Erhversakadami Sjælland

K19s Venner Member System

Supervisor: Michael Claudius

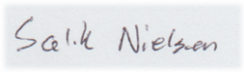
Name of education: AP Degree in Computer Science
Project period: 7th April - 10th June
Exam: 24th - 25th June

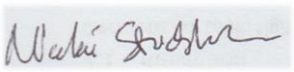
Nicolai Strudsholm
Salik Henrik Nielsen
Katarzyna Grabowska

10-06-2015

We hereby grant permission for this project report to be used by Erhvervsakadamiet Sjælland and permits the copying of this report as long as the content remains the same.

Signatures:





Salik Henrik Nielsen	*Katarzyna Grabowska*	*Nicolai Strudsholm*

Date: 10/06/2015

Title: K19's Venner Member System

Supervisor: Michael Claudius

Project period: 7th April – 10th June

Exam: 24th – 25th June

Keywords:

Dissertation, SharePoint Online, ASP.NET, Bootstrap, C#, Azure, SQL, Microsoft

Abstract:

This is a dissertation report for the AP Computer Science programme. It covers the development process of a provider-hosted application for SharePoint Online. The project is finished with help from agile software development methods such as SCRUM and Extreme programming. The system is developed in C# and ASP.NET with help from JavaScript, Ajax and Bootstrap.

Table of contents

1 Project Establishment

Written by: The team

1.1 Intro

We are students from Zealand Institute of Business and Technology and this is our final project for AP degree.

The point of the project was to prepare some kind of software for a real-life company. We decided to choose the company one of us interned it, as we were already familiar with how they work and what their needs are.

The purpose of this project is also to increase the knowledge in new technologies, which we have not worked with before. Adapting to new and perhaps better technologies is a key factor in this field of work. We'd like to find a technology which is relevant to the industry and adapt to it, document how we work with it and hopefully learn a lot about it.

The purpose of this project is also to increase the knowledge in new technologies, which we have not worked with before. Adapting to new and perhaps better technologies is a key factor in this field of work. We'd like to find a technology which is relevant to the industry and adapt to it, document how we work with it and hopefully learn a lot about it. finish with a working and ready-to-use product for the costumer and a thorough description of the process developing it.

This project should finish with a working and ready-to-use product for the costumer and a thorough description of the process developing it.

1.2 Plan for the project establishment

We did not have a specific plan for the establishment, but we spent the first week[1] on going into detail of what this project should contain, which methods we wanted to use and the tools to create this solution.

The two first meetings were used to create the foundation of the project establishment, and the details were discussed remotely within the team through Facebook, as there was no need for us to meet every

[1] Appendix – Meeting Reports (p. 7-8)

day. The idea we had was quite simple to understand and after the first general discussion, we did not have any doubts as to what to do and what our role is.

1.3 Description of project partner

1.3.1 The organization

Our customer is the organization K19's Venner. The organization supports a department of the FDF (Frivilligt Drenge- og Pige-forbund), called K19 Vanløse. They support the department through their subscription of their own members, giving work force to events, fundraising etc. The organization is independent from K19, so they can support the projects they want. Everyone can be a member of the organization for a yearly fee of 50DKK.

Their website is http://www.k19svenner.dk

1.3.2 Product owner

We don't work directly with the organization, but with one of their board members. His name is Meik Børner, and he is a treasurer of K19's Venner. He is the person we will show our product to during the process of development, and he will be giving us tasks, requirements and suggestions. Meik has an AP degree in Computer Science, so he can be a big value to our team with his knowledge.

Meik will always be available to contact, and he will set time off for when we need to see him and show our work. He will be referred to as “product owner” throughout this report.

1.4 Project description

The organization has a decent amount of members (around 80 members), and while increasing in size, the organization needs an IT solution that will work better than their current solution of handling members.

As of right now, the members are handled in Excel files with all their information. This requires a lot of work to keep files updated with members etc. A lot of problem arises when handling members in files. If a member changes email, address or phone number, the board members will have to change it manually.

At our first meeting with the product owner, he talked about all this and how he wants to improve this system. Overall, he wanted a member system of some sort with very few specific requirements. Meik has not supplied us with a project proposal, so while satisfying the few requirements; we are free to do what we want.

1.4.1 Requirements

- Product should be an app for SharePoint Online
  - See 'SharePoint Online' under Method, 'Technique and tools'
- Needs a database
  - Needs a well-designed database with member information etc.
- Everything should be in the cloud[2]
  - Product owner specializes in working in/with the cloud for his own company, so it is something he wants for this product as well.

[2] http://www.microsoft.com/enterprise/it-trends/cloud-computing/default.aspx#fbid=gt2BxNAWXIL

- Should work with computer, tablets and phones
    - The design of the app should work and look nice on every device.

1.5 Problem definition

1.5.1 Problem statement

Having few members in an organization is easy to maintain manually, but as the organization grows, so does the need for a better and more automatic system for administrating members. The team is needed to solve this problem with their expertise, since the solution requires knowledge within web development, software design and programming in general.

The customer is K19's Venner. The members of this organization and the board members needs this system to make the everyday usage easier for the whole organization. A board member, Meik Børner, is the product owner and the person who decides how the problem is fixed and when it's fixed.

While having limited time for this project, the team will strive to complete every requirement set by the customer. The project team will have its first meeting on April the 7th and the project will end on May the 29th. The time in-between will be used to fulfill the requirement set by the customer, which is a working IT system integrated to SharePoint Online.

1.5.2 The problem

- How can we develop an appropriate IT administration system to K19's Venner?

1.5.3 In depth questions

- Which development method is suitable?
- Which tools are proper for this project?
- Why choose a cloud solution instead of a server solution?
- How do we create an IT solution that's compatible on every device and browser?

1.5.4 How to solve and answer the problems

To answer these problems, the team has to increase their knowledge within web development. Web development is a very broad subject, and to further specify, what the team has to work with, go to 'Methods' and read the section 'Techniques and Tools'.

The team thinks that this is a manageable project, and their knowledge in C# is going to take them a long way into learning web development and ASP.NET. Knowledge about Microsoft products such as SharePoint Online and Azure can be acquired from Microsoft's Virtual Academy[3] and from the product owner.

As for the development method for this project, the team has been taught different methods at the education, which makes them capable of choosing a method for this project.

[3] http://www.microsoftvirtualacademy.com/

1.6 Project scope

The development of the product will take around 7-8 weeks of the project and this includes the project establishment and the sprints.

The development of the product includes writing the program, testing and react to feedback from the product owner. While having a limited time, it's hard to know if we can finish every requirement.

We want to present the limited time to the product owner, and we will ensure that we get the core functions of the product, so we can focus on them and get them thoroughly tested. Depending on the severity on the core functions, we will try to have the lesser important functions developed with a good standard and tested a lot.

We expect that we will be using a lot of time writing documentation and report besides developing the product, so we will not be making a guide in form of a manual for the product owner because of the tight time schedule.

According to browser statistics[4], a high percentage of people are using Google Chrome, so our main focus is to make this program work in Google Chrome. We will try to make it fully compatible for other browsers, if time allows it.

1.7 The Group

Our group consists of 3 young talented people: Nicolai Strudsholm, Salik Nielsen and Katarzyna Grabowska (Kasia). We are eager to do as much as we can to provide our customer with a fully working, useful and easy to use product.

1.7.1 Strength and weaknesses

- Our strengths: we are hardworking, talented and ready to put everything we have into this project. We do not have trouble communicating, we discuss every part and every problem we encounter and like spending time on the project.
- Our weaknesses: we are still young, inexperienced and not fully aware of the mistakes we make so we need to ask our supervisor and the customer to give us guidelines in how to work.

Name	+	-
Nicolai	Good at programming, cares about the project	
Salik	Skilled at databases, coding	Not good in GUI design
Kasia	Good sense of esthetics, design skills	Absence, not a good programmer

1.7.2 Role Distribution

Name	Role	Contact Info

[4] (http://www.w3schools.com/browsers/browsers_stats.asp

Nicolai Strudsholm	Programmer	nicolai.xs@gmail.com
Salik Nielsen	Database, programmer	saliknielsen@yahoo.dk
Katarzyna Grabowska	Designer, Report	kwmgrabowska@gmail.com

1.7.3 Risks

- Working with same documents (Word Online)
    - We would like to avoid a situation where two or more members of our team edit the same thing in the report. We are using Word Online which allows to edit the document both online and offline, depending on preferences. We will be trying to discuss the changes everyone made so that we do not get confused about the contents of our project report.
- Working on same code (VSO)
    - Similar to the previous point, but the solution is to share the code a few times a day, so that everyone is up to date with all the changes. This method is called Continuous Integration.
- Absence
    - This risk is unavoidable since we are all just humans and we get sick or have other important things to do besides the meetings. We will however try to be present as much as possible, and in the event of absence, we will contact each other to get updated.
- Losing files
    - Losing files will always be a risk when working with computers. The solution is to have everything online and often save the changes we make.

1.7.4 Conclusion of the team

The individual team members have their own strengths and weaknesses. The team members strengths compensates for other team members weaknesses, so on paper this looks like a group that will work well together.

The team members has not worked together on a project before, so it's hard to say how the process will be. We have around the same expectations for the project, and we all seem focused on finishing with a good product and report.

1.8 Method, Techniques and tools

The second meeting and in the end of the project establishment phase, we decided on which methods, techniques and tools we wanted to use for the first sprint.[5]

As written in our meeting reports, we all decided to work with the SCRUM methodology and for the first sprint, we wanted to try out some agile practices and see how they fit into the work in the group.

For a further detailed discussion of our chosen software development methodology and its practices, see the section 'Method - Development methods'.

Tools we will use:

[5] Appendix – Meeting Reports (p. 8)

- Visual Studio
    - We will be using Visual Studio to develop our solution. Visual Studio also has an Online feature (Visual Studio Online) which we also will be using a lot in our project.
- Office365
    - Since our project has to be on SharePoint Online (a part of Office365), we decided on using Microsoft's Office365 products as a part in creating this project. The Office365 products offers Word(writing report, notes etc.), SharePoint Online(using as a team intranet for keeping all our files) and Excel(charts). Word and Excel both have online versions, so we can always work on our project wherever we are. SharePoint Online is a cloud based solution, so every document is in the cloud and secured with back-ups.
- Skype/Facebook
    - We will use these for communication within the team. If it's necessary to share the desktop screen of your laptop, we will use Skype but generally we will communicate through Facebook.
- Microsoft Azure
    - This tool offers an SQL service and offers to host our apps on 'web apps' within Azure.

For a more thorough description of the tools, go to 'Method - Techniques and tools'.

1.9 Project plan

Our first week started at the 7th of April. We will hand in the report at the 10th of June.

	Week	1	2	3	4	5	6	7	8	9	10
Phase / Sprint											
Establishment											
Sprint 1											
Sprint 2											
Sprint 3											
Report											
Hand in											

1.10 Evaluation of the project establishment

We didn't fully understand the concept of making a project establishment. When we did the actual project establishment, we weren't thinking about it being in the report but that it should only work as guidelines for the group.

Our conclusion of this phase is that we didn't do it thorough enough because of lacking knowledge about this part of the project. We had to write it in the report during the first sprint, and it moved some of our focus to this phase instead of us having 100% focus on the first sprint. We had a few notes for the different points of the project establishment, but we still missed some we didn't think of during our "establishment" meetings. The most important thing we missed was identifying risk and how to deal with them.

After creating the project establishment, we saw that we missed some opportunities to increase the quality of the project, but generally, it didn't hurt the quality of the product.

Creating a project establishment has given all members of the team a good idea of the project and its guidelines. Moving into the project, we can always go back and see what our plan with this project was so we can keep the project inside its scope and follow the thread of consistency.

2 Method

Written by: The Team

2.1 Development methods[6]

2.1.1 Intro

When it comes to types of development methods according to Wikipedia we have a few:

- Waterfall development - requires finishing one phase (Requirements, Design, Implementation, Verification, Maintenance) before starting the next one, may result in time delays due to the necessity of going back to fix problems from previous phases.

- Iterative and incremental development (Unified Process)[7] - splits the development process into iterations (increments), every one of which delivers a part of the project thanks to team co-operation. Additionally, the Unified Process groups iterations into 5 phases: inception, elaboration, construction, and transition. The main advantage of this process is the fact that the team working on a project is able to detect and solve problems early, preventing wrong assumptions that could ultimately lead to catastrophic results.

- Agile development - based on iterative development, a flexible methodology where the whole project evolves gradually as every part of it is being developed. It includes many teams that co-operate with each other to deliver a final version of the software. Agile development is similar to iterative development, but it is lighter and more focused on the people who work on the project, instead of the project itself.

- Others - such as Spiral Development, Prototyping, Rapid application development etc.

2.1.2 Elements of the development method

There are many elements in Agile development, but we use only a few of them, such as:

- Extreme programming
- SCRUM meetings
- Some parts of other elements such as Kanban and Agile modeling

For a further description of practices, go to 'Sprint Zero' and the section 'Agile Practices'.

[6] http://en.wikipedia.org/wiki/Software_development_process
[7] http://en.wikipedia.org/wiki/Iterative_and_incremental_development

2.1.3 Discussion

We had a discussion between the team members as to which development process to choose, but it did not take long as we all quickly agreed on Agile development.

2.1.4 Final decision

Agile development was the quickest and most suitable way to develop our project. During the meetings the customer told us which parts should be fixed or changed, and Agile model gave us a chance to go back and do it. If we chose Waterfall model, we would not be able to go back so easily, because the project would be finished in a bigger part, or maybe even as a whole.
Using parts of Agile development such as SCRUM meetings allowed us to discuss every aspect among the team and with the customer, avoiding developing faulty parts of the project.

2.2 Techniques and tools

2.2.1 MVC architecture[8]

Model-view-controller is an architecture we used for our code to make it clearer and more readable for us and other people who are going to view it. This kind of technique consists of 3 components: controller, responsible for sending commands to model, which stores the data it received. It then sends it to view, which presents that data in a way understandable to the user.

We also have a data access layer, which works with the controller layer. It is much like the work between the logical tier and a data tier in a three-tier architecture9, where the logical tier requests data, and the data tier Query this from the database.

We use a Data Access Object file (DAO), which is the one querying the database and returning data to controller.

2.2.2 ASP.NET[10]

ASP.NET is a framework used for web development. It is server-side, so it's different from e.g. JavaScript, which is client-side. When creating an ASP.NET web application, you are presented with a markup file and a code-behind file. The idea is to create markup that interacts with the code-behind, which is a powerful tool to have especially if you already now C#.

In our code-behind we use C#, and the markup has different events (Button events) that interacts with the code and fire methods.

ASP.NET is made by Microsoft, so it already have good interactions with SharePoint Online. Therefore it is an obvious choice for us to make, because we already now C#, and C# can work with ASP.NET.

The choice stood between PHP and ASP.NET for our server-side scripting. We went with ASP.NET because the code looks familiar and it interacts better with SharePoint Online.

