Tehetena Masresha

Changing Desktop Application to Real Time Web Application

Metropolia University of Applied Sciences

Bachelor of Engineering

Degree Programme

Thesis

10 August 2018

Metropolia

Author	Tehetena Masresha
Title	Changing Desktop application to Web application
Number of Pages	35 pages
Date	10 August 2018
Degree	Bachelor of Engineering
Degree Program	Information Technology
Professional Major	Mobile Solutions
Instructors	Pasi Siitonen, IT Solution Architect Ilka Kylmäinen, Principal Lecturer

Desktop applications are applications that run on several platforms of a desktop environment. These standalone applications are powerful tools that have been in use for several years.

The thesis is done for a company named Semel Oy. The aim of this thesis project is to produce a Realtime web application that can replace a desktop application. The case company has developed a Windows form desktop application around ten years ago. Since the technologies in the desktop application are outdated, the case company has demanded a new fast and efficient Realtime web application that can be used instead of the desktop application.

This thesis project has produced a Realtime web application built with angular JS and bootstrap frameworks. The web application has taken expectations of the old desktop application users and new users in consideration. The user interface is designed by compromising both user group expectations. To enhance the performance and the effectiveness of the web application, recent libraries and software have been used. These libraries are used in a way that the interface and behavior of the application will not deviate from the desktop application and create confusion for users. SignalR and Kendo UI grid are among the libraries used in this web application. Their collaboration in making a Realtime application are explained.

Keywords	SignalR, UI, Kendo UI

Metropolia

Contents

Introduction 1

2 Desktop application and web application 3

2.1 Pros and cons of Desktop and web application 4
2.2 Windows Forms Application 6

3 Realtime web application development and tools 8

3.1 Designing 9
3.2 Development 10
3.3 Deployment 13

4 Semel Oy Contacts web application Development 14

4.1 Overview 15
4.2 Requierment 18
4.3 Process 20

5 Implementation 21

5.1 Early phase 21
5.2 Process 23
5.3 CSA 25

6 Problems and Proposed solution 27

6.1 Design 28
6.2 functionality 31

7 Conclusion 32

References 34

Metropolia


Metropolia

List of Abbreviations

VSO	Visual Studio Online Git
UI	User Interface
GUI	Graphic User Interface
WWW	World Wide Web
CERN	"Conseil Européen pour la Recherche Nucléaire"
CGI	Common Gateway Interface
C#	C Sharp
SPA	Single Page Application
JS	JavaScript
npm	Node Package Manager
ADB2C	Active Directory Business to Consumer
API	Application Programming Interface
CRUD	Create Read Update and Delete

Introduction

In the last few years we have witnessed rapid development of web applications. Since web technologies are being adopted from big and complex companies to startup companies, there are several opportunities and potential career paths for web developers all around the globe.

The purpose of this thesis is to produce a web application. It is commissioned by Semel Oy. The case company is a private held mobile information systems and software provider. It also works with mobile system ventures demanding system and software integration services.

The thesis project is done by taking part in making a web application based on an old Windows Forms application developed by Semel about 10 years ago. The project is developed by a team of two. One is developing the backend of the new web application and the author will be working on the front end. A GitHub and VSO Git (Visual Studio Online Git) repository is created and in use to manage the project and develop in the team.

The purpose of this thesis is to produce a web application by joining a team of developers in the Semel Oy. The thesis will be based on this web application project. It will discuss the technical and theoretical details related to Web development and design based on a preexisting Windows Forms application that has been in use for quite a long time.

The web application is developed in visual studio code development environment for the frontend. Angular JS is used as the development framework. Other than Angular JS, jQuery and JavaScript are also the development languages used. The backend is developed in visual studio pro 2017. Asp.net Core was planned to be the development language for the backend but due to problems that occurred in the problem .NET console application was needed to be developed. Both backend applications are being used for the server-side data manipulation and interaction.

The thesis will discuss the process of the web development and briefly the tools that are used in the practical work. These are AngularJS, SignalR, Kendo UI, VSO and GitHub.

The process of the development mainly includes designing, Angular JS programming, testing and deploying the web application to Semel's test environment.

There are two main difficulties in this process. The first challenge is part of the designing process, which is going to be done in Bootstrap. How to keep the web application good looking and at the same time not extremely different from its older version for the target group and how to balance the web application for both old users and new users interesting? In other words, for the website to neither be so different and confusing, nor bad looking and boring.

The second difficulty is more technical. The existing Windows Forms application has event driven features. The web application is planned to be Realtime application when needed. Integrating the latest Realtime technologies and libraries to this website has led to several incompatibilities with the older database. Furthermore, this process may also cause the web application to lose a resemblance to the older design.

By referring to the technical documentations, advice from the project manager and other references, the author believes to have found solutions for these problems and briefly discuss and present them in the thesis for others to refer to and learn from it.

2 Desktop application and web application

A Desktop application is a software that runs on a desktop environment. Desktop environments can be of different platforms. Among these platforms Windows/Linux and the Mac platforms can be good examples. The Development of desktop application has been introduced to the world before web applications emerged. Among many reasons, GUI (Graphical User Interface) developed by Apple in the 1980th, made desktop applications far easier and cleaner to develop. Programmers were able to develop a desktop application with about 95% of the programming with built in graphical elements and without writing or bothering to recall the codes and programming language commands. [1]

Desktop applications are also named Standalone applications since both the client rendering commands and the data processing codes can be executed on a single desktop application. [1]

After the invention of the internet and especially after the WWW (World Wide Web), programs and software started being developed it to web applications. As demands for functionalities and supports became numerous that desktop application production started becoming complex. It also leads desktop applications to be dependent on third party customs and libraries which is costly and heavy on the environment it is running. This and several other reasons have led the concept of web applications and thin client architecture to be more adoptable and famous. [2]

The WWW was invented by Tim Berners-Lee who was working for CERN, a science laboratory located in Geneva Switzerland. The main reason for the invention of WWW was the need of transporting files and graphics using the internet by CERN. [4]

The introduction of the concepts of hyperlinks and a common UI (User Interface) throughout an application triggered the development of CGI (Common Gateway Interface). The CGI was a way of enabling browsers run components and resources on a web server. From here on programmers keep on finding ways and invented new designs and architecture for data processing and manipulation. Making a smooth connection and dataflow with a database all around the globe was also possible. This made websites step several steps forward towards what the present-day advancement. One big drawback on the websites at this point was that web servers were only capable of delivering static web pages. When a query for a static page was made by the client side, the server will then read the query and look for the correct web page and delivers the requested page to the client browser. [1,3]

The static web pages were replaced by dynamic webpages through time and this leads to the invention of Application Servers. Application servers are frameworks through which request and respond from a webserver are processed back and forth. When a request for a dynamic web page from the web server is received the requested page will be first delivered to the application server. So that it would read and loads it on the web server as a static page. After receiving this static web page, the web server will then render it accordingly on to the browsers window. [3, p.111]

The application server also functions as a layer while using server resources. Mainly the database. Communication with the database and the application layer is done via database drivers such as APIs that are serving as translators to the data and requests sent back and forth. [3]

2.1 Pros and cons of Desktop and web application

Before starting any programming development several factors have to be taken in to consideration in order to decide what would work better to the developer's advantage and offer effective, easy and smooth functionality architecture. Both web and desktop applications have their own advantages and downsides depending on the kind of functionalities and products a developer would want to achieve. This chapter of the thesis discusses common and basic pros and cons of desktop applications and web applications concerning the concepts of designing, centralization performance and portability respectively.

