Western Norway
University of
Applied Sciences

BACHELOR'S THESIS

Digitalization of the medicine chart at Gatehospitalet in Bergen

Sunniva Hillesøy and Alvin H. Gusfre

Computer Science
Faculty of Engineering and Science
Supervisor: Carsten Gunnar Helgesen
18.06.2021

Western Norway University of Applied Sciences

Faculty of engineering and science
Department of Computer Science,
Electrical Engineering and Mathematical Sciences

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: Digitalization of the medicine chart at Gatehospitalet in Bergen	*Dato:* *18.06.2021*
Forfatter(e): Alvin Haukedal Gusfre and Sunniva Hillesøy	*Antall sider u/vedlegg: 45* *Antall sider vedlegg: 49*
Studieretning: Dataingeniør	*Antall disketter/CD-er:* 0
Kontaktperson ved studieretning: Carsten Gunnar Helgesen	*Gradering:* Ingen
Merknader:	

Oppdragsgiver: Gatehospitalet i Bergen	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson: Paal Nygaard	*Telefon:* 90756995

Sammendrag: Bachelorprosjektet går ut på å lage en digital løsning på medisinkurven hos Gatehospitalet i Bergen. Programmet skal hente informasjon om medisinene til en pasient fra en database, og brukerne skal kunne registrere at medisinen har blitt gitt til pasienten.

Summary: This bachelor thesis is about creating a digital solution for the medication charts used at Gatehospitalet in Bergen. The software should fetch information about a patient's medication from a database, and the users should be able to register it when a patient is given their medicine.

Stikkord:

Windows Presentation Foundation	Database, Web API	Medication Chart

Høgskulen på Vestlandet, Fakultet for ingeniør- og naturvitskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

Preface

This software development project was done as a part of our bachelor's thesis at Western Norway University of Applied Sciences and focused on developing a medicine chart for Gatehospitalet Bergen.

During this project we have had to learn several new skills such as GUI design and creating a RESTful Web API. The most important skill we have had the chance to improve is the development process as a whole – from start to finish. Being able to talk to the project owners and industry professionals has been an incredible experience, and it has also enlightened us and made us aware of the difficulties with software development, as well as taught us how to handle them.

There are many people to whom we owe our thanks to, but first and foremost Carsten Gunnar Helgesen - our supervisor at the faculty, and Paal Nygaard at Gatehospitalet for clear and concise guidance and feedback throughout the project.
Other people we would like to thank are:

- Johannes Straume at Infodoc for helping us get the database data right.
- Remy Andre Monsen for answering questions and helping with technical difficulties.
- Kimmy Reikerås Davidsen for letting Alvin borrow his laptop while he was stuck at the Covid Hotel.
- Friends and family for proofreading and all the moral support.

Finally, we would like to thank each other for a great partnership and support through difficult periods of the project.

Table of Contents

1 Introduction

1.1 Motivation and Goal

Wrongful distribution of medicine is a major problem in the health care sector, and with inadequate solutions for registering the distribution the risk increases even further (FDA, 2019). At Gatehospitalet in Bergen there are already digital systems in place for most of the matters the employees need, such as information about the patients and the medication they have been prescribed. However, there is no good solution in place for registering which patient is given what medicine, or who is responsible for distributing it. The consequences of giving the incorrect medicine to the wrong person can in a worst-case scenario be fatal, which is why this is consider an important issue to look at.

At Gatehospitalet they are currently using pen and paper to register the medication a patient uses, as well as when it is distributed throughout the day on a medication chart. A medication chart is a form where nurses and doctors can sign off what medicine is given to which patients and is one of the most important forms in a hospital (Noddeland, 2016). If the medication chart is messy it may lead to wrongful distribution of medicine, and when writing it by hand the probability of this increases.

The main task of this project is to digitalize Gatehospitalet’s existing “pen and paper”-solution for distribution of medicine.

The goal is to create a fully functioning module within the existing system, Infodoc Plenario, which is easy to understand and use for everyone, especially the nurses who will be the main end users. Both people who are comfortable utilizing digital resources and those who are not as comfortable with it should be able to effortlessly use this program. The graphical user interface (GUI) should consequently be designed with input from the end users.

Sub-goals:

- Security regarding patient privacy.

- Good usability for the end users.
- Works on both desktop and tablets.
- Collects the right data and saves it correctly to the database.
- Checks that the input is correct.

The program should never compromise any patient information or any other potentially harmful information. It is therefore necessary to pay extra attention to the security aspect of the module. In addition, it should be easy to maintain and make changes to the module at a later point in time if that is needed.

1.2 Context

At all hospitals and other health care institutions a great amount of medication is being distributed every day. Traditionally this has been done with the help of charts that the nurses and doctors have had available, and in many cases, these charts would be handwritten.

If the medication chart is poorly written or hard to read it might lead to patients receiving the wrong medication at the wrong time. In a worst case scenario this might be fatal.

Other hospitals and institutions have implemented different digital solutions to this problem. For example, at Haukeland University hospital they use a program called Meona to track a patient’s medication.

At Gatehospitalet they use Infodoc Plenario for their day-to-day work, and this works excellent in many ways, but it lacks a medication chart. The goal of this project is therefore to make a medication chart module that can be integrated with Plenario.

1.3 Limitations

Due to poor communication with Infodoc, developers contracting Covid-19 and time limits, there was a need to cut down on the scope of the project. Instead of making an integrated module within Plenario, the focus has been to make a program that works with test data from a database that simulates the one in Plenario. This will hopefully make it possible for either Infodoc or another group of students to continue this work.

1.4 Resources

Technical

For the project, a test environment of the system they use at Gatehospitalet, Plenario, is needed. Infodoc Plenario is a complete and safe journaling system in the cloud and Infodoc has been an important partner to the Norwegian health care system for over 40 years. Plenario is run at the well-known cloud platform Windows Azure, and it has a lot of useful features that are aimed at general practitioners and other short-term offices (Infodoc, u.d.). It is not made for hospitals and long-term institutions, but since Gatehospitalet need a lot of the features it offers this was still the best option for them.

An insight in Infodoc's database and how it is designed, will also be needed to display the right data at the right place in the module.

The Graphical User Interface will take inspiration from the existing software and layout of Infodoc Plenario. The programming language will be .NET/C# and Visual Studio is used to write the code. To make drafts of the GUI Figma, a tool for designing prototypes for webpages, mobile apps etc. (Figma, u.d.), is used.

Knowledge

To gain information, get technical recommendations as well as tips on how to keep a good workflow a supervisor at HVL, Carsten Gunnar Helgesen, and one at Gatehospitalet, Paal Nygaard have made themselves available. In addition to this, some of the lecturers are used for help on the more technical aspects of the program.

To obtain knowledge about technologies used, database creation and other, multiple literature resources are used. These include, but are not limited to, (Deitel & Deitel, 2016) for C#, (Elmasri & Navathe, 2011) for database and (Joshi, 2019) and (Kanjilal, 2013) for Web API.

Access to people

For the evaluation and testing of the module access to the end users of the program is needed. This is an important part of the project to make sure that the module works for the nurses and doctors at Gatehospitalet. For this purpose, the doctors and nurses at Gatehospitalet have all expressed that they are willing to aid this project.

1.5 Organization of the Report

In the second chapter there is information about the project and project description, as well as some practical background information.

The third chapter is describing different approaches to the project as well as the risks involved each step of the way. In addition, it will specify the development tools we will be using and how the development process will be handled as a whole.

Chapter 4 will explain the detailed design with the help of diagrams and figures, and chapter 5 will describe the evaluation of the project.

In chapter 6 there is a detailed description of the results of the project, while chapter 7 contains discussion around the process. The last chapter will hold the conclusion and a description of further work.

At the end of the report there are two appendixes, a Gantt diagram and a list of the risks.

2 Project Description

2.1 Practical Background

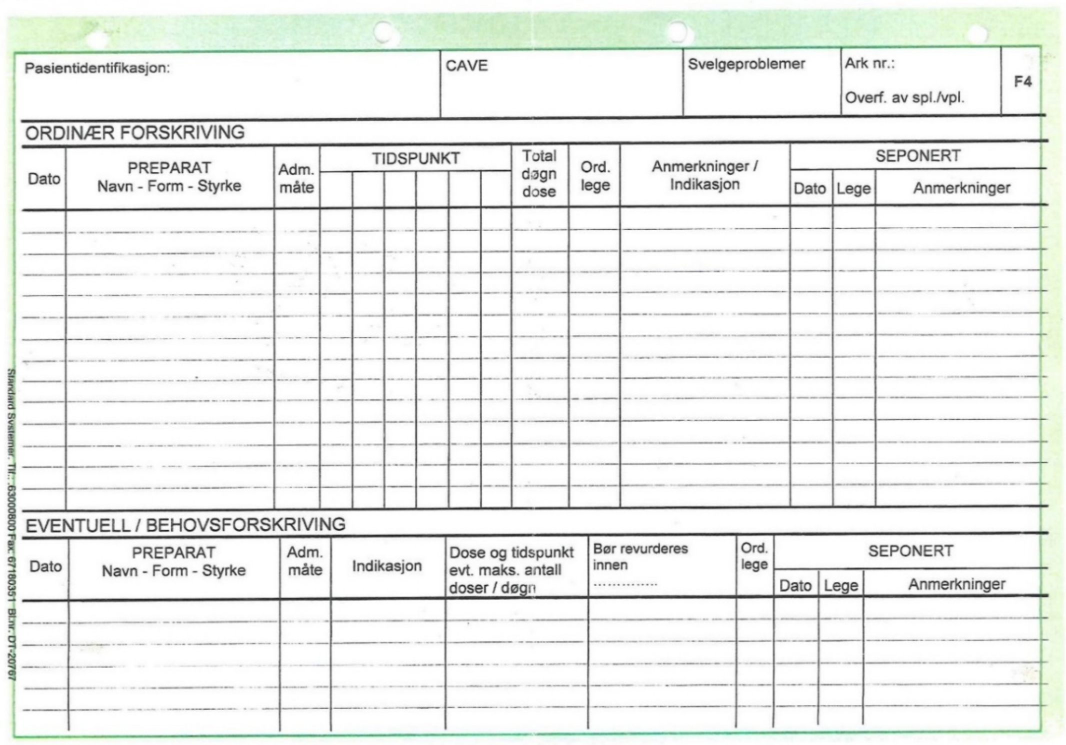
2.1.1 Project Owner

Gatehospitalet in Bergen opened in 2020 and is a health care service for people with severe drug addiction, that also have other health issues. Those health issues include cancer, COPD (Chronic obstruction pulmonary disease), malnourishment, and other. It is the Salvation Army in Norway that runs Gatehospitalet in Bergen on behalf of the Ministry of Health and Care Services. (Frelsesarmeen, u.d.)

The clinic is located at Garnes outside of Bergen. The normal time spent here is 2 to 3 weeks, but this depends on each individual case. There are two doctors and multiple nurses at Gatehospitalet who take care of the patients.

2.1.2 Current Situation

Gatehospitalet is using Infodoc Plenario to keep track of patient data. This cloud-based program has everything they need, except a medication chart. As of today, they record the distribution of medicine by hand, on paper. Figure 1 and Figure 2 below shows how the papers look like. Because of patient confidentiality it is not possible to show an example of how messy these charts tend to get.