When ASP.NET has to talk between client and server, it required a post back. The post back is like a small refresh (F5), and when developing in ASP.NET, the code reloads when a post back happens losing

[8] http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
[9] http://en.wikipedia.org/wiki/Multitier_architecture#Three-tier_architecture
[10] http://en.wikipedia.org/wiki/ASP.NET

data stored in variables etc. This is usually fixed by checking in the code if a post back is happening or not. We also have the opportunity to use Ajax - explained below.

2.2.3 JavaScript, JQuery and Ajax[11]

JavaScript is a dynamic programming language for web development, which is supported by many web browsers. It is mostly used as a client side, so it can interacts with the user, communicate asynchronously and content that is displayed. It can also be used as a server side, which we don't use.

JQuery is a fast, small JavaScript library. JQuery is used for handle HTML content, event handler, animation and Ajax.

Our main reason for using JavaScript is to avoid post back. Post back refresh the page, and that can be confusing for the user. Therefore, we decided to use Ajax to communicate with the server. Ajax can send JSON, and therefore will work almost the same as ASP .NET.

2.2.4 Bootstrap[12]

Bootstrap is a free open-source HTML, CSS and JavaScript framework.

It is easy to learn, good documentation, short code lines and great browser support/compatibility. Basic touchscreen friendly component with modern design like dropdown button, panels, tables, etc.

Bootstrap have 12-column grid system for responsive design. You decide how much column each component should have.

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1
span 4			span 4			span 4					
span 4			span 8								
span 6						span 6					
span 12											

There are 4 screen sizes that are supported. Large(larger desktops), medium(desktops), small(tablet) and extra small(phones). You decide how many columns each component should have for every screen size.

If a table has span 6 then it will fill half of the screen in width.

2.2.5 SharePoint Online[13]

SharePoint Online(SPO) is a product by Microsoft which is a part of their Office365. It works like an intranet, where your company/organization/etc. can have a private website with many different tools (apps) to use if you want. An app could be a document library, a calendar or something third. In SPO an

[11] http://en.wikipedia.org/wiki/JavaScript and https://jquery.com/

[12] http://www.quora.com/What-are-the-pros-and-cons-of-using-Bootstrap-in-web-development

[13] https://products.office.com/en-us/sharepoint/sharepoint-online-collaboration-software

app is a very broad term. Many big companies use SharePoint Online, and they often use it for having an intranet for their departments, where they can upload document, see news from the department etc.

SPO allows implementing your own apps, which opens up for customization and independent developers to earn money through developing apps for companies. SPO has a special framework when developing, so there are limits to what you can do.

It is a requirement by the product owner, that the system is running on SharePoint Online, so there are no other alternatives for us for choosing a platform for our app. But to mention a few, we could have developed an app for Android and/or IOS that could do the same.

2.2.6 Visual Studio Online[14]

Visual Studio Online is a cloud service to share software code within the developers. It works with many IDE, like Visual Studio, Eclipse and others. Visual Studio Online supports any language like C# and Java. It have version control, so that you can get your old code back, if you need them. Visual Studio Online also have tool for agile, so you can track your user stories and backlogs.

It is free to use if you are five member or lower. You have to pay per month if you want more members.

The reason we are using Visual Studio Online is continuous integration, so we can share are code between each other.

2.2.7 Microsoft Azure[15]

Microsoft Azure is a cloud computing platform and infrastructure, for compute, storage data, networking and app. It support any operating systems, language, tool and frameworks. It means that you can store C#, Java or something else, and many kinds of databases. We are storing ASP .NET, C# and SQL Server.

You have to pay every month to use it, but you do not have to maintain anything. Microsoft make that everything is backed up in another server, so you do not lose anything.

2.2.8 iTextSharp[16]

iTextSharp is .NET PDF library, to create, adapt, inspect and maintain pdf document.

We are using this because it is free to use and to create invoices.

3 Sprint Zero

Written by: The team

3.1 Intro

We decided on calling this phase the Sprint Zero, because this is what we did before actually starting on the first sprint. Coming into this sprint, we wish to plan a lot of what we are going to do and how we are

[14] https://www.visualstudio.com/en-us/products/what-is-visual-studio-online-vs.aspx

[15] http://azure.microsoft.com/en-gb/overview/what-is-azure/

[16] http://sourceforge.net/projects/itextsharp/

going to do it. We want to discuss the thoughts the individually team member has about the project and any concerns they might have.

3.2 User stories

At the meeting with the customer, we sat down and talked about the specific functions the system should have. We worked out a list of functions, which the team should transform into a list of User Stories and then get acceptance from the customer to start the development.

The functions[17] were written by the customer after a thorough discussion of the individual functions so we were sure that we were on the same page of what the system should do.

3.3 Prioritizing User stories

At the meeting with the customer, we decided to ask him right away about prioritizing the functions, so we knew what to focus on first.

The team and the product owner decided not to assign business value to the individual User Stories, because the business is not going to profit from this system and a lot of the functions are "nice to have" functions which are not 100% required.

The functions that are not fully required we will still aim at completing, but the ones that are actual requirements, we will focus on first.

The functions listed at the number 1[18] are functions that needs to be in the system as a minimum. The higher the number, the lower prioritization it has in the system.

3.4 Product Backlog

After meeting with the customer and after rewriting requirements into User Stories, we got the following backlog:

	User stories
1	As a member, I can login **Conditions**: Needs to check if user is an administrator or normal user
2	As an administrator, I can create members **Conditions**: Member has signed up and paid the fee
3	As a member, I can edit my data **Conditions**: Member can only edit some data.
4	As an admin, I can edit members
5	As an admin, I can delete/archive members **Conditions**: Member resigns from the organization, member has not paid membership after reminders
6	As a member and/or administrator, I can upload documents **Conditions**: Member has his own document library (SharePoint function)
7	As an administrator, I can upload documents concerning members
8	As an administrator, I can create an invoice and start a subscription
9	The system needs to automatically send out invoices when membership runs out
10	As an administrator, I want to send out reminders, if a member is missing a payment

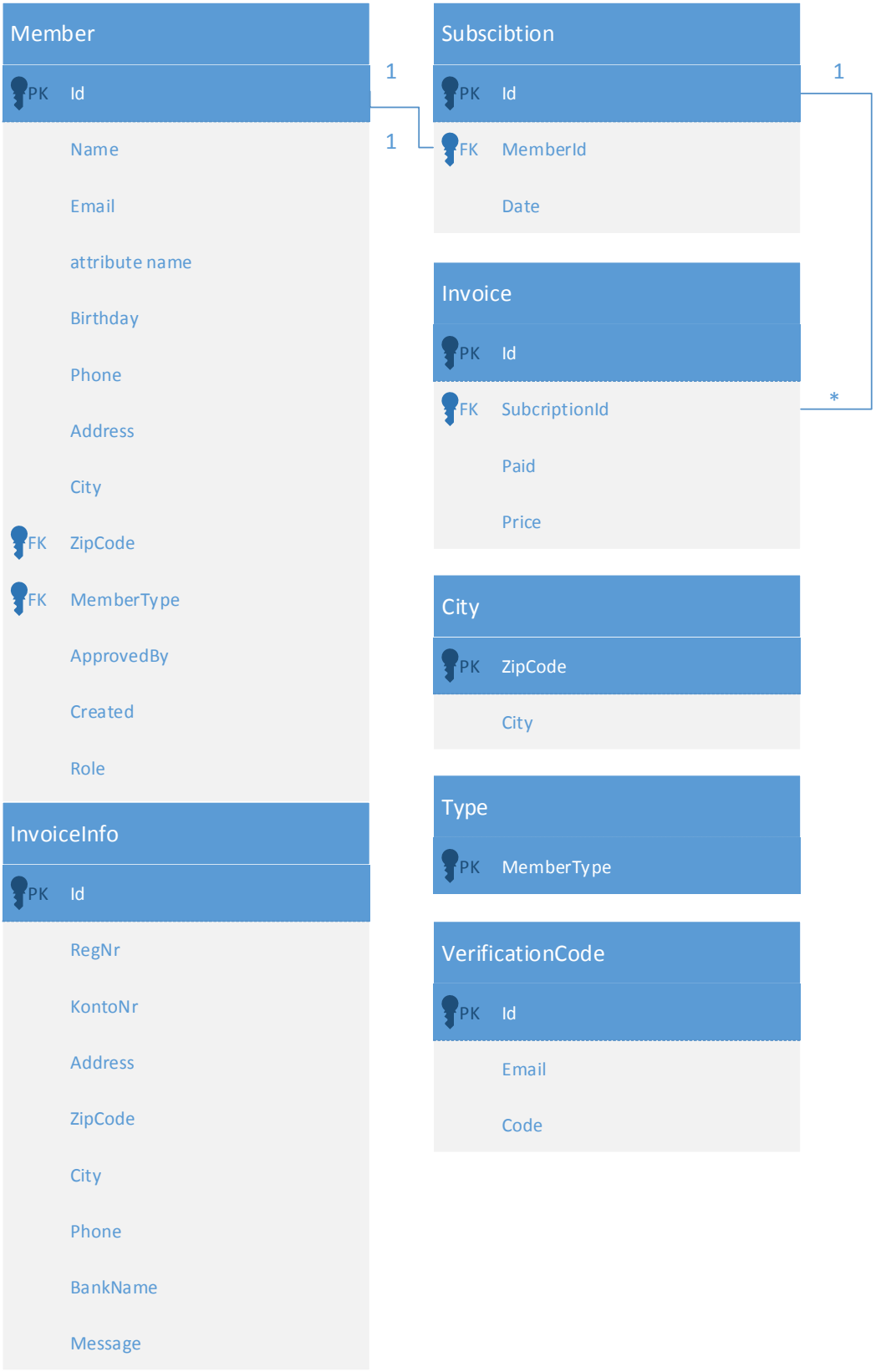
[17] Appendix – First Meeting with costumer (p. 22)
[18] Appendix – First Meeting with costumer (p. 22)

11	As a non-member, I want to sign up as member
12	As an admin, I want to approve/decline signed up members
13	The system needs to send changes toMailChimp, if a member edits his e-mail address.
(14)	As an admin, I can search for members
(15)	As a member, I can see a contact list of the board members

The conditions in some of the User Stories were further requirements set by the product owner when he described his requirements.

User Stories in parentheses are User Stories which were added during the sprints. Those User Stories were oversights by the team and the product owner, and they were discussed during Sprint Review Meetings and added into the backlog.

3.5 Database design



3.6 Agile Practices

We are not going to use all the agile practices but only a few of them, some of which may be dropped later as we go due to not being as useful as we supposed. As of now, we are going to use:

- Pair programming
    - Programming in pairs, allowing us to code faster and with a use of all the knowledge we have, discussing the code as we write it.
- Planning game
    - A good way to start a project. Writing out all User Stories we need with their priority and difficulty. During this part we also estimate our Velocity.
- Whole team
    - Every member of our team is giving something to the project, discussion is our main focus. We try to complete each other and the gaps in knowledge each of us has.
- Continuous integration
    - We can save our time by synchronizing the code very often, which will prevent us from developing errors and mistakes.
- Small releases
    - We will meet with our customer after every sprint, also keeping in contact during the sprints so that they are always updated on the project. Small releases allow us to find mistakes and misunderstandings early on, and to fix them before they go too far.
- Collective code ownership
    - Every team member is allowed to write/change the code. What is more, they are encouraged to do it every so often.
- Test-driven development
    - As we started, we thought it would be a useful pracitce for us, but it turned out later it was only holding us back.
- Backlogs
    - In Sprint backlogs we can present what we did during each sprint. It lets us review the progress we make in relation to time we have left
- Scrum events
    - Mostly Scrum meetings, reviewing what we did before and what we are going to do every time we meet, discussing ideas and our input in the project.
- User story
    - One of the most important parts of our project, without User Stories we would not be able to develop anything as we would not know what we are supposed to do.
- Retrospective
    - Looking back at every Sprint to estimate how it went. We can see which practices and methods were useful to us and how fast we are. We can also estimate if we are going to deliver the project on time.
- Velocity tracking
    - A part of agile development that allows us to be up to date with the progress we make during the project.

3.7 Estimating velocity

Web Development mixed with Software Development is a new to all in the group, so it's impossible to know how fast we will work, the problems we will face and how big an individual User Story actually is.

We looked at the project as a whole, and decided that we wanted to try and finish the product and all of its User Stories as it seems that we could manage it.

After estimating Story Points for the individual User Stories, we found the sum of estimated points and divided it with the amount of sprints we wanted.

Since the Product Owner is busy in his work time, we didn't want to do weekly sprints but decided on a sprint length of 2 weeks.

The sum of estimated Story Points came to 64, which is about 21-22 story points each sprint.

3.7.1 Estimating story points

For each User Story we had to estimate an amount of story points that we would give the User Story. While being inexperienced with web development, we chose a scale from 1-10, where 1 is the easiest and 10 is the hardest.

The team had a meeting where we started by discussing each User Story. We gave each other our suggestions and knowledge, and we ended up wanting to find the easiest User Story and compare the others to that. So for the easiest one that gave 2 points, the 4 points stories should be twice as hard, and 10 points should be 5 times as hard.

Throughout the meeting, we overall agreed on every estimation for each User Story. The story points can be found at the intro for each sprint, where we discuss which User Stories we are doing for that specific sprint.

3.8 Quality Factors

After having the first meeting with the product owner, we didn't directly talk about quality factors, but he mentioned requirements which we fulfill with focus on some quality factors. Below there is a list of quality factors, their importance, and what we will do to achieve a quality system:

3.8.1 Scalability - High

One of the most important factors in having a member system is scalability. The system should be able to scale according to the amount of users using the system. We will focus on the product owner being able to scale up and down according to what he needs.

Scalability in this system is achieved through Azure and its SQL service. The system architecture and coding quality should also keep up with the amount of users using the system, but Microsoft's SharePoint Online features also solve some of those problems.

3.8.2 Usability - High

As mentioned in the requirements, we want the system to be used on every device that can use an internet browser. This is a quality factor we will strive hard to focus on.

We also want to keep the UI simple and good. Minimizing effort to complete jobs such as approving members, edit data, handle users etc.

3.8.3 Reliability - High

While this is a very important quality factor in a member system, we rely on Microsoft's uptime of their products, both SharePoint Online and Azure. We will of course try to make our system fail proof so that it will not crash but continue to work even though errors may happen.

3.8.4 Flexibility - High

Our system should be flexibility, both while we work with it and when we deliver the final product to the product owner. There is a chance, that we can't finish every single requirement by the customer, so it should be easy to modify our system with changes without it all breaking down.

This is achieved through a good system architecture, well documented and well managed code.

3.8.5 Maintainability - High

As written in flexibility, our code should be well documented, so it's easy to maintain if an error should arise. This is especially important with Microsoft products, since they will roll out updates to their products, which we heavily rely on. An update on SharePoint Online can break some functionality, so this is also an important factor we need to focus on.

3.8.6 Security - Medium

Our system needs to be secure for every type of member there is. SharePoint Online will handle some security, but we still need to focus a bit on security ourselves so that people will not gain access to parts of the system, they shouldn't have access to.

Being new to ASP.net, we will have to gain knowledge on how to properly secure web applications.