Desktop applications provides a powerful control when designing the UI. A programmer can build a UI exactly as intended with less coding and more graphic and visual supports. This being strongest sides of desktop application it can also be too much especially if the numerous third-party controls are considered. A time and money costing trainings and researches will have to be done to be more effective. On the other hand, web applications allow programmers to use UI elements that are familiar to most of the target users. Desktop applications can be complicated and less intuitive for most of the users. [5]

As downsides to HTML UI elements in web application, it's browser dependent behaviors are worth mentioning. UI elements in web application may appear differently depending on the browser it is running on. Besides the UI, certain functionalities are not supported in all browsers. Another big problem concerning the web UI is that absolute positioning of elements is not supported, which makes designing look less elegant. when Desktop applications grant rich UI set up controls. [5]

Regarding centralization of application deployment and updates web applications are far more convenient and effective. If a certain web application is deployed only to on central server, it would be possible to allow users all around the globe to access the web application via protocols and networking setups. Updating too is done easily by just updating the web application that is located at the central server and users would get the updates automatically or on demand depending on the way it is designed. On the contrary, deployment in desktop application is done by burning CDs or setting up a push server to every single user all around the world. The same applies for updating a desktop application; the same deployment process will be done all over again. [5]

Desktop applications have higher performance speed than web applications in general, because UI components are drawn just one time when the application is fired. In web application every time the client requests a page the page UI will be requested from the server and fetched and will be redrawn for the user to see. Desktop applications can interact and integrating with other products and hardware devices more easily and effectively that web applications. In web applications the programming is long and complex to achieve this functionality. Working with excel word and similar other documents is strait forward with desktop application while it is not in web applications. Also, web application development is highly dependent on web browsers. Therefore, clients are forced to use specific browsers against their will. [5]

The final aspect to be discussed is portability. For a certain application to be used by remote users, web applications are the way to go. This is mainly because of the secure and effective potential of the http protocol in transferring file and data to and from any part of the world. While desktop application could be a bit hard for remote users. There needs to be a connection established to the main network to get the application working. And this connection would require extra setups and security risks as well. In addition to connection if users must work on a different workstation, the desktop application has to be installed and all the saved changes preferences and histories of the previous work-stations will not be accessed form the new work station. [6]

To sum up both desktop and web applications have their own drawbacks and advantages as discussed above. But this doesn't mean that one is better and the other is less effective. It is up to the developer to make use of the advantages provided by both application types in accordance with the type of product being is built, the size of the data being used, the target clients to whom the application is built for and so many other reasons. [5]

2.2 Windows Forms Application

As elaborated in the introduction the project on which this thesis based on, is a desktop application built with Windows Forms. It was built about ten years ago with the advancement of Windows Forms available at that time. Pasi Siitonen is the developer who created this desktop application from start to finish. A detailed overview about this desktop application will be given in later chapters. In this chapter Windows Forms will be discussed.

Windows Forms application is an object included in the System namespace. To elaborate this a bit more, all components and objects in C# (C Sharp) are arranged and systemizes in to namespaces. These namespaces contain objects that will also include other sub objects inside of them. The namespace organization helps components of the same features, functionalities or even names to be unmistakable. An input field can be a good example to justify this. Both the System.Web.UI.WebControls and System.Windows.Forms namespaces have input field components. An input field in System.Web.UI.WebControls stands for an input field inside of a web page while input field in System.Windows.Forms represents an input field on a windows form UI. The Forms component is nested in windows which represents the framework. Again, the Windows in included in the System namespace. Therefore the forms can be accessed in a C# project using the call method System.Windows.Forms. [3, p. 12]

The System.Windows.Forms also referred simply as the Windows Forms is a graphical library class that enables programmers to build the user interface components and functionalities of windows applications. As it is a part of the windows namespace, it is executable on the windows operating system. The Windows Forms has a forms class embedded in to it. The forms class supplies Form Methods, Form properties and Form event classes. These three main classes include numerous methods that will govern almost all the behaviors of the windows forms UI. [6]

3 Realtime web application development and tools

Web application development can be grouped in two kinds to get a clear understanding on how it works and how developers design and develop a web application. The first one is the traditional web application and the second one is SPA (Single Page Application). The coming paragraphs will briefly discuss these web application development technique types.

The traditional web application development is mainly done with server-side scripting languages. Java, .NET, Ruby, Perl and PHP and be mentioned as examples of these languages. The development architecture behind the traditional web development is similar in all the scripting languages. When a web page is requested by a client on a browser the web server will receive the request and send the request to the database. The web server will get the resulting page from the database and build the page template and dynamically render the page on to the client browser screen. The web server has to take care of the request response process and again loading and rendering the requested page on every request made by the client. This will lead to a need for several web servers sharing this task since most of the job is done by the web servers. [8, p. 33]

Single page application uses a different code flow in rendering web pages and handling request and responses than the traditional way. When An application is fired in SPAs all templates including the HTML, JS (JavaScript) and the CSS files are loaded on the client browser. Therefore, the web server does not have to render templates as in the traditional development. When a page query is made to the web server it only must fetch contents of the web page form the database. The contents then will be built and rendered on the browser. Even though the browser would be handling the heavier tasks, the web servers would not be overloaded and bottlenecked as they might be in traditional web architectures. [8, p. 31-32]

Realtime web is a way of letting users get all updates and changes of the data in the application whenever it happens. Users will not have to do anything to acquire these changes instead they will be integrated without the user’s knowledge and change on the UI. [9, p. 26]

Realtime updates and changes can be integrated to a web application through several ways. The recent and advised way of doing it is using Web Sockets. This is done through a connection initially made between the client and the server. This connection channel is a socket connection. For the security this connection also supports protocol. If this connection is sustained updates will be sent via this connection if needed with a frequency set. [9, p.30]

The coming subsections of this chapter are sorted depending on the authors experience and approach in developing this web application. The three sub sections will give a short briefing about the major parts of web development.

3.1 Designing

The concept of web designing has not been consistent since it arises. Depending on the times technologies and programming architecture its definition well changes accordingly. In the older times web pages were developer through static HTML whose functionalities and logics were hard codded in the HTML page. At this point web designing was defined as building the backend functionality of a web page. [10, p. 2]

At this point the previous definition for web designing would be ambiguous. Hence a better definition needs to be made because the advancement of web application development process and development tools. A suitable definition for web designing at this point of time can be the making of the user interface looks and the detailed artistic feels of a website so that it would reflect the sense of the purpose it was built for. [11, p.203]

To design effective and attractive websites one has to master both the technical aspects and the artistic implementation of these technicalities on to a certain website. To make websites reach the current standards websites have to work effectively and smoothly with a database. The data sent and received to and from the database should be displayed in a format that is intuitive good looking and attractive. So that users can feel comfortable with the website. Therefore, the best websites are constructed by fulfilling both the technical requirements and the artistic feels. [8, p. 2-3]

In this thesis project designing the UI looks and feels was a little out of the norm. This is because the design of the web application was based on the former desktop application. There was no UI/UX design protocols or mockups or user experience cases. This is mainly because the old desktop application has been used for the past ten or more years by similar users, the UI looks, and feels are familiar to users. Completely changing the UI design would make target users feel uncomfortable and the web application might seem strange too. For this reason, it was decided that the new web application to look similar to the old web application so that users could find it easy to use.

