

FMH606 Master's Thesis 2021

<IIA>

The development of a metering reporting system



The screenshot displays the website interface for INEOS and inovyn. At the top, the INEOS logo is on the left, and navigation links for 'Info', 'Manuell registrering', 'Perioderapporter', 'Årsrapporter', and 'Innlogging' are on the right. Below the navigation bar, the INEOS logo and 'Olefins & Polymers Europe' are on the left, and the inovyn logo with 'An INEOS company' is on the right. The main heading is 'Metering rapporter', followed by the subtitle 'Ny web applikasjon for metering rapporter'. A large aerial photograph of an industrial facility with a ship docked at a pier is centered below the text. At the bottom of the screenshot, the INEOS and inovyn logos are repeated.

Erlend H. Hestnes

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

Course: FMH606 Master's Thesis, 2021

Title: The development of a metering reporting system

Number of pages: 93

Keywords: Study of programming languages, Development process, Web Application, React, C#, design of an application.

Student: Erlend H. Hestnes

Supervisor: Hans-Petter Halvorsen

External partner: Ineos Rafnes

Summary:

The current Metering Reports application at Ineos Rafnes and Inovyn will be outdated when Microsoft ends internet explorer support. No commercially available off-the-shelf product seems to be available for a new Metering Reports application. Hence a new custom-made solution shall be developed.

Ineos Rafnes and Inovyn want a solution that pursues the following design for a new Metering Reports application: 1) A one to one replacement that is sustainable for the future 2) A modern application that can be easily modified and maintained 3) An application developed in programming languages known and used by employees of Inovyn and/or Ineos Rafnes.

A prototype for a new custom-made solution Metering Reports application has been created based on the current system and user input. A programming language study has been conducted to ensure a sustainable and enduring alternative is selected for the prototype. The Scrum method has been utilized to develop a new prototype where the aim has been to develop a new version of the prototype per iteration. Each version of the prototype has been deployed on a server and tested by the current application users.

The result of the development process is a flexible design. The flexibility is arranged in the different operations by having one process for performing desired result dependent upon the information. Resulting in the maintenance and modifications are a more straightforward procedure.

The prototype is a custom-made solution that can replace the current system when it is completed. In addition, the prototype has the potential of being a better system in terms of functionality than the current system.

Preface

Choosing development as a topic for the master thesis has its foundation in the advancement of processes is an eternal chapter.

The motivation for the master thesis is to enhance and empower the skills needed for developing new web applications.

Thanks to Ineos Rafnes for making it possible to complete the industrial master's program.

A special thanks to:

- Helle Manger, for taking the initiative for the master thesis task. The invaluable support and guidance throughout the whole process of the master thesis.
- Ole Petter Gusfre, for being interested and giving valuable input for the project
- Tommy Borgen, for the inspiration and facilitate the needed part for the project.

Porsgrunn, 18.05.21

Erlend H. Hestnes

Contents

Preface	4
Contents.....	5
Nomenclature	8
1 Introduction	9
1.1 Background	9
1.2 Problem description	10
1.3 Scope	10
1.4 Intended Audience.....	10
1.5 Thesis Structure.....	10
2 Working methods.....	11
2.1 Data collection	11
2.2 Scrum	11
2.3 Unified Process.....	11
3 System requirement specification.....	12
3.1 Main requirements	12
3.2 Introduction to the current system	12
3.2.1 <i>Monthly report</i>	13
3.2.2 <i>Update a Monthly report</i>	14
3.2.3 <i>Yearly report</i>	16
3.2.4 <i>Manual registration</i>	16
3.2.5 <i>Login</i>	17
3.3 Databases	18
3.4 Functional Design Specification	18
4 Selection Phase	20
4.1 Definitions	20
4.2 Web application	21
4.3 Web browser	21
4.4 Web server.....	22
4.5 Programming language.....	23
4.6 Client-side programming language	25
4.6.1 <i>Website design pattern</i>	26
4.6.2 <i>SPA Options</i>	27
4.6.3 <i>Choice</i>	28
4.7 Server-side programming language	29
4.7.1 <i>Framework</i>	30
4.8 Interface	30
4.9 Development tools.....	30
4.10 Changes.....	31
4.10.1 <i>Utilize the Phdapinet</i>	31
4.10.2 <i>Communication with UPHD</i>	32
5 Design	33
5.1 System	33
5.2 Architecture.....	33
5.3 Use cases	34

5.4 Design patterns.....	34
5.5 UI design.....	36
5.6 First draft.....	36
5.7 Development plan.....	36
6 Prototype.....	38
6.1 Get a monthly report.....	38
6.2 Update a monthly report.....	40
6.3 Check stored values.....	40
6.4 Get a yearly report.....	41
6.5 Manual registration.....	42
6.6 Login.....	43
7 Frontend.....	45
7.1 React.....	45
7.2 The structure.....	46
7.3 Navigation.....	46
7.4 Functions.....	47
7.4.1 Side menu.....	47
7.4.2 Fetch.....	47
7.4.3 Presenting table with data.....	48
7.5 Components.....	49
7.5.1 Monthly report.....	49
7.5.2 Yearly report.....	51
7.5.3 Manual registration.....	51
7.5.4 Login.....	52
8 Developed Backend.....	53
8.1 Structure.....	53
8.2 Operations.....	55
8.3 Functionality descriptions.....	58
8.3.1 Access Layer.....	58
8.3.2 Business layer.....	59
8.3.3 Data access layer.....	60
8.3.4 Parameters.....	61
8.4 Communication with UPHD.....	61
8.4.1 Posting data.....	61
8.4.2 Extraction of data.....	62
8.4.3 Data storage.....	62
8.4.4 Filtration.....	62
9 Testing.....	65
9.1 UI.....	65
9.2 web API.....	65
9.3 Version test.....	65
9.4 User test.....	65
10 Maintenance.....	66
10.1 Version updating.....	66
10.2 Modifications.....	67
10.3 Change of reports.....	67
11 Comparison.....	68
11.1 Technology.....	68
11.2 Design.....	68

11.3	Performance	68
11.4	Maintenance	69
11.5	Functionality	69
11.6	Status	70
11.7	Security	70
11.8	Future work	71
12	Discussion	72
12.1	Study for selection	72
12.2	Development process	73
12.3	Frontend development	74
12.4	Backend development.....	74
12.5	Undeveloped function	75
12.6	Testing	75
12.7	Documentation.....	76
13	Enhancement	77
13.1	Database	77
13.2	Correction calculation.....	77
13.3	Automated verification.....	78
14	Conclusion	79
	References	80
	Appendices	87

Nomenclature

ASP	Active Server Pages
API	Application Programming Interface
COTS	commercially available off-the-shelf
CSS	Cascading Style Sheets
DLL	Dynamic-Link Library
FDS	Functional Design Specification
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IIS	Internet Information Services
JSON	JavaScript Object Notation
MPA	Multi-Page Application
OS	Operating System
SEO	Search Engine Optimization
SPA	Single Page Application
UI	User Interface
UPHD	Uniformance Processes Historian Databases
URL	Uniform Resource Locator
XSS	Cross-Site Scripting

1 Introduction

At Ineos Rafnes, there are currently many different Information Technology (IT) solutions in use. Some IT are specific for Ineos Rafnes, while others are shared with other companies inside the Ineos Group. As part of daily operation, most of these IT are used and maintained. Whenever a change of an IT is happening, the rest of the IT bound to this IT has to adapt to one that has changed.

Microsoft has announced that Internet Explorer will not be supported after 17 of August 2021. [1] This implies IT using ActiveX, [2] and VBScript [3] for presentation in a web browser needs to be modified and upgraded to present in another web browser or be replaced. Currently, Ineos Rafnes has an IT that is currently affected by this change, is called Tarragon. The User Interface (UI) of Tarragon relies on Internet Explorer.

1.1 Background

The Tarragon application is used for two purposes at Ineos Rafnes and Inovyn; 1) process overview and main production parameters at business level and 2) fiscal metering reporting. Ineos Rafnes and Inovyn are looking for replacements for these applications and this report describes the development of a new application for the fiscal metering reporting.

The metering reports are the basis for financial settlement for products being imported and exported to the plants at the Rafnes site. Daily and monthly totals are verified and approved via the metering report application. All data are collected from and stored back to the process historian databases for Ineos Rafnes and Inovyn.

The purpose of the Metering Reports is to display monthly and yearly reports of products being imported and exported to the plants. Each of the monthly reports is used to approve the previous day's measurements for flows of products. When a month has passed all the monthly report needs to be approved, because they are used in a financial settlement between Ineos Bamble, Ineos Rafnes, Inovyn, and Yara. Ineos Rafnes and Inovyn are the two companies that perform the different flows of products and approve the monthly reports. At each Site, key personnel maintains the measurement equipment and approves the monthly reports they are in charge of.

The Metering Reports web application consists of "Avregning" as the frontend and "TaraStore" as the backend. It was created around 2005 and has been used since then and maintained by a contractor. The technology used for the frontend is Microsoft Active Server Pages (ASP) [4]. Each of these reports has its ASP file that contains:

- ASP structure for logic
- HyperText Markup Language (HTML) [5] for static presentation
- JScript [6] and VBScript [7] for dynamic presentation
- Cascading Style Sheets (CSS) [8] for styling.