3.8.7 Testability - Medium

The system will be thoroughly tested through the development phase, and at each sprint review, the product owner will run his acceptance tests.

We are not so experienced in ASP.net, but we will try to focus a fair bit on testing, because it increases the quality of the product for future use. Testability also helps maintainability and flexibility.

A logging system will be our goal to increase the quality of this factor.

3.8.8 Platform Compatibility and Portability - Low

This is something we will not put effort in doing. SharePoint Online will do most of the work, but we will of course make sure that the system will look the same in different browser types and devices.

3.9 Evaluation of Sprint Zero

After finishing our Sprint Zero, we feel confident going into Sprint 1. This 'sprint' has given a lot of guidelines about how we want to work in the everyday on this project, and it has given us a decent estimation of what we at least should expect to finish when we are done with the last sprint.

We will use this sprint in the evaluation of sprint 1, to see if we are on the right track when choosing agile practices and estimation of story points. This will be further evaluated in the Sprint Retrospective of sprint 1.

4 Sprint 1

4.1 Intro

Written by: Kasia

Sprint 1 started on April 14th and continued throughout the next two weeks, ending with a sprint review meeting with the costumer and a sprint evaluation by the team.

We will start to shape the product with the right architecture and setting up everything such as SQL database and Visual Studio Online and afterwards we will start developing the user stories for this sprint.

4.2 Sprint Backlog

Written by: The team

Our backlog for this sprint is the following User Stories:

	User stories
1	As a member, I can login Conditions: Needs to check if user is an administrator or normal user Story Point: 3 **Tasks:** Code: Salik and Nicolai Database: Salik
2	As an administrator, I can create members Conditions: Member has signed up and paid the fee Story Point: 4 **Tasks:** Design: Kasia Code: Salik and Nicolai Database: Salik
3	As a member, I can edit my data Conditions: Member can only edit some data. Story Point: 5 **Tasks:** Design: Kasia Code: Salik and Nicolai Database: Salik
4	As an admin, I can edit members Story Point: 5 **Tasks:** Design: Kasia Code: Salik and Nicolai Database: Salik
	As an admin, I can delete/archive members

5	Conditions: Member resigns from the organization, member has not paid membership after reminders Story Point: 3 **Tasks:** Design: Kasia Code: Salik and Nicolai Database: Salik

We picked the User Stories this sprint according to the prioritizing the Product Owner gave us. These User Stories gives the sum of 21 estimated Story Points, and as explained in Sprint Zero, we need to aim at around 21-22 Story Points each sprint, so this fits perfectly.

4.2.1 Dividing User Stories into tasks

Written by: The team

For this sprint, we wanted to divide the user stories into tasks, so everyone would work on every user story. Kasia will mainly work in the design of the functionalities. Salik and Nicolai will both work on the code while occasionally use pair programming if it's needed. Salik will work on the database, work on setting it up, keeping it updated and connect the functionality with the database through SQL commands.

- Kasia will mainly work in the program Axure while setting up design. Axure helps to set up the look while also generating the HTML and CSS.
- Salik will work with Microsoft Azure and also spend some time learning how to work with it.
- Nicolai will work with SharePoint Online and make sure the program is integrated when we meet with the customer for the first sprint review meeting.
- All team members will work with Visual Studio and Visual Studio Online for developing the product and testing it.

4.3 User Stories

Written by: The team

A brief description of each User Story. We will not go through all the code for each User Story, because it will become redundant talking about the same functionalities. We will go through the most relevant code for each User Story. If the code is too big, it will be moved it to the appendix.

During the next sprints, we might want to change something about the User Stories from previous sprints. This could be based on finding better solutions or costumer feedback. We will write this in a form of outlook; what we could have made better, if we had the time.

4.3.1 User Story 1

Written by: Nicolai

One of the first requirements of a membersystem is a login before you can access the system. The login helps make the user unique and with help from sessions, he will browse the system from his point of view and be able to make changes to his account etc.

We don't have to create the logic for a login because the user will have to login through a Microsoft account to access the SharePoint site. Since our system is an app and is inside SharePoint, we will need a way to authenticate that login with the SQL database, so it can give him access to the membersystem.

We have chosen to create a way to authenticate him with the help from SharePoint ClientContext class. [19]That class represents the context of SharePoint objects and operations. This includes the member that is currently logged in.

The system should authenticate him through email, since the username for a Microsoft account is his email (and it's unique). The database needs a unique email field, and then it is easy to authenticate him:

```
1   var spContext = SharePointContextProvider.Current.GetSharePointContext(Context);
2
3   using (var clientContext = spContext.CreateUserClientContextForSPHost())
4   {
5      Session.Add("clientContext", clientContext);
6      clientContext.Load(clientContext.Web);
7      clientContext.ExecuteQuery();
8      clientContext.Load(clientContext.Web.CurrentUser);
9      clientContext.ExecuteQuery();
10      email = clientContext.Web.CurrentUser.Email;
11      Session.Add("UserEmail", email);
12
13   }
14
15   String memberType = _memberController.GetMemberType(email);
```

Microsoft does some standard authentications, which is already added when you create a SharePoint App in Visual Studio. This authentication happens from line 1-9.

On line 10 we requests the email from the current logged in user and save it in a field. We add this email into a session with the id "UserEmail" on line 11.

On line 15, there is an example of how we use the email to authenticate which membertype the current logged in member is.

And then we can do basic operations such as giving him access to parts of the system, if the member is an administrator.

An example would be our front page menu. There are icons displayed for normal users and those icons are also displayed for administrators together with all their functions only for them.

```
if (memberType == "Medlem" || memberType == "Bestyrelse")
```

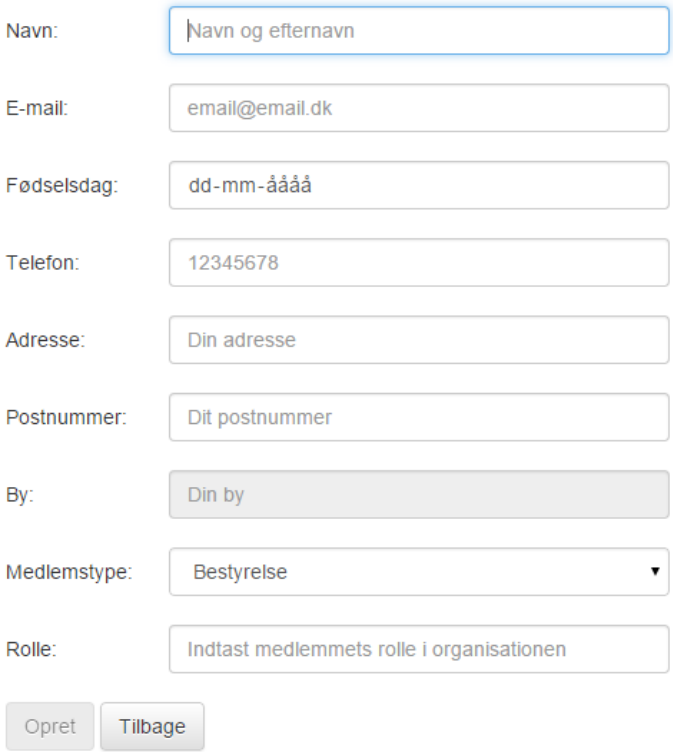
[19] https://msdn.microsoft.com/library/office/microsoft.sharepoint.client.clientcontext%28v=office.15%29.aspx.

```
{
//Dynamically create icons for members and administrators
 }
if (memberType == "Bestyrelse")
{
//Dynamically create icons for administrators
 }
else
{
//User is not in the system/User is archived/User is deactivated.
 //Display Error message
}
```

We authenticate through email and membertype. He can still be in the system, but if he is archived or not activated yet, he will still not have access to the system.

4.3.2 User story 2

Written by: Salik



The admin can add a new member. If the new user is “Bestyrelse”, they can add a role, so other members can see what he does in the company. When all input is filled out, the “Opret” button can be pressed. Every time the user write something in the input, CheckInput will be call.

```
function CheckInputs() {
   var name = $("#Name").val();
   var email = $("#Email").val();
   var birthday = $("#Birthday").val();
   var phone = $("#Phone").val();
   var address = $("#Address").val();
   var zipCode = $("#ZipCode").val();
   var city = $("#City").val();

   if (name.length === 0 || email.length === 0 || birthday.length === 0 || phone.length
=== 0 || address.length === 0 || zipCode.length === 0 || zipCode === "By ikke fundet" ||
city.length === 0) {
       $("#Submit").prop('disabled', true);
   } else {
       $("#Submit").prop('disabled', false);
   }
}
```

When the user is finished, a model of the member will be created in the controller, and will be passed to the database. The add member method from dao (“data access object”-class) is in appendix[20].

We are using SqlConnection to connect to the database, and SqlCommand to read and write.

```
private static string sqlString = @"Data Source=qjauqbgpqz.database.windows.net;
Initial Catalog=membersystem; Integrated Security=False;
User ID=membersystem; Password=pass@word1;Connect Timeout=60;
Encrypt=False; TrustServerCertificate=False";
private readonly SqlConnection _sqlConnection;

private DAO()
{
    _sqlConnection = new SqlConnection(sqlString);
}
```

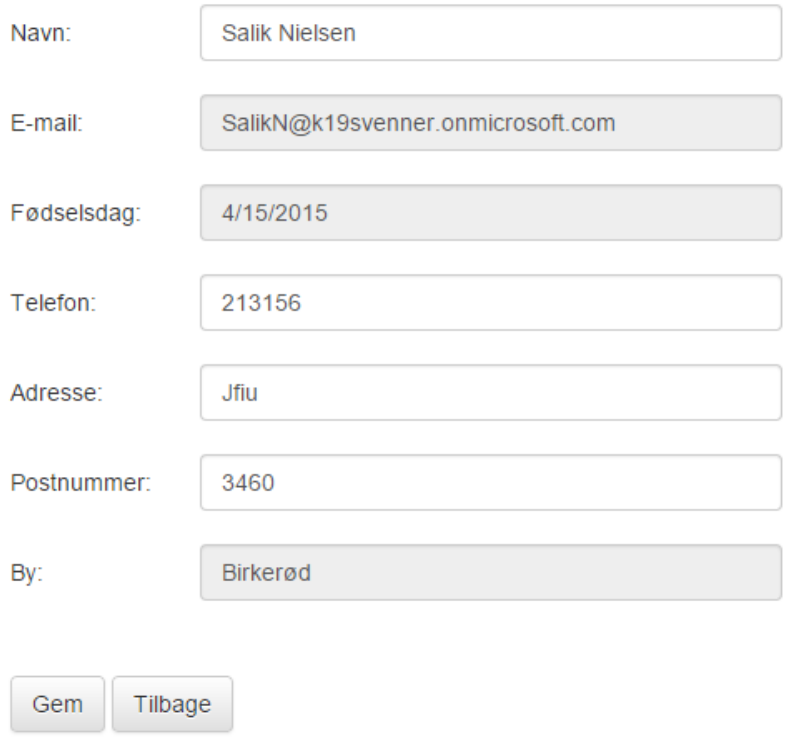
4.3.2.1 Outlook

The company wants the birthday to be three input for day, month and year instead of date picker. When the user has been created, a visible alert should be shown. Bootstrap alerts can be used to notify. The three input should be displayed as dropdowns, so people can’t be typing in wrong dates e.g. if it was a textbox. The picture below is an example of how it should be implemented:

[20] Appendix – User story 2 (P.2)

4.3.3 User story 3

Written by: Salik



If a user wants to edit their information, they will be shown their current information. E-mail and birthday is not editable because the e-mail is associated with SharePoint login and your birthday does not change. If the user wants to change e-mail or birthday, they will have to contact the company.

The user cannot write in the city input, because K19s Venner wants zip code input to auto check city. When the have typed 4 digits, it will automatically check if the city exist. If it does exist, then the city will be shown in the city input. Our first solution was with ASP .NET postback. Every time the method is call, the page will “refresh”, and it can be confusing to the user when that happens. To avoid that, we decided to use Ajax call.

Here can we see how we are using Ajax:

Client side:

```
$.ajax({
    type: "POST",
    url: "EditProfile.aspx/ZipCode_OnTextChanged",
    data: "{'zipCode':'" + zipCode + "'}",
    contentType: "application/json; charset=utf-8",
    success: function (data, textStatus, jqXHR) {
        if (data.d === "City not found") {
            $("#City").val("By ikke fundet");
            $("#Submit").prop('disabled', true);
        } else {
            $("#City").val(data.d);
            CheckInputs();
        }
    }
});
```

Zip code will be sent to the server and the city will be returned.

Server side:

```
[WebMethod]
0 references | Nicolai Strudsholm +1 | 2 changes
public static string ZipCode_OnTextChanged(string zipCode)
{
    string city = _memberController.GetCity(zipCode);
    return city.TrimEnd();
}
```

The method has to be static and declared as a WebMethod[21]. The database table 'City' is from Post Danmark[22].

4.3.3.1 Outlook

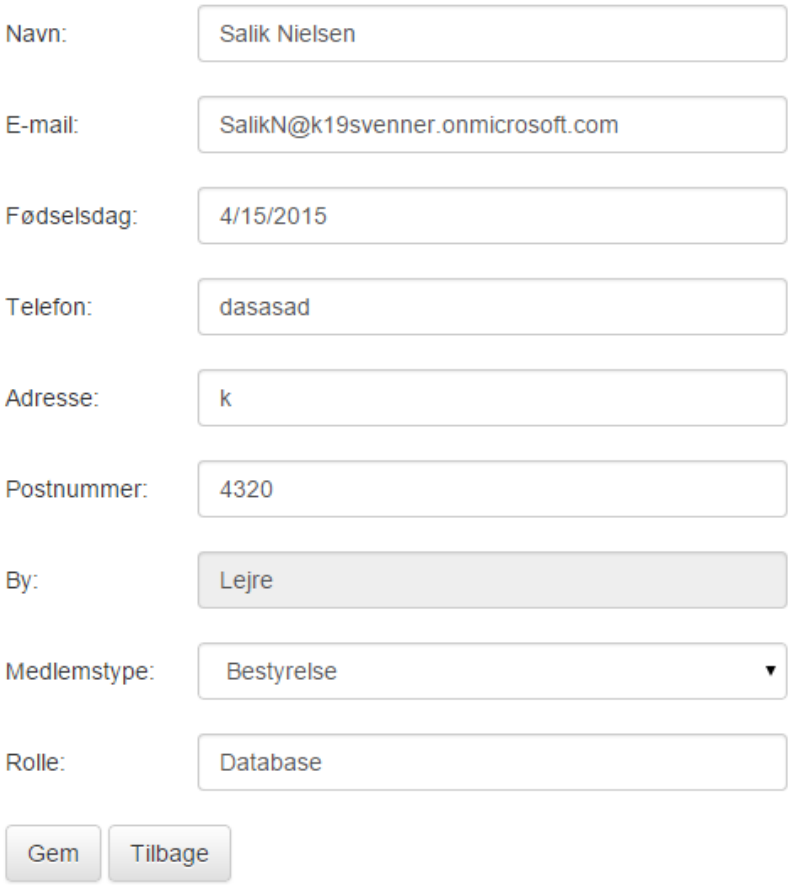
As in user story 2, the company wants 3 input for birthday. We could have used Bootstrap's alert with green background so it is easier to see that the system writing something at the screen.