To achieve the needed UI for the web application, several tools and technologies have been brought to use. Bootstrap integrated with HTML and CSS was used to develop the frontend framework. Since the old application includes lots of grids in most parts using normal tables would be ineffective and time consuming. Therefore, a grid system framework that suits the application well was purchased. This grid system is called Kendo UI. All the tables in the old application are replaced by Kendo grids. which looks better and made the programming a lot easier. Detailed explanation of the Kendo UI framework will be provided in the upcoming chapters.

3.2 Development

The next important step in website development after the design is giving the components of the design actual purpose and functionality. This step has several parts in it. The basic part is structuring the web application. As mentioned above this application is structured to be a single page application. After structuring and building a skeleton or a frame for the web application in a suitable order giving functionality for all UI components follows. Finally, after everything is in place making data on the UI change every time there is an update in the server-side data in Realtime. This part is a basic and important part since the rest of the components are developed on top of it.

As mentioned in previous chapters this thesis project is a single page web application. To make this application, the JavaScript framework used is AngularJS. The main reason for this framework to be chosen was the authors personal interest and prior experience on this area. AngularJS by its nature a simple to use framework supporting CRUD

operations. since this project itself requires CRUD operations angular has worked nicely with it.

Angular JS or angular 1 is the first version of the angular families. It is a JavaScript based framework that was developed in 2009. It provides an MVC (model view controller), also addressed as MVW (model view whatever) development architecture. It's components as scope, controllers, rotoscope and so on enable two-way data binding in development. Which is quite helpful in changing data in both the UI and the script. Development is simplified and modularised in angular JS mainly because of the ability to reuse codes in a precise and clean way. This is due to Angular JS has a introduced the concept of directives. Directives made it possible to write a more elegant and divided code. [7, p. 69-77]

Throughout the development process there are several functionalities requested by the application. Amongst them was real-time data update. In this project most of the data coming from the database are changing constantly. Also, those changes are mandatory for the clients in making the decisions and provide the service. In the old desktop application clients had to request updates whenever a latest data was needed. Furthermore, in some parts the application was forced to fetch updates from the database in a given range of time. The same approach could have been used in the web application. since this would give the web servers a lot to work, an alternative and better approach has to be used. This approach was using SignalR to handle the Realtime updates and data flow.

SignalR is an open source library that takes care of real-time data flow between web servers and Clients. The SignalR makes things a lot easier since it uses WebSocket to establish a persistent connection in the beginning. The connection is very useful in monitoring data flow. This connection management lets the server to recognize connected users and decide which data shall be accessed or be hidden to which customer. This improves the security of the application in general. [12]

SignalR enables a persistent real-time data transfer via a hub API. This API is a platform that enables methods defined on the server side to be called and used from the client side. And the reveres holds true as well. With just using key words like server or client, methods can be defined on both sides and those methods that are public will be accesses from both sided. The rest of the piping is taken care of by the SignalR

library. SignalR hub API also has methods called when a new connection is made or when a device disconnects. Tracking connection plays a great deal of role in data handling, authorising and dealing with errors. [13]

Another essential development process in this thesis project was localisation. The clients using this application are based in three different countries with three different languages and date and time format. Therefore, to satisfy the needs of all those clients the application has to support all three language systems. These languages are Finnish Swedish and English.

To implement localization the tool used in this project is the Angular-Translate module. Angular-Translate is an angular module that can be installed via bower or npm (Node Package Manager) and injected as a dependency. This module enables programmers to modify parts of it in a way that suits the application better. The functionalities are flexible when it comes to error handling, Storage providing and loaders. translations can be made via filters or directives depending on the code structure or programmer's personal choice. It also loads the i18n data asynchronously which makes it work faster and efficient to work with. [14]

The last development feature that also has a significant role for this thesis project is authentication. In this thesis project clients should be authorized to get access to the application. There were no authentication systems on the old desktop application since the application was manually installed to authorised computers on every possible location. But for web applications anyone with the correct URL can access the application without authorization system enabled.

Even though handling user directories and identification is possible to be handled on the database, it adds load to the database and server. It increases the amount of time and work in the development process. To help ease this process tools are available from several sources that can handle user authentication in a secure and reliable manner. The tool chosen for this thesis project is developed by Microsoft. It is called the Microsoft Azure ADB2C (Active Directory Business to Consumer) authentication system.

Microsoft Azure ADB2C is a customer identity service for several categories of application types. Web application or JavaScript clients are in this category. ADB2C allows customer identification service and access management as well. Login system can be done

via company or personal email and any social accounts. It allows users to set a policy that is unique for a certain organisation and reuse it for several applications developed under the company. [15]

3.3 Deployment

Websites are deployed to a web server so that they can be assessable for users who are authorized. Before the deployment in this project the web application was tested by serving it from the local machine. To do this a simple node server called HTTP Server was used. HTTP Server is available as a npm package and it runs from the command prompt. The HTTP Server has worked as needed without any difficulties throughout the project. After making sure there are no bugs or errors on the code the project was deployed to the case company's test server. The case company uses windows iis (internet information services) server. [16]

4 Semel Oy Contacts web application Development

As mentioned in the introduction this thesis project is done for a company named Semel Oy. The case company is involved in development, maintenance and serving large scale user software. There are several software products developed in the case company that are used in so many countries in Europe. There is a large scale of users covered in all services given by the software that are produced by Semel. The case company has several offices in all the countries it is giving services. The headquarter of based in Helsinki, Finland. This is where the IT department is also found. [17]

The web application produced on this thesis project is currently named Contacts. The Contacts web application is developed based on an old desktop application also named Contacts. The old desktop application serves the same purpose the web application is intended to provide. Before stepping to the Contacts web application development process, a brief overview of the old Contacts desktop application is necessary.

Contacts is a desktop application among several software produced by the case company. It has been in use for the past ten years and still being used. Contacts is developed by Pasi Siitonen the case company's Software architect and the project manager of this thesis project is. The Contacts application is entirely written in C#. To be more specific it is a windows forms application. Where the UI is made with just one windows form.

SQL database is used for handling the company's data. The application has several tables in the database. A huge amount of data is stored in these tables. These tables handle data needed for several purposes. Among these purposes few are mentioned as follows. Data of cars information, zones of every car and necessary data concerning every zone, shifts of every drivers and cars, messages sent and received to drivers, routes of the cars, payments made and invoices, status of every car drivers and zones, bookings made and details about every bookings are few of these data processes handled by the database.

The source code of the Contacts application has several files that take parts in building the software's logic and looks. The source code contains few folders which are organised

in a manner that the author can understand it. In other words, the setup of the code does not follow any guidelines by the case company or any other. For example, one folder contains fractions of UI elements that are added on top of the applications parent UI to make the complete look of the application. Another folder includes the configuration files. Also, commonly known as config files. The config files handle security and authentication for the database connection. This folder also contains other start up configurations for the application to follow certain preferences.

There are also translation files placed in one folder. The files are used in localization of the application. Images and logos are also placed in another folder. The rest of the folders include files that are planned to be implements in the future for further functionalities and modifications. But most are not implemented, and some are not completely developed. They contain codes that are not complete, personal comments reminders and TODO lists.

The entire logic of the contacts application rests in one C# file named Form1.cs. In this file the application is controlled from start to finish. The code entirely is 6703 lines long. The application is not separated in to parts or modularized. The whole functionalities, input output controls, UI implimentations are handled with in this one big C# file. Further insights will be briefly discussed in the chapters to come.