Pasientidentifikasjon:	CAVE	Svelgeproblemer	Ark nr.: Overf. av spl./vpl.	F4

ORDINÆR FORSKRIVING

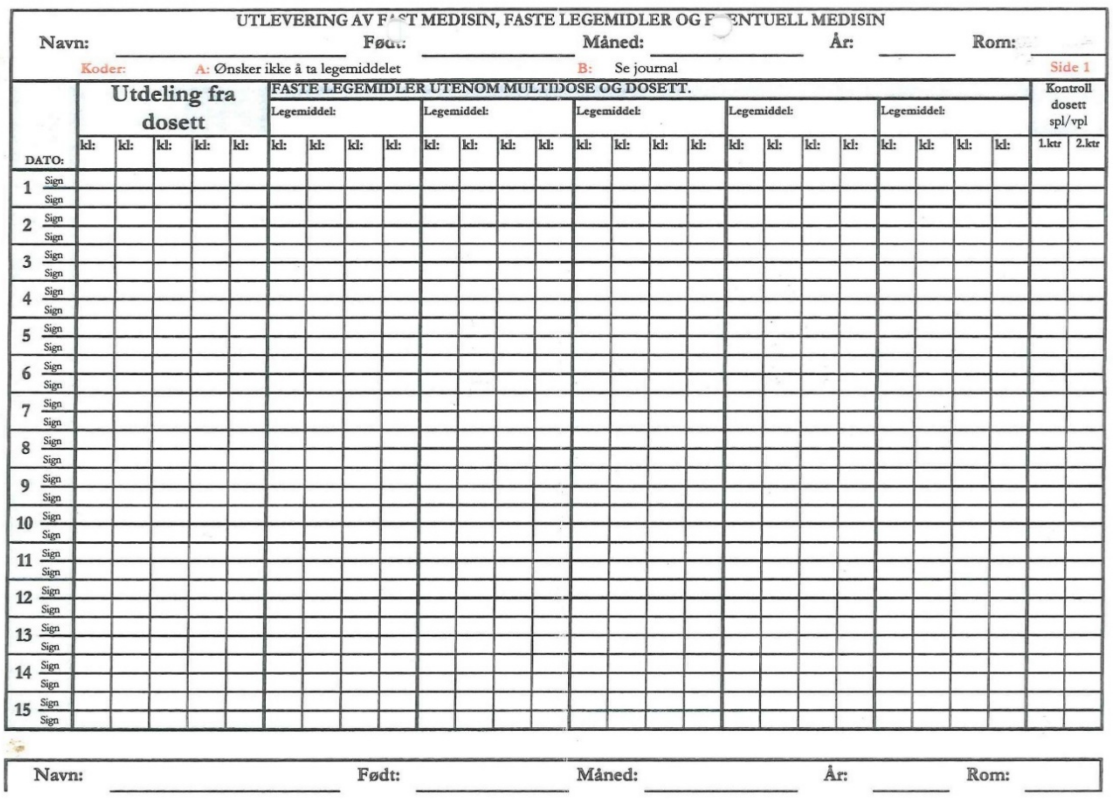
Dato	PREPARAT Navn - Form - Styrke	Adm. måte	TIDSPUNKT						Total døgn dose	Ord. lege	Anmerkninger / Indikasjon	SEPONERT		
												Dato	Lege	Anmerkninger

EVENTUELL / BEHOVSFORSKRIVING

Dato	PREPARAT Navn - Form - Styrke	Adm. måte	Indikasjon	Dose og tidspunkt evt. maks. antall doser / døgn	Bør revurderes innen …………	Ord. lege	SEPONERT		
							Dato	Lege	Anmerkninger

Standard Systemer. Tlf.: 63000800 Fax: 67180351 Bl.nr. DT-20767

Figure 1 - In this form the patient's medication is listed.



UTLEVERING AV FAST MEDISIN, FASTE LEGEMIDLER OG EVENTUELL MEDISIN

Navn: ______________ Født: ______________ Måned: ______________ År: ________ Rom: ________

Koder: A: Ønsker ikke å ta legemiddelet B: Se journal Side 1

DATO:		Utdeling fra dosett					FASTE LEGEMIDLER UTENOM MULTIDOSE OG DOSETT.																				Kontroll dosett spl/vpl	
							Legemiddel:				Legemiddel:				Legemiddel:				Legemiddel:				Legemiddel:					
		kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	kl:	1.ktr	2.ktr
1	Sign																											
	Sign																											
2	Sign																											
	Sign																											
3	Sign																											
	Sign																											
4	Sign																											
	Sign																											
5	Sign																											
	Sign																											
6	Sign																											
	Sign																											
7	Sign																											
	Sign																											
8	Sign																											
	Sign																											
9	Sign																											
	Sign																											
10	Sign																											
	Sign																											
11	Sign																											
	Sign																											
12	Sign																											
	Sign																											
13	Sign																											
	Sign																											
14	Sign																											
	Sign																											
15	Sign																											
	Sign																											

Navn: ______________ Født: ______________ Måned: ______________ År: ________ Rom: ________

Figure 2 - This form is used to distribute medication throughout the day.

When a medication is discontinued it is simply crossed out with a marker, and when one is added it is written below the existing medicines. Additionally, the dosage may change and therefore need to be changed in the chart. This will quickly result in a somewhat messy chart and the risk of wrongful distribution because of human error increases.

If the medication chart gets too chaotic, they need to write a new one. This is extremely time consuming as one must be exact and double check every medicine. Furthermore, the writer needs to have a readable handwriting.

All of this can be avoided by implementing a secure and easy to use interactive module in Plenario. There are a lot of considerations to take when making such an important tool for the clinic. Some of these are listed in the next chapter.

Infodoc do not know of any other user of theirs that need this, and therefore they have not used any resources to develop it themselves. This means that there is no previous work done. Some of the elements in this program may take inspiration from Infodoc's current solutions.

2.1.3 Initial Requirements Specification

The project description as well as initial conversation with Gatehospitalet and Paal Nygaard provided a list of requirements for the project:

- A solution that combines the information that is available in Plenario with a calendar function for day-to-day work.
- A solution that fulfills the governments demands to patient records and medicine handling.
- A solution that is developed for the users.
- A solution that gives a good overview of the medicine use from the screen.
- A simple and intuitive solution.
- A solution that works even when Plenario is regularly updated.

- A solution that can open inside a Plenario patient record. The signed in employee is identified and can sign on the distributed medication.
- A solution with enough space for additional info.
- A solution that is stored with other Plenario notes.
- A solution that can get the medication list from PLL (Pasientens legemiddelliste) instead of Plenario if PLL is introduced.
- A solution that can transfer the medication list to Plenario when a patient is discharged.

Gatehospitalet has been a big part of the project and it has given feedback on what works and what needs to be changed or added. Therefore, the specification on the end product may differ from this initial list.

2.1.4 Initial Solution Idea

From the initial requirements and talks with Gatehospitalet the initial solution idea is to make a digital medication chart that is integrated with Plenario. This means that any logged-on user will be able to open the medication chart and register the distribution of medication.

The module will open from within a patient record and contain all the information on that patient's medicine. This includes:

- The name, form and strength of the medication.
- The startup date for the medication.
- Which doctor prescribed the medication.
- What type of medication it is; regular medicine, medicine by need or cures.
- What the dosage is.
- When it is discontinued, and by whom.

It should be possible to register when medication is distributed and add comments on the effect of the medicine or other information. This information needs to be stored and it must be possible to access it at a later point.

3 Project Design

3.1 Possible Approaches

Figure 3 shows four different approaches to problem. The first two having the medication chart application integrated with Plenario or not, and the second two are saving data from the medication chart in Infodoc's databases or having the data in a separate database.

3.1.1 Alternative Approach 1

Creating a fully integrated module within Plenario would make everything available in the same software, and with no extra windows to manage on the screen. The new module will have to look and behave pretty much exactly as the already existing ones. Retrieving and posting information to and from the database might be easier with this approach, but the difficulty of making it fully integrated will be more challenging than creating an external program. Extra authentication will not be needed as the user is already logged into the system with a secure two-step authentication process.

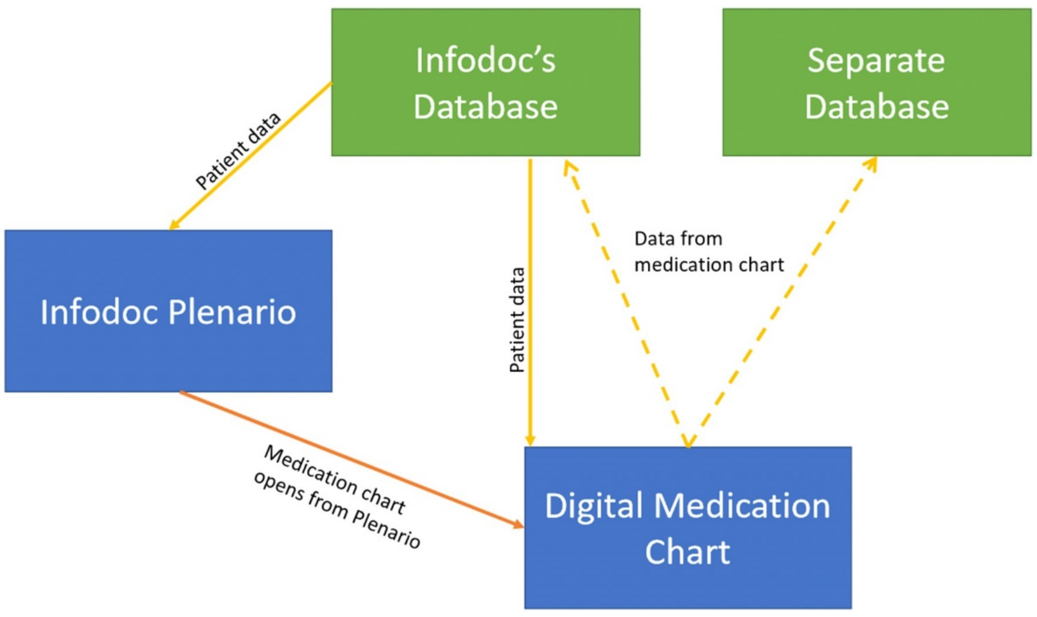


Figure 3 - Outline of how the module might communicate with the database and Plenario.

3.1.2 Alternative Approach 2

Creating a separate program that communicates with Plenario is the alternative solution. The external software should be opened from within Plenario and a new screen will appear on the user's screen. Having an external program might make it more challenging to post and retrieve information to and from the database on a technical aspect, however this should not affect user experience. Extra authentication will not be needed if the program is only accessible for users logged into the medication records.

3.1.3 Approach on Handling Databases

Whether or not the medication chart is to be fully integrated into Plenario, there is still the handling of patient data to take into consideration. As Infodoc's databases does not contain a table for handling the registration of distributing medicine, there are two possible approaches to handling this – storing the data in an external database or in one that Infodoc supplies.

3.1.4 Discussion of Alternative Approaches

While making a fully integrated module might improve the look and feel of the software, it does not necessarily make it better in terms of performance and overall user experience. After discussing this with Paal and two of the nurses from Gatehospitalet it was found that making an external program that works together with Plenario is the best solution. This is because it is considered easier to develop and therefore there will be more time to get the most important functionalities to work properly without making it any significant amount harder to use. The time spent on communication with Infodoc to establish the needed connectivity to their databases and other systems was a major factor in this decision. In a development process which is as short as this one, communication would have to be almost instantaneous, which is not realistic in such a project.

Regarding the data storage approaches it is favorable to have Infodoc supply a database for this purpose by either expanding the already existing one or creating a new one solely for the purpose of this data. By having an established software company handling the data, it will in almost all cases improve security and stability.

3.2 Specification

The main goal is to create a functioning program that works well together with Plenario. Even though it will not be fully integrated into the existing system it will look and feel the same way and therefore easy to operate. The users should be able to do the following:

- Open the software from within Plenario patient record.
- See what medicines and cures the patient is currently on.
- Sign off when the pill organizer has been prepared.
- Sign off when a medication has been taken by the patient, including the option of adding a comment.