[21] http://www.aspsnippets.com/Articles/Calling-ASPNet-WebMethod-using-jQuery-AJAX.aspx

[22] http://www.postdanmark.dk/da/Documents/Lister/regionsopdelt-postnummer-excel.xls

4.3.4 User story 4

Written by: Salik



The user story 4 is almost the same as user story 3. But user story 4 is meant to be used by admins, so they can edit a member's e-mail, birthday and member type. If the member type is change to "Bestyrelse", they can add role to the member, because they will be shown in "Bestyrelsen" page.

This page can only be accessed if the admin have search for the member and then clicked the member to edit their info.

We are using the same page as user story 3. The page load checks if the admin is about to edit a member's info or if a member is editing themselves.

We check the HTTP query string variables. If it is empty, then it means that a member is about to edit their own info. If it not empty, then it checks if you have the right to edit other members info.

```
string edit = Request.QueryString["edit"];

                if (edit != null)
                {
                    if ((string)Session["MemberType"] == "Bestyrelse" &&
edit.Equals("member"))
                    {
                        string email = Request.QueryString["email"];
                        _memberInfo = _memberController.GetMember(email);
```

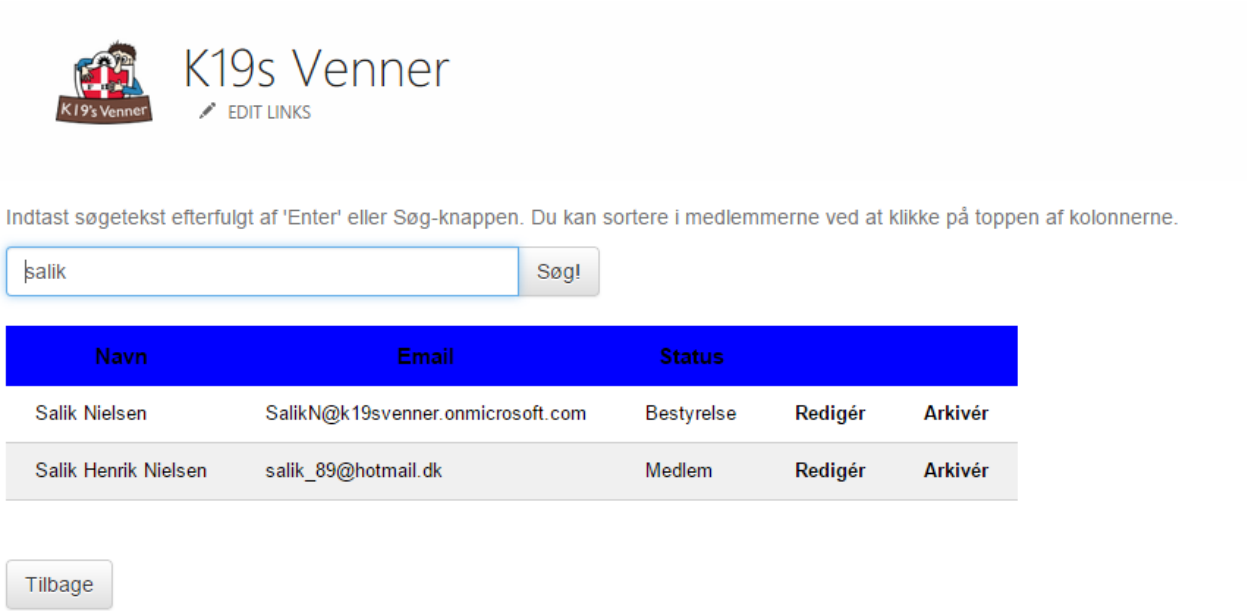
```
            Email.ReadOnly = false;
            Birthday.ReadOnly = false;
            MemberType.Enabled = true;
            MemberType.Visible = true;
            MemberTypeText.Visible = true;
            Position.Visible = true;
            PositionText.Visible = true;
        }
    }
    else
    {
        string email = (string)Session["UserEmail"];
        _memberInfo = _memberController.GetMember(email);
    }
```

4.3.4.1 Outlook

The same as user story 3, because they use the same page (editmember.aspx).

4.3.5 User story 5

Written by: Salik



The admin has to search for members by their names, if they want to archive members. There are two options to archive members. The first option is in search result. Here they can press "Arkivér" to archive member. Using JavaScript, a pop up message will ask, if you want to archive the member. As you can see in the code down below, we are using query string[23] to pass the member's e-mail.

```
<script type="text/javascript">
    function Confirm(email, status) {
        var confirmValue = document.createElement("INPUT");
        confirmValue.type = "hidden";
        confirmValue.name = "confirm_value";
```

[23] http://en.wikipedia.org/wiki/Query_string

```
        if (confirm(status + " " + email + "?")) {
            confirmValue.value = "Yes";
            window.location.href = "ArchiveConfirm.aspx?email=" + email;
        } else {
            confirmValue.value = "No";
        }
        document.forms[0].appendChild(confirmValue);
    }
```

4.3.5.1 </script>

When we pass the e-mail, we will redirected to ArchiveConfirm.aspx. It will tell you that the member is now archived and you will be taken back to search member after 3 seconds.

Here you can see how we get the e-mail from query string.

```
string email = Request.QueryString["email"];
```

The second option to go in edit member page, by pressing "Redigér" button. In the edit member page, there is a dropdown button to change the member type and select archive.

4.3.5.2 Outlook

The admin have to remember the members name to archive a member. We could have included "Redigér" and "Arkivér" buttons from search page to member list page.

4.3.6 User Story 14

Written by: Nicolai

This User Story was added by the team because of an oversight from everyone including the team and product owner.

We wanted to use this functionality to work with other functions such as Archive User and Edit Profile / Edit Member.

Instead of having a search for each those individual functions, we had one search function where there would be different links to go to the function needed.

We used the ASP.NET tool called "GridView"[24] which is a Grid where you can create a view from a data source. We have an SQL database as data source, and we create a view based on the things we want displayed.

The markup for the GridView does most of the job of filling the Grid with information:[25]

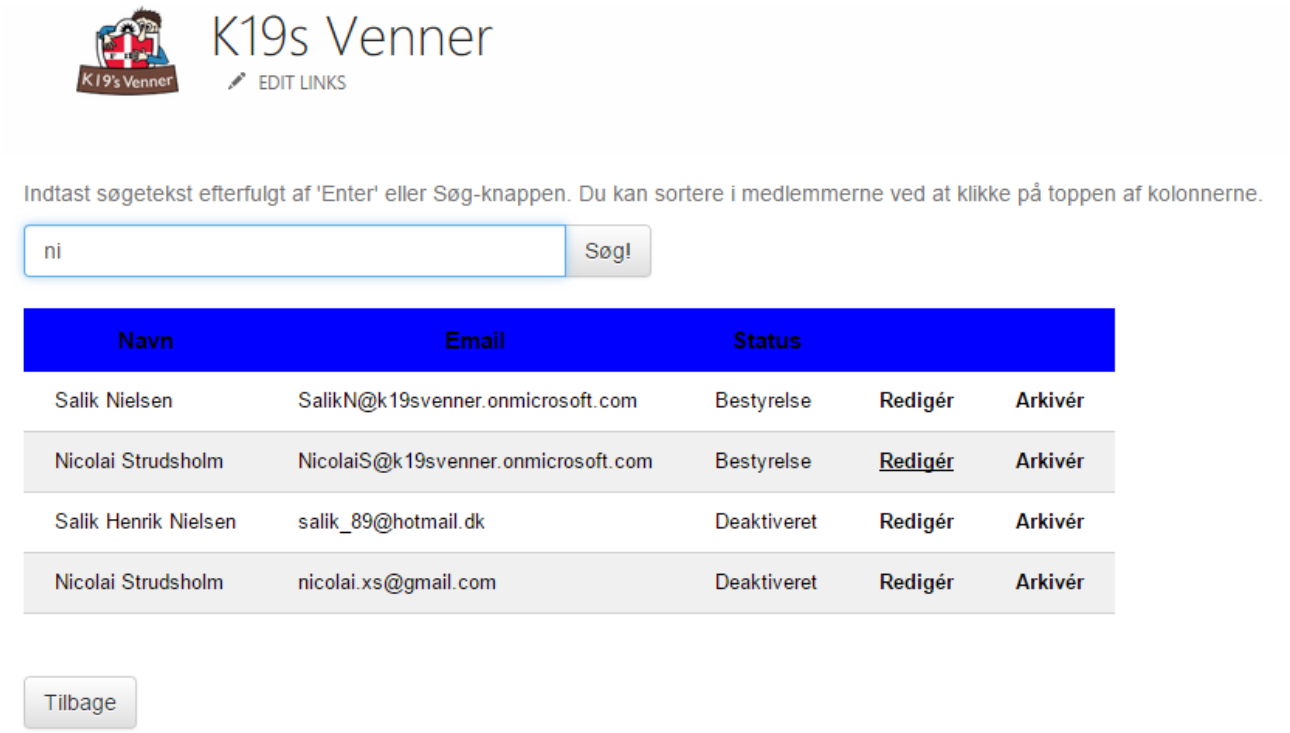
The BoundFields are fields bound to some kind of data. Then we have a HyperLinkField, which is not a bound field, and we can fill in what we want.

The "SqlDataSource", marked with yellow, is where it will connect to the database with id and password. It also creates the view in the "SelectCommand" attribute.

[24] https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview%28v=vs.110%29.aspx
[25] Appendix – User Story 14

The “ControlParameter”, marked with green, is the parameters it asks for in the “SelectCommand” attributes. So ‘Name’ from the SQL database has to contain what is inside the “MemberNameTextBox”.

The GridView in action:



You can sort the columns (ascending and descending) by clicking the header of each column.

“Redigér” and “Arkivér” fields are added dynamically. The GridView has different event handlers attached to it, and when we want something added dynamically, it should be added after the “DataBound” event, which is every time it binds data to the GridView.

```
        protected void GridView1_OnRowDataBound(object sender, GridViewRowEventArgs e)
        {
            if (e.Row.RowType == DataControlRowType.DataRow)
            {
                string status = "Arkivér";
                if (e.Row.Cells[2].Text == "Arkiveret")
                    status = "Åben";
                HtmlAnchor popupLink = new HtmlAnchor() { HRef = "#", InnerText = status
};
                popupLink.Attributes.Add("onclick", "Confirm(\'" + e.Row.Cells[1].Text +
"\', \'" + status + "\')");
                e.Row.Cells[4].Controls.Add(popupLink);
            }
        }
```

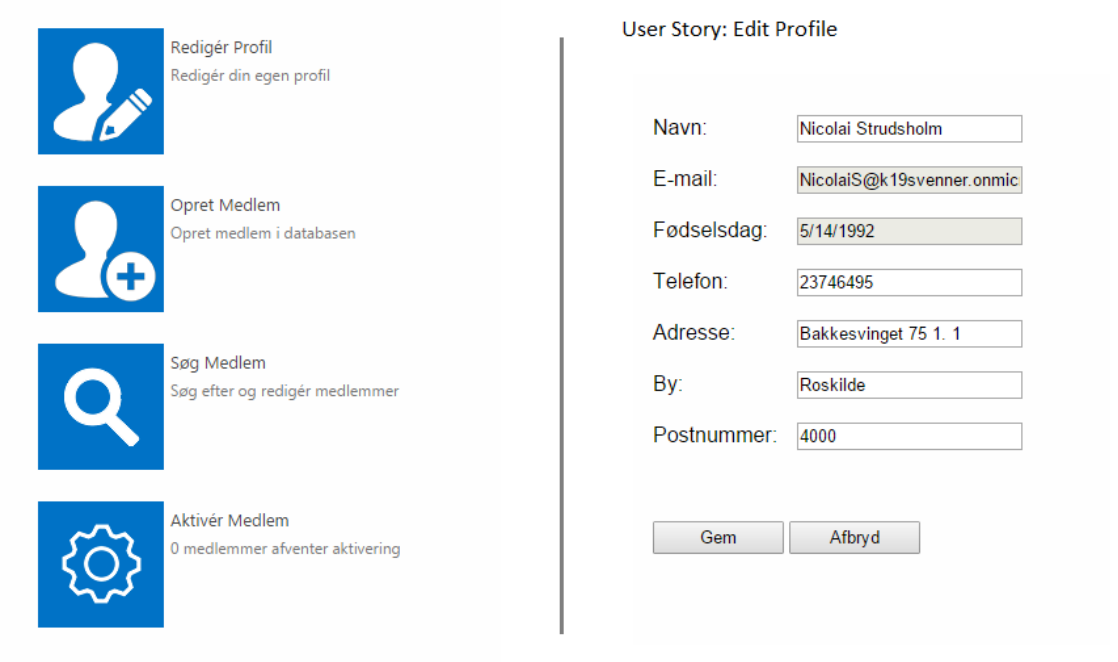
It will insert into each row in 5th cell every time the DataBound is fired.

4.4 Evaluation of Sprint 1

Written by: The team

We decided on taking a screenshot of the system's front page and a screenshot of a User Story/function to show any progress from sprint to sprint.

On the left side the front page is displayed. On the right side we have a function displayed, in this case the Edit Profile User Story.



4.4.1 Sprint Review Meeting

Written by: Salik and Nicolai

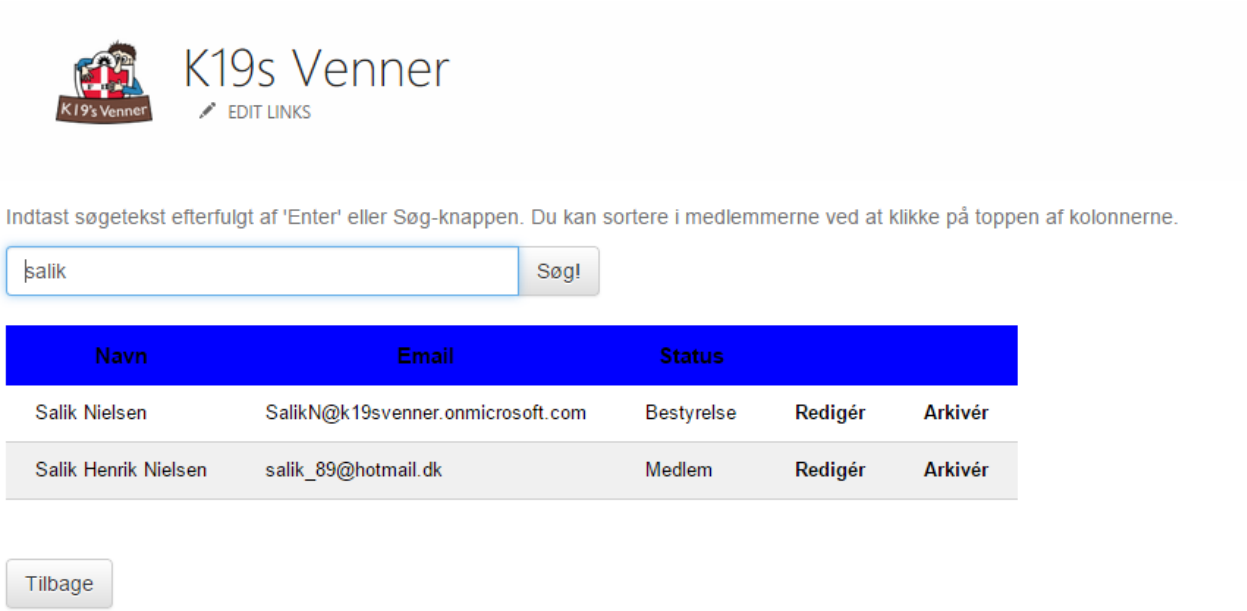
The meeting started out by showing the product owner the User Stories, which we made ready for release.