4.1 Overview

Before starting the actual designing and development of the new web application a thorough understanding of the old application was indeed essential. Knowing the demanded functionalities, types of data transported, tools used, outcomes needed, and nature of user groups helped a great deal in taking the development process several steps ahead. All this information and even more helpful knowledge about the old desktop application are without doubt gained through testing and using the old desktop application.

The old desktop application was a bit complicated to understand in the beginning. Since it is designed for a specific purpose and only used by authorized users, the application was not straightforward for users who does not have prior briefing and understanding of the purpose of the application.

The general look and feel of the desktop application tell that the application was developed a while ago. There are certain decorative and technical advancements missing from it. These defects can be seen in Figures 1 and 2.

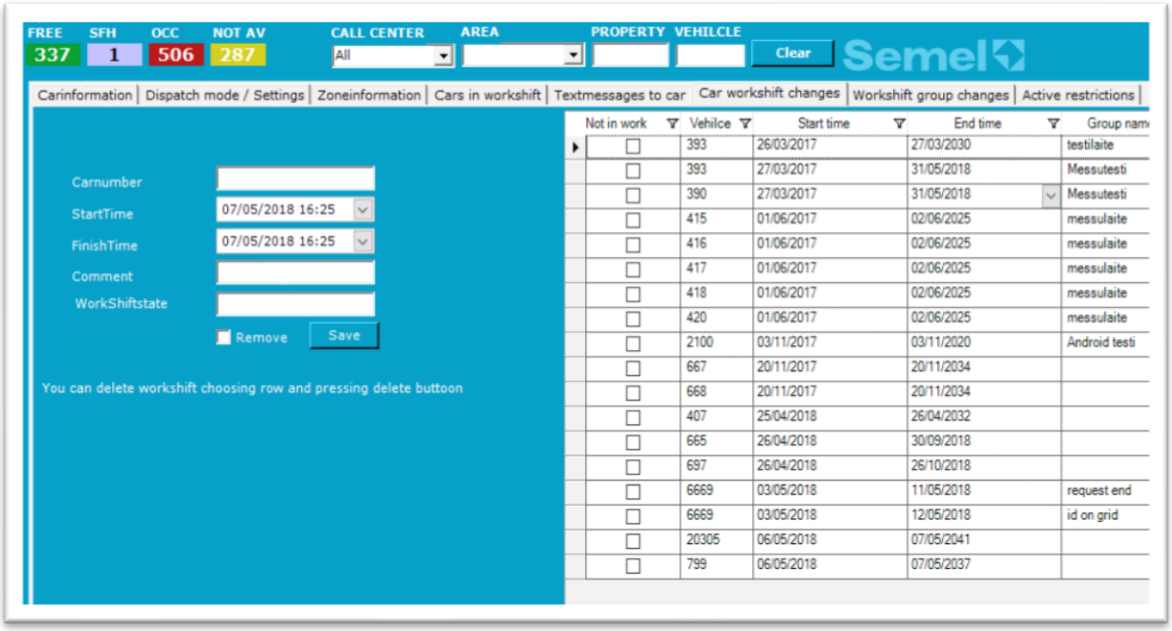

Figure 1 Contacts desktop application UI

Figure 2 illustrates the responsiveness of the desktop application when a page is minimized or opened on small screen devices.

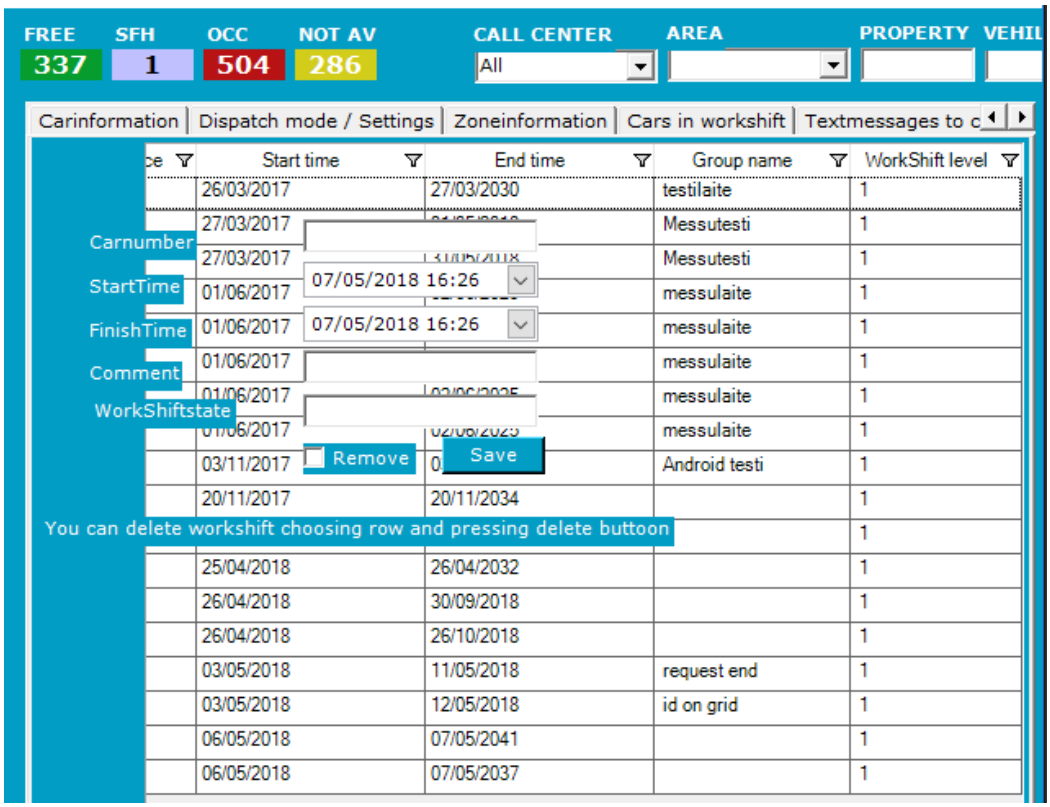

Figure 2 Non-responsive UI Contacts desktop application

As illustrated in Figure 1 the desktop application's design looks quite old. Besides the look the application is not intuitive in some respects too. To mention few as examples, it seems hard to tell on which navigation tab the current interface is on, unless users are familiar with the application or unless the application has not been used couple of times.

The desktop application also has features that are complicated than they should be. As shown in Figure 1 deleting a row is implemented in a complicated way where users must press the delete button on the keyboard after selecting a row on the table. This whole process could have been avoided with just a single delete button.

Figure 2 demonstrates the non-responsive property of the application. If the screen size of the desktop window gets smaller elements on the view will also start to overlap on each other. This will disrupt the entire functionalities and look of the application.
It is also visible that the size of the fonts and the elements are not flexible at all. A better approach would be to make these elements change their size as it would fit the screen size.

There are also other features which could not be captured in an image but where improvements should be done. For example, if users click a cell in the grid the cell automatically become editable. Users can then change the data recorded on the cells. But the problem is editing and changing data is not implemented on the server
therefore, the changes will not be saved in the database. Although the data on the grid changes, there is no way users would know the changes are temporary unless the application is fired again, or users navigates to another view page and loads the grid back again. This is misleading and confusing for users. These and other defects are addressed and tried to be done in a better way in the web application.

4.2 Requirement

The production of this thesis project required several tools for certain purposes. The first and most needed tools for communication and code sharing are Git and VSO. The backend code is saved in VSO so that new changes can be pulled and used while developing the frontend. the front-end code is saved to a git repository. But at some point, all the codes were deployed to the test server and were served from there.