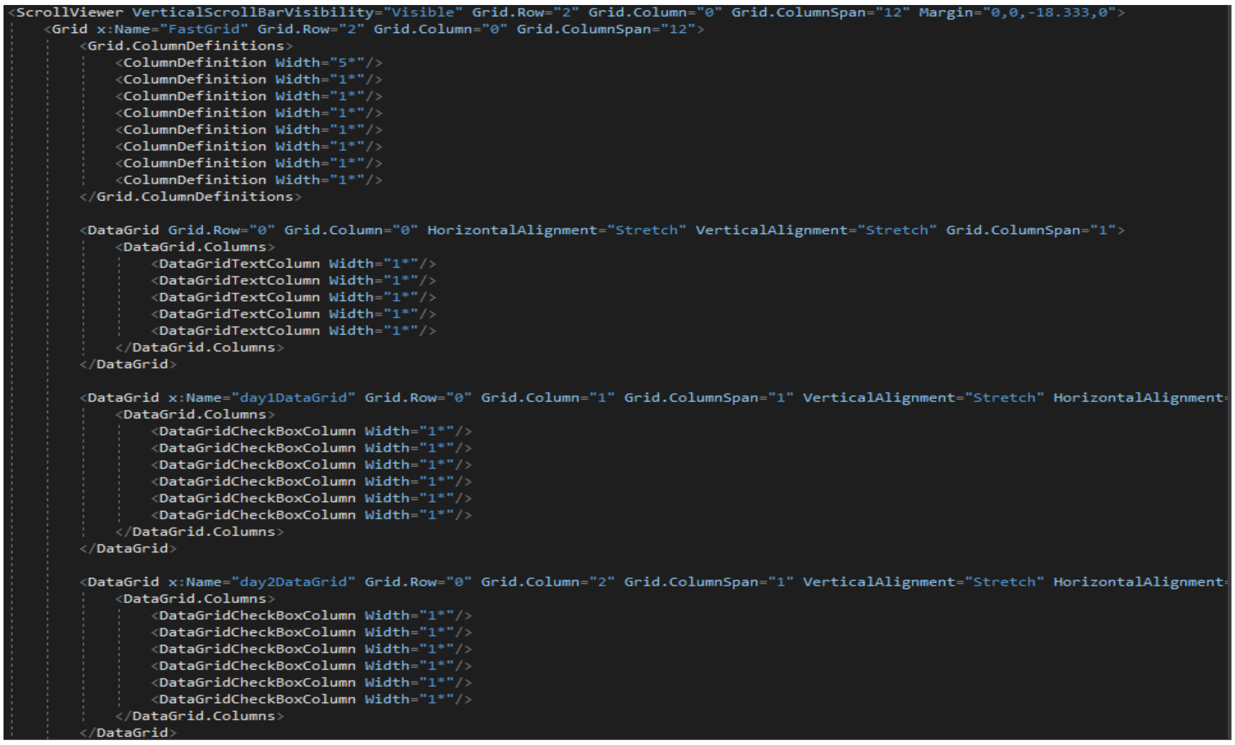
- Sign off that a medication has not been taken by the patient, including the option of adding a comment as to why.
- See previous records of medication by either scrolling or searching by date.

3.3 Selection of Tools and Programming Language

3.3.1 C# In Visual Studio

As Plenario is a .NET based application the languages available is either F#, C# or Visual Basics. This program will be developed in C# as it is the language the team have most experience in. The software used for developing will be Visual Studio as this is the standard .NET IDE.

At first it was decided that the program should be developed as a Universal Windows Platform (UWP) application, a platform for creating applications for the Windows operating system (Microsoft, 2020). The development of the GUI itself almost got to the point where it was ready for the data from the database to be loaded. However, the development in this environment came to a halt because of lack of knowledge on how to manipulate cells into showing data in the desired way. After getting feedback from professionals, it was decided that a switch to developing as a Windows Presentation Foundation (WPF) was the best way to go. This change was done mostly due to the fact that WPF supports DataGrids which is a XAML control that displays data in an easily customizable grid. A lot of the XAML code from the UWP application could also be copied directly so that the workload did not increase significantly.



```
<ScrollViewer VerticalScrollBarVisibility="Visible" Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="12" Margin="0,0,-18.333,0">
    <Grid x:Name="FastGrid" Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="12">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="5*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
        </Grid.ColumnDefinitions>

        <DataGrid Grid.Row="0" Grid.Column="0" HorizontalAlignment="Stretch" VerticalAlignment="Stretch" Grid.ColumnSpan="1">
            <DataGrid.Columns>
                <DataGridTextColumn Width="1*"/>
                <DataGridTextColumn Width="1*"/>
                <DataGridTextColumn Width="1*"/>
                <DataGridTextColumn Width="1*"/>
                <DataGridTextColumn Width="1*"/>
            </DataGrid.Columns>
        </DataGrid>

        <DataGrid x:Name="day1DataGrid" Grid.Row="0" Grid.Column="1" Grid.ColumnSpan="1" VerticalAlignment="Stretch" HorizontalAlignment
            <DataGrid.Columns>
                <DataGridCheckBoxColumn Width="1*"/>
                <DataGridCheckBoxColumn Width="1*"/>
                <DataGridCheckBoxColumn Width="1*"/>
                <DataGridCheckBoxColumn Width="1*"/>
                <DataGridCheckBoxColumn Width="1*"/>
                <DataGridCheckBoxColumn Width="1*"/>
            </DataGrid.Columns>
        </DataGrid>

        <DataGrid x:Name="day2DataGrid" Grid.Row="0" Grid.Column="2" Grid.ColumnSpan="1" VerticalAlignment="Stretch" HorizontalAlignment
            <DataGrid.Columns>
                <DataGridCheckBoxColumn Width="1*"/>
                <DataGridCheckBoxColumn Width="1*"/>
                <DataGridCheckBoxColumn Width="1*"/>
                <DataGridCheckBoxColumn Width="1*"/>
                <DataGridCheckBoxColumn Width="1*"/>
                <DataGridCheckBoxColumn Width="1*"/>
            </DataGrid.Columns>
        </DataGrid>
```

Figure 4 Example of XAML code with DataGrids in WPF.

WPF is a UI framework for creating desktop applications (Microsoft, 2018). It is a part of .NET and uses the Extensible Application Markup Language (XAML) for “defining and arranging GUI controls without any C# Code” (Deitel & Deitel, 2016). WPF can handle both audio and video as well as 2D and 3D graphics without incorporating any other technologies.

WPF makes graphics based on vectors rather than pixels. Vector-based graphics are defined by mathematical models and are resolution independent, which means that the quality stays the same even when the window is resized, or the resolution is changed.

Since WPF generates XAML markup when the designer and toolbox is used to make the GUI, it is easy to manually write the XAML markup to create the GUI. This allows for a clear separation between the program’s design and the logic behind.

3.3.2 Database

The database will have a conceptual design in the form of an ER diagram, that will be mapped to a relational schema. It will be created in Microsoft SQL Server. SQL Server is a relational database management system, or RDBMS from Microsoft Windows, and is built on top of SQL. (Sqlservertutorial.net, u.d.).

3.3.3 GitHub

For hosting the blog website and updating the code GitHub will be used. GitHub is an online service for version control and code hosting (w3schools, u.d.). It also has a service for hosting your own website which was used to host the development blog.

3.3.4 GUI Planning

Figma is used to create a sketch of the GUI. It allows the user to easily create, showcase and update the graphical user interface very quickly without having to write any code at all. Figma made it possible to collaborate on the design easy on the Web. It also made it possible to turn the static design into interactive prototypes (Figma, u.d.).

3.3.5 Web API

An Application Programming Interface, API, is an interface that allows different applications to talk to each other. A Web API can be accessed by many different HTTP clients, for example desktop and mobile apps. In this project ASP.NET is used to create a RESTful Web API.

Representational State Transfer (REST) is an architectural style that has become very popular for designing communicating applications (Kanjilal, 2013). It is often combined with the HTTP protocol to make stateless client-server applications. The client asks for data or resources with requests and the server provides the data with responses. REST uses the common HTTP verbs for CRUD (Creating, reading, updating and deleting). These include GET, PUT, POST and DELETE.

The data that is transferred between the client and server is usually in JSON or XML format. This Web API will be using JSON. JSON stands for JavaScript Object Notation and is a lightweight, text-based data interchange format (Joshi, 2019). It supports the most common datatypes, such as strings, integers, Booleans, arrays and objects. An object consists of multiple name-value pairs and is enclosed with curly-brackets. It is possible to transfer an array of objects with JSON by using square brackets around multiple objects.

3.4 Project Development Method

3.4.1 Development Method

The development process will be done iteratively to make sure that the end users get what they need and not what we think they need. Iterative development is a way of developing software that splits the project into multiple iterations that are like smaller projects. Through these iterations the application grows incrementally (Larman, 2004). The iterations will be one to two weeks long with feedback given directly by the staff at Gatehospitalet. More specifically each iteration will include planning at the start, getting the necessary information and/or access if needed, implementing the changes, testing the changes, and finally evaluation (see Figure 5).

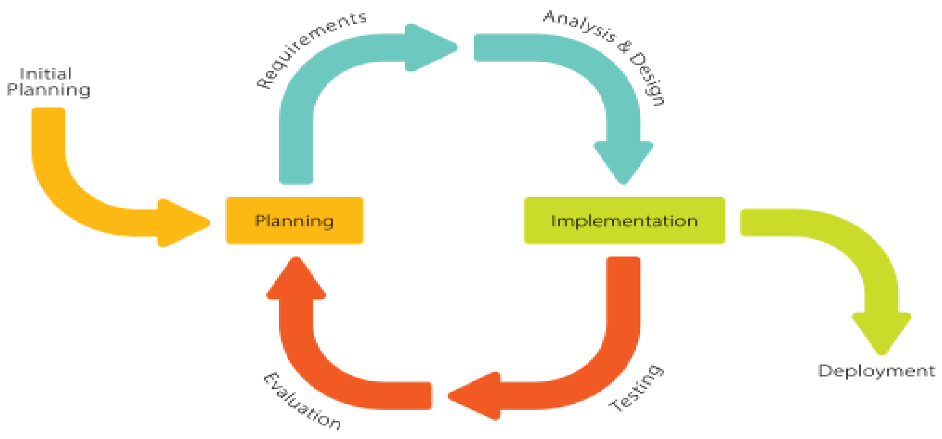


Figure 5 - Iterative development model, fetched at [https://commons.wikimedia.org/wiki/File:Iterative_Process_Diagram.svg] with license [https://creativecommons.org/licenses/by-sa/4.0/]

3.4.2 Project Plan

A GANTT diagram was made for mapping out the project. See Appendix B.

3.4.3 Risk Management

The risks involved vary greatly in both probability and consequence and therefore it is convenient using a risk matrix where each risk is measured individually in both aspects (Figure 6). Each case has a description which says something about what could go wrong, and what the consequences would be. All the individual cases have two values – one measuring the severity of the consequence, and one measuring the probability of it happening. The total risk value is calculated by simply multiplying the probability and consequence values, and the higher the risk value, the more it needs to be paid attention to. In the figure below you can see how risk values are considered according to their score. In addition to the risk matrix there are also thoroughly written precautions (Appendix A) to ensure that none, or at least as few as possible, of the consequences must be taken into consideration.

		Consequence				
		1 Negligible	2 Minor	3 Moderate	4 Major	5 Catastrophic
Probability	1 Very unlikely	1 Low	2 Low	3 Low	4 Medium	5 Medium
	2 Unlikely	2 Low	4 Medium	6 Medium	8 High	10 High
	3 Possible	3 Low	6 Medium	9 High	12 High	15 Severe
	4 Likely	4 Medium	8 High	12 High	16 Severe	20 Severe
	5 Expected	5 Medium	10 High	15 Severe	20 Severe	25 Severe

Figure 6 - Risk matrix to compute the risks of different scenarios. Made with inspiration from a figure on RecourceGate(Gulsum, 2018)

3.5 Evaluation Method

To test the system, the plan is to create automated test cases as well as trying out the system manually to check if the system does what it is intended to do. Having the actual end users try out the software is also a great way to get feedback from those who are meant to benefit from this module. In this case it will primarily be the nurses at Gatehospitalet, but the doctors will also give feedback and help with the design.

The manual testing performed by the end users will hopefully be performed in person at Gatehospitalet to make it easier for the people working there to be available. There will be three main steps to each test, and the goal is to be able to have one test at the end of each iteration.

1) **Presentation:**

The developers will present the program using language which is easily understandable. There will also be a short demo run through of the program to showcase how it works and what everything looks like.

2) **Testing:**

The testers will be given access to the program to try it out themselves with some dummy data. They will be able to test all the old (if any) and new features of the program after an iteration has ended.

