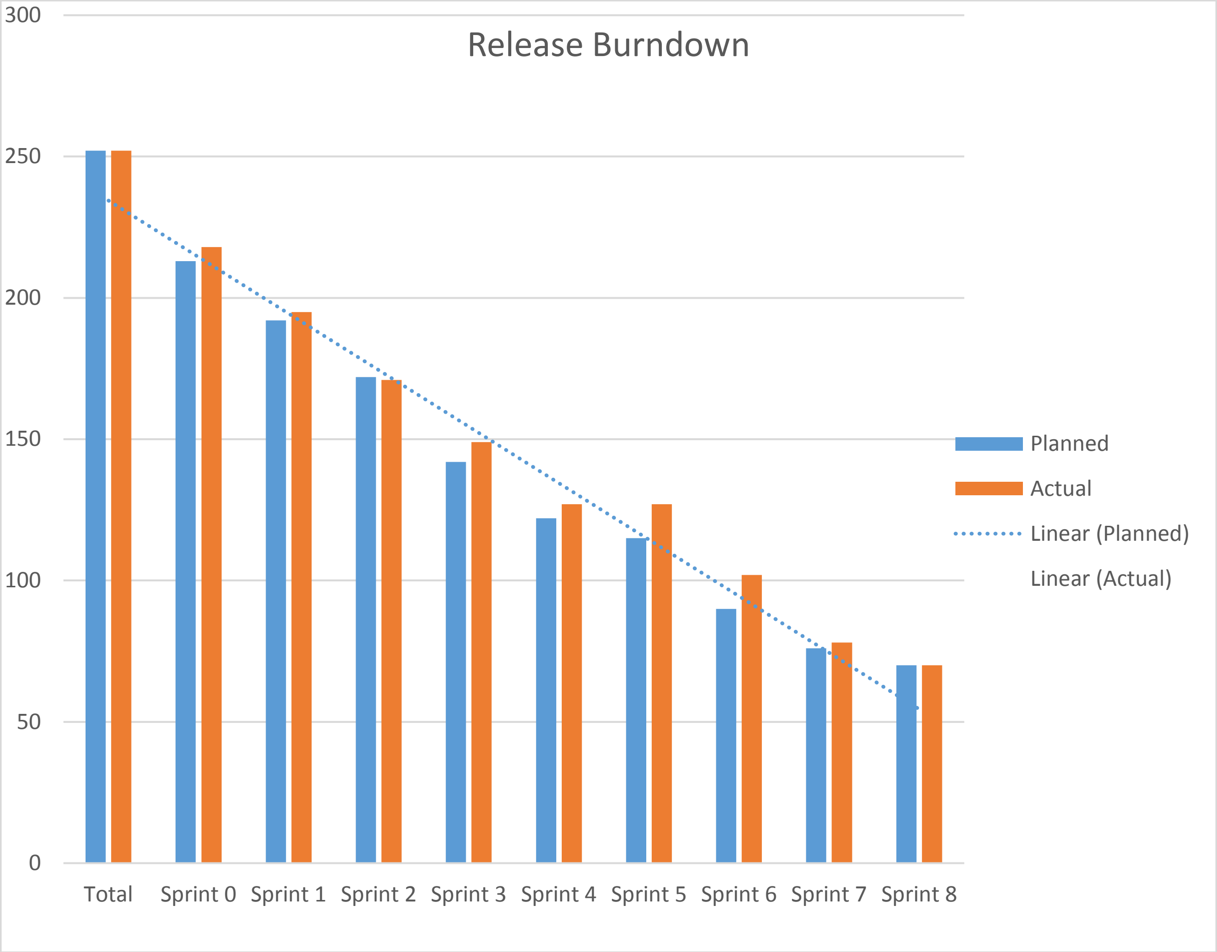


Figure 10 – Release Burn-Down

Figure 10 shows the release burndown chart of our project.