Since the nature of the application requires number of data to be formatted in to a table Kendo UI suits this project better than other grid systems. This is because the Kendo ui has built in support for SignalR. The connection and data transfer including CRUD operations are taken care of within the kendo without writing extra commands to do it.

The code snippet below shows how a signalR connection is stablished in Kendo ui and how changes and updates will be integrated in the grid.

```
var hubUrl = "hub url goes here ...";
var connection = $.hubConnection(hubUrl);
var hubStart = connection.start();
var hub = $.connection.textMessageDetailTickerHub;
$("#grid").kendoGrid({
    height: 550,
    editable: true,
    sortable: true,
    columns: [],
    toolbar: ["create"],
    dataSource: {
        type: "signalr",
```

```
        autoSync: true,
        schema: {},
        transport: {
            signalr: {
                promise: hubStart,
                hub: hub,
                server: {
                    read: "read",
                    update: "update",
                    destroy: "destroy",
                    create: "create"
                },
                client: {
                    read: "read",
                    update: "update",
                    destroy: "destroy",
                    create: "create"
                }
            }
        }
    }
});
```

As stated in the introductory chapters signalR uses a web socket connection to stream data through Kendo grid would also first establish a socket connection then the hubStart promise is fulfilled.

If the connection is made the kendo will start to listen to the hub assigned to tat specific update signalling. Since by rule the structure of the updates and the structure of a kendo grid row are similar whichever row has been updated will be traced with its id and its updated properties will be bounded t the grid table and the UI will be redrawn without even the user noticing anything on the grid. [18]

As long as the grid is being displayed and as and as the connection is still on the updates will keep being posted to the grid even is the updates are coming from the user's grid or other users working on the same data. [18]

This code snippet also shows how straightforward and easy Kendo UI makes making a WebSocket connection. Not only the connection but also making grid is a lot easier in Kendo UI grid. Array of fields and their titles are the only required data to be provided for Kendo to generate a table.

The database and server are also tools needed in making of this application. The same database is used as the one used in the old desktop application so that exact data can be displayed and to reduce errors.

4.3 Process

This thesis project took several steps to get to the development stage it is at now. Since there are no guidelines to follow during development phases or even on how cods should be formatted and written in the case company, developers have to decide what to do and how to do it based on their personal knowledge and standards.

This helped the programmers taking part in this thesis project experiment and test without any restrictions. This freedom let developers learn a lot from all the trials and errors. But at the same time general process can be a bit time consuming at points. Also, codes written with kind of freedom are not easily understandable for a programmer who did not take part in the development.

Generally, the development process of this web application was accelerated due to several reasons. One and most important reason was that there was a complete project to base the thesis project on. Several parts of the development were easily implemented so developers do not have to start development from scratch. Since there is a functional application most functionality development processes were just duplicating the functionalities in a more effective and faster way. As an example, translation is among processes that were easily implemented. The old application was designed to be used by three language user group. The new desktop application will also be used by the same user group. Therefore, the old translations are directly used for the new application too.

There were no mockups or user experience-based designs implemented. The old desktop application’s looks and feels are highly viewed in the web application as well. The elements of the UI are also used in the new application. The major changes made in the UI development is to change the colors to a web safe contrast. More components were added in order to make the functionalities more logical and straightforward. Detailed insight about the entire process of the development and the current status of the application will briefly be discussed in the next chapter.

5 Implementation

To As stated in the previous chapters the case company does not have or follow any development process guidelines. Therefore, it was decided by all developers including the project manager that the First thing to be done was test and try understanding the desktop application without any guidance after a short verbal introduction of the application. It was difficult to understand the application without any help. But at the same time, it helped the developers get good in site of the defects and ideas for improvements.

After this it took numerous meetings with the project manager and briefings to be able to use the desktop application and use in independently. After understanding the application and before the development started the project manager suggested that among the eight components of the desktop view pages the Messages tab has to be implemented first. This was because most of technologies and development components used throughout the desktop application, are also implemented in the Messages tab. Therefore, completing this functionality will let developers build the rest of the application components based on it.

In general, the implementation was a bit disorganized and messy in the beginning. Mainly because of the lack of guidelines and the time spent on understanding the desktop was long. But as the project progressed, steps were taken according to what is best and effective for the development.

This chapter discusses the entire project implementation process in depth. The processes are classified in three sections as the Early phase, Process and Current status of the application. The whole development process following the above decision will be discussed in these sections according to their orders.

5.1 Early phase

After Understanding the desktop application and knowing where to start, the first part of the development was to decide what framework language to use. As a result, it was decided to be Angular JS with bootstrap. This decision was made by the author based on the application type and personal preferences.

After this the first few weeks of the development were focused on building the modularized application skeleton. A web application with dummy components or just descriptive texts as placeholders with all the eight navigation view pages and the index page was developed. This web application was used to develop the rest of the components and the functionality on top of it. The web application is structured in a way all eight of the components are modularized separately with their own respective controllers using angular JS. After this all the routing sequences were implemented to look similar to the desktop application.

The next step was to design the UI. The UI of this application is made with the Bootstrap framework. Since the new application has to look familiar for users the UI was developed based on the old desktop application. There were two top navigation bars. One for navigation through the eight view pages and the other for filtering the grids found on three of the view pages. The filter bar also includes four colored text boxes where real time count of four car groups is shown. Each color representing specific car groups.

When the navbars were completed as discussed in the previous chapters the first view page to be developed was the Messages. This page is basically used to send individual or broadcast messages for the devices in the network. Also, there is a portion of this page where all messages sent by the current user and other users, are displayed in a grid and updated in Realtime.

The first part of the messages page functionality implemented was the message sender form. In the desktop application the labels of the form inputs are confusing and packed in a small space. Therefore, to avoid this the form on the web application is designed to be spacious, with descriptive labels on top of every input fields.

By the time the UI for the message sending form was completed the API (Application programming interface) for sending messages was ready. Therefore, this API was used to send the user inputs from the form to the database where it would be processed and sent to the devices as requested by the user in the form.

After completing this part of the messages, the next was to get the Realtime updates of the messages displayed in a grid. Therefore, Kendo grid was brought to use to make this happen on the web application**.** This process and the rest of the processes are discussed in the next sections of this chapter.

5.2 Process

Once the message sending form was functional, logging the message exchanged on to the user screen was following step. The messages have several attributes that define them, and all those attributes are important to be seen by users. Therefore, using a grid was the preferred way to display the messages. In the web application the grid was made using the Kendo UI tool. This tool not only makes the grids look nice but also responsive and flexible.

Most importantly developers do not have to listen to events, collect form data, and write the code for processing the data to perform the CRUD (Create Read Update and Delete) operations. To enable a kendo grid, possess the CRUD functionalities and integrate then to the table is easily achieved by setting a certain property to be true of false. The code snippet below clearly shows how easy and strait it is in kendo. [20]

```
var grid = $("#grid").kendoGrid({
      dataSource: {
        transport: {
          read: {
            url: root,
            data: { format: "json" },
            dataType: "json"
          }
        }
      },
      toolbar: ["create"],
      filterable: true,
      resizable: true,
      sortable: true,
      pageable: true,
      columns: [
        { field: "ismanual", title: "ismanual", hidden: true },
        { field: "workShiftState", title: "Workshift State" },
        { command: ["edit", "destroy"] }
      ],
      editable: "popup"
    });
```

Just by setting certain keywords to true or adding a certain key word to the code will automatically enable these operations on the grid. This is demonstrated in the following code snippet and Figure 3 shown below.