3) **Evaluation:**

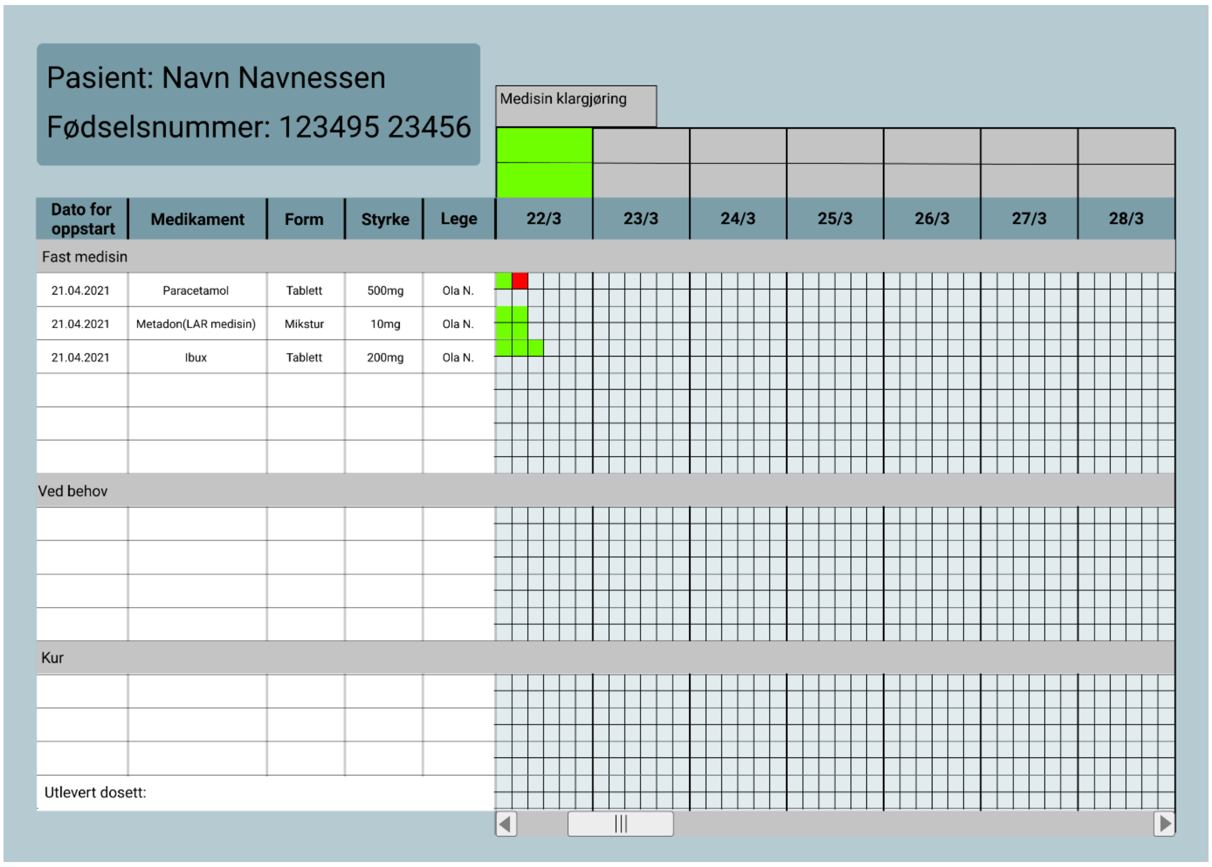
After the test has ended the testers will be given the opportunity to give feedback right there and then and it will be noted by the developers. In addition to that, there might be a written form they have to fill where the goal is to get a quick overview of how the program is performing, how easy it is to use and what needs improvement.

4 Detailed Design

4.1 Planning the Graphical User Interface

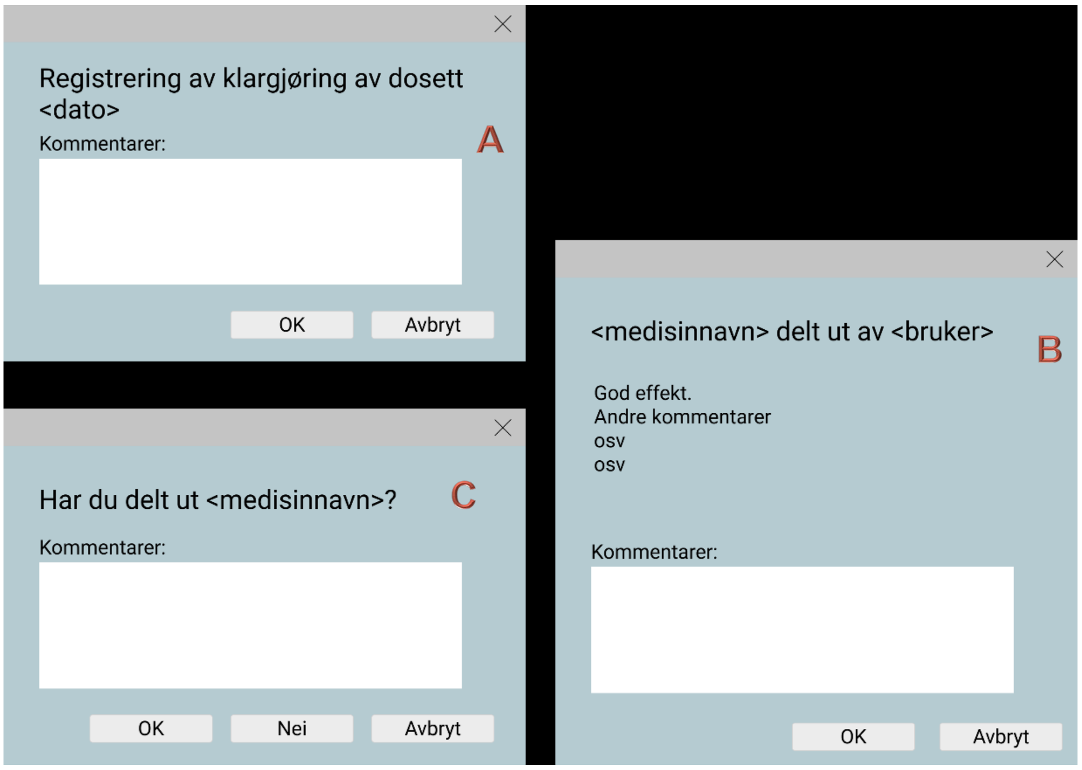
The first step of the project was to understand the requirements of the project and Gatehospitalets needs and wants for the program. From this an outline of the GUI was made using Figma. This outline was showed to the staff at Gatehospitalet, which then gave feedback and suggestions to changes. This process was repeated to make sure the GUI was understandable and containing all the elements needed.

In Figma there is also an option to simulate the flow of the program, with different windows and elements. This feature was used to ensure the right flow and that the functionality was easy to understand for the end users. The prototypes of the GUI from Figma is showed in Figure 7 and Figure 8.


Pasient: Navn Navnessen
Fødselsnummer: 123495 23456

Dato for oppstart	Medikament	Form	Styrke	Lege	22/3	23/3	24/3	25/3	26/3	27/3	28/3
Fast medisin											
21.04.2021	Paracetamol	Tablett	500mg	Ola N.							
21.04.2021	Metadon(LAR medisin)	Mikstur	10mg	Ola N.							
21.04.2021	Ibux	Tablett	200mg	Ola N.							
Ved behov											
Kur											
Utlevert dosett:											

Figure 7 - Outline of the distribution chart. Made in Figma.


Registrering av klargjøring av dosett <dato>
Kommentarer:
OK
Avbryt
A
<medisinnavn> delt ut av <bruker>
B
God effekt.
Andre kommentarer
osv
osv
Kommentarer:
OK
Avbryt
Har du delt ut <medisinnavn>?
C
Kommentarer:
OK
Nei
Avbryt
Figure 8 - Outline of the other windows in the program. Made in Figma.

The boxes that are colored green indicates that the medication has been distributed. If the box is red, it indicates that something might have happened, and the medication has not been distributed. To open the dialog boxes shown in Figure 8, the squares for the date and time one wants to register on have to be clicked. If a box that is not filled in is clicked, dialog box C pops up, but if a green or red box is clicked, dialog box B pops up. If one is to register the preparation of the pill organizer the box on top of the dates must be clicked, and dialog box A will appear.

In each of the dialog boxes, the only thing to fill out is the comments but this is also optional. When the OK button is clicked the distribution is registered and sent to the database (dialog box A and C). In dialog box B it is possible to add comments at a later time to a medication that already is distributed. This is especially beneficial with the medication that is only given when needed to document the effects of the drug.

4.2 Database

The initial plan was to use a database provided by Infodoc, which contained two fictional patients with test data. This would have made it easier to integrate the program with

Infodoc, but unfortunately, Infodoc stopped responding to e-mails and questions. Therefore, it was decided to make another database to simulate the data coming from Infodoc and Plenario.

To make this database an ER diagram was created and reviewed to make sure the data was mapped correctly. The ER diagram describes the data as entities, relationships and attributes (Elmasri & Navathe, 2011). Since the concepts in ER diagrams do not include implementation details, they are easier to understand for people without a background in computer science. This made it easy to discuss the diagram with the doctors and nurses at Gatehospitalet to be sure that the concepts and relations at the hospital was correctly mapped. Figure 9 shows the final ER diagram with the blue rectangle with the registration entity representing the distribution data, and the relationship with patient, prescription line and nurse, indicates what would be saved in a separate database. The rest of the data in the ER diagram should come from Infodoc's database.

A relational database schema was then made to map out the ER diagram for the database (Figure 10).

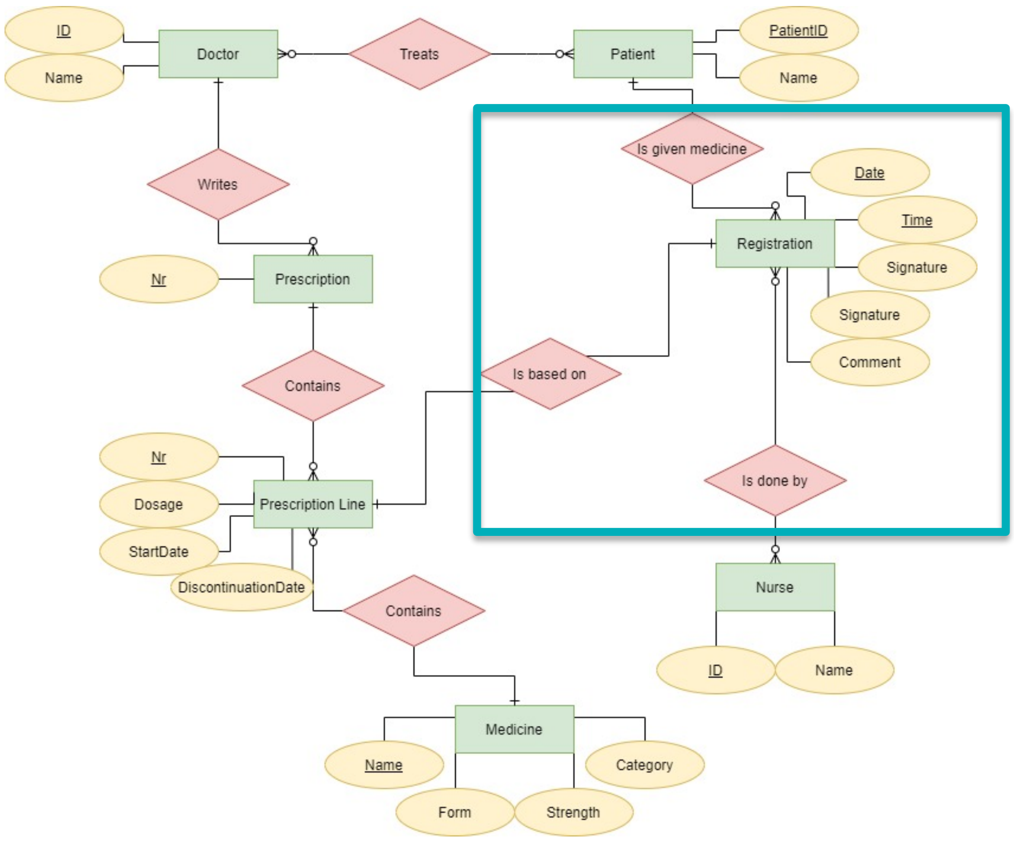


Figure 9 - ER diagram, made with draw.io, with the data to be stored in a distribution database.