We talked a lot about the future functions and about the project. The product owner gave us a lot of tips to the product and to our project. He talked about some practices that we should use, as the practices would give him more insight in the project when we meet for the next review, and he can give us better feedback that way.

He liked the way the product started out, and we are planning to start doing the things he mentions, but it's something we will discuss in the sprint retrospective.

When searching for a member, it just showed a list of text with a member on each line of text. He wanted a better solution for this, where it shows a table and where you can click the different columns to sort in the list of found members.

It should be looking something like this, where you can click on "Navn" or "Email" and it will sort ascending or descending based on how many times you click the header.



4.4.2 Sprint Retrospective

Written by: The team

Start doing:

- Database design
    - We need to have a design of the database that explains how we set up the SQL database. The customer wants to see this, and it will benefit the group having this in our project.
- Sprint backlog
    - For the next sprint review, the customer would like to see a visual presentation of the sprint backlog. We could create a SCRUM board online somehow.

Stop doing:

- Test-driven Development
    - We're all very new to web development in ASP.net, so it's too difficult to write the tests before we even begin to code. We want to stop the test-driven development, because it will take a lot of our time and we feel that to write tests before you even code, you need to be experienced in the framework, you work in.

Continue doing:

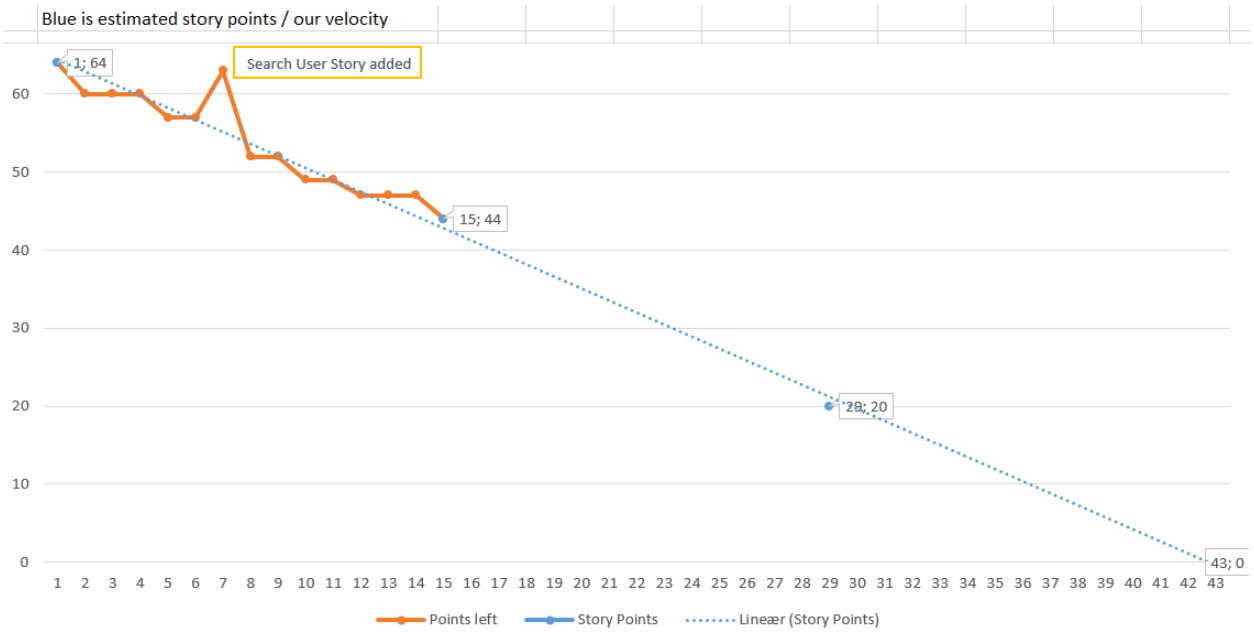
- SCRUM meetings
    - SCRUM meetings that are documented work as a good documentation for what we did every single day we meet. When writing the report, we can look back and see what we did each day, what problems we had and how we solved them.
- Pair programming
    - We will sometime do some pair programming during the day, and it works very well to get a good understanding of the code for everyone.
- Continuous integration
    - Sharing the code multiple times during the day means there are a minimum of errors in working on the same code. Visual Studio Online helps us fix the problems on working on the same code and helps merge our code multiple times during the day.
- Velocity tracking
    - Keeping track of the velocity in form of a burn down chart helps the team if they will need to speed up, take on more tasks for the sprint or work at the same current speed. In sprint 1, we made some tasks for a User Story in sprint 2, because we could see that we would come out ahead at the finish of sprint 1.
- SCRUM in general
    - We are happy to keep working with SCRUM and most of the practices it contains. It helps to keep us motivated and give a structure on the project. Dividing the User Story into tasks also helps us dividing the workload and giving everyone an opportunity to work on the project.

4.4.3 Velocity tracking

Written by: The team

In the middle of our sprint, we had an extra User Story added that both the product owner and we did not think about. That added some extra work that required us to work a bit overtime, but when we first managed that, the User Stories fit perfectly into our Velocity.

We don't see any need to make a new estimation of our velocity.



5 Sprint 2

5.1 Intro

Written by: Kasia

We will try to repeat the success from Sprint 1, but the User Stories will be more difficult this sprint, so it's difficult to tell if we'll have time for everything.

We'll start out by fixing all the errors we found at the sprint review meeting, and the meeting also had a lot of feedback from the costumer we will try to react upon.

5.2 Sprint Backlog

Written by: The Team

Our backlog for this sprint is the following User Stories:

	User stories
6	As a member and/or administrator, I can upload documents Conditions: Member has his own document library (SharePoint function) Story Point: 2 **Tasks:** SharePoint configuration: Nicolai
7	As an administrator, I can upload documents concerning members Story Point: 2 **Tasks:** SharePoint configuration: Nicolai
8	As an administrator, I can create an invoice and start a subscription Story Point: 4 **Tasks:** Code: Salik and Nicolai Database: Salik
11	As a non-member, I want to sign up as member Story Point: 5 **Tasks:** Code: Salik and Nicolai Database: Salik
12	As an admin, I want to approve/decline signed up members Story Point: 3 **Tasks:** Code: Salik Database: Salik
13	The system needs to send changes to MailChimp, if a member edits his e-mail address. Story Point: 8 Tasks: Code: Salik and Nicolai Database: Salik

Our Story Points for this sprint has a sum of 24 points. This is a bit higher than our previous sprint, and a bit higher than our estimation for each sprint. This is mainly because we have no idea how to create User Story 13, so we are not sure if we even will try to implement that function in the system. The customer has made it clear, that this function is nowhere near a requirement, and it's a "nice to have"

function. If this is not possible for us to do, it's okay to remove it from the backlog and do lesser story points for this sprint.

5.3 Dividing User Stories into tasks

Written by: The team

For this sprint, Kasia will be finishing her internship in Poland, which leaves the sprint for Salik and Nicolai to finish. Kasia will try to take part in group meetings as much as possible, through Skype and Facebook.

We have decided to implement Bootstrap this sprint, because our designer is absent for around two weeks. The Bootstrap will help us not worrying about using a lot of time for designing so we can focus on the development of the functions.

We (Nicolai and Salik) will work a lot on pair programming this sprint, since last sprint was a success for us both to work together on the code. User Story 6+7 and User Story 12 will be developed separately at home to make up for the time that we lose for only being two members present this sprint.

5.4 User Stories

5.4.1 User Story 6

Written by: Nicolai

Since the membersystem should be a part of SharePoint Online (SPO), we wanted to make some functions that could work with the SPO functions such as uploading documents on the website.

We discussed this with the costumer, and how we should complete this User Story. He wanted the system to create a Document Library automatically when a member is created in the system or accepted into the system. This Document Library should have permissions set to administrators and the 'owner', which is the user the library, was created for.

To achieve this, we need to use the ClientContext class (as described in User Story 1). Using the ClientContext, we can access and use other SharePoint classes. In this case, we use the ones to create a new Document Library.

```
        public static string CreateMemberList(string email, string name, ClientContext
cc)
        {
            try
            {
                Web web = cc.Web;
                cc.Load(web);
                cc.ExecuteQuery();

                ListCreationInformation lci = new ListCreationInformation();
                lci.Title = name;
                lci.TemplateType = (int) ListTemplateType.DocumentLibrary;
                List list = web.Lists.Add(lci);

                list.Description = name + "'s dokumenter. Email: " + email;
                list.Update();
                cc.ExecuteQuery();
```

```
            web.Update();
            cc.ExecuteQuery();
            return "success";
        }
        catch (Exception ex)
        {
            return ex.Message;
        }
    }
```

The “Load” method of the ClientContext is used to request something to use. In this case we need to use the Website, so we initialize it and request it. “ExecuteQuery” is used to execute the requests you make, which could be changes to the website or new things that you want to add. This is the way to work when developing in SPO.

ListCreateInformation is a class with information as fields used to create a list. There are minimum requirements in this class’ fields that needs to be filled. We need to give it a name and a template type.

The CreateMemberList method takes a name as parameter, and that is the name of the member together with his User ID to make it 100% unique and avoid people with same names to have same libraries.

5.4.1.1 Uploading documents automatically

As an addition to this User Story, we wanted it to upload documents automatically if something important needs to be saved. This includes the invoices that is created in a later User Story. We had to go back to this User Story at the time we created the User Story 8, to make the upload-function.

We had a little extra time on our hands because we devoted a whole sprint for User Story 8, so it was possible to make this without it disturbing the project. The following function we implemented:

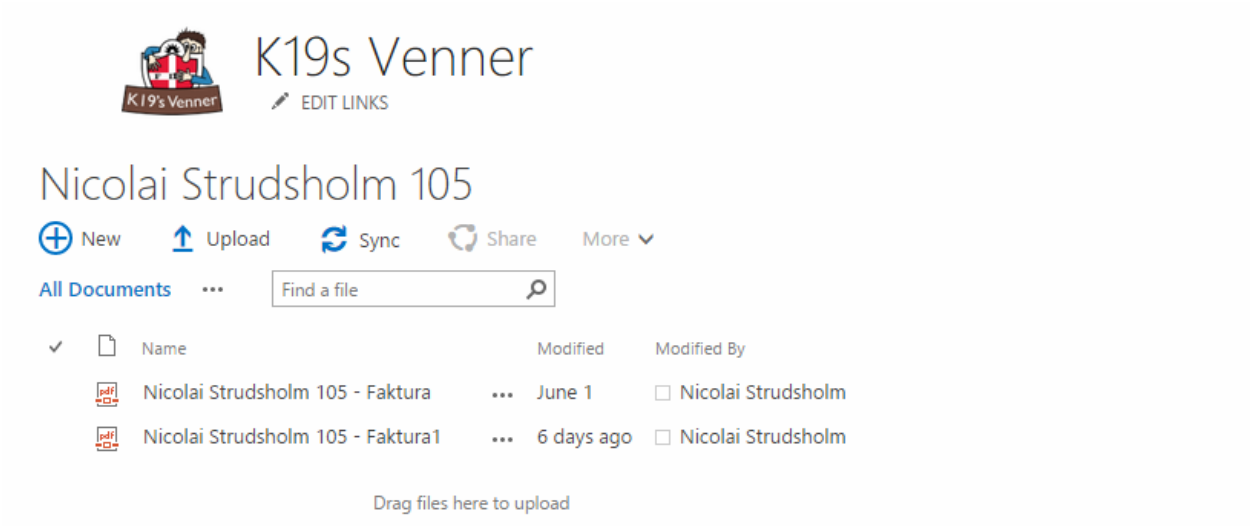
```
    public static string UploadInvoice(string filepath, string listName,
ClientContext cc)
    {

            try
            {
                Web web = cc.Web;
                cc.Load(web);
                cc.ExecuteQuery();

                List list = web.Lists.GetByTitle(listName);
                cc.Load(list);
                cc.ExecuteQuery();
                FileCreationInformation fci = new FileCreationInformation();
                fci.Content = System.IO.File.ReadAllBytes(filepath);
                fci.Url = listName + " - Faktura.pdf";
                fci.Overwrite = true;
                cc.Load(list.RootFolder);
                cc.ExecuteQuery();
                cc.Load(list.RootFolder.Files);
                cc.ExecuteQuery();
                int itemsCount = list.RootFolder.Files.Count;
                if (itemsCount > 0)
                {
                    fci.Url = listName + " - Faktura"+itemsCount+".pdf";
```

```
                }
                list.RootFolder.Files.Add(fci);
                cc.ExecuteQuery();
                return "success";
            }
            catch (Exception ex)
            {
                return ex.Message;

            }

        }
```

We get the list by its title after loading the Web class. The name is the automatically created list from before. We use a “FileCreateInformation” class just like the one for List creations. When creating an invoice, it is uploaded to the server on Azure where the website is. We want to delete this file after using it, because we don’t want to make it waste space on the website server. We take the path of that file and copy it into a SharePoint File. We upload the document to the library, and if the library contains more than one file, we modify the name, adding a number which makes it unique and not overwriting previous files. The number we get from the list.Count method, so the number will be “1,2,3,4” etc.

Site Contents:

Inside the Document Library:



5.4.1.2 Outlook

SharePoint Online doesn’t like ’ÆØÅ’, and since the organization is Danish, this is crucial for every function to include the Danish letters. When creating a Document Library including those, it simply

removes it from the URL and the automatically functions we have for uploading will not work, because they will include the Danish letter, and not find the list they need.

Our product owner's name is 'Meik Børner', and when we tested his name in the system, it made a list with the URL containing the list name as 'Meik Brner'.

5.4.2 User Story 7

Written by: Nicolai

This User Story doesn't need an thorough explanation, since it's pretty much the same functionality as User Story 6.

When creating this User Story in our backlog, we had a misunderstanding with the customer. We thought that the administrators should have a "secret" Document Library for each member, where they could upload documents regarding this member.

What he actually meant was that they both, administrators and the Document Library owner, should have access to the library.