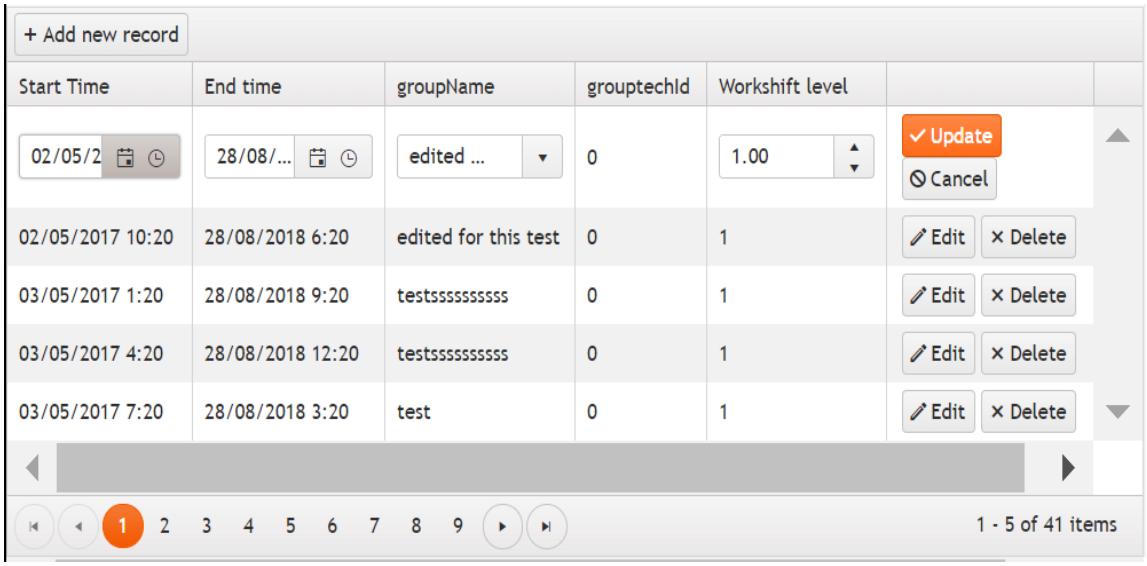

Figure 3 CRUD operations on Kendo grid

As shown in the snippet filtering sorting paging and resizing the data and the table is done very easily just by setting the key words for the respective functionality to true. And Figure 3 demonstrates it how smoothly the paging and sorting are performed on the grid without any further programming. And for the CRUD operations if the command strings create, edit and destroy are included in the program as showed in the snippet, then the grid will have “add new record”, “edit” and “delete” buttons that would send the relevant data when clicked. Figure 3 shows how these three buttons will look on the grid. The edit button will also make the grid row editable for users also a cancel option as demonstrated.

When the messages grid was done the development of the SignalR hub and the streaming functions from the backend were also complete. After this, integrating the SignalR hub to the kendo grid was done simply as shown in the previous chapter. This enabled the grid to keep listening to changed row signals and update the changes on to the grid without refreshing or reloading the state of the grid.

Once the message view page was developed since, the other view pages contain only a grid, the rest of the development was very swift and smooth. After all the view page

grids were developed, localization of the application was done next. For some of the web components Angular Translate was used as mentioned above. Since this translation tool is developed specifically for angular JS applications, it worked nicely with this web application. Localization of all the Kendo grids is done with a built-in feature for Kendo. Based on the chosen language by the user, Kendo enables the grid to change the language and formats of the grid accordingly.

The development process proceeds to deploying the application in to the test company's remote server after the localization was complete. It went very smoothly following the guidelines set by an employee in the case company. After deploying the application, the Azure ACB2C authentication was implemented on top of the web application. This enables the application to be accessed only through login by authorised users.

After completing these steps, the application was functional but left with some detailed fixes, additional functionalities and final touches to be implemented. These details and fixes will be discussed in the coming chapter.

5.3 CSA

At this point of the development the application is functional with all the grids and the SignalR real-time updates display working as intended. There are certain functionalities that has not been completely implemented yet and some that needs modification or improvements.

Currently the development team is working on the final view page which is the zones information page. This page processes the largest data amongst all the rest. The grid behaves slow and the updates fail to be displayed on time before a new update is received via the SignalR listener of the grid. The team is working on avoiding this heavy data traffic and improving the performance of this page at this point of time.

When this phase is complete the development team will continue to add detailed functionalities and minor fixes. Some of this fixes and functionalities are mentioned as follows. The first one is giving messages a name so that a message can be displayed by a short descriptive name to help users reduce time and effort wasted on identifying a message by reading the whole text. Another Additional implementation is finding a way to

save standard messages in to the database so that users can save and reuse messages that has been composed once. The final fix on the TODO list of the development team is to make a custom login page for the application. The Azure ADB2C authentication provides a login page by default. And all the applications developed by the case company use the default login page provided by Azure. This has made all applications login page to look similar with each other when users are logging in. To avoid confusions caused due to this implementation the development team will make a custom login page specific just for the web app with descriptive texts and logos.

The final part of the development as planned by the development team will be adding functionalities that has not been implemented in the desktop application but were planned to be implemented in the future. The major functionality planned to be added is restriction of cars and drivers from the system. Also, to implement a grid showing the restricted cars' and drivers' information. This requires designing the UI and planning the grid on the frontend and a working API from the backend.

By the time all the TODOs listed in this section are completely implemented in the web application the application will be ready for testing by another team in the case company. After testing the web application will be published and be ready for users to use it instead of the old desktop application.

6 Problems and Proposed solution

The aim of this thesis project is to build a web application that can substitute an old Windows Forms application. The old desktop application has been in use for the past ten years and still being used. Due to this, the users have been familiar with this application that, using this desktop application is almost natural. Since similar users will also be using the new web application, two main challenges arise in the development process. These problems will be elaborated, and the solutions proposed will be discussed in this chapter.

The first problem was keeping the old desktop applications feels and looks in the new web application. This was a bit challenging to achieve without making the interface of the web application boarding and unpleasant mainly because the desktop application was developed with older design concepts and guidelines which are outdated at this moment. Since the new web application will be used by new users who are more familiar with modern web applications and good-looking websites the new web application has to also satisfy the eyes of these users as well. Developing a UI for the website that can balance these two extreme user experiences was the first challenge throughout this thesis.

The second and most challenging problem solved while working on this thesis project was enabling the changing parts of the application to be displayed in real time without stressing the performance of the web application. The application makes sure users are informed about the status of every devices cars and drivers with minimum delay. The old desktop application integrates both the database changes or the backend and the frontend in one. Therefore, when an application is deployed in to a machine it would have control over both the frontend and the backed through the programs integrated in it. This has made real-time updated to be caught right with in the application. But in the web application the backend which is kept in a separate server application has to broadcast the updates and the web application will catch the updates and display then on the application interface.

There are several medium and methods to make this work on the web application, but the performance would be slowed down since there are numerous components in the application that needs to be updated in real time. Both challenges were solved in an effective way that the application at this moment is working as intended with a satisfying performance and a better-looking UI. The measures taken to overcome these challenges will be discussed thoroughly in the coming two paragraphs.