Both JScript and VBScript are dependent on components included in the web browser Internet Explorer 11 [11] for running.

TaraStore is currently integrated with two plant historian databases delivered by Honeywell. Inovyn will change their process historian to a database from AspenTech.

1 Introduction

Ineos Rafnes and Inovyn are searching for a commercially available off-the-shelf (COTS) product to replace the current Metering Reports application. The arguments for a COTS product are that the software is updated and maintained by a supplier, and it is not dependent upon specific developers/persons. A couple of suppliers have been asked if they have a suitable application. However, none of them have shown a solution covering all the required functionality. A new custom-made solution is therefore being considered.

1.2 Problem description

No COTS product seems to be available for a new Metering Reports application. Hence a new custom-made solution shall be developed. Ineos Rafnes and Inovyn want a solution that pursues the following design for a new Metering Reports application:

- A one to one replacement that is sustainable for the future
- A modern application that can be easily modified and maintained
- An application developed in programming languages known and used by employees of Inovyn and/or Ineos Rafnes

1.3 Scope

The task for this master thesis is to design and develop a new Metering Reports application. The scope for this master thesis is the following:

- Perform a study on different programming languages for developing web applications with a focus on evaluating which of them will endure into the future.
- Set up a development schedule based on given information for a new metering reporting system, and make it open for changes as the development process unfolds.
- Create a web application for the presentation of the metering reports and test it in cooperation with the users of the system.
- Integrate the web application for presentation of the metering reports with the Processes Historian Database system at Ineos Rafnes.
- Test the new metering reporting system and create a test report.
- Create design and user documentation of the new system.

By reformulating the tasks for the thesis, it can be viewed as a study of programming language and a full-stack development of a web application for a customer.

1.4 Intended Audience

The master thesis is intended for colleagues at Ineos Rafnes.

1.5 Thesis Structure

The structure of this master thesis is based upon the IMRaD.

2 Working methods

The utilized working methods for the development process are Data collection, Scrum, and Unified Processes.

2.1 Data collection

Data collection has been elected to perform the study on the programming languages. Data collection can be view as a mixed study of both Quantitative and Qualitative data, perform as a survey [9, pp. 204-206]. Quantitative data has been used to determine the viable options to choose. Qualitative data has been used to present the advantages and disadvantages of the viable options.

2.2 Scrum

Scrum is an agile development method. The agile methods emphasize face-to-face communication over written documents. Scrum advocates a team size of 5 to 9 and consists of inspections and sprints. Inspections mean checking the work, and the goal is to detect undesirable. A sprint is the timeframe of 14 days or a month to accomplish the set development plan. The cycle of the Scrum method is first a sprint, then an inspection, and then it iterates as the project continuous [10, p. 21]. Scrum has been selected as the primary method for the development method.

2.3 Unified Process

The Unified Process is an iterative and incremental software development process framework [10, p. 23]. The used methods from the Unified Process framework are:

- Take care of the most important, and often the riskiest areas
- Continuous check of quality with much testing (in each iteration)
- Engage the users for evaluation, feedback, and requirements.

3 System requirement specification

A new custom-made solution requires a set of defined requirements for development.

3.1 Main requirements

The problem description is translated into useable requirements for the development process.

1. A one to one replacement that is sustainable for the future

The term one to one replacement gives the requirement of all the current functionality shall be replacement. The statement sustainable for the future gives that the replacement shall be useable for some time into the future.

2. A modern application that can be easily modified and maintained

A modern application gives a requirement of modern technologies shall be used. Easily modified means that it shall be simple to make a change to the application. Easily maintained says that it shall be simple to update the application.

3. An application developed in programming languages known and used by employees of Inovyn and/or Ineos Rafnes

When it comes to a custom-made solution, someone creates it, so the expertise should be in-house to maintain it. The chosen programming languages shall be known to some employees of Inovyn and/or Ineos Rafnes. So, the maintenance can be performed by them.

Furthermore, each of the scope statements gives out what is to be achieved and guidelines for the development process. The scope can be viewed as an extension of the requirements above.

3.2 Introduction to the current system

The current metering system is the foundation of functionality due to the one to one replacement. An introduction to how the current system works and looks is required. For showing what is to be developed as a replacement for it. Figure 3.1 The front of the current system.

3 System requirement specification



Figure 3.1 front of the current system.

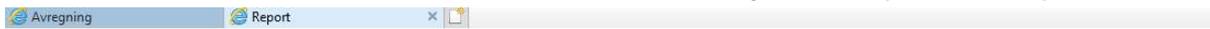
3.2.1 Monthly report

A monthly report can be selected from the folder "Perioderapporter." Once a link for a month report is clicked, a new tab is opened to select the time interval of the selected monthly report. Figure 3.2 shows the new tab for the selection of time intervals of the selected monthly report. The monthly report is fetched when the button "Hent rapport" is clicked. Figure 3.3 displays the monthly report of the chosen monthly report. The comment of daily measurements can be viewed as the link reference at the bottom of the page while holding over the date.



Figure 3.2 the new tab for the selection of time intervals of the selected month report.

3 System requirement specification



Perioderapport: Matevann til VCM

Dato	Egetforbruk Noretyl	FIQ8305	FIQ012	FIQ012-FIQ8305		FIQ8305	
	tonn	Råverdier tonn	tonn	tonn	Relativt avvik (4%) %	Korreksjon tonn	Verifisert tonn
01.05.2021	319,316	1966,183	1954,438	-11,745	-0,60	0,000	1966,183
02.05.2021	321,912	1795,659	1783,125	-12,534	-0,70	0,000	1795,659
03.05.2021	394,311	1209,075	1194,438	-14,637	-1,21	0,000	1209,075
04.05.2021	2351,231	1083,281	1067,375	-15,906	-1,47	0,000	1083,281
05.05.2021	2810,210	1453,618	1438,250	-15,368	-1,06	0,000	1453,618
06.05.2021	2677,330	1660,502	1646,250	-14,252	-0,86	0,000	1660,502
07.05.2021	1454,232	1955,822	1942,250	-13,572	-0,69	0,000	1955,822
08.05.2021	1114,524	1727,604	1714,250	-13,354	-0,77	0,000	1727,604
09.05.2021	1108,836	1577,877	1565,250	-12,627	-0,80	0,000	1577,877
10.05.2021	1116,962	1566,592	1553,813	-12,780	-0,82	0,000	1566,592
11.05.2021							
12.05.2021							
13.05.2021							
14.05.2021							
15.05.2021							
16.05.2021							
17.05.2021							
18.05.2021							
19.05.2021							
20.05.2021							
21.05.2021							
22.05.2021							
23.05.2021							
24.05.2021							
25.05.2021							
26.05.2021							
27.05.2021							
28.05.2021							
29.05.2021							
30.05.2021							
31.05.2021							
Forelepig sum	13668,863						15996,213
Måneds korreksjon							
Sum	13668,863	15996,213	15859,438	-136,775	-0,86	0,000	15996,213

Månedskomentar:

Kommentar :

Figure 3.3 the presentation of a monthly report.

3.2.2 Update a Monthly report

The update a month report function is displayed if the user is login and then opens a monthly report. The update menu is displayed below the current report. Figure 3.4 shows the updated menu of the monthly report. Verification value, correction values, and comments can be entered here, and all values are sent to storage when the button "Oppdater" is clicked.

3 System requirement specification

Avregning
Report
✕

Perioderapport: Matevann til VCM

Dato	Egetforbruk Noretyl	FIG8305		FIG012	FIG 012- FIG8305		FIG8305	
	tonn	Råverdier		tonn	Relativt avvik (4%)		Korreksjon	Verifisert
	tonn	tonn	tonn	tonn	tonn	%	tonn	tonn
<input checked="" type="checkbox"/> 01.06.2021	319,316	1966,183	1954,438	-11,745	-0,80		0,000	1966,183
<input checked="" type="checkbox"/> 02.06.2021	321,912	1795,659	1783,125	-12,534	-0,70		0,000	1795,659
<input checked="" type="checkbox"/> 03.06.2021	394,311	1209,075	1194,438	-14,637	-1,21		0,000	1209,075
<input checked="" type="checkbox"/> 04.06.2021	2351,231	1083,281	1067,375	-15,906	-1,47		0,000	1083,281
<input checked="" type="checkbox"/> 05.06.2021	2810,210	1453,618	1438,250	-15,368	-1,06		0,000	1453,618
<input checked="" type="checkbox"/> 06.06.2021	2677,330	1660,502	1646,250	-14,252	-0,86		0,000	1660,502
<input checked="" type="checkbox"/> 07.06.2021	1454,232	1955,822	1942,250	-13,572	-0,69		0,000	1955,822
<input checked="" type="checkbox"/> 08.06.2021	1114,524	1727,604	1714,250	-13,354	-0,77		0,000	1727,604
<input checked="" type="checkbox"/> 09.06.2021	1109,836	1577,877	1565,250	-12,627	-0,80		0,000	1577,877
<input checked="" type="checkbox"/> 10.06.2021	1116,962	1566,592	1553,813	-12,780	-0,82		0,000	1566,592
<input type="checkbox"/> 11.06.2021								
<input type="checkbox"/> 12.06.2021								
<input type="checkbox"/> 13.06.2021								
<input type="checkbox"/> 14.06.2021								
<input type="checkbox"/> 15.06.2021								
<input type="checkbox"/> 16.06.2021								
<input type="checkbox"/> 17.06.2021								
<input type="checkbox"/> 18.06.2021								
<input type="checkbox"/> 19.06.2021								
<input type="checkbox"/> 20.06.2021								
<input type="checkbox"/> 21.06.2021								
<input type="checkbox"/> 22.06.2021								
<input type="checkbox"/> 23.06.2021								
<input type="checkbox"/> 24.06.2021								
<input type="checkbox"/> 25.06.2021								
<input type="checkbox"/> 26.06.2021								
<input type="checkbox"/> 27.06.2021								
<input type="checkbox"/> 28.06.2021								
<input type="checkbox"/> 29.06.2021								
<input type="checkbox"/> 30.06.2021								
<input type="checkbox"/> 31.06.2021								
Foreløpig sum	13888,883							15998,213
Månedss korrelasjon								
Sum	13888,883	15998,213	15868,438	-138,776	-0,86		0,000	15998,213