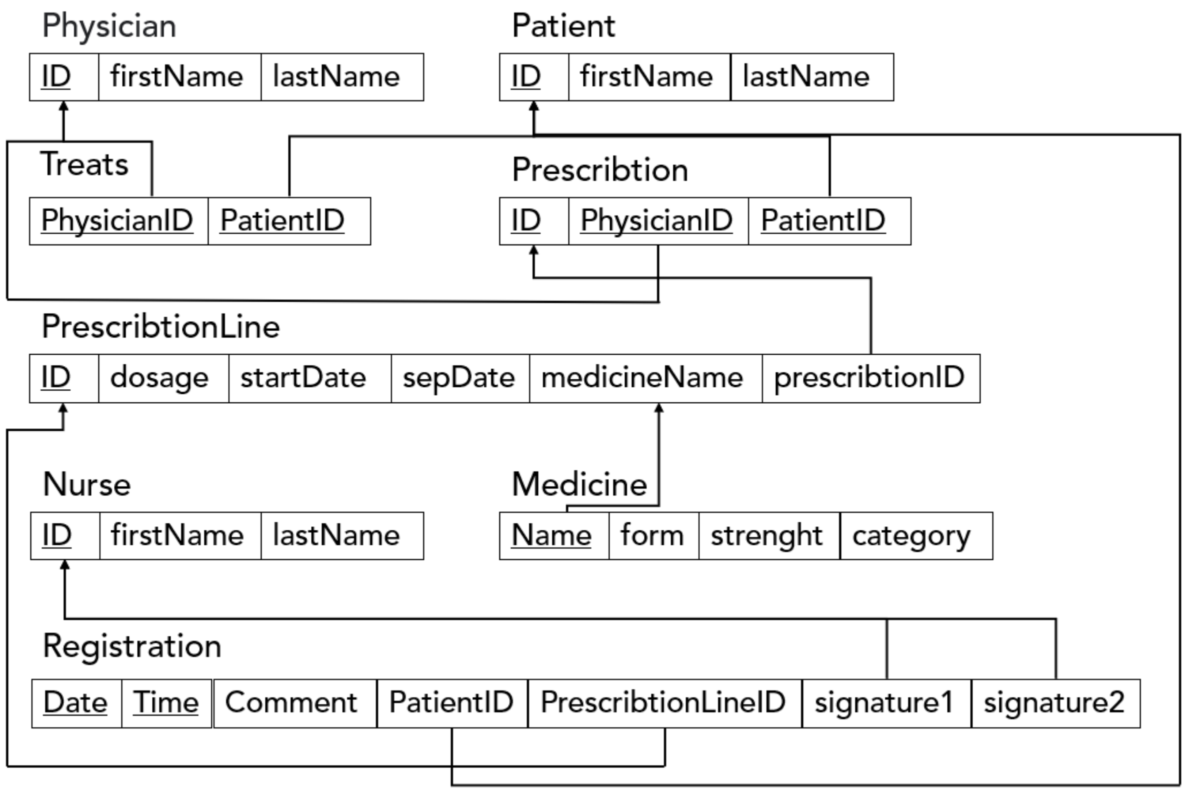
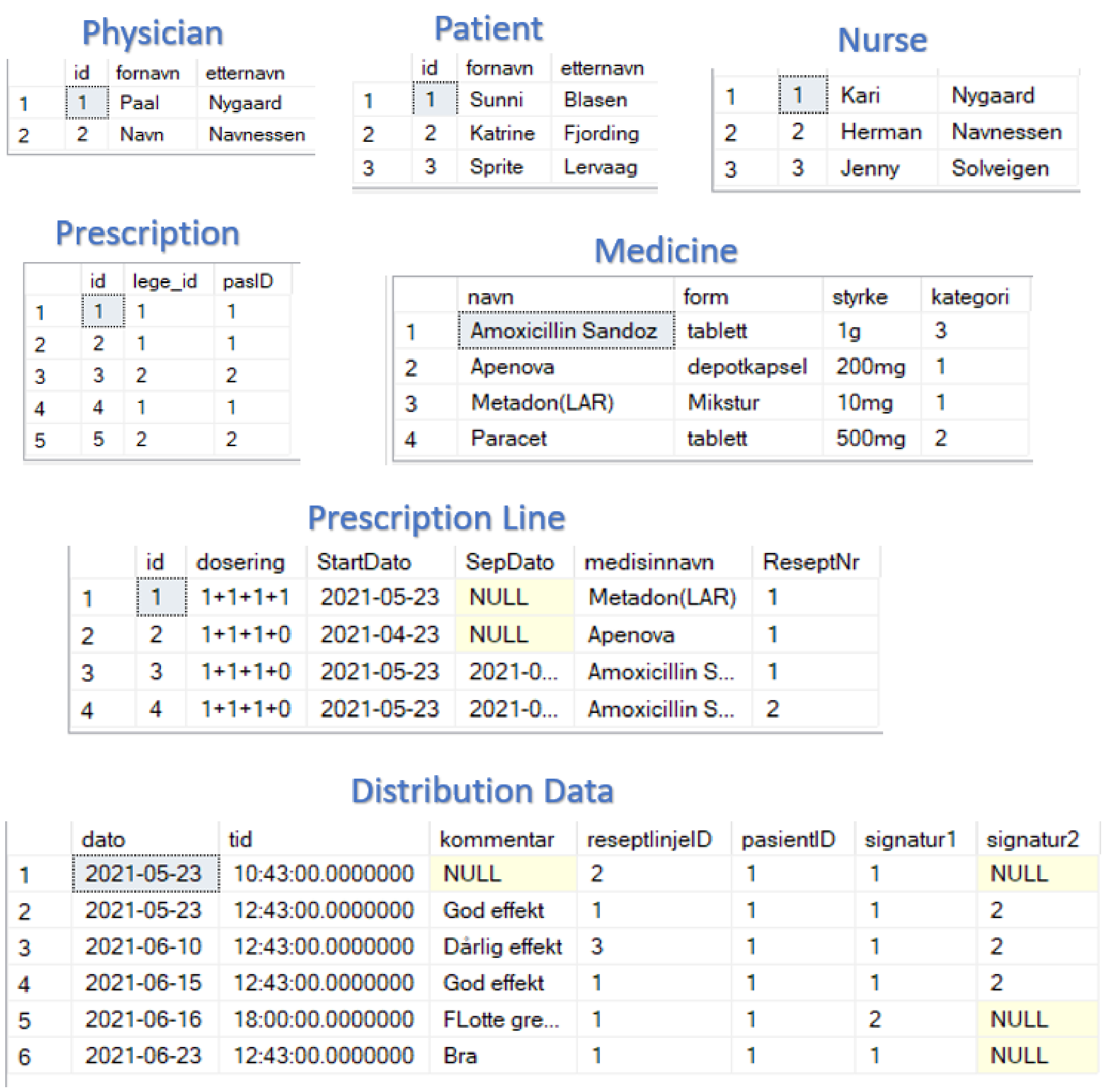

Physician
ID | firstName | lastName
Patient
ID | firstName | lastName
Treats
PhysicianID | PatientID
Prescribtion
ID | PhysicianID | PatientID
PrescribtionLine
ID | dosage | startDate | sepDate | medicineName | prescribtionID
Nurse
ID | firstName | lastName
Medicine
Name | form | strenght | category
Registration
Date | Time | Comment | PatientID | PrescribtionLineID | signature1 | signature2

Figure 10 - Relational schema

The next step was to create the actual database. This was done in Microsoft SQL Server with the ER diagram and schema above. Figure 11 shows the database tables with example data. This data is made up and is only used to illustrate how real data might look like.



Physician

	id	fornavn	etternavn
1	1	Paal	Nygaard
2	2	Navn	Navnessen

Patient

	id	fornavn	etternavn
1	1	Sunni	Blasen
2	2	Katrine	Fjording
3	3	Sprite	Lervaag

Nurse

1	1	Kari	Nygaard
2	2	Herman	Navnessen
3	3	Jenny	Solveigen

Prescription

	id	lege_id	pasID
1	1	1	1
2	2	1	1
3	3	2	2
4	4	1	1
5	5	2	2

Medicine

	navn	form	styrke	kategori
1	Amoxicillin Sandoz	tablett	1g	3
2	Apenova	depotkapsel	200mg	1
3	Metadon(LAR)	Mikstur	10mg	1
4	Paracet	tablett	500mg	2

Prescription Line

	id	dosering	StartDato	SepDato	medisinnavn	ReseptNr
1	1	1+1+1+1	2021-05-23	NULL	Metadon(LAR)	1
2	2	1+1+1+0	2021-04-23	NULL	Apenova	1
3	3	1+1+1+0	2021-05-23	2021-0...	Amoxicillin S...	1
4	4	1+1+1+0	2021-05-23	2021-0...	Amoxicillin S...	2

Distribution Data

	dato	tid	kommentar	reseptlinjeID	pasientID	signatur1	signatur2
1	2021-05-23	10:43:00.0000000	NULL	2	1	1	NULL
2	2021-05-23	12:43:00.0000000	God effekt	1	1	1	2
3	2021-06-10	12:43:00.0000000	Dårlig effekt	3	1	1	2
4	2021-06-15	12:43:00.0000000	God effekt	1	1	1	2
5	2021-06-16	18:00:00.0000000	FLotte gre...	1	1	2	NULL
6	2021-06-23	12:43:00.0000000	Bra	1	1	1	NULL

Figure 11 - The database tables with example data.

To show what data was needed from Infodoc's databases a data flow diagram (DFD) was made (see Figure 12). This includes information about the prescription, the medicine in question, the patient's name and ID, as well as information about the doctor and nurses. The DFD diagram also shows what data is to be stored in a distribution database, which include the date and time of the distribution, the prescription id and patient's id and up to two nurse ids.

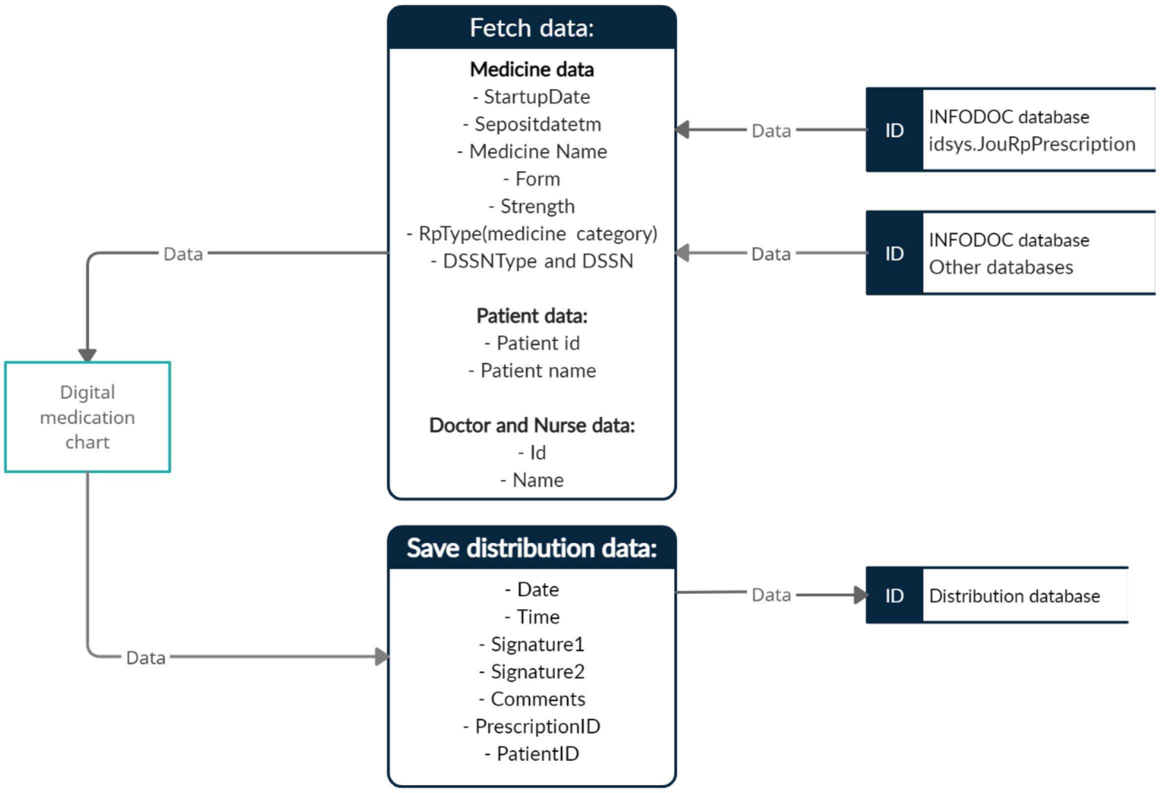


Figure 12 - Data flow diagram of the system

4.3 Windows Presentation Foundation (WPF)

The WPF App for this project is made using .NET (Framework), which means the GUI is designed in the declarative markup language XAML and the controls for it is written in C#. The project is meant to consist of four XAML classes, one for each of the windows needed for this application. Each of these have their respective C# classes for controlling their behavior once the program is running. Describing the XAML and C# files as separate classes working together is not completely right though, as a pair of XAML and C# files are actually two partial definitions of the same class (Microsoft, 2021). This makes developing easier than having everything in one main class as code readability is much better when the GUI elements and logic is separated in the IDE.