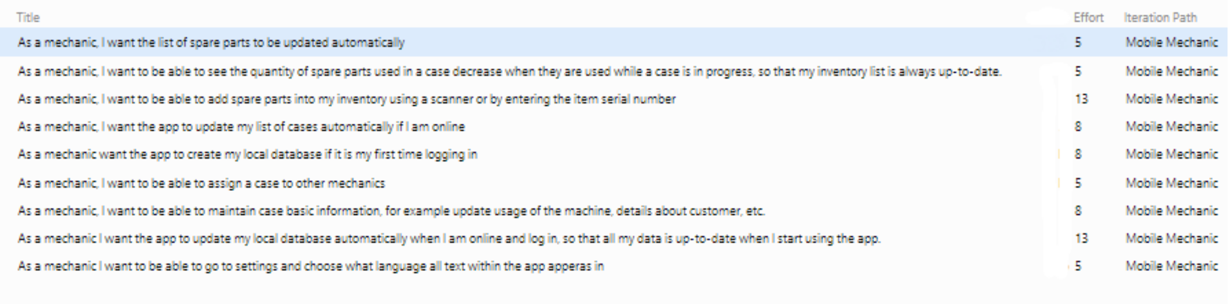
Below, in Table 3, you can see the story points completed for each sprint. The “Planned” row shows the amount of story points that the team had planned to finish for a sprint, whereas the lower row shows the “Actual” story points finished.

	Total	Sprint 0	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8
Planned	252	213	192	172	142	122	115	90	76	70
Actual	252	218	195	171	149	127	127	102	78	70

Table 3

We hit some problems during sprint 4 and 5 due to exams and other projects at school but got on track again during sprint 6, 7 and 8.

The team was not able to finish all the stories in the product backlog, this was due to various reasons, whether it was lacking an internet connection, some will not be implemented and others pushed to the future work of Mobile Mechanic.



Title	Effort	Iteration Path
As a mechanic, I want the list of spare parts to be updated automatically	5	Mobile Mechanic
As a mechanic, I want to be able to see the quantity of spare parts used in a case decrease when they are used while a case is in progress, so that my inventory list is always up-to-date.	5	Mobile Mechanic
As a mechanic, I want to be able to add spare parts into my inventory using a scanner or by entering the item serial number	13	Mobile Mechanic
As a mechanic, I want the app to update my list of cases automatically if I am online	8	Mobile Mechanic
As a mechanic want the app to create my local database if it is my first time logging in	8	Mobile Mechanic
As a mechanic, I want to be able to assign a case to other mechanics	5	Mobile Mechanic
As a mechanic, I want to be able to maintain case basic information, for example update usage of the machine, details about customer, etc.	8	Mobile Mechanic
As a mechanic I want the app to update my local database automatically when I am online and log in, so that all my data is up-to-date when I start using the app.	13	Mobile Mechanic
As a mechanic I want to be able to go to settings and choose what language all text within the app apperas in	5	Mobile Mechanic

Figure 11 Left overs from the TFS2012 Product Backlog

In Figure 11 we can see what stories were not implemented, story one and six were actually completed through other stories.

7.3 WORK HOURS IN TOTAL

Time estimate was around 500 to 600 hours man hours for the project. Total hours spent on the project during sprints were 720 hours.

8 FACILITES

Annata opened up their office for the team, which is located in Mörkin 4, Reykjavík. The team got their own work space and the majority of work on Mobile Mechanic was done there. The team had access to this office all hours of the day and during weekends.

9 FUTURE WORK

In this chapter we will look at the future of the project as well as discussing possible the defects of our solution.

9.1 FUTURE OF THE PROJECT

Mobile Mechanic is still a work in progress, there is still plenty of work that needs to be done as this final project is only one of at least two iterations the development of Mobile Mechanic needs to go through.

Here are the main features that need to be implemented in the future.

- Create a local database for the application using most likely SQLlite
- Connect the application to Annata´s IDMS connector via the internet
- Implement an inventory system for the mechanics
- Add modules to allow the usage of bar code scanners
- Add localization to the application
- Thoroughly test the application with mechanics in the field and get their feedback
- Add application settings, such as larger font and change of colors
- Allow automatic updates to the application while connected to a web service

The next iteration will occur right after iteration 1 and will last about 3 months where the above features will be implemented.

Annata is planning to release the application to its customers at the end of 2013 or beginning of 2014. It will be side-loaded onto tablet computers through Annata's IDMS package instead of being available directly through the Windows 8 store. The reason for this are security and competition concerns.

9.2 SYSTEM OVERVIEW

As seen in Figure 11, this was the original system overview plan for Mobile Mechanic, to the team's knowledge this plan has not been deviated from.