Månedskomentar:

Editer

Godkjent Oppdater

FIG8305 korrigerig

Kommentar

Månedss korrigerig

Egetforbruk Noretyl

Månedss korrigerig FIG8305

Månedss kommentar

Figure 3.4 a monthly report with the update menu displayed.

3 System requirement specification

3.2.3 Yearly report

A yearly report can be selected from the folder "Årsrapporter." The selection menu for selecting a year appears on the same page. Figure 3.5 shows the selection menu for selecting a year for a yearly report. The year report is fetched on the selected year when the button "Hent rapport" is clicked. Figure 3.6 Shows the presentation of the yearly report.



Figure 3.5 the selection menu for selecting a year for a year report.

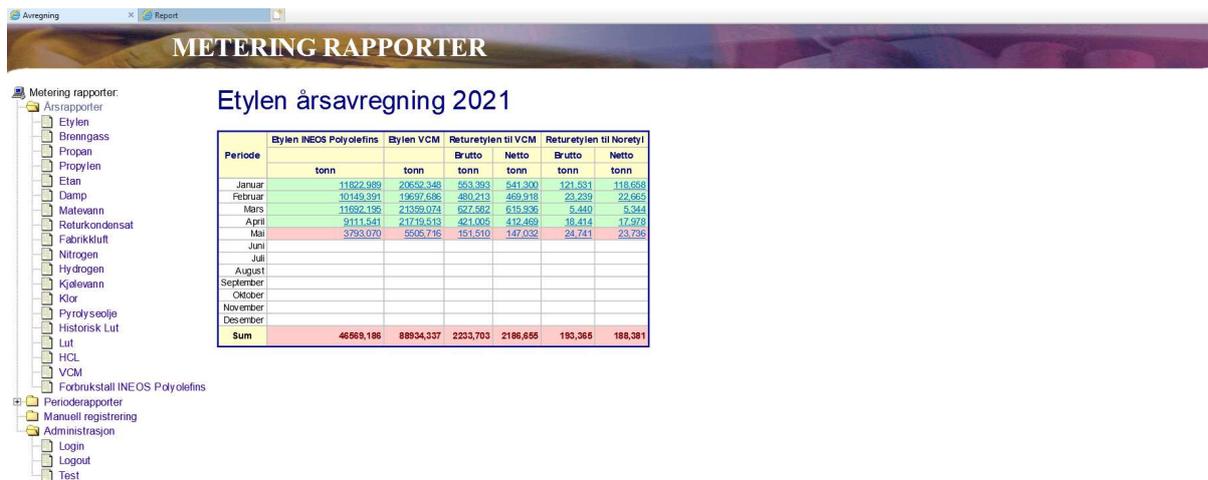


Figure 3.6 the presentation of the yearly report.

3.2.4 Manual registration

The measurement to manually register can be selected from the folder "Manuell registrering" if the user is login. Figure 3.7 shows the presented page to register a measurement with all the values registered one year back in time. The little table is calculated values between two registered values. Figure 3.8 shows the registration menu for entering a value and date-time value.

3 System requirement specification



Figure 3.7 the presented page to register a measurement with all the values registered one year back in time.

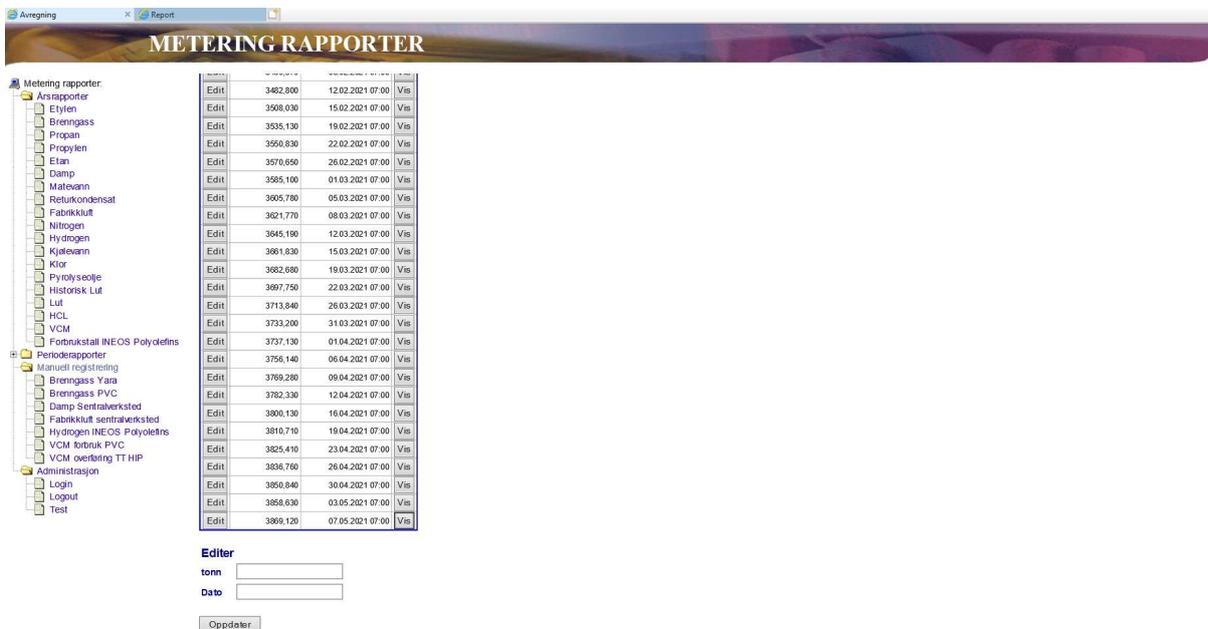


Figure 3.8 the registration menu for entering a value and date-time value.

3.2.5 Login

The login is inside the folder "Administrasjon." Figure 3.9 shows how the login page is and where it is possible to log in. The logout function is linked below the login. If it is clicked, the user is logged out even if the user was not logged in.



Figure 3.9 the login page.

3.3 Databases

The data is stored inside two different Uniformance Processes Historian Databases (UPHD). The names of these two UPHD are UPS_RAF and UPS_RAF2. Each of these UPHDs is running on two different servers. UPS_RAF is running on server FELLEES_RA09PHD, and UPS_RAF2 is running on server ETY_RASV15PHD. Each Process Historian Databases are products by Honeywell inc. The UPS_RAF2 is version 400, while UPS_RAF is version 321.

There are two different systems because UPS_RAF2 belongs to Ineos Rafnes and UPS_RAF belongs to Inovyn. Ineos Rafnes choose to continue to use the UPHD product from Honeywell and upgraded the version this winter. In contrast, Inovyn has decided upon changing out the UPHD to an IP21 Historian Database by Aspen Tech this summer. Currently, all the data used for the metering reports are stored in the two UPHD.

3.4 Functional Design Specification

A Functional Design Specification (FDS) was written on how a custom-made solution of a new Metering Reports application shall work, with the baseline as a one to one replacement.

The features from the FDS are:

- Login
- Presentation of 19 yearly reports
- Presentation of 29 monthly reports
- Manual registration
- Verification of daily measurement
- Correction of daily measurement
- Comment upon a daily measurement
- PHD interface
- Calculation

The non-functional requirements are:

- Available for 7/24 hours
- Termination shall result in a safe system state

3 System requirement specification

- Shall provide fault containment mechanisms to prevent error propagation
- Shall provide error handling to support safety-critical functions
- The user shall be preloaded
- All account/user modifications shall be logged
- Usability shall be measured via user surveys
- Performance of functions shall be tested for timing adequately
- The functionality shall be evaluated via user feedbacks from the system
- Security tests shall be performed sufficiently

File to be integrated for communication with both UPHD:

- Phdapinet.dll

4 Selection Phase

This chapter will be focused on investigating different technologies and selecting different technologies for a web application with consideration of the specification for a custom-made solution of a new Metering Reports application.

4.1 Definitions

Frontend

The frontend is everything the user interacts with of a website. In programming terminology, the frontend is the presentation layer [11].