The GUI itself is composed of one big grid which has many columns and rows, making the resizing of the elements automatic when something is put inside the cells. The headers were created by wrapping Label elements inside Viewboxes to enable font resizing. The Gray horizontal bars are rectangles which are layered between the main grid and the viewbox.

DataGrids were used to fit the data onto the screen in the main parts of the GUI, however this turned out to be a bit challenging to get right. Alligning the DataGrid to the main grid could not be done by simply having it as part of the main grid as the columns turned out skewed when resizing the window. The solution turned out to be to create a Grid inside the main Grid to hold the DataGrids. Each of the three main parts are made up of eight DataGrids – one for the description of the medication and one for each of the seven days. Effectively that means that the columns on the “date-grids” needs to correspond with the rows on the “description-grids”.

There is also a “DatePicker”-object in the top right of the interface where it is possible to select a date of which to look at. The program will then fetch and display the data from the selected date and one week forward from that date.

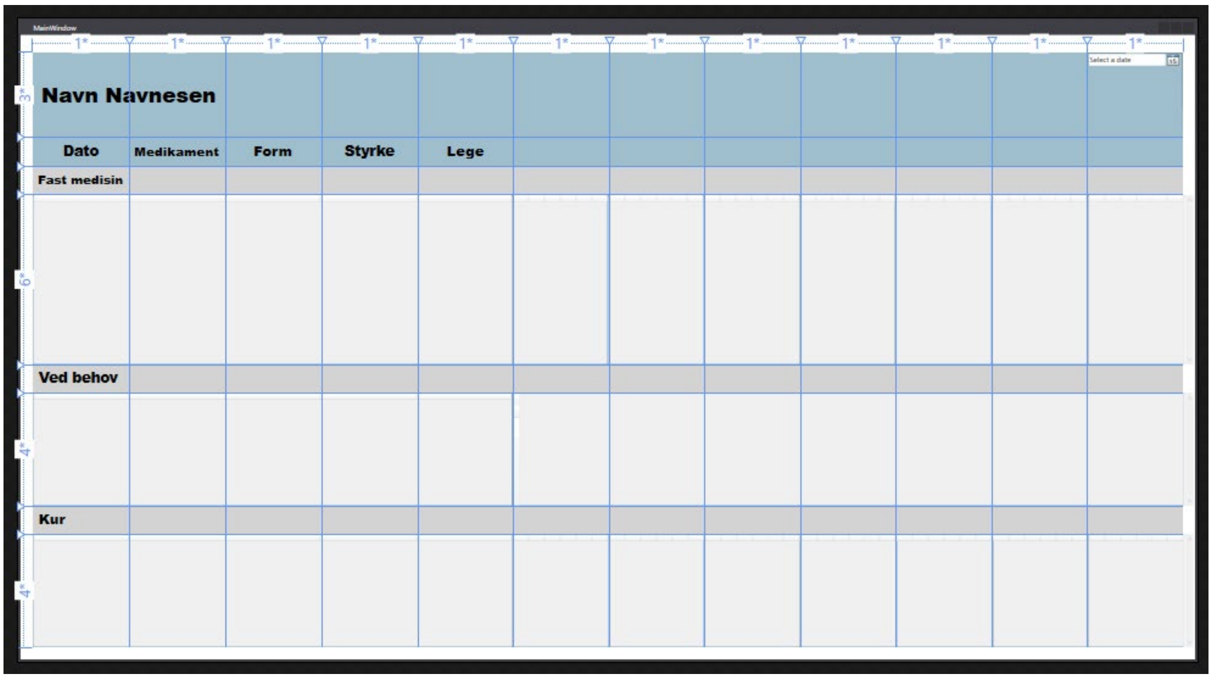


Figure 13 - Example of the GUI without any data loaded.

4.4 Web API

The Web API was developed using ASP.NET with Entity Framework. The solution consists of two projects, the data access class library, and the API. The data access class was connected to the database and then entities was generated from the database tables using Entity Data Model from Entity Framework. Figure 14 shows the generated data model with relationships and attributes.

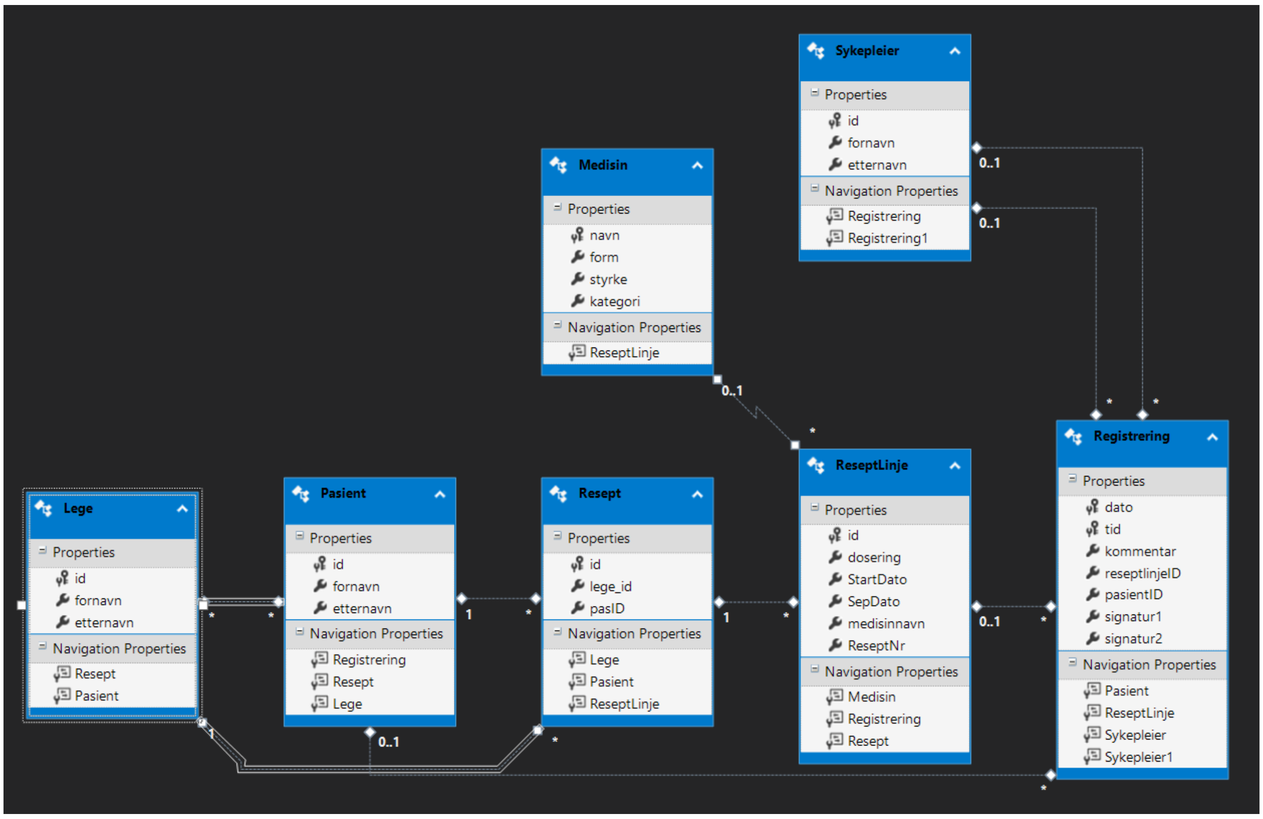
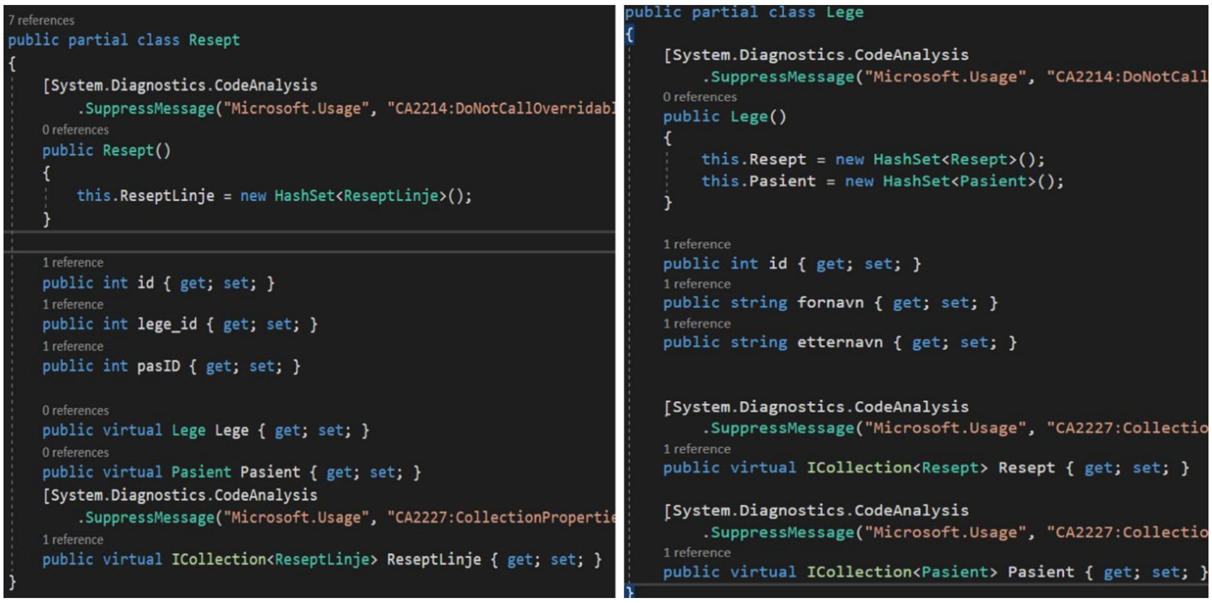


Figure 14 - Data Model generated from the database tables.

The model also generates C# classes with the attributes and relationships found in the database. This also creates a lot of virtual collections and attributes, that will make the API go in a loop when trying to fetch data.



```
7 references
public partial class Resept
{
    [System.Diagnostics.CodeAnalysis
        .SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridabl
    0 references
    public Resept()
    {
        this.ReseptLinje = new HashSet<ReseptLinje>();
    }

    1 reference
    public int id { get; set; }
    1 reference
    public int lege_id { get; set; }
    1 reference
    public int pasID { get; set; }

    0 references
    public virtual Lege Lege { get; set; }
    0 references
    public virtual Pasient Pasient { get; set; }
    [System.Diagnostics.CodeAnalysis
        .SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertie
    1 reference
    public virtual ICollection<ReseptLinje> ReseptLinje { get; set; }
}
```

```
public partial class Lege
{
    [System.Diagnostics.CodeAnalysis
        .SuppressMessage("Microsoft.Usage", "CA2214:DoNotCall
    0 references
    public Lege()
    {
        this.Resept = new HashSet<Resept>();
        this.Pasient = new HashSet<Pasient>();
    }

    1 reference
    public int id { get; set; }
    1 reference
    public string fornavn { get; set; }
    1 reference
    public string etternavn { get; set; }

    [System.Diagnostics.CodeAnalysis
        .SuppressMessage("Microsoft.Usage", "CA2227:Collectio
    1 reference
    public virtual ICollection<Resept> Resept { get; set; }

    [System.Diagnostics.CodeAnalysis
        .SuppressMessage("Microsoft.Usage", "CA2227:Collectio
    1 reference
    public virtual ICollection<Pasient> Pasient { get; set; }
}
```

Figure 15 - Prescription and Physician class to illustrate looping problem.