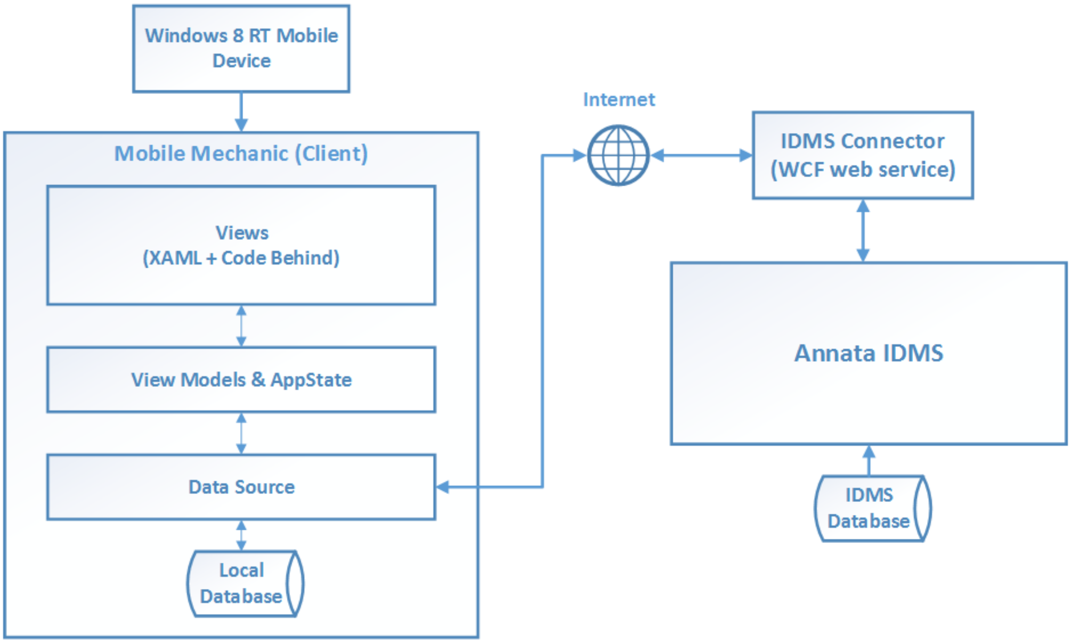


Figure 11 - System Overview Mobile Mechanic

9.3 KNOWN DEFECTS

The biggest defect is that Mobile Mechanic is lacking connectivity through the internet to Annata's IDMS connector, not only does this make it hard to test the application at its current stage but also there could arise design issues with the interfaces and code. Right now all the information is stored locally and then instantly arrives to the screen when called upon, this makes it hard for us to test the application to see how latency issues will play out.

For more details about known defects can be found in the Mobile Mechanic Developer Guide.

10 CONCLUSION

Our work on the project mostly went well. We found it was hard to plan scrum sprints down to the hour due to other factors, such as workload from other courses, the learning curve of a new programming environment and other delays.

Also, considering the lack of common and important UI controls in the Windows 8 RT development environment that are common in WPF applications, such as date pickers, developers are forced to create or find their own – which again can impede development speed, especially for a small team of two, and even produce problems that are difficult to solve.

10.1 WHAT THE GROUP LEARNED

When looking back we learned that communication is a key component in software development, even if it needs to be painstakingly persistent. Lack thereof can have serious consequences. In our case it resulted in the application not being connected to the internet and we had serious delays due to waiting on entity classes that were either missing or had missing properties, to be handed to us.

We should have pressed more frequently for a database from the start and not let things slide for as long as they did – i.e. hoping that these things were in the process of being provided, which wasn't the case.

Giving up on the unit tests was a mistake as well. However it did not result in any game breaking situations, just that it would have helped us dealing with the bug fixing during the last two sprints and possibly lead to less code-smell.

This experience of analyzing, designing and developing enterprise software to meet up with standards will help the team it their future work in the business. Often, people need to experience what works and what doesn't work in order to grow. We are truly thankful for the experience as it already has helped us making decisions for our careers as software developers.

10.2 Thanks

We would like to thank Árni H. Reynisson, Hannes Pétursson and Hallgrímur Arnalds for helping us through this final project and Reykjavik University in general for being a fantastic University.