Backend

The backend is any part of a website that the user does not see. In programming terminology is the data access layer [12].

Website

A website may be a webpage or collection of webpages [13].

Webpage

A Web page is written in HTML (hypertext markup language) and translated by a Web browser [14].

Client-side

Client-side refers to operations that the client performs in a client-server relationship of a computer network [15].

Server-side

Server-side refers to operations that are performed by the server in a client-server relationship in a computer network [15].

Programming language

A programming language is a computer language used to develop software programs and scripts for a computer to execute. A programming language has its syntax. The code written in a programming language is referred to as source code. The source code may be compiled into machine code for being useable for a computer [18].

Framework

A framework is a platform for developing a software application. It may include a set of software libraries, compiler, interpreters, or an API. Each framework is based around a specific programming language [19].

Library

A library is a collection of pre-compiled and non-volatile routines for a programming language [20].

4.2 Web application

A web application consists of frontend and a backend.

The frontend consists of:

- A web browser
- Client-side programming language

The backend consists of

- A web server and Web hosting service.
- Server-side programming language

All these parts must be investigated frontend and backend and selected.

It is predefined that the server application and the UPHD must be interfaced through an API. The API for communication with the UPHD is a Dynamic-Link Library (DLL) called Phdapinet. The Phdapinet is a client portal for interaction with UPHD. The Phdapinet relies on some other DLL to be utilized, and they are Network, phdknl, phdrapi, and PHDSecurity. Honeywell created all the DLL and has been chosen for the newest version, which is 400.1.

4.3 Web browser

The user's interface and interaction point will be through a web browser. Figure 4.1 shows the development of the usage of significant web browsers through the last decade [19].

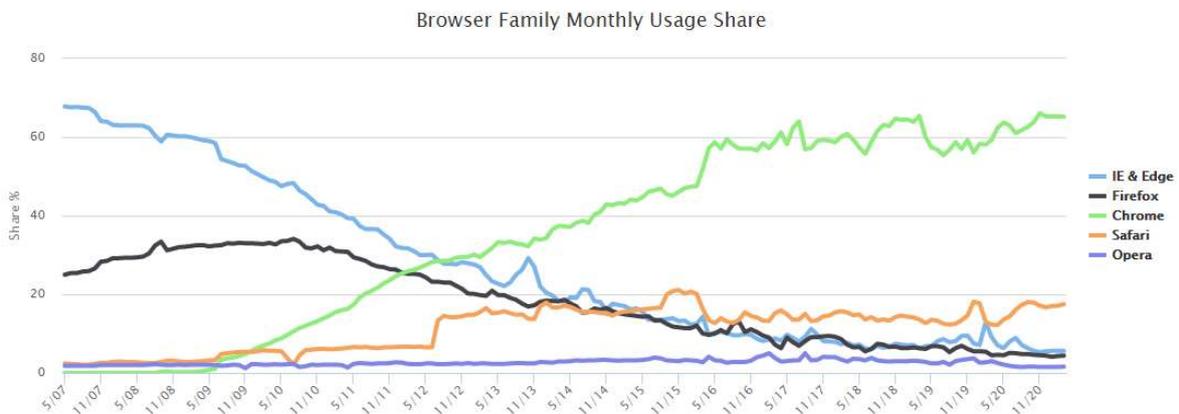


Figure 4.1 Web Browser Usage Trends [19].

The selected web browsers the frontend shall be compliant with is Google Chrome and Microsoft Edge. The two arguments for this choice:

- Google Chrome has the largest marked share of users.
- Google Chrome and Microsoft Edge are the two available options as default in the Software Center at Ineos Rafnes and Inovyn.

4.4 Web server

A server is a computer or system integrated into a network that provides data, programs, resources, or services to other computers through the network. For this project, a web server is needed for hosting the web application. [20]

A web server hosts programs and data across the internet or an intranet. The interaction from a client to the webserver happens through a browser on the client computer that sends a Hypertext Transfer Protocol (HTTP) request. The web server replies with a web page with the requested content, and it is displayed in the browser on the client computer. [21]

Every web server needs software for hosting web applications. Currently, there is different software for a web server: Apache, Microsoft Internet Information Services (IIS), NGINX, OpenResty, or others. The two alternatives that have been chosen to investigate are Apache and Microsoft ISS. The argument for investigating these two alternatives was because of the most used web hosting services over time. Figure 4.2 shows the market share of active sites of web hosting services from 2019 [22].

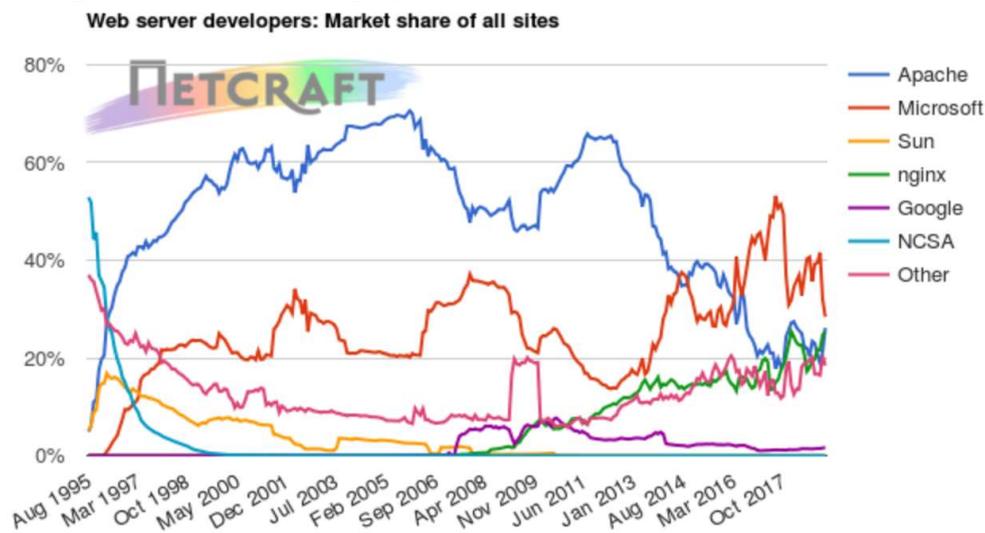


Figure 4.2 market share of all sites of web hosting services from 2019 [22].

Table 1 shows the difference between Microsoft IIS and Apache [23].

Table 1 IIS vs. Apache [23].

Features	Microsoft IIS	Apache
Supported OS	Windows	Linux, Unix, Windows, macOS
User support & fixes	Corporate support	Community support
Cost	Free, but bundled with Windows	Completely free
Development	Closed, proprietary	Open source
Security	Excellent	Good
Performance	Good	Good
Market share	32 %	42 %

The web hosting service is selected to be Microsoft IIS. The primary argument is that a corporation supports it. Supported by a corporation gives the benefit of support if it fails. IIS

4 Selection Phase

has better security than Apache. The downside with ISS is that it only can run on Windows Operating System (OS), but most computer and servers at Ineos Rafnes mainly runs with Windows as the OS.

The IT department has provided the server and installed Windows Server 2019 as OS with Microsoft ISS for hosting the web application.

4.5 Programming language

There is a whole range of programming languages available for creating different types of programmed applications.

Widely used

To determine if a programming language is widely used, internet searches were used as an indicator. The TIOBE¹ Programming Community index is a good indicator of programming languages most used amongst people searching code. Figure 4.3 shows the top 10 programming languages that have been searched for over time [24].

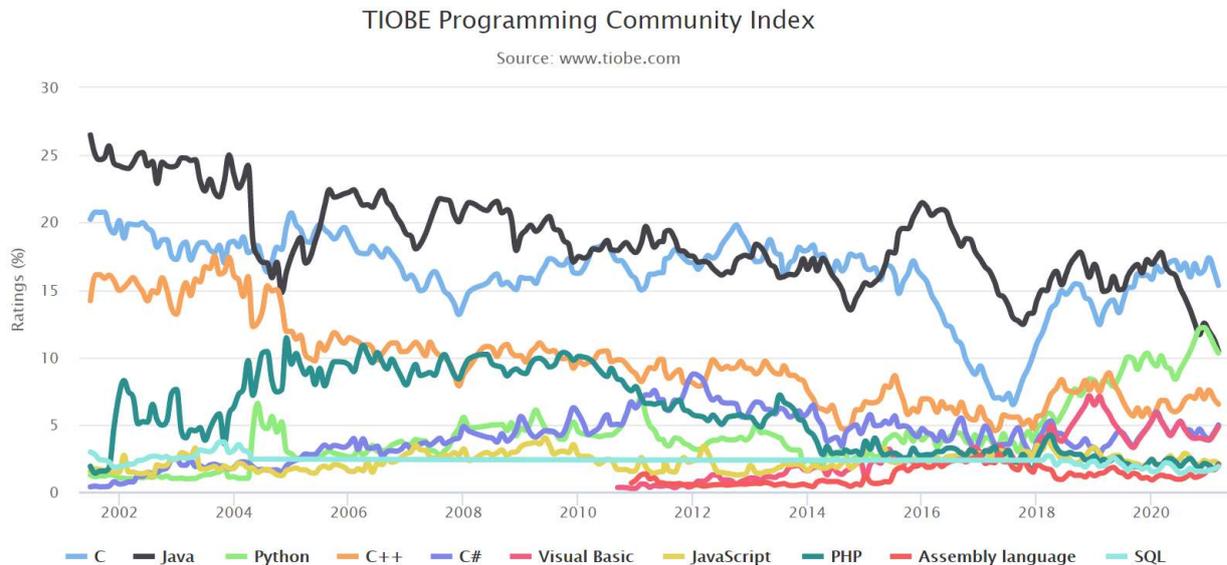


Figure 4.3 The TIOBE Programming Community index shown as a graph through time [24].

The graph shows that the top three widely used programming languages for the server-side are Java, Python and C# (C-scrap). The widely used programming language for client-side from the graph is JavaScript.

Popular amongst developers

To determine if a programming language popular amongst developers, the communities and storage places for code and coding had to be investigated. The largest community for developers is Stack Overflow [25], and the most prominent place to create and store code is

¹ TIOBE (The Importance Of Being Earnest) Software BV

4 Selection Phase

GitHub [26]. Figure 4.4 shows a regression model of the programming language between Stack Overflow and GitHub. Figure 4.4 reveals which programming languages are popular to create projects with on GitHub and the popularity of discussed programming languages by developers using Stack Overflow.

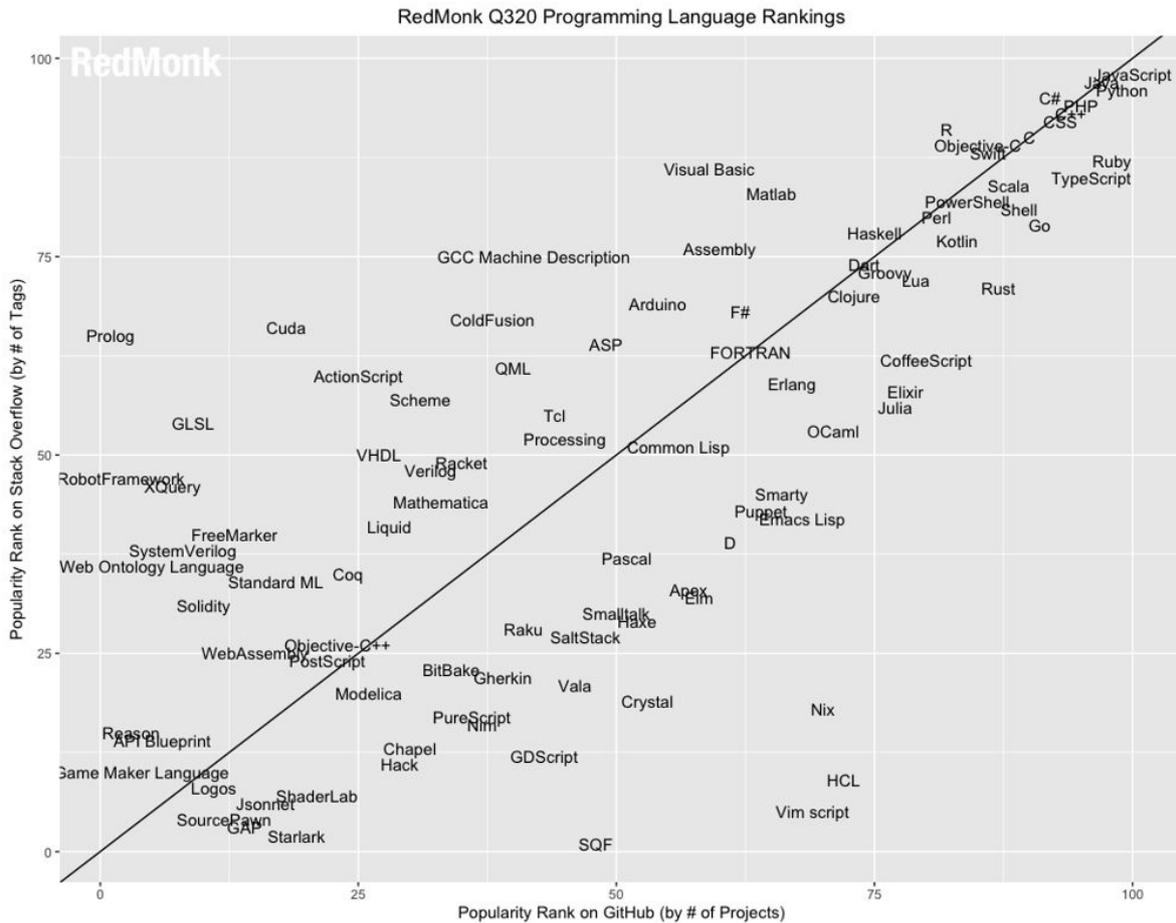


Figure 4.4 regression model of the programming language between Stack Overflow and GitHub [27].

The regression model shows that the top three most popular programming languages for the server-side are:

- 1) Java
- 2) Python
- 3) C#

For the client-side, the top two most popular programming languages are:

- 1) JavaScript
- 2) TypeScript

Known to the same developer.

It is common for a developer to use or know more than one technology. Stack Overflow does a survey every year where they ask developers which database, framework, language, and platform they use. Based upon the feedback of the survey, Stack Overflow creates a cluster

4 Selection Phase

diagram. Figure 4.5 shows technologies cluster together into related ecosystems that tend to be used by the same developers [28].

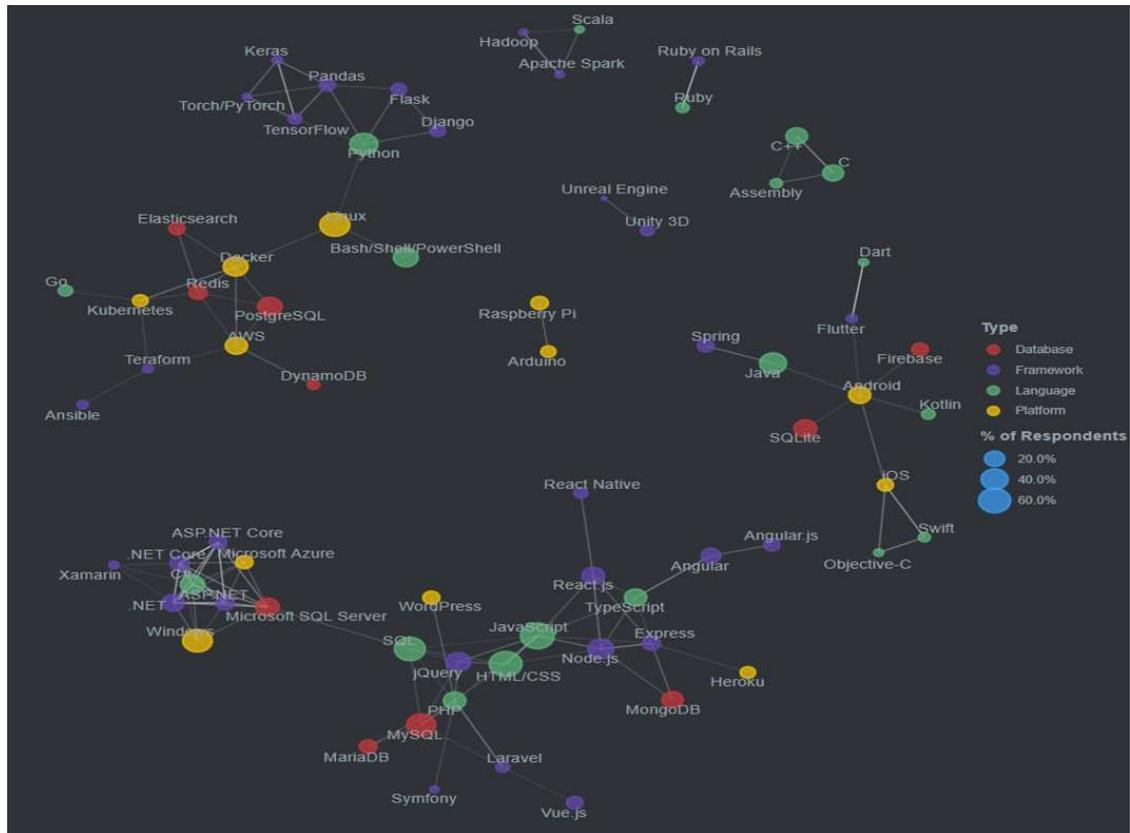


Figure 4.5 Technologies cluster together into related ecosystems that tend to be used by the same developers [28].

The cluster diagram indicates that the most popular and widely used programming languages for server-side and client-side do not tend to be used by the same developer.

4.6 Client-side programming language

A web page consists of HTML and CSS can be added for styling. Neither HTML and CSS are technically programming languages because they are page structure and style information [29]. As for the client-side programming language is JavaScript. Currently, 97.3% of all the web sites use JavaScript [30]. The client-side programming language is JavaScript, but there is a different programming language for the client-side programming language, but all of them are compiled into JavaScript. Examples of other pure client-side programming languages are TypeScript and Elm. In addition, there is a whole range of frameworks and libraries for client-side programming languages. A web design pattern can be selected to eliminate some of the options of a client-side programming languages, frameworks, and libraries.

4.6.1 Website design pattern

There are two main design patterns for a website: single Page Application (SPA) and Multi-Page Application (MPA). An MPA is considered a traditional website, where the web pages are requested from the server each time. In comparison, a SPA is loaded dynamically. Each of these design patterns has different advantages and disadvantages, as shown below [31].

Advantages for an MPA:

- Each page has its own Uniform Resource Locator (URL). This makes it easy to bookmark the specific page user wants to load [31].
- It is more Search Engine Optimization (SEO) friendly. [32]
- More accessible to scale since there is no limit to the number of pages it can contain. [32]
- Insightful data analytics to how the users use the pages like how long time they spend on each page, numbers of daily and monthly visited each page. [32]

Cons for an MPA:

- The development time increases in proportion to the number of pages to be built and the functionality to be implemented. [32]
- Since each page requires being loaded when the user wants to see them, this may be perceived as a slow web application. Also, the load on the server will be higher if many are using it simultaneously. [31]
- The frontend and backend development are tightly coupled, and the developer needs to use frameworks for either server-side or client-side [33]

Advantages for a SPA:

- Fast and smooth switching between pages without reloading the page due to most resources being loaded once. [31]
- The development is much faster since fewer pages are created and benefit from being easier to maintain. [32]
- It separates the frontend from the backend. [31]
- SPA can cache any local storage effectively and work offline with the data that has been loaded. [32]
- The debugging of a SPA can be done in Google Chrome. [31]

Cons for A SPA

- It is developed using JavaScript frameworks, so if the users have disabled JavaScript in their browser, it will only show an empty page. [31]
- It is slow to download the first time because a large amount of data is loaded to the client in the first it is loaded. [33]
- It is more vulnerable to Cross-Site Scripting (XSS) [34]. The reason behind this is that some features and logic are moved to the client-side. The consequence of this can be a leak of user sensitive information [32]

The selected design pattern has been selected to a SPA. The arguments are:

- Easier to maintain.
- Fewer web pages need to be created.
- It makes the frontend and backend separated.

4.6.2 SPA Options

The design pattern SPA gives some option to use as of programming language with framework or libraries. The selected options to review have been chosen to be Vue, Elm, and React. Selecting these options is because Vue and React are the most popular and wanted framework and library to use [28], and Elm performs better in comparison [35].

Vue

Vue is a progressive framework for JavaScript to build web interfaces and SPA. Evan You is the original author of Vue, and it was released in 2014. The origin of Vue is from Angular, where Evan You attempted to build a tool of the best parts of Angular [36]. Vue has both advantages and disadvantages, and they are:

Advantages:

- Documentation of Vue is extensive and concise. In addition, the documentation gives a comparison to other frameworks and libraries [36].
- The data binding of Vue is two-way, giving the benefit of real/time updates [37].
- Vue is both User- and beginner-friendly when having a good understanding of JavaScript [38].

Disadvantages:

- Language barrier as many discussions, plugin descriptions, and instructions are in Chinese for Vue [36].
- Vue has excessive code flexibility, and giving room for different programming approaches can be applied simultaneously [38].
- Vue is still a young framework and is still evolving fast [39].

Elm

Elm is a newcomer and is a programming language compiled to JavaScript, and that makes it a framework. Elm is created by Evan Czaplicki, which initially started as his master thesis at Harvard University [40]. In common with React, Elm is created for developing SPA. Elm has been investigated due to its performance outrank other frameworks and libraries [35]. Elm has both advantages and disadvantages, and they are:

Advantages:

- Compiles to JavaScript, making it ideal for building fast-executing UIs with zero errors at runtime [35].
- It is a functional programming language that is strongly typed and compiled [41].
- Elm is fast and easy to learn [42].

Disadvantages:

- The source code is not open source and is still being developed by the creator and trusted people [43].
- The community is small and is not popular amongst developers [44].
- Most of the tutorials are outdated, and the documentation is incomplete [43].

React

React, also known as ReactJS, is a JavaScript library for building User Interface (UI) specifically for single-page applications. React was first created by Jordan Walke, a software engineer working for Facebook and Facebook's newsfeed, in 2011 [45]. Facebook released React as an open-source framework in 2013 and is still being developed and maintained by Facebook. Since the release date, React has become one of the tops loved and wanted frameworks by developers due to its focus on user experience and development simplicity. [46]. React has both advantages and disadvantages, and they are:

Advantages:

- Easy to Learn: Anyone with a basic understanding of HTML and JavaScript can quickly startup developing with React. This is also the reason behind React is popular than other JavaScript libraries and frameworks. [47]
- Components-based architecture: React components that can be independently created, maintained, and reused. This gives the ability to have multiple components inside the same web page who work independently. [47]
- Large community: React has had a rapid and is still a growing community. [46]
- Helpful tools: React Developer Tools is an extension to Google Chrome and Mozilla Firefox, making it easier to check the code when tested. [46]

Disadvantages:

- The high pace of development: React is constantly evolving and changing, which can be seen as both negative and positive. In the negative senses, one has to relearn how React works constantly, and it may be challenging to keep up with all the changes. [48]
- Lack of proper documentation: As a result of the high pace of development of React, the documentation is not always been up to date on the newest version. [48]

4.6.3 Choice

The choice has fallen upon React to create the frontend. The arguments are:

- Use JavaScript as the programming language, which is the most popular and used.
- It has the highest likelihood of being known by other developers.
- It is easy to learn if JavaScript is known.
- The documentation is the most updated and in English compared with the other alternatives.
- It is known to a developer at Inovyn.

The latest version with complete documentation of React was selected and was version 16.14.0.

4.7 Server-side programming language

The programming language for the server-side is responsible for interacting with other applications and databases. The three most prominent programming languages for server-side are C#, Java, and Python. Java and C# are relatively equal, while Python differs more from them. Table 4.1 presents factors for each programming language and two unique factors for Ineos Rafnes and Inovyn [49] [50] [51] [28].

Table 4.1 factors for each programming language and two unique factors for Ineos Rafnes and Inovyn.

	Java	C#	Python
Used by Professional Developers	38.4%	32,3%	41,6%
Liked by developers	44,1%	59,7%	66,7%
Supported by	Oracle	Mircosoft	Community
Used for creating	Android apps, large websites	Windows apps, large websites (Unity games)	Math scripts, websites
Used especially by	Large companies (Banks, eCommerce, Google, etc.)	Large companies (Microsoft, healthcare, etc.)	Academics, startups, Google
Top Web Framework	Spring MVC	ASP.NET MVC	Django
Performance	Compiled	Compiled	Interpreted
Variable usage	Strongly Typed	Strongly Typed	Dynamic-typed
Difficulty level	Easy to moderate	Easy to moderate	Easy
Compliant with react	Yes	Yes	Yes
Deployed with ISS	Yes	Yes	Yes
Used by employees at Ineos and Inovyn	No	Yes	No
Procedure for data extraction from UPHD	No	Yes	No

C# been selected as the programming language. The two last rows of the table give the main reasons. In addition, both Ineos Rafnes and Inovyn uses mainly products from Microsoft or integrated with Windows OS.

4.7.1 Framework

The chosen framework for C# is ASP.NET, the only framework for web development. In addition, there are two platforms to choose from for ASP.NET. the platforms are .NET Core and .NET Framework. Table 4.2 gives essential arguments for and against both NET Core and .NET Framework [54].

Table 4.2 arguments for and against both NET Core and .NET Framework.

A high-performance and scalable system without UI	.NET Core is much faster.
Docker containers support	Both, but .NET Core is born to live in a container.
Heavily rely on the command line	.NET Core has better support.
Cross-platform needs	.NET Core
Using Microservices	Both, but .NET Core is designed to keep today's needs in mind.
User interface centric Web applications	.NET Framework is better now until .NET Core catches up.
Windows client applications using Windows Forms and WPF	.NET Framework
Already have a pre-configured environment and systems	.NET Framework is better.
Stable version for an immediate need to build and deploy	.NET Framework has been around since 2001. .NET Core is just a baby.
Have existing experienced .NET team	.NET Core has a learning curve.
Time is not a problem. Experiments are acceptable. No rush to deployment.	.NET Core is the future of .NET.

The platform has been selected to .NET Core because it is faster, better support, and the future of .Net. The latest version of ASP.NET with .NET Core is ASP.NET Core 5.0 and is selected.

4.8 Interface

The interaction between the React and ASP.NET Core is set to the default APIs. React has an inbuilt standard API library. The ASP.NET Core is the ASP.NET Core MVC for Web API. The protocol between them is HTTP with format JavaScript Object Notation (JSON). The argument for choosing this interface is that it is updated simultaneously when the version of the framework or library is updated.

4.9 Development tools

For creating, writing, test, compile, debug, and publish the code, an Integrated Development Environment (IDE) is needed [53]. For C#, the natural choice fell on Visual Studio 2019 as the IDE. Visual Studio 2019 has the possibility for creating, compile, test, debug, and publish C# code. [54] Furthermore, a version control tool is needed for storing and handle the C# code. For this purpose, GitHub was selected.

4 Selection Phase

React can be created, compile, test, and publish by Visual Studio 2019, but it is not suited to write and debug the code. Visual Studio Code was chosen for writing code. Visual Studio Code is a code editor that can import libraries for types of frameworks and libraries to highlight that code is written correctly. [55] For debug purposes, Google Chrome was selected. Hence, it can write out error messages, and it has an inbuilt console application where it is possible to log statements. As Visual Studio 2019 was used for creating it, GitHub was selected as the version control.

The Web API's has been tested with software called Postman. The purpose of Postman is to send HTTP messages to see the response. [56]

The API DLL for UPHD has been view with JetBrains Dotpeek. JetBrains Dotpeek is software that can decompile DLL files. [57]

The current metering report application has been viewed with Note++.

4.10Changes

The selected framework .Net Core, and version of the Phdapinet had unforeseen issues. The two main issues were utilizing the Phdapinet.dll with .NET Core and communication with UPHD with the Phdapinet version 400.1.

4.10.1Utilize the Phdapinet

The first issue to arise was to utilize the Phdapinet. The .NET Core was not able to utilize the Phdapinet, but the .NET Framework was able. There were some alternatives to solve this issue, and the alternatives were:

1. Change the whole web application to .Net Framework.
2. Split the application into two parts, with two different frameworks.
3. Create a new project .Net Framework inside the same project and pass the data between two applications.

Each of the alternatives has their advantage and disadvantage. Table 4.3 shows the advantage and disadvantages of the alternatives.

Table 4.3 the advantage and disadvantages of the alternatives.

	Alternatives 1	alternatives 2	alternatives 3
Advantage	One application	Separate the application into two independent projects	One application
		Frontend is cross-platform	Fully Cross-platform
		It makes the applications more open to being used by other applications and changeout	
Disadvantage	Older version	Two applications	Higher update frequency on imported libraries

4 Selection Phase

	It can only run on Windows OS	The backend requires Windows OS	It makes the application more complex
--	-------------------------------	---------------------------------	---------------------------------------

The selected alternative was to split the application into two parts, with two types of frameworks. The reason for selecting this alternative is the uncertainty of what is required for implementing the new database to Inovyn. Hence, having two applications increases the chance to handle the implementation and keeping one part as new as possible.

The selected alternative has been created to a different version where the frontend is .Net Core with ASP.NET Core 5.0, while the backend is .Net framework version 4.6.1 with ASP.NET Core 2.1.

4.10.2 Communication with UPHD

The second issue to arise was communication with UPHD. The Phdapinet version 400.1 was able to request data but was not allowed to send back data to the UPHD. To clarify what the issue was, Honeywell was contacted. A Honeywell expert on the UPHD informed that they had some issues with version 400.1. When a computer has installed Uniformans Processes Historian Database Client 321.21² tries to use version 400.1 of the Phdapinet. Honeywell recommended using version 321.21 because it worked with all the versions of a UPHD.

² A software module belonging to Unifromance Process Studio for displaying data from UPHD by Honeywell.

5 Design

The custom-made solution of a new Metering Report has had a preliminary design as the development processes evolved. It has been several prototypes for a new Metering Report custom-made solution, but most have kept the design.

5.1 System

The system is a web server, and the given name is NoRafDev03. NoRafDev03 had to be reachable for the user at Inovyn and Ineos Rafnes and connect to both UPHDs. Figure 5.1 shows an overview of the system. The web server is set to only be reachable on the intranet as a safety precaution. The intranet is divided into levels, and the level NoRafDev03 operates on is level 4.0. Level 4.0 is the business or office level of the intranet.

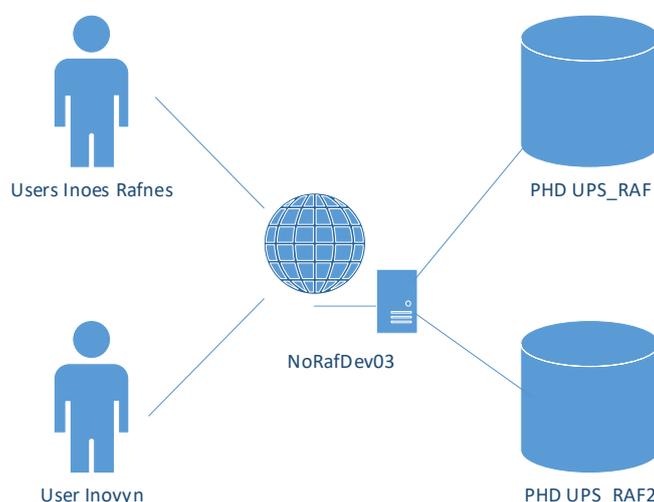


Figure 5.1 System overview

5.2 Architecture

The Architecture of the web application has been selected to a two-layer application. This separated the frontend and the backend as two independent development, with an interface between them for data exchange. The arguments for the two-layer structure of the web application are to make each part's maintenance more uncomplicated and the possibility for changing out one of the parts. Figure 5.2 shows the architecture of the two layers with the underlying structure of each of them.

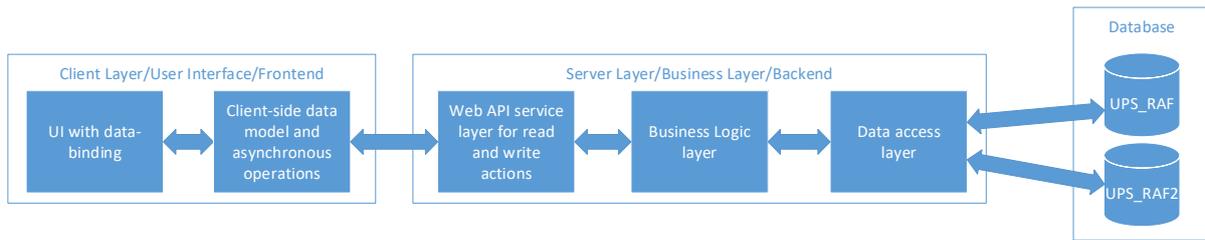


Figure 5.2 overview of the principal structure of a custom-made solution.

The client layer consists of two parts, UI and code-behind. The UI task is to let the user interact with the system. The code behind is all predefined structures, operations, and data exchange based on the user input.

The server layer has been selected into three parts, Web APIs, Business logic, and data access. Web APIs are the entrance for the frontend to pass and get data. The Business logic is the code for determining how data is created, stored, and changed. The Data access layer handles the interaction against the databases.

5.3 Use cases

The requirements of a new custom-made solution have been set to use cases. Figure 5.3 shows the use cases with actors. Each of the use cases has been interpreted for creating the web application.

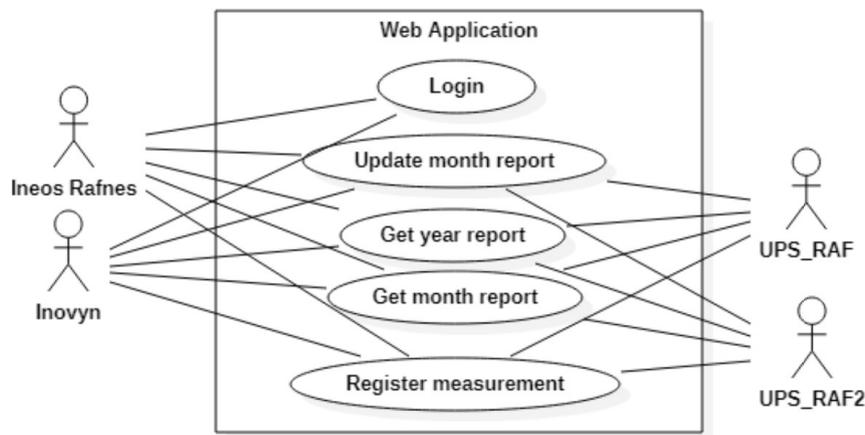


Figure 5.3 use case diagram of the web application.

5.4 Design patterns

It has been used in design patterns to create the structure of the web application. The design patterns are from General Responsibility Assignment Software Patterns (GRASP) and Gang of Four (GoF) [10, pp. 173-175]. Table 5.1 and Table 5.2 shows the used design patterns for the web application.

Table 5.1 used design patterns from GRASP.

GRASP patterns overview

5 Design

Name	Problem	Solution
Controller	Starting point of the application or use case, what controls the use-case?	Assign the responsibility to an object who: 1. represent the overall system (root object), 2. represent (control execution of) the use-case.
Information Expert	Which object is responsible of knowing the specific information?	Assign the responsibility of knowing to the class holding that information.
Creator	Which object creates another object?	Assign the responsibility of creation an object to: 1: Another object who contains it, 2: Another object who records it, 3: Another object who uses it (closely, meaning a lot), 4. Another object who knows the initialization parameters of it.
Low Coupling	How to reduce the impact of changes in the code?	Avoid back and forth messages, one way communication between objects. Minimum interaction/knowledge /dependency between objects, tree structure.
High Cohesion	How to keep objects focused, understandable, manageable?	Objects should be focused about the contained task and not have multiple responsibilities, unless they belong together (tasks closely related), Contain related functions in only one object, never store data in different locations, Objects should cooperate to solve the overall task, but be responsible for their own part, assign responsibilities to suitable objects, or create new classes if no suitable object exists.
Polymorphism	How to handle methods with different types of variables but same function?	Use polymorphism overloads, and keep the function code only one place.

Table 5.2 used design patterns from GoF.

GoF patterns overview		
Name	Problem	Solution
Singleton	How to implement a class when only one instance of the class should exist?	Use a static method to return the instance and make constructor private so it is impossible to create multiple objects of the class.
Strategy	How to design for varying (able to change), but related, algorithms/policies?	Define each algorithm/policy/strategy in a separate class with a common interface
Facade	How to avoid multiple connections between multiple objects between subsystems or an application?	Define a single point (or a few) of contact between subsystems/layers, as interface objects or wrapper objects.

5.5 UI design

The UI design has aimed to create a simplistic and intuitive website. It has been chosen to have a header, footer, content, and a sidebar to change the content if needed. The color resolution for the UI was elected to be light colors and colors matching the logo of Ineos Rafnes or Inovyn. It has been attempted to selecting components for user input commonly seen in other applications to make it intuitive.

5.6 First draft

A first draft of the application was created based on the requirements. Figure 5.4 shows the first draft of the classes for the use cases inserted into the architecture. The first draft has been used as the foundation as the development has moved forward. The cycles of the web application start with user input triggers an event that loops through the system to return the desired operation based on the user input.

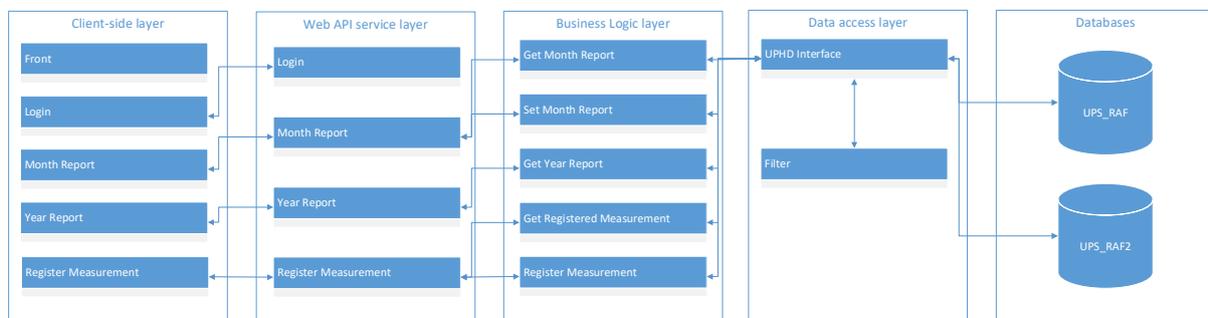


Figure 5.4 the first draft of the classes for the use cases inserted into the architecture.

The principal idea is that each use case starts at the UI. The client-side sends either a request for getting or setting data to the Web API. As the Web API contains two functions, it has been split into a function for each. These functions are the business logic, which controls the use case. The UPHD interface is responsible for handling data from and to the two UPHD.

5.7 Development plan

It was made a development plan to have a course for the prototype. The plan was created concerning the use cases and the users. Figure 5.5 shows a block diagram of the plan for each part of the web application. The idea was to start with the UI to get the users interested in the development and then implement one after one of the use cases.



Figure 5.5 a block diagram the plan of each part of web application.

5 Design

The Scrum method of sprint was planned to be utilized with a new version of the prototype each 14 day until the project was finish. Figure 5.6 shows a block diagram of the planned first sprints and the iteration of the sprints for the project.

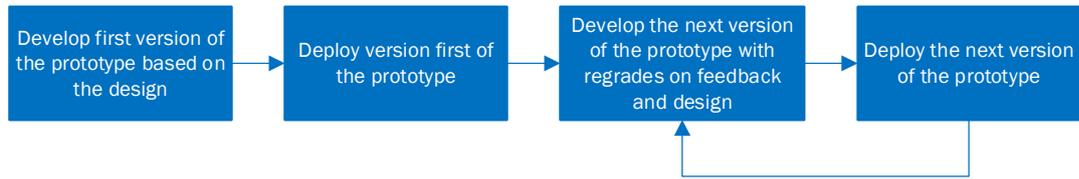


Figure 5.6 a block diagram of the planned first sprints and the iteration of the sprints for the project.

6 Prototype

The current prototype of a custom-made solution for a new metering system has been made from the requirement. The prototype runs on a server at Ineos Rafnes and is accessible to anyone connected to the intranet. Figure 6.1 shows the front page of the prototype.



Figure 6.1 Front page of the prototype.

6.1 Get a monthly report

The monthly report page can be reached by clicking the "Perioderapporter" link on the navigation bar. All the types of monthly reports are inside the side menu. The side menu for monthly reports is divided into Ineos Rafnes and Inovyn, with the monthly reports belonging to each of them. Figure 6.2 shows the start points for selecting a monthly report. When a monthly report is selected for the first time, the time interval is set to the current year and month. Figure 6.3 shows the monthly report "Pyrolyseolje" when it is chosen as the first monthly report. When a monthly report has been selected and displayed, it is possible to change the time interval. The time interval may be chosen between January 2005 and the present time. Figure 6.4 shows the monthly report of "Returkondensat" at a select time interval of February 2005.

← → ↻ ⚠ Not secure | 172.24.22.87:5545/yearly ☆ 🌐 📄

INEOS
THE WORLD FOR CHEMICALS

Info Manuell registrering Perioderapporter Årsrapporter Innlogging

Årsrapporter

Årsrapport for Etylen 2021

Periode	Etylen Ineos Bamble		Etylen VCM		Returetylen Til VCM Brutto		Returetylen Til VCM Netto		Returetylen Til Ineos Rafnes Brutto		Returetylen Til Ineos Rafnes Netto	
	Tonn	Tonn	Tonn	Tonn	Tonn	Tonn	Tonn	Tonn	Tonn	Tonn	Tonn	Tonn
Ineos Rafnes	11822,989	20652,348			553,393		541,3		121,531		118,658	
Brenngass	10149,391	19697,686			480,213		469,918		23,239		22,664	
Etan	11692,195	21359,074			627,582		615,936		5,44		5,344	
Etylen	9111,541	21719,513			521,005		512,469		18,414		17,978	
Damp	1463,682	2460,285			55,931		54,312		12,236		11,663	
Fabrikkluft	0	0			0		0		0		0	
Ineos Bamble Forbrukstall	0	0			0		0		0		0	
Matevann	0	0			0		0		0		0	
Nitrogen	0	0			0		0		0		0	
Propan	0	0			0		0		0		0	
Propylen	0	0			0		0		0		0	
Pyrolyseolje	0	0			0		0		0		0	
Inovyn	0	0			0		0		0		0	
Sum	44239,798	85888,906			2238,124		2193,935		180,86		176,307	

INEOS
Olefin & Polymers Europe

INEOS
THE WORLD FOR CHEMICALS

inovyn
THE WORLD FOR CHEMICALS

Figure 6.8 the yearly report of "Etylen" for 2021

6.5 Manual registration

The manual registration page is accessed by clicking the "Manuell registrering" link on the navigation bar. All the types of manual registration are inside the side menu. When a manual registration is selected from the side menu, the operation interface is displayed. Both the registered values and the calculated values are displayed for the current month. If the last registration happened longer ago than a month, the last value registered is presented. The registration interface is placed below the table of registered values. To register a new value, a date and a value need to be inserted, and click the button "Oppdater telleverksverdi." Then both tables are refreshes with corresponding values. Figure 6.9 the manual registration page of "Damp."

There are two possible registration scenarios:

- Register a new value of a new date
- Change the current registration

When either of the scenarios happens, the application recalculates the calculated values. The calculation is display as the user enter values or changes the date. The limitations for the user is:

- Not possible to enter a date before the last registration.
- Not possible to enter a lower value or higher than two times the currently registered value

If the last limitation is valid, the displayed calculation is mark red. Figure 6.10 shows the indication of a value outside of limitation in the registration interface.

The screenshot shows the 'Manuell registrering' (Manual Registration) page for 'Damp Sentralverksted'. The page is divided into several sections:

- Manuell registrering:** A list of items including Brenngass Yara, Brenngass PVC, Damp Sentralverksted (selected), Fabrikkluft sentralverksted, Hydrogen INEOS Polyolefins, VCM forbruk PVC, VCM overføring TT HIP, and Test.
- Registrert verdi for EP-FIQ302.MASS_CUM_EOD:** A table with columns 'Avlesningstidspunkt' and 'Telleverksverdi'.

Avlesningstidspunkt	Telleverksverdi
30.04.2021 07:00:00	3850,84
03.05.2021 07:00:00	3858,63
- Tidligere kalkulererte verdier EP-FIQ302.MASS_PDAY:** A table with columns 'Dato' and 'tonn'.

Dato	tonn
30.04.2021 07:00:00	3,519998
01.05.2021 07:00:00	2,596637
02.05.2021 07:00:00	2,596637
03.05.2021 07:00:00	2,596637
- Form fields:** 'Velg Dato' (05/03/2021) and 'Ny telleverksverdi' (3858.63).
- Text:** 'Differanse mellom datoene