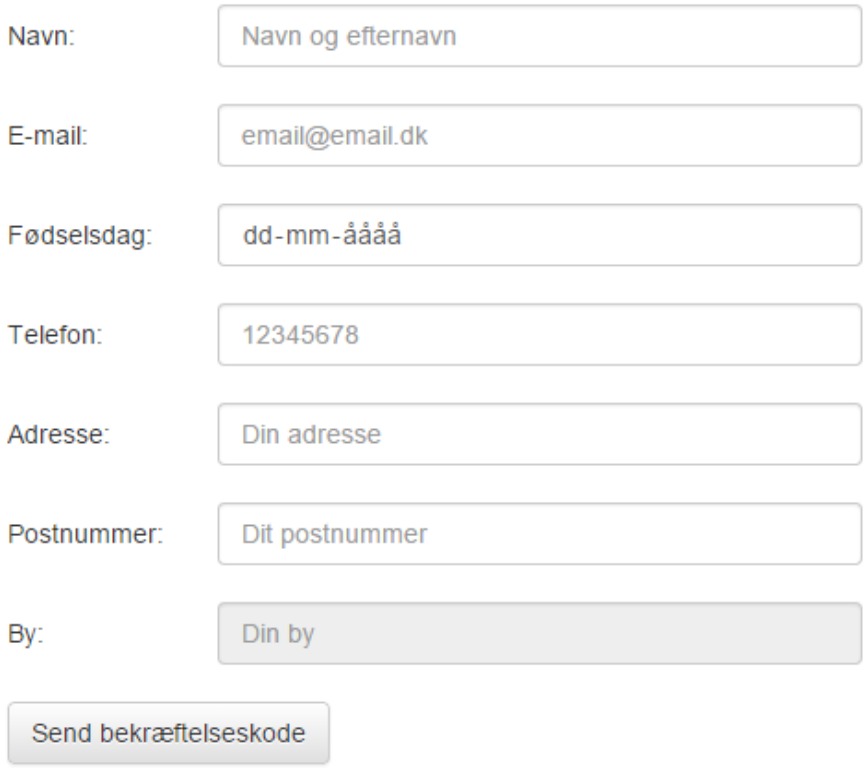
This function we actually spent time on and implemented it, but as the product owner said at the sprint review, this was not necessary, and it was later removed completely from the project.

5.4.3 User story 11

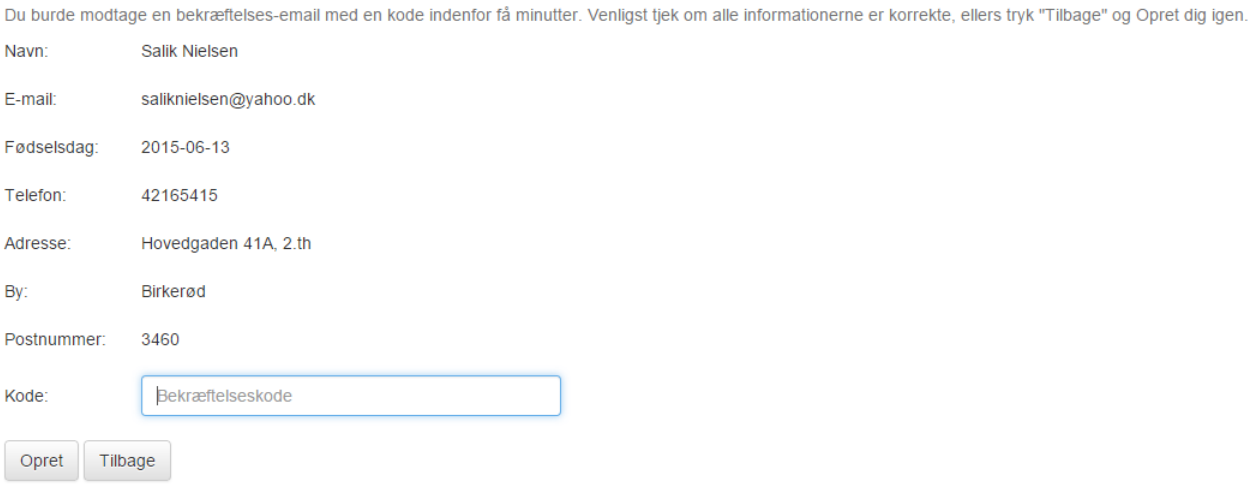
Written by: Salik

We created a new project in Visual Studio for this user story, because this is going to be in K19's Venner web site and not in there SharePoint. But it uses the same database as other user stories.

In other user stories we used model-view-controller, but not in this one, because this project is small, and easy to program.



The user has to type their information about themselves. When the user clicks “Send bekræftelseskode”, they will get an e-mail with random verification code and will be redirected to verify page, where they can see the data they have typed.



After they have typed the correct verification code, they will get this message:

Velkommen til K19s venner, Salik Nielsen.
Vi har sendt dig en email med informationer omkring dit nye medlemskab.

They will have to wait for accept from admin, to login to SharePoint page.

Here can we see how save the verification code we send the verification code:

```
private void SendVerificationCode(string to, string verificationCode)
{
    try
    {
        _dao.VerificationCode(to, verificationCode);
    }
    catch (Exception)
    {
        Response.Write("Database problems");
    }
    MailMessage mail = new MailMessage("admin@k19svenner.onmicrosoft.com", to);
    SmtpClient client = new SmtpClient
    {
        Port = 587,
        EnableSsl = true,
        DeliveryMethod = SmtpDeliveryMethod.Network,
        UseDefaultCredentials = false,
        Credentials = new NetworkCredential("admin@k19svenner.onmicrosoft.com","******"),
        Host = "smtp.office365.com"
    };
    mail.Subject = "Bekræftelseskode til K19s Venner";
    mail.Body = " Kopier og indtast koden på hjemmesiden for at gennemføre din
oprettelse: " + verificationCode;
    client.Send(mail);
}
```

The code down below shows us how we create random verification code. It makes sure to remove all dot and the length is no longer than 10.

```
public static string RandomStr()
{
    string rStr = Path.GetRandomFileName();
    rStr = rStr.Replace(".", ""); // For Removing the .
    rStr = rStr.Remove(10);
    return rStr;
}
```

In down below can we see how we insert verification code to the database. ExecuteScalar() executes the query and returns the first id it found. If the returned value is zero then it means that verification code was not found.

```
public bool CheckVerificationCode(string iEmail, string iVerificationCode)
{
    try
    {
        String SQLQuery = "SELECT COUNT(*) FROM VerificationCode WHERE Email = @Email AND
Code = @Code";
        SqlCommand command = new SqlCommand(SQLQuery, _sqlConnection);
        SqlParameter param = new SqlParameter("@Email", iEmail);
        command.Parameters.Add(param);
        param = new SqlParameter("@Code", iVerificationCode);
        command.Parameters.Add(param);
        _sqlConnection.Open();
        Int32 length = (Int32)command.ExecuteScalar();
        if (length > 0)
        {
            return true;
```

```
        }
        return false;
        }
        catch (SqlException sqlEx)
        {
            throw new Exception(sqlEx.Message);
        }
        finally
        {
            _sqlConnection.Close();
        }
    }
```

5.4.3.1 Outlook

- When there are message from the system we could have used Bootstrap's alert, so they are easier to see, instead of ASP.NET's Response.Write().
- In verify page the birthday should be dd-mm-yyyy, instead of yyyy-mm-dd.
- If the admin rarely goes to their SharePoint page, they won't see that there are new users who wants to join their membership. The solution can be that the admin gets an e-mail notification.
- After verification code. It should also tell the user that the admin has to accept their sign up.

5.4.4 User story 12

Written by: Salik

When a user has signed up in K19s Venner main site (not SharePoint), the admin can either accept or decline the new members. When a user has signed up, they can't use the system. The admins can see how many that needs to be accepted or declined in the frontpage, so they don't have to go in the approve page.

The approve page shows the name of the new members. If they click a name, it will expand, so it shows info about the user. Bootstrap collapse is used to expand info's. E-mail, birthday, phone, address, city and zipcode will be shown. Accept and decline button will also be shown.

An e-mail will be sent to the user when the admin accept the member's sign up.

The server side gets the new users, and dynamically creates html content and handlers for new users and puts them in panel. The code for creating html content can be seen in appendix[26].

We are using Microsoft Outlook to sent e-mail. The e-mail contains condition for membership and the company's logo. The e-mail's body is created with html content, because that was the way we learned how to put an image in the e-mail body.

5.4.4.1 Outlook

The accept and decline buttons should have Bootstrap design.

5.4.5 User Story 15

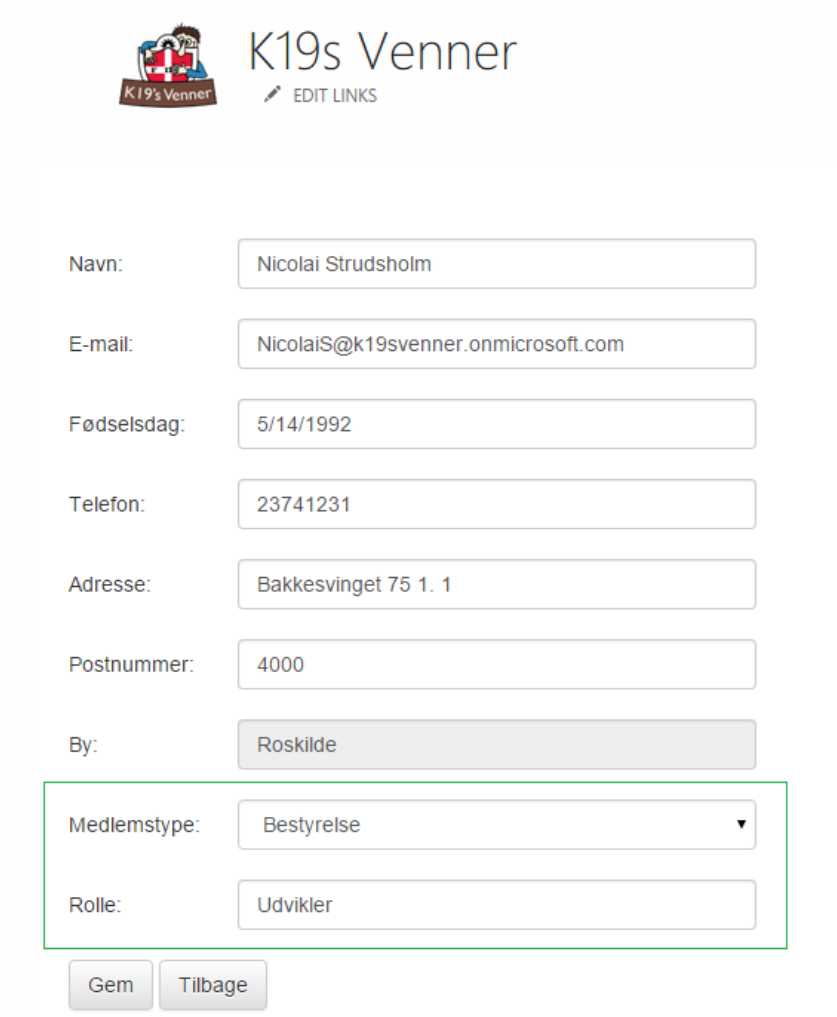
Written by: Nicolai

The Contact List is using the same functions as User Story 14. In this GridView, the view has a where clause where MemberType should be "Bestyrelse".

When we implemented this function, the product owner wanted a way to see which role each individual member was. So that you could contact the right member based on the role. If you had Invoice problems, it would be a good idea to contact the Treasurer etc. This was added the day after the Sprint Review Meeting for this Sprint.

When you edit a member, you can assign role if he is "Bestyrelse".

[26] Appendix – User story 12 (p.5)



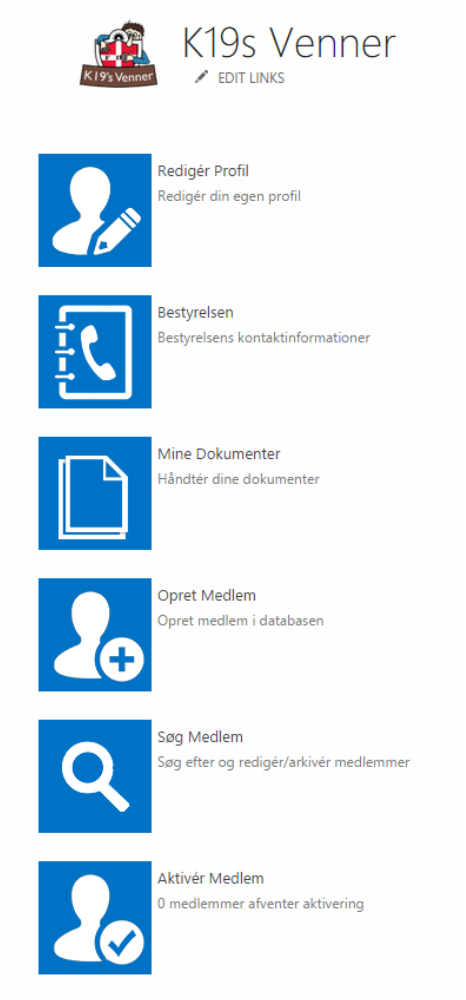
5.5 Evaluation

Written by: The team

Compared to the screenshot from Sprint 1, it is easily seen that there has been many changes to both the amount of functions and the design of those.

Both the team and the product owner agrees that the system has taken shape and is a lot better looking than the system we showed him in Sprint 1.

We implemented Bootstrap, which is the reason why the design has changed for the better. We used Axure in the past to create the User Interface, but using Bootstrap we can save a lot of time, it will look better and the CSS is generally easier to handle using Bootstrap and not Axure.



User Story: Edit Profile

Navn:	Nicolai Strudsholm
E-mail:	NicolaiS@k19svenner.onmicrosoft.com
Fødselsdag:	5/14/1992
Telefon:	23741231
Adresse:	Bakkesvinget 75 1. 1
Postnummer:	4320
By:	Lejre

Gem Tilbage

5.5.1 Unfinished user story

Written by: The team

The reason why we can't finish user story 8 is, that we were going to use an Excel ark for invoice template and Azure do not support Microsoft Office[27] and the company has to pay if they want to save something in Azure. We are also inexperienced in ASP .NET, which makes it harder to come up with another solution. There may be another way to create an invoice, but we need to check if the customer has paid, using NETS. We do not have an experience with NETS, and we decided not to try it.

If we can't make invoices, then we can't finish user stories 9 and 10.

5.5.2 Sprint Review Meeting

Written by: Salik and Nicolai

As the sprint review meeting from sprint 1, we started out with showing the part of the product to the product owner, which was ready for release. We talked about the different functions, the likes and dislikes and what to improve.

The unfinished User Stories were explained to the product owner at the sprint review meeting for this sprint. We had a long discussion of different solutions that we could implement to fulfill the requirement of this User Story. We didn't talk with the customer about the User Story number 13, because we had our hands full with the User Story number 8.

In the end, we came to an agreement of a solution that we, the team, are confident that we can implement. Together with the product owner, we decided to make some changes to the product backlog, and move this User Story to next sprint. This will be further explained in the section of Sprint 3.