6.1 Design

As mentioned above the first challenge was deciding the looks of the application to satisfy the expectation of both current and new users. After several researches and discussions this was solved by keeping all the components of the old application's interface as similar as possible in the new web application. The colors of certain components are completely changed while some are replaced by a web safe but similar looking colors. Fonts and sizes of all the text components in the web application are changed to better and bigger fonts. It was aimed to not change the look of the application but just to make it look better and sharp.

Even though there are other elegant ways of implementing some features in the application, some compromises had to be done on the extent to use them. This is mainly because these implementations and tools could change the look of the web applications to have a big difference form the desktop application. After a lot of trials and experiments the web application has been developed to look as the improved version of the desktop application.

The figures below demonstrate this and responsive behaviors by showing the messages view page and one of the grids view pages, of both the desktop and the web application.

Figure 4 shows how the look and feel of the entire UI has changes on the new web application.

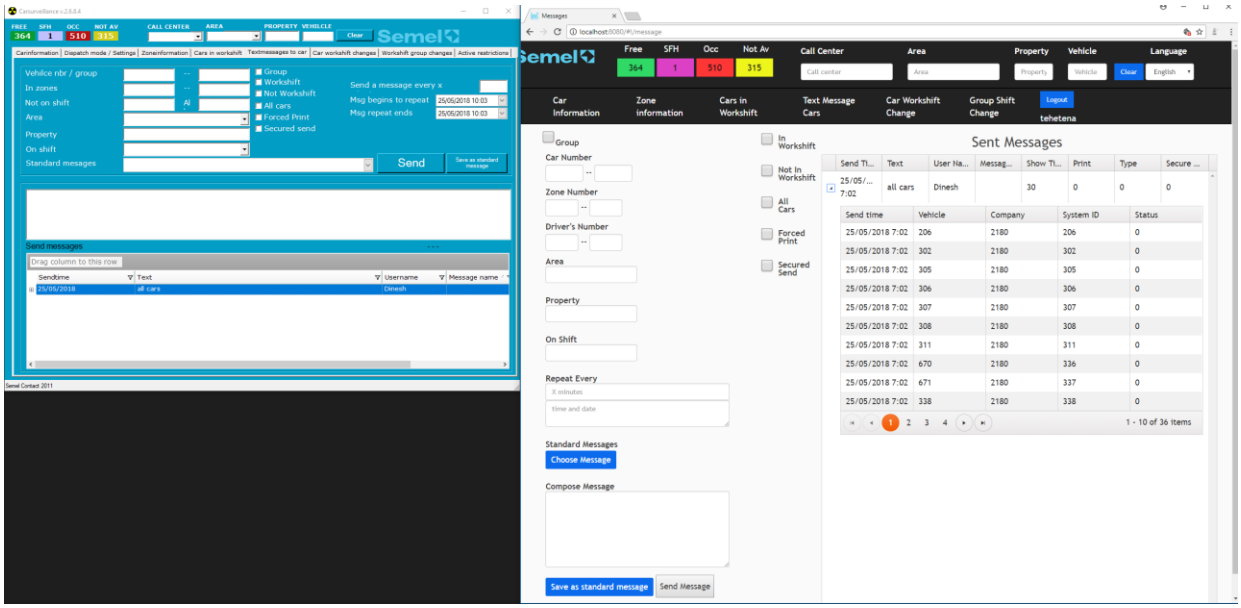

Figure 4 Messages view page of the Desktop and Web applications

Figure 5 illustrates the fixes that has been implemented on certain UI components of the old desktop application to improve the UI features and user experiences that were not sufficient in looks or usage before.

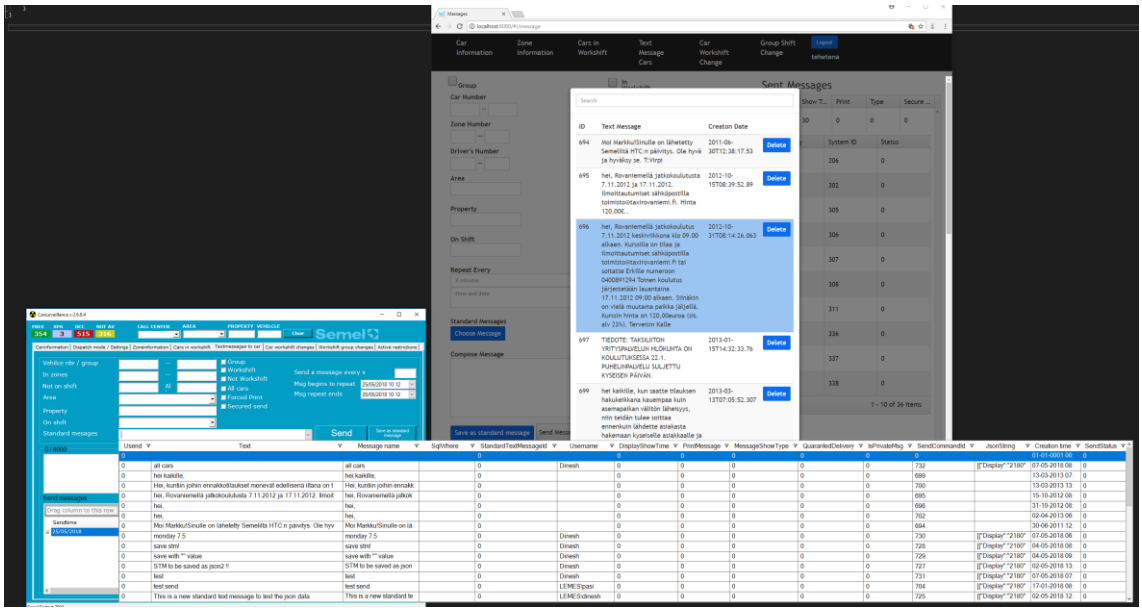

Figure 5 Standard message implimentations

Figure 6 shows the responsive behaviour of the grid that were not even in the question in the previous desktop application and also the integration of the forms on to the grid making the user experience very high and easy.