When creating a Physician (Lege) object the program will try to create a Collection of Prescriptions (Resept) (see Figure 14 for the classes in question). For each prescription in the collection, it will try to create a physician, which then will try to create a collection of prescriptions, etc. To solve this looping problem a [JsonIgnore] annotation was added to all the virtual data members. This entails that the data is not included, but it is not necessary to send all the patients and prescriptions when we are fetching a physician.

Furthermore, the Web API project is created, and the data access class library is added to it as a reference. To collect the right data that is to be displayed in the medication chart, a new class is added to represent one line in the chart. This class combines physician id, patient id, prescription id and info about the medication, such as name, form, strength, etc.

A ChartController class was made to fetch the chart data. In this class there is only one HttpGet method to collect the chart data for a given patient. Language Integrated Query, LINQ, is used to create chart objects for each of the prescription lines a given patient has. LINQ makes it possible to make queries that looks similar to SQL queries. It can be used to retrieve data from different data sources, such as lists, arrays and as in this case databases (Deitel & Deitel, 2016). A list of all the chart objects is then returned.

In addition to the chart data, data about previous distributions also needs to be fetched from the database. Therefore, another controller, DistributionController, was made. This controller will get the distribution data for each prescription line, and only the data from a given time period, again using LINQ. As of now, this time period is set to be between now (as in DateTime.Now) and seven days ahead.

Distribution data must be saved to the database, and for this purpose there is a POST method in the distribution controller. This takes in a distribution object in JSON format and adds it to the table in the database. Then it will try to save the changes to the database, but if another row in the database has the same date and time the method will return a Conflict.

There is also a PUT method that can be used to alter a row in the distribution table. This is needed if for example a nurse or doctor wants to comment on the effect of the medication after they have given it to the patient.

5 Evaluation

When evaluating a piece of software like this it is equally important to build the product right as it is to build the right product, and there are a couple of reasons for this:

- The program will handle extremely sensitive information, and so it is of great importance that none of the information is available anywhere else than where it is intended.
- It must be able to work together with other systems, mainly Infodoc Plenario, and so therefore it is important that it is built in a way where they can communicate easily.
- Another reason for creating this software is that the workers at Gatehospitalet, mainly the nurses, get an easier way to register medication that is given to the patients. Therefore, it should be built in a way that makes it very user friendly in terms of both the GUI and performing the tasks needed effectively.

5.1 Evaluation Method

Unit and module testing is important to ensure both the safety and functionality of the program. By using these kinds of testing it is possible to simulate situations which can be difficult to do by running user tests, such as a lot of people using the system at once. It is significantly more time efficient and does not require the developers to have anyone at hand to test the program for them, making it more available and flexible.

By using an iterative development method, the project is ensured to get multiple collections of feedback directly from the end users which can be applied towards the final product one piece at a time. See Figure 5 for a rough, visual representation of the development process.

More specifically the user tests have mostly been done by simulating the program in Figma (Figure 7 and Figure 8) and this is mostly due to the Covid-19 epidemic which has, on many occasions, made it very difficult to meet in person for the testing.

It was always the intention to use Figma as a tool for getting early feedback from the end users as this is extremely valuable information to have before starting programming the software itself. However, it was not the plan to be doing almost all the testing by using the sketch.

5.2 Evaluation Results

Evaluation is crucial if a good final product is to be delivered to the project owners in time. Therefore, it was decided upon frequent feedback from the end users themselves as well as unit testing done by the developers. By using two quite different testing methods to evaluate the software, it ensures that both the technical aspects and the user friendliness of the program will be satisfactory. Due to lack of time the unit testing was unfortunately not completed, but the many rounds of meetings with Paal Nygaard and nurses at Gatehospitalet provided great feedback when planning the GUI.

5.2.1 The First Evaluation

In a zoom meeting with the project leader at Gatehospitalet, a non-interactive sketch of the GUI was shown in Figma. The comments were valuable, and identified the following missing features:

- A comment section for each medicine.
- A comment section for each distribution of medicine.
- A column which says which doctor has prescribed the medication.

5.2.2 The Second Evaluation

The second evaluation was also done after a zoom meeting, but this time two nurses were also present. The end users seemed to understand the layout of the GUI clearly and several different options on how to solve the comments section where being discussed openly as there was not enough space on the main frame to always have them visible. A solution was decided upon where you can get comments on both the prescription and the distribution by clicking on their respective squares (Figure 7 and Figure 8). Other than that, there were still a couple of things that were missing:

- A section for cures.

- The ability to see past dates.

5.2.3 The Third Evaluation

Once again it was done over zoom due to the Covid-19 pandemic and this time an interactive Figma-model of the GUI was showcased. The feedback was very positive, and the GUI seemed completely satisfactory to the end users. The only challenge that came to mind was the number of doses of medicine that could be registered each day. This is a problem because some medications could be given up to six times a day, and it would need more squares to represent that. This could be made dynamic so that the number of squares changes based on the medicine, but that can potentially clutter up the GUI quite a bit. Neither solution was decided upon in this iteration, so the development of the actual software started with every medicine having six squares, which is the maximum amount.

5.2.4 The Fourth Evaluation

As with the previous evaluations it had to be done online due to the pandemic. An almost fully developed GUI was presented, and the feedback was positive. The response was that it was still easy to understand, which is natural considering it is almost an exact copy of the sketch shown in the previous evaluation meetings. This was unfortunately the last evaluation before the product was to be delivered, mostly due to delays because of bad communication from Infodoc’s side and both developers falling sick to Covid-19.

6 Results

Through many meetings with the doctors and nurses who provided a lot of good feedback, the outline of the GUI was made using Figma. Across several iterations of developing and presenting the sketch to the end users it got to the point where all involved parties were satisfied with the look and planned functionalities of it. Since then, this is something that has been used as a reference throughout the project.

In the end a piece of software was developed and many of the functionalities from the project description and meetings are present. The user interface is intuitive and easy to use, quick and responsive. Information about medication is being retrieved from the database through the RESTful API and can be displayed on screen in a test program. It is possible to select a date from the DatePicker-control and the dates on the GUI will update, starting on the selected date and displaying that and all the dates one week ahead in time. The one significant aspect that is missing in terms of the program itself is the registrations. It does not display on the GUI and it is not possible to register that a patient has been given his or her medication.

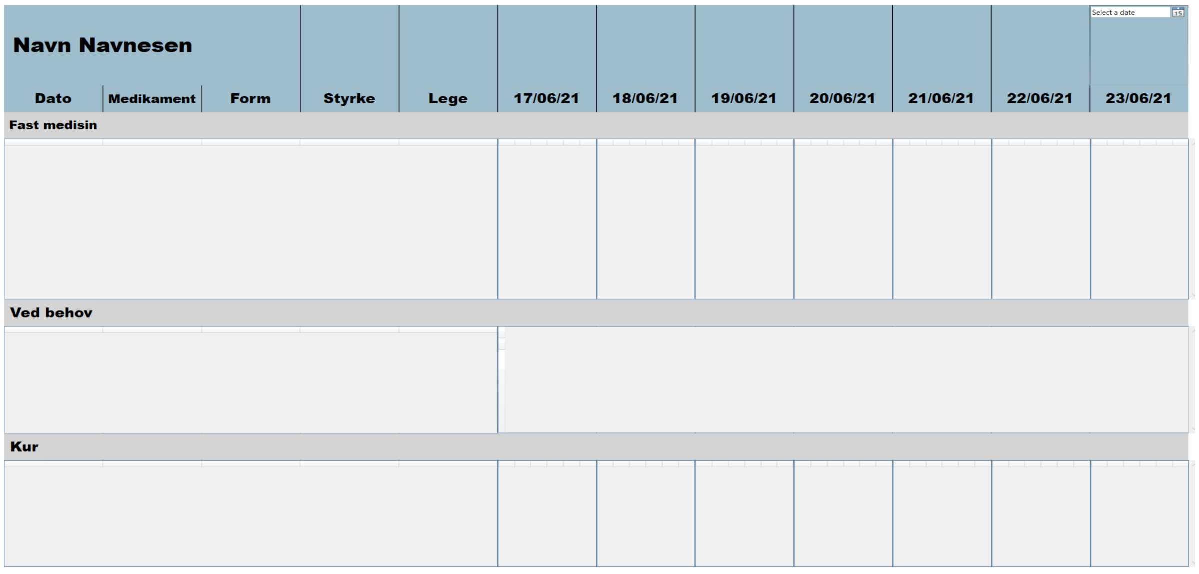


Figure 16 - GUI displaying selected dates.

The application is in no way connected to Infodoc's software or databases, so the only data seen when running the program is dummy data created by the developers of this application in a local database. This should not be difficult to change should that be of interest to Infodoc and Gatehospitalet since a fully working RESTful API has been created to send and retrieve data from the database. The database itself is modelled after the specifications Infodoc sent, so it should look similar to the one used now, except for the registration data which is supposed to be stored in a new database table Infodoc does not have yet.

7 Discussion

7.1 Limitations to the Work

Due to unforeseen circumstances many of the initial goals have not been reached and the scope of the project has changed from what it was initially. At the start of the project, it was envisioned that Infodoc would play a bigger role and have more time to aid the project. This would not actually be the case, as Infodoc ended up with a bigger workload than expected, and the developers there simply did not have the time to assist this project.

There was spent some time waiting for Infodoc to respond but in the end, it was decided to develop the program without their help. This entailed that the data that was supposed to come from Infodoc's database had to be simulated by making another database with corresponding data. Ideally it should not have been spent as much time waiting for a response from Infodoc, but rather start developing the software at an earlier time.

Not all the tools and technologies used in this project have been familiar to the team members. A great deal of time was spent on obtaining knowledge and information on different approaches to the various problems. If the tools and technologies had been chosen at an earlier time, less time would be spent towards the end of the project acquiring the knowledge needed, and there would have been more time developing the software.

Covid-19 has also played a big role on how far along the project has come. Sadly, both team members contracted the virus in mid-May. This resulted in about two weeks of little to none work, due to both members feeling the exhausting symptoms of the illness and had to stay at the Covid Hotel in isolation with poor working conditions. The aftereffects of Covid have also been noticed with the team getting tired much faster than usual, and therefore the periods of work have had to be shortened.

As a result of the limitations and situation there are some features missing from the application. An area to check off if the pill planner has been stocked for the day and the dialog boxes to register distribution are some of the things that have not been implemented yet. The functionality of adding a comment to an already registered distribution is also not finished.

7.2 Chosen Approaches

By using Figma to develop a prototype of the Graphical User Interface (GUI) and the functionality of the application, the end users could quickly give feedback on what worked and what needed to change. Meetings was held sometimes as frequent as weekly, to get the outline ready quickly. This resulted in a good plan of the GUI that matched the end user's expectation and made a guide for developing the GUI in a WPF application.

The development of the database took a lot of extra time since it is a complex database with a lot of tables and relationship. It was also important to get the entities and relations right, so the ER-diagram was reviewed with the doctors and nurses at Gatehospitalet multiple times to make it as correct as possible. This delayed the work on the actual program, and the development started very late.

Because of the delayed start on the development, there was not a lot of time to figure out and decide on what technology suited the application best. It was first chosen to use Universal Windows Platform (UWP), since Gatehospitalet had a tentative plan to purchase tablets for this purpose. This was later reviewed with some lecturers at HVL, and with their input it was agreed to switch to Windows Presentation Foundation (WPF) development. This of course means that in this round the program will not work on tablets, but due to time and knowledge limitations regarding UWP, WPF was still considered to be the best option.