[27] http://debugmode.net/2011/08/28/creating-and-updating-excel-file-in-windows-azure-web-role-using-open-xml-sdk/

5.5.3 Sprint Retrospective

Written by: The team

Continue doing:

- Pair programming
    - Pair programming was quite successful in our case, which encouraged us to do it more. We decided there was no point in changing this part of our teamwork in the middle of the project, especially knowing how fruitful it was for us in the past.
- Sprint backlog
    - This is an essential part of developing any project and our customer also requested for us to do it after every sprint. It allowed us to see the growth of our system.
- Database design
    - Another important part of the system we developed. We decided it would not be a good idea to give up on this.
- Continuous integration
    - We could not program the system without this component. Sharing the code allowed us to see which parts of the code had been changed already and what errors we received. There was no time for mistakes and continuous integration turned out to be very useful.
- SCRUM in general
    - As our project progressed and was being developed continuously, it was important for us to be able to shape the project as we go.

Stop doing:

- SCRUM meetings
    - The decision was to stop SCRUM meetings because the system was basically shaped by that point and the team worked together on every thing. There was no need to discuss what had been done because everyone was up to date with the progress of the project.
- Velocity tracking
    - We gave up on velocity tracking because we figured it was a waste of our time. We knew what we had to do and the points we were close to finishing the project so this was one of those not necessary things that would only take the time otherwise devoted to finishing the system.

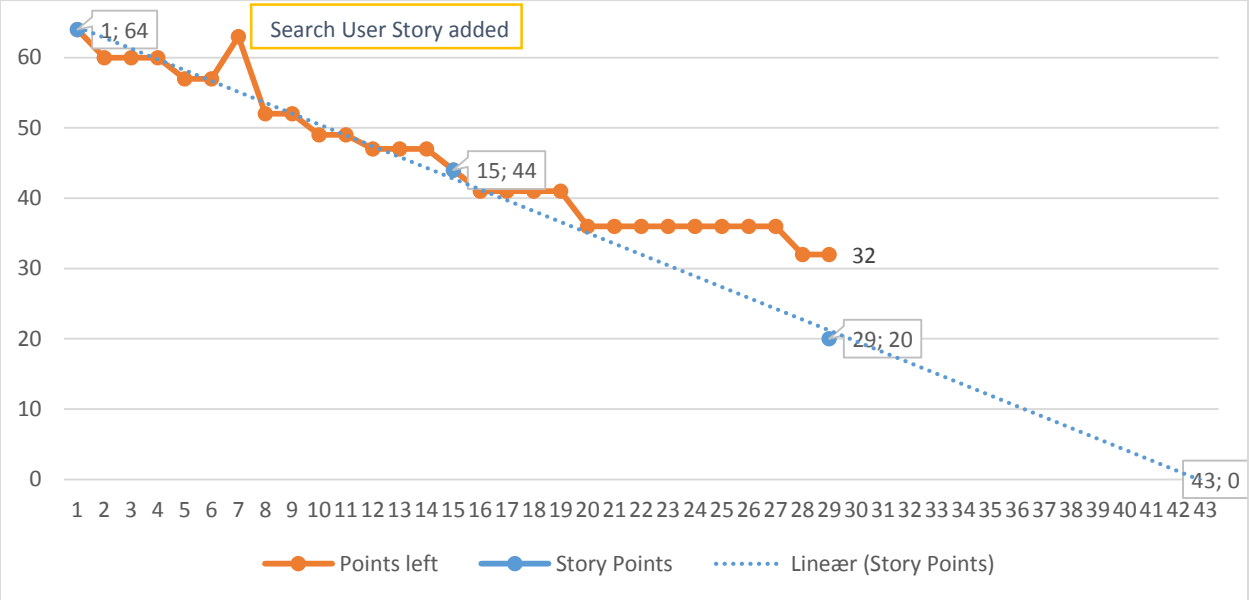
5.5.4 Velocity Tracking

Written by: The team

We don’t see any need to change the velocity even though we are behind the schedule. We mostly didn’t finish User Stories because of inexperience and some doubts about functions, which we needed to clear up with the customer.

The missing two User Stories, number 8 and 13, have a sum of 12 Story Points, which are displayed in the graph below. The number 13 was a bonus User Story that we wanted to make, if the number 8 User Story was easy to do. We had no experience in the requirements for implementing the functions to create number 8, so the Story Points assigned to that User Story was a wild guess.

In the graph, there is a period of 6-7 days that shows no progress in the project. To clear things up in this period, we did work as much on the project as we did last sprint, but the work was not 100% related to the User Stories but more related to fulfilling quality factors described in the project establishment. In this period we had 1 member working on the User Story and the others working on increasing the quality of the product. During that time, we implemented logging, compatibility for different browsers and devices and Doxygen documentation.



6 Sprint 3 / Release Sprint

6.1.1 Intro

Written by: The team

For the last sprint (May 13th - 31st), we decided on using it as a mix of a normal sprint and a release sprint. A release sprint is mainly used to test the system thoroughly and make sure everything works before handing it over to the costumer. We also want to work on implementing the last User Story as much as time allows it.

After this sprint, we will move all focus to writing the report and finishing our project. If the costumer has any errors with the system, we will try to respond to them and fix them if time allows it.

6.2 Sprint Backlog

Written by: The team

	User stories
8	As an administrator, I can create an invoice and start a subscription Story Point: 4 **Tasks:** Code: Salik and Nicolai

	Database: Salik Design: Kasia

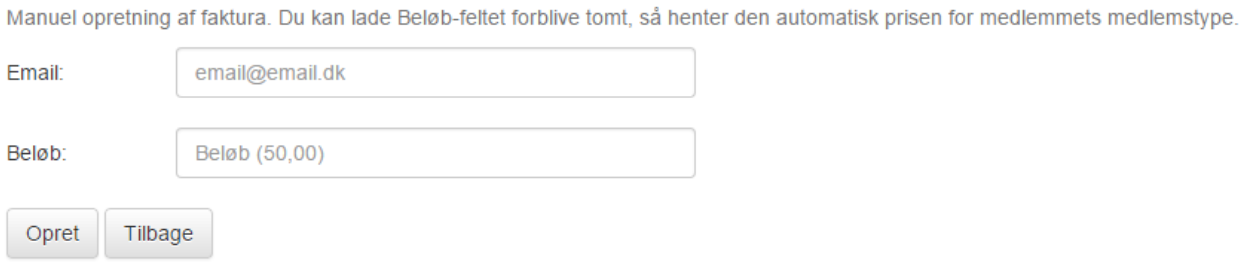
We will work on finishing User Story 8 for the first part of this sprint. As evaluated and discussed in Sprint 2, we decided on this being our last User Story, because even if we tried to read and learn how to finish those User Stories that are not so important for the customer, we do not have enough time.

For the last half of this sprint, the release sprint, we will mostly be working on making the product ready for release. We also need to improve the maintainability and the reliability of the product, which we have been putting focus on through the whole project.

6.3 User Stories

6.3.1 User Story 8

Written by: Salik



The admin have to type the member's e-mail to create invoice and sent in by e-mail. They can put price for the invoice or leave it empty. If it is empty, then the system will check the price by checking the member's member type, because each member types have their own price.

K19s Venner

Medlems nr: 10
Salik Henrik Nielsen
Hovedgaden 41A, 2.th
3460 Birkerød
salik_89@hotmail.dk

K19s Venner
Addresse: Fjeldhammervej 17
By: 2610 Rødovre
Telefon: +45 154848

Faktura nummer: 83
Dato: 6/8/2015
Bank: Nordea Bank
Reg. nr: 12341234
Konto nr: 567890324

Faktura

Betegnelse	Beløb
Medlemskab	0,00

Total DKK 0,00

Du skal betale.

We are using iTextSharp to create invoices and save them as pdf.

The invoices contains member's info on the left and the company's info on the right. In the middle is the price for the membership. At the button is the company's message for the member.

The method to create a pdf file is appendix[28]. Here you can see how to put the name and path for the file with Server.MapPath(). Using iTextSharp's Paragraf you can put some text to the pdf file.

Using HTMLWorker we created HTML table to put the price for the membership. Using StringReader the HTMLWorker can parse it to HTML content.

```
var worker = new HTMLWorker(doc);
worker.StartDocument();

string total = "<br/><table border='0'><tr><td width=\"80%\" style=\"text-
align:right\"><b>Total DKK</b></td><td width=\"20%\" style=\"text-align:right\">" +
newPrice + "</td></tr></table><br/>";

TextReader reader2 = new StringReader(total);

worker.Parse(reader);
```

[28] Appendix - User story 8 (p.3)

6.3.1.1 Outlook

- We could have included “Create new invoice” in the member list page, so the company doesn’t have to type every e-mails to create invoices.
- There is should be a check if the e-mail exist.
- More input like name or member id could be implemented, so the company doesn’t have to remember their members e-mail. When you type in member name, it could give suggestions just like searching on Facebook and Google.
- The invoices should have CVR numbers, because of the law.
- When the amount is big, an error occurs.

6.4 Evaluation

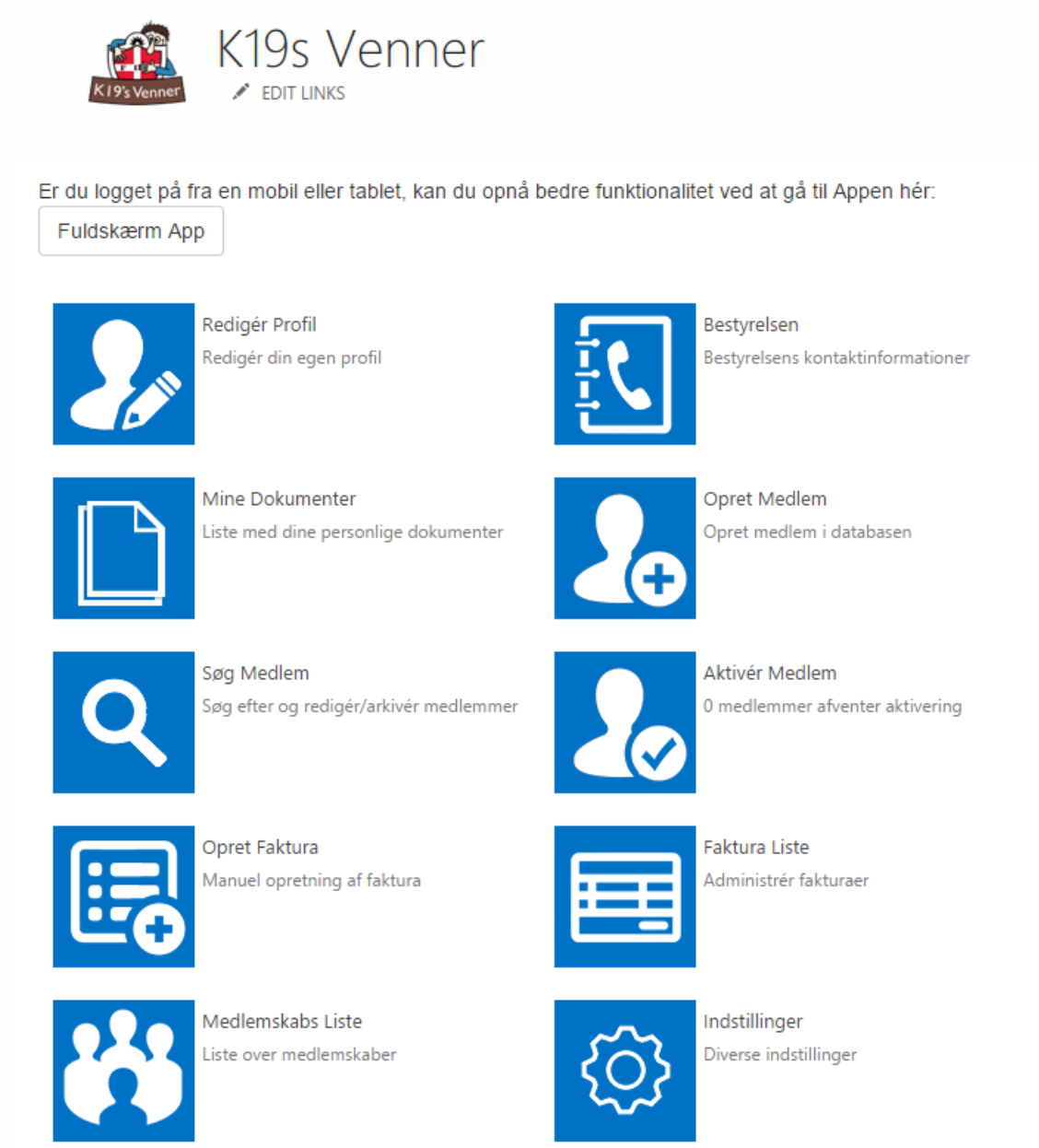
Written by: The team

There has been added more functions to our system as it is shown in the screenshot below. The invoice functions are done together with the membership function.

We have decided to implement Bootstrap in our front-page icons, so it will create more columns for each icon based on how big the screen is. So for mobiles, the icons will be in one column going vertical, and for a big screen with full HD, the icons will have a column each with one or two rows going horizontal.

This is something we discussed with the costumer at the end of sprint 2, and it is something that he would like seeing in the system.

When we finished the first half of the sprint, we decided on working more on the release sprint part, which were about error fixing and making the design better and more user friendly.



6.4.1 Sprint Review Meeting

Written by: Salik and Nicolai

The last sprint review is where we hand the system over to the customer, and in a perfect project, the product would be done and running with no problems.

We started out by showing the whole product to the customer. We showed the first functions from sprint 1 and how they work with the rest of the system. He saw how the system worked as a whole, where in the first two sprints, he has only seen bits of that.

When showing him the different functions, we ran into some minor problems with date conversions and some User Interface related problems. It was very small problems, and the core functions of the system worked perfectly.

We took him through the whole process of:

1. (User) Signing up for member system
2. (User) Receiving Verification Code through email
3. (User) Finish signing up
4. (Admin) Send invoice to member before activation
5. (Admin) Accept Invoice
6. (Admin) Activating members who signed up
7. (User) Signing in with new account

That process worked flawlessly, but some other minor functions of the system had some small problems, and it is something that we would like to fix when writing the finishing documentation of the system.

The User we created for the product owner was used to test the system alone without the team guiding him. He wrote some feedback after the sprint review meeting, where he wanted to see changes, where he found errors and what caused them.

The sprint review meetings has been a very good method to include the product owner in the project, and the team and the product owner think it has been very productive having these meetings. Without these meetings, the team would probably has continued to use poor user friendliness throughout the development and the product owner would maybe not be satisfied with the outcome.

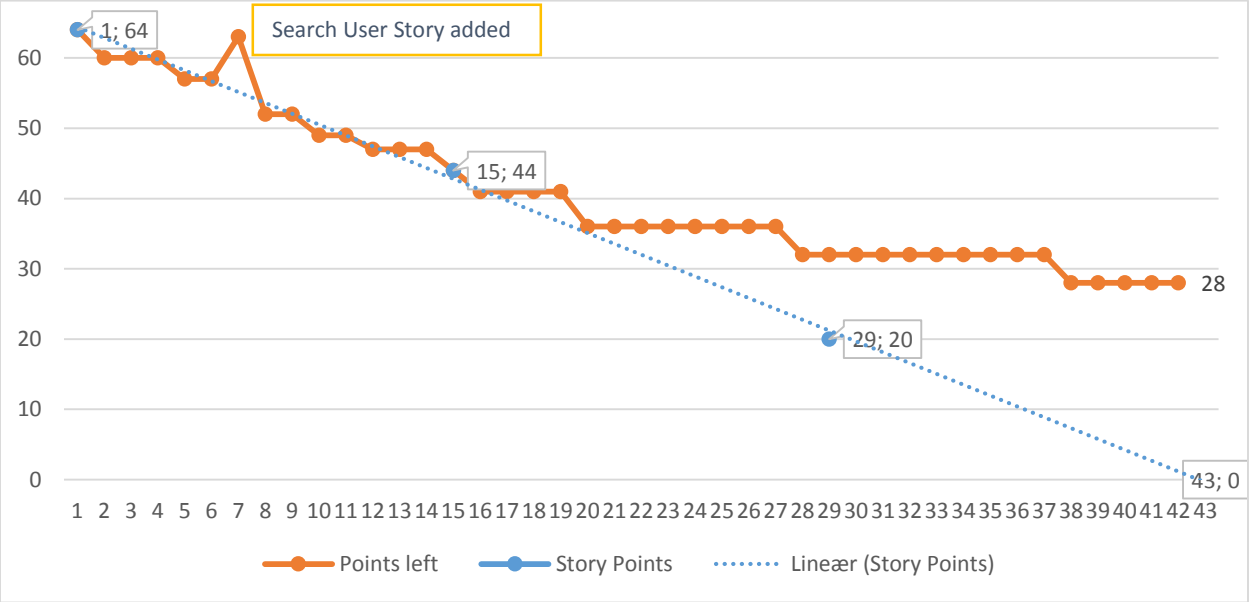
6.4.2 Velocity Tracking

Written by: The team

The velocity tracking is something we stopped doing at the end of sprint 2, because as it is shown below, it didn't make sense tracking it anymore, since the User Stories were too difficult, and we decided together with the customer to only work on very few functions towards the end.

But to show how much we did and how much we have left, we decided on quickly finishing the velocity tracking graph at the end of sprint 3.

The end point has a value of 28 point, which is the amount of story points there are left in the product backlog. We failed to estimate the amount of User Story points on number 8, and it delayed the rest of the User Stories, so they couldn't be finished. We estimated the User Story 8 to four points, and since we used a whole sprint on finishing this, it should at least have had the same value as the sum of all the User Stories in sprint 1.



7 Conclusion

Written by: The team

7.1 Reflection

The planning we did and the decisions we took were overall very good. We managed our time responsibly, tried to contact the customer as often as it was possible. The meetings in Sprint Zero, which purpose was to plan ahead, were successful, even though we could not possibly foresee the problems we would have later. The problems we encountered were solved quickly, either with or without consultations with the customer.

In general, the development process went smoothly and the results were satisfactory for both the team and the customer. We tried our best and the only things we did not do were due to our lack of experience (not being able to finish User Stories 9, 10 and 13). If we had more time, we could have tried to fill the gaps in our knowledge and finish even those User Stories that were not a 'must have', but in this situation, given the time span we had for the project, it was everything we could do.

When the team was finally together, the teamwork was good enough. There were some problems with communication, mostly because of the language barrier, but we managed to overcome it and discuss the points we had made during the development process. Every team member did their best to keep the project going, whether it was by adding new lines of code or polishing the details or even by keeping the good mood among us. It was a very pleasant experience to work together and create the product we delivered.

7.2 Outlook

We think that our system could have been quite successful if we had had the time and skill to finish it. It could have been changed to be more general and useful to other companies too. We hope that the work we have done will be used anyway, even if it meant other people working on finishing it. It is not perfect but we are proud of it.

7.2.1 Ideas:

- If we wanted to take this project to the next level, we could have made it into a generic member system product which could be tailored to companies by themselves. Right now, the system is tailored to K19's Venner, and it wouldn't make sense for other companies to use this member system right now.

- When we created the invoice function, we made it possible for the company to change the information on the invoice itself. We could easily take this function alone, develop further on it and make it a stand-alone app for SharePoint Online. This would give the opportunity for companies to send out invoices and save them online.

- The invoice list and the invoice functions could be used to develop an ERP[29] like system. This would require a lot of development and not really have anything to do with the member system.

[29] http://da.wikipedia.org/wiki/Enterprise_resource_planning

7.2.2 What comes next?

If all errors were corrected, and the system would run flawlessly, there could be many ways to take the next step.

- The product owner talked about making this system almost fully automatically. This would mean that the system would run a job on a daily basis perhaps. This job would check the membership status of each member, and send out invoices automatically, if the member's membership was about to run out. If the member forgets to pay, it will send reminders to him.

- Payment system. If the organization contains many members, the next optimal function could be a payment system, where the system will accept payment, check payments and extend the membership of a member if the payment went through. Through this function, it could also accept donations etc.

- The most ideal would be a mix of the two previous mentioned functionalities. If they could work with each other, there would be no need for many of the functions we created. This would mean there wouldn't be need for much administrative work and since it's an organization that doesn't pay its board members, this would be of high value.

The functionality mentioned above costs money based on how much you use it in Azure. A web job that needs to run on a daily basis could cost a few Danish kroner each month, and together with payments to SQL database and SharePoint Online subscription, it can be quite costly.

Perhaps the functions will be cheaper or free in the near future, it's hard to say with Microsoft. They change their prices on their products sometimes. As of right now, you can have up to 10 free web apps on Azure, where in the past; it would not have been free.

7.3 Evaluation by individual team member

7.3.1 Salik Nielsen

I started with no experience ASP.NET, but it was easy to learn how use it, because I already know C#, HTML and JavaScript. I like how we worked as a team. It was easy to communicate with each other and helping out with the code. Working with agile made it easy for me what we should do.

I like that we have a company to work for, so we can get more experience on how to communicate with them and make better products.

I don't think I'm going to use Microsoft Azure, because it was slow and we have to paid for many thing like database. I would prefer to have my own local server, so I don't have rely on others, and I only have to pay once for the equipment.

Using Visual Studio Online was great to share our work with each other.

7.3.2 Katarzyna Grabowska

The project was in my opinion quite successful. We delivered the product that was not perfect but satisfactory for us, considering how little time we had and our lacking skills.
I myself learned a lot about working in a team, how important it is to discuss everything and not be ashamed to speak out my mind. I liked the rest of my team and the fact that they accepted me and did

not try to diminish my role in the work we did just because I am not so good at coding. I tried to give my best and even though I was not always able to be present on the meetings, I always got updated by the other members. I also tried to be up to date with the code and why they used the methods they did, which taught me something about coding too.
I think I learned many things about teamwork which will help me in the future - acceptance, patience and, most importantly, communication. It is vital for a team to work fluently and in peace, trying to solve any arguments or misunderstandings.

7.3.3 Nicolai Strudsholm

I've spent three months in an internship where I studied SharePoint Online together with ASP.NET. This project was about trying to create a process of creating a new product based on the things I learned together with learning a lot more about the subjects. At the end of the project, I've learned a bit more about ASP.NET, but I've learned the most from the experience on working together on a project, making decisions together and having a costumer connected to the project as well.

I've never really worked on a serious project, where the product would actually matter when we were done. Working in a group with agile methods and practices has been very exciting, and I've come to understand the power of agile practices, especially pair programming. The project was a big one, and we didn't finish all the User Stories for the product, so if we could have been a bigger team, it would have been more enjoyable and some of the agile practices would have made more sense.

7.4 Ending conclusion

We have divided the ending conclusion into several sections. These sections should each answer to some of the problems in our problem definition finishing with a section that will answer the overall problem.

7.4.1 The Process

The product requirements we got from the customer was a very good start to our project. We could see that the project was big enough to create a good product, and there was enough for us to do to work a lot each sprint.

A lot of the requirements we had no idea how to implement it, but with the help of pair programming and increasing knowledge within the chosen tools such as ASP.NET, we came up with solutions to a big part of the requirements. If we had any problems and were stuck, it was easy to talk with the costumer to refine the solution to make it something that we could do.

We had to deal with a bit of absence throughout the project, but we tried to make up for the lost work through communication on Facebook. If someone was absent, we had to sometimes cover for the work they were missing, and it caused a bit of increased work load and a different priority delaying the project a bit.

We focused on having a good relationship with the costumer. We kept our appointments and if he requested changes, we would prioritize to fix that for the next meeting. We also had several meeting with our supervisor giving us advice for the project and the report, which helped us a lot shaping this project.

7.4.2 Techniques and tools

Using SharePoint Online (SPO) and Microsoft Azure was a requirement by the costumer. Our problem was to find the most suited tools and techniques to work with these two products.

We decided on going with ASP.NET, because it was also made by Microsoft, and therefore it would have good synergy with SPO. We also deciding on working with Visual Studio (VS) as our integrated development environment. All the above products are managed by Microsoft and therefore they had good synergy and worked well together.

Microsoft Azure was very slow when working together with VS, and it delayed our working process a little bit. We had the cheapest version of an SQL database, so that might have caused the problems.

We used Visual Studio Online when integrating each others work into the main project. It worked almost flawlessly, we had minor problems with merging the code sometimes, but we expected a lot more problems from earlier projects using VS Online.

Microsoft products and tools has been satisfying to work with, and the online community using Microsoft products is very big, so it's very easy to get help with problems that might occur.

To make the system responsive and a lot less confusing, we decided on implementing AJAX, JQuery and JavaScript. ASP.NET has post backs, as we described in the section with Techniques and Tools. The overall implementation of AJAX went well, and the system ended with being a lot more responsive. Using AJAX together with AJAX controls, some of our labels stopped working, and it is mostly because we are inexperienced in implementing it.

To make the solution compatible on every decide and browser we decided on learning how to use Bootstrap at the start of Sprint 2. We ended up integrating Bootstrap in almost every control in our project, because it worked so well. There was several problems making the design compatible with Internet Explorer, and it took some of our time finding the right solution. But overall, Bootstrap was the right choice for us, and the system looks fine both on PC, mobile and tablets. For mobile and tablets, Bootstrap makes the components bigger and more touch-friendly.

Cloud solution

Having a small company or organization, it might not be optimal to have a room full of servers. A cloud solution doesn't take any space in your office, since everything is online. There are a lot of advantages in using cloud services instead of a server solution. The most important advantages for the company would be that it's very cost efficient and it's easy to access from everywhere.

The organization is not very big and doesn't have a big budget, so it's optimal for them to pick a cost efficient solution. The organization members should be able to connect from everywhere, since they want to do organizational work from home because of not meeting frequently.

The cloud service might have technical problems once in a while, making it hard to reach documents or work in SharePoint Online. Using Microsoft Cloud service you chose to trust the workers of Microsoft that they will not sell your personal information to other companies.

Overall a cloud solution is better for K19s Venner, since the information is not very sensitive information such as social security numbers. If a technical problem should occur at the cloud service, the work of the organization can be postponed since their board members work for free.

We worked with the cloud on this project, and we had very few problems with it. A day or two we had problems connecting to Microsoft Azure and SharePoint Online, but it was not a long downtime it had.

7.4.3 Methods

We chose to work with agile software development methods for this project, since we had good experience working with it from earlier projects. The team has been taught in Unified Process some semesters ago, but since we all have been taught in agile recently and we enjoy it more, we chose that method.

Agile doesn't require heavy documentation such as Unified Process, and we wanted to work more on the product, since the costumer was more interested in the product instead of the documentation.

The good thing with agile and its practices is that we can shape the way we want to work as the project moves on. We do this in the sprint retrospective, where we pick up practices we want to try out or stop doing the ones that are a waste of time for us.

Using SCRUM it is required to have a lot of contact with the customer. The customer is invested in the work we do, when after each sprint, we have a review meeting with him. Using Product backlog and Sprint backlogs, it gives the team members and costumer more insight in what we need to do, and what to expect for the sprint review meeting.

Using agile practices helped us prioritizing our work and increase the quality of the project and the product.

7.4.4 Product

Using the appropriate techniques, tools and methods, we finished a major part of the requirements set by the costumer. Throughout meetings with the costumer at the end of each sprint, we shaped the product to his liking. The system has all the core functionalities the customer wanted. We added some functionalities ourselves such as Search Member and a Contact List.

While having all the core functions and a lot of "nice-to-have" functions, the team and the costumer agreed that the User Interface and the user friendliness of the system could have been better. Some places it doesn't make sense from a user's point of view, but it would make sense from a developer's point of view. If our system would have been tested by a group of users, both admin and regular members, we think that the feedback could have helped the system a lot. We relied on feedback only from one person, and when not in contact with this person, we relied on our own experience in working with User Interfaces.

Working in a group has also increased the quality of the product, since we would read each others code and comment and give feedback if we thought of an idea.

Since the product had to work with SharePoint Online, we had some difficulties making it work together. We had problems making the Microsoft Account interacting with our system and SharePoint at the same time. We decided on removing a lot of SharePoint buttons through CSS and JQuery so that the user would be forced to use our app and not being able to click around.

We are very happy with the outcome of this project, and we spent a lot of time commenting the code and making it look nice so that it can be developed further on by others.

8 Bibliography / Webography

Websites used for theory:

- http://www.microsoft.com/enterprise/it-trends/cloud-computing/default.aspx#fbid=gt2BxNAWXIL
- http://www.microsoftvirtualacademy.com/
- http://www.w3schools.com/browsers/browsers_stats.asp
- http://en.wikipedia.org/wiki/Software_development_process
- http://en.wikipedia.org/wiki/Iterative_and_incremental_development
- http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
- http://en.wikipedia.org/wiki/Multitier_architecture#Three-tier_architecture
- http://en.wikipedia.org/wiki/ASP.NET
- http://en.wikipedia.org/wiki/JavaScript
- https://jquery.com/
- http://www.quora.com/What-are-the-pros-and-cons-of-using-Bootstrap-in-web-development
- https://products.office.com/en-us/sharepoint/sharepoint-online-collaboration-software
- https://www.visualstudio.com/en-us/products/what-is-visual-studio-online-vs.aspx
- http://azure.microsoft.com/en-gb/overview/what-is-azure/
- http://sourceforge.net/projects/itextsharp/
- https://msdn.microsoft.com/library/office/microsoft.sharepoint.client.clientcontext%28v=office.15%29.aspx.
- http://www.aspsnippets.com/Articles/Calling-ASPNet-WebMethod-using-jQuery-AJAX.aspx
- http://www.postdanmark.dk/da/Documents/Lister/regionsopdelt-postnummer-excel.xls
- http://en.wikipedia.org/wiki/Query_string
- https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview%28v=vs.110%29.aspx
- http://debugmode.net/2011/08/28/creating-and-updating-excel-file-in-windows-azure-web-role-using-open-xml-sdk/
- http://da.wikipedia.org/wiki/Enterprise_resource_planning

Online courses:

- http://www.microsoftvirtualacademy.com/
- https://channel9.msdn.com/