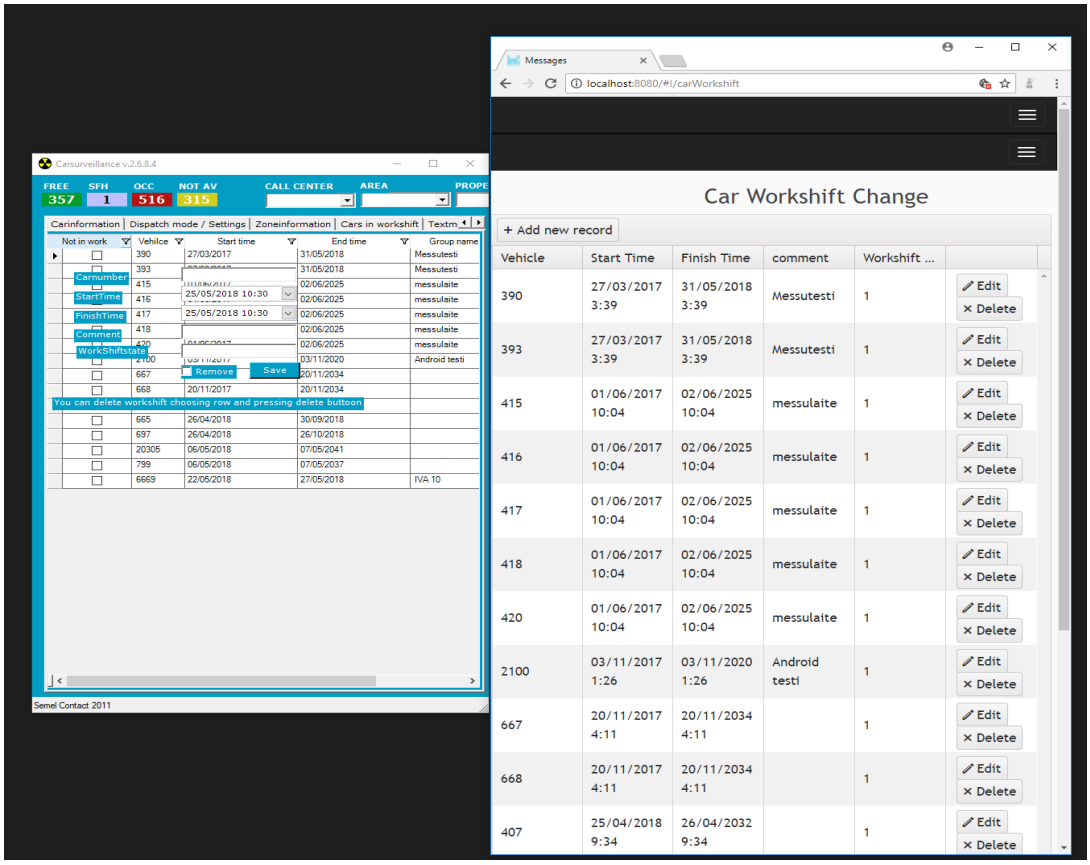

Figure 6 Responsive behaviours of both applications

As illustrated in all the above figures the web application has a different colour and font choices than the one from the desktop application but still tried to keep the appearance of the desktop application. This hopefully will help users to feel familiarised with the web application. The grids are also responsive and intuitive as shown in Figure 7. CRUD operations are implemented with external form on the desktop application while in the web application the operations are performed on the grid itself without any external forms needed.

Making the UI components fully functional using recent technologies was the next challenge that will be discussed in the coming section.

6.2 functionality

In this application almost, all the grids contain information that are changing every few seconds and these changes are very crucial in making this whole system work. Therefore, displaying all the changes as they happen to every authorized user should be implemented effectively and sharp.

As stated above, the team had to find a new way to update the grid and other components in real-time without sending get requests to the database again and again. This was possible to be done using one of the recent technologies of SignalR. SignalR basically reads through a database table and signals only the rows that has been changed with the new or updated data in them. The frequency of this signaling can be set by the developer as often as needed.

After the changed row's data are being signaled the client-side program has to compare the changed data and find them and update the changes. SignalR has saved the client side from sending requests and redrawing the UI to display it. But the one catch in using SignalR was the client side has to still iterate through the entire data and match the changed values and update it with the Signaled changes. This can slow at times and impossible depending on the size of the data and the frequency of the signals.

Luck for the team since the data are displayed on a grid to avoid the hit on the performance and make the updates visible with a minimum delay, a tool was needed to take care of this long and heavy process with built-in program. Kendo UI grid was the perfect solution for this issue. Kendo grid not only provides a grid with a good-looking interface but also supports SignalR. It provides and a way to read through all the signaled changes and update the rows in the grid without any noticeable delay.

The updated are being displayed every 5 seconds for most of the grid data but in some cases where, the data transported is heavy, the frequency is slowed to few seconds more to ease stress on the application and increase the performance. The kendo grids are also easily editable with high level of responsiveness.

7 Conclusion

This thesis project has successfully produced a functional web application to replace an old Windows Forms desktop application. The web application serves the same purpose as the old application but the interface and the entire logic to do it has been done in a different platform and language. It is aimed to help users of the old desktop application to use this projects application as a replacement without any confusion and having familiar feeling while using this application.

Latest and effective tools have been brought to use in the development of this application. It was also possible to make this application support real-time updates and data streams for certain parts of it.

This thesis project has let the author learn a great deal of things both in programming and teamwork aspects. The application was done without any provided base codes or any guidelines. Even though it gets confusing and tough to make decisions and take the development to a smooth phase it makes developers learn new things along every steps of the way. There are several logics learned in order to make the application work in general. Most of these logics have not been used or worked on before with the authors experience. There are also tools and frameworks that are used in this thesis project which are firs time experiences then but mastered ay the author now. The things learned through this thesis project are with no doubt important in future developments.

Two developers were involved in this project developments, including the author, who have to communicate well in order to reach a common understanding to decide the next step of the project. There were no rules or procedures on how to undertake these communications. But several communication tools have been involved. Git and VSO are the frequently used communication tools used. Experiencing to use those tools was an interesting part of the project. It has Helped improve communication skills and team work technique knowledge.

Finally, there are things that can be learned from this thesis project and certain recommendations that can be made. The first one is that in projects involving large grids to be processed and perform CRUD operations, it is a good idea to take Kendo UI grid systems into consideration since kendo has a built-in and simple support for all the above

operations and a lot more. Moreover, kendo also has a smart and fast way of handling signals sent via SignalR and displaying the changes on the grid.

References

1 Martin Löfberg. Patrik Molin. Web vs. Standalone Application. Sweden: Blekinge Institute of Technology: 2005.

2 Yaapa Hage. Express Web Application Development, Packt Publishing Ltd, ProQuest Ebook Central, 2013

3 Erik Brown. Windows Forms programing with C#. 209 Bruce Park Avenu Greenwich: Manning Publications: 2002.

4 W3School. History of the Web. USA: Oxford Brookes University: 2002

5 Microsoft Developer Network. Designing for web or Desktop? [Online]. 2018.

[link] https://msdn.microsoft.com/en-us/library/ms973831.aspx

6 Microsoft Docs. Develop Windows Doc Application [Online]. 2018.

[link] https://docs.microsoft.com/en-us/windows/desktop/choose-your-technology

7 Natalie Matkovska. WinForms vs. WPF [Online]. 2016.

[Link] https://www.linkedin.com/pulse/let-battle-begin-winforms-vs-wpf-natalie-matkovska?trk=portfolio_article-card_title

8 Rufus Vinci. AngularJS Web Application Development Blueprints. Olton: Packt Publishing: 2014.

9 Rai Rohit. Socket.io Real-time Web Application Development. Aprix publication: 2014

10 Eccher Clint. Professional Web Design: Techniques and Templates (CSS and XHTML). Boston: Course Technology: 2008.

11 Kaplan Dean. Web Designers Application Sketch Book.

12 Microsoft Developer Network. Introduction to SignalR. [Online]. 2016.

[Link] https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr

13 Ingebrigtsen Einar. SignalR. Real Time Application Development. Packt Publishing Ltd: 2013.

14 Pascal Precht. angular-translate. [Online]. 2016.

[Link] https://angular-translate.github.io/

15 Microsoft Developer Network. Azure Active Directory B2C Documentation. [Online]. 2018

[Link] https://docs.microsoft.com/en-us/azure/active-directory-b2c/

16 npm Community. http-server

[Link] https://www.npmjs.com/package/http-server

17 Semel. Official Website

[Link] http://semel-en.sivuviidakko.fi/

18 Kendo UI for jQuery Documentation and API references. [Online].

[Link]https://docs.telerik.com/kendo-ui/api/javascript/data/datasource/configuration/transport.signalr#transport.signalr

19 Ganatra Sagar. Kendo UI Cookbook. Packt Publishing Ltd: 2014.

20 Adams John. Learning Kendo UI Web Development, Packt Publishing Ltd, 2013.