The team agreed that to get the data from the database into the application, the best option was to make a Web API. This was decided before the decision on technology for the GUI was made, and a Web API makes it possible to use with a variety of technologies. This also made it easier to extract the right data on the right format. One disadvantage on choosing to make it was that the team lacked knowledge on the topic, but by using both online resources and lecturers at HVL that obstacle was overcome. A RESTful Web API developed with the HTTPS protocol also made the security around the data extraction better.

8 Conclusions and Further Work

8.1 The Goals

One of the goals of the project was good security regarding patient privacy. Since the end goal is to integrate the application in Plenario, the focus has not been as much on security as first assumed. By using a RESTful API with the HTTPS protocol, the transfer of data is secure. When it is implemented with Plenario the security should be good, but at this point the application is just a prototype with test data. The database should also ideally be integrated with Infodoc's databases, to make sure the security is good enough.

Good usability for the end users was an important goal, that got a lot of attention during the project. Through many meetings with the doctors and nurses at Gatehospitalet the outline of the GUI was made. They came with great feedback, that was used to alter the GUI before the actual development started. This was an important part of the project to make sure the look and feel of the application matched the end user's expectations.

Because of the choice on WPF the application does not currently work on both tablets and desktop. Since Gatehospitalet currently only have laptops, and not tablets, this is not a big issue now. With the Web API the application can collect the right data as well as saving new entries to the database.

8.2 Limitations and Deficiencies

As the previous sub-chapter describes, some of the goals have been reached but some have not. There are multiple reasons for this, one of them being that the project had a slow start with a lot of waiting for Infodoc to reply and give the information needed. When the decision about doing the project without the help from Infodoc came, the biggest limitation was the time. The workload of the project was significant as both the database and application needed to be created.

Another big reason for the project not being finished was that both team members contracted the coronavirus. This resulted in two weeks without much work in the middle of the project.

The application is missing a few details from the planned GUI. This include an area to check off if the pill planner has been stocked for the day and the dialog boxes to register distribution. The functionality of adding a comment to an already registered distribution is also not finished.

8.3 Further Work

This application can potentially be useful for many other health institutions that use Infodoc Plenario today, not only Gatehospitalet. Therefor it would be in many people's interest to see this application finished, and especially for the people at Gatehospitalet who currently only have a pen and paper solution to this problem. Earlier in the this report it is stated that no other customers of Infodoc had asked for something like this, but that does not mean it is not a welcomed addition of functionality for others as well.

The remaining functionality such as the pill planner, registration of given medicine and the dialog boxes needs to be implemented. Furthermore, the program should be integrated with Plenario to some extent to test the functionality on actual data. This means that it is necessary to make changes to the Web API to connect to the right database so that the right data can be retrieved. The distribution database should also be moved to Indofoc's systems to ensure good security of the data.

The GUI needs a visual update to better fulfill the expectations given from the Figma sketch to make it easier to work with as well as more pleasing to the eye. While updating the GUI it is important that it remains very clear and responsive.

9 References

Deitel, P. & Deitel, H., 2016. *C# 6 for programmers.* 6. ed. Boston, MA: Addison-Wesley.

Elmasri, R. & Navathe, S. B., 2011. *Database Systems.* 6th ed. s.l.:Pearson.

FDA, U. F. a. D. A., 2019. *Working to Reduce Medication Errors.* [Online]
Available at: https://www.fda.gov/drugs/information-consumers-and-patients-drugs/working-reduce-medication-errors
[Accessed 17.06 June 2021].

Figma, n.d. *Figma.* [Online]
Available at: https://www.figma.com/
[Accessed 29 03 2021].

Frelsesarmeen, n.d. *Frelsesarmeen - Gatehospitalet.* [Online]
Available at: https://frelsesarmeen.no/rusomsorg/gatehospitalet
[Accessed 13 April 2021].

Infodoc, n.d. *Infodoc.* [Online]
Available at: https://www.infodoc.no/tjenester/infodoc-plenario/?gclid=Cj0KCQjw9YWDBhDyARIsADt6sGalNbo0Dc9GJKSQ8sFXan7LYoSjwFUm-uFVwEWydbo5TBbm-YmKu-waAmsYEALw_wcB
[Accessed 20 03 2021].

Joshi, B., 2019. *Beginning Database.* s.l.:Apress.

Kanjilal, J., 2013. *ASP.NET Web API.* s.l.:Packt Publishing.

Larman, C., 2004. *Agile & Iterative Development: A Managers Guide.* s.l.:Addison-Wesly.

Microsoft, 2018. *Get started with WPF.* [Online]
Available at: https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2019
[Accessed 11 June 2021].

Microsoft, 2021. *XAML overview (WPF .NET).* [Online]
Available at: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-5.0
[Accessed 17.06 Juni 2021].

Noddeland, H., 2016. *Helse Sorost.* [Online]
Available at: http://admininfo.helse-sorost.no/digitalfornying_/Documents/Prosjektbegrunnelse%20Regional%20kurve%20og%20medikasjon%20v1%200.pdf
[Accessed 12 April 2021].

Sqlservertutorial.net, n.d. *What is SQL Server?.* [Online]
Available at: https://www.sqlservertutorial.net/getting-started/what-is-sql-server/
[Accessed 27 May 2021].

w3schools, n.d. *What is GitHub.* [Online]
Available at: https://www.w3schools.com/whatis/whatis_github.asp
[Accessed 17 June 2021].

Table of figures

10 Appendixes

Appendix A – Risk list

Description	Probability	Consequence	Risk	Precautions	Responsibility
Lack of communication with Infodoc	2	3	6	We will try to maintain continuous contact with our contact at Infodoc. If he should stop responding to us, we will contact Paal Nygaard who has helped us with this issue before.	Sunniva
The architecture of the module is unclear.	3	4	12	We will look at the existing architecture in Plenario and plan our module and architecture well with UML diagrams.	Alvin
Inadequate communication with the database	3	5	15	We will make diagrams of how the system should work, and fetch data. As well as early implementation of the module to catch problems early.	Sunniva
Loss of backup of our code	5	1	5	Always have a backup of the backup and store everything in a GitHub repository.	Alvin
Illness in the group	2	3	6	We will continue taking proper precautions regarding infectious deceases, as we have learned this last year. But if this should occur, we need to use zoom and try to communicate well.	Sunniva and Alvin
If the school closes again	3	2	6	This is out of our control, but if this happens, we will have at least one zoom meeting at the beginning of the week and one at the end. Then we can plan and review our progress and	-

				discuss elements of the project.	
Bad communication with the end users	3	4	12	We need to maintain the good communication we already have with Gatehospitalet, as well as have short iterations and show the program regularly to get input from the end users.	Sunniva

Appendix B – Gantt Diagram

Digitalization of the medicine chart at Gatehospitalet in Bergen

D14

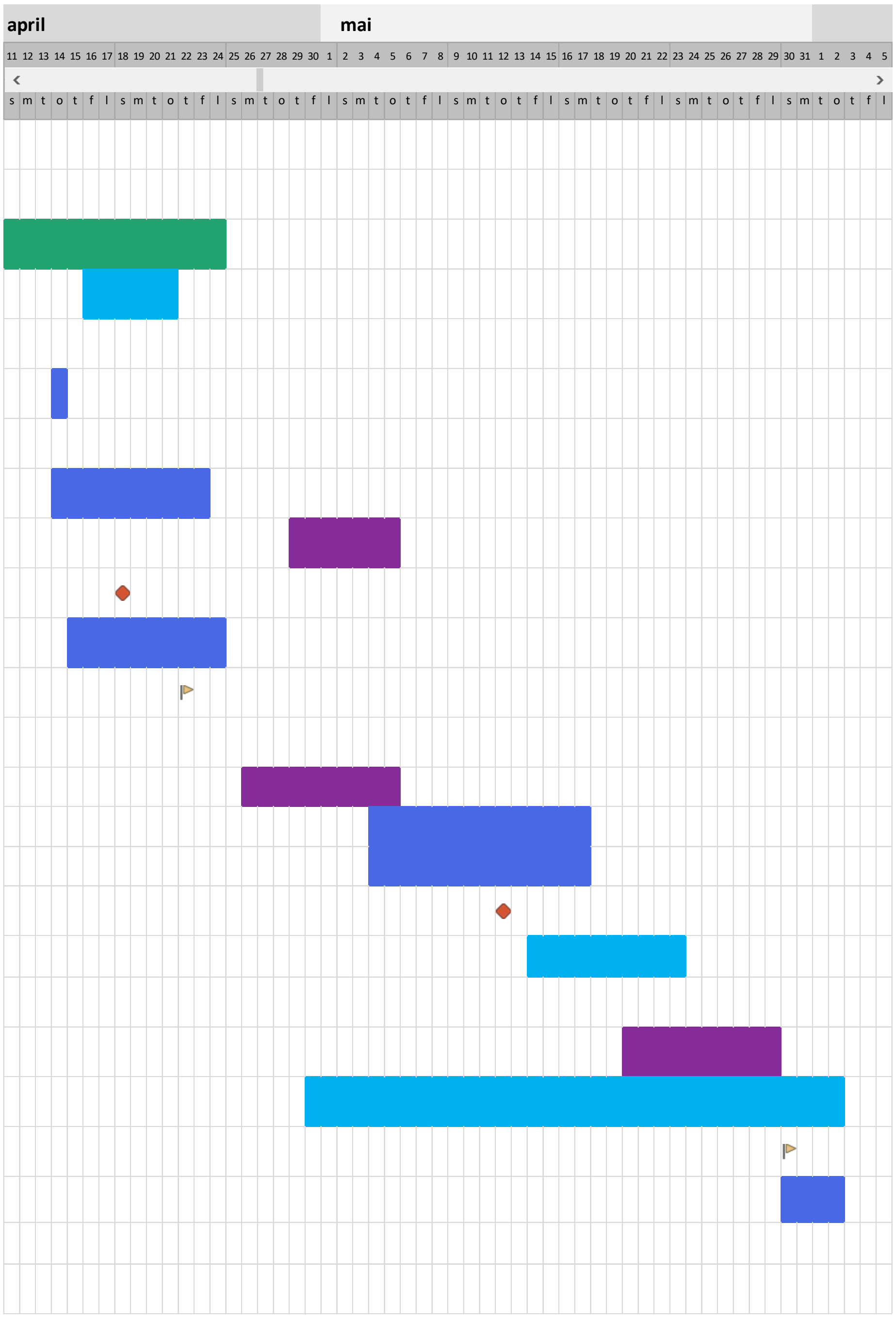
Project lead

Project Start Date:

Scrolling Increment:

Legend: 🟩 On track 🟦 Low risk 🟦 Med risk 🟪 High risk

Milestone description	Progress	Start	Days
Startup			
OA 1-OA 6	100 %	01.01.2021	60
Goal and method	100 %	10.04.2021	15
Make a draft of the solution	100 %	16.04.2021	6
Establish contact with Infodoc	100 %	22.03.2021	7
Present the draft for Gatehospitalet	100 %	14.04.2021	1
Planning and writing			
Update draft with input from the end users	100 %	14.04.2021	10
Model the databasesystem	100 %	29.04.2021	7
Pre-project report	100 %	18.04.2021	1
Make UML diagrams of the system	100 %	15.04.2021	10
Preproject presentation	100 %	22.04.2021	1
Developing			
Connect to the database	100 %	26.04.2021	10
Make the GUI	50 %	04.05.2021	14
Write backend code	50 %	04.05.2021	14
Present the prototype for Gatehospitalet		12.05.2021	1
Update the module		14.05.2021	10
Writing the report and compliting the program			
Integrate the module with Infodoc	0 %	20.05.2021	10
Write remaining chapters in the report	100 %	30.04.2021	34
Present the module for the end users		30.05.2021	1
Update the module	100 %	30.05.2021	4
Submit the report	100 %	18.06.2021	1
Presentation of the final report		29.06.2021	1


april
mai
s m t o t f l s m t o t f l s m t o t f l s m t o t f l s m t o t f l s m t o t f l s m t o t f l s m t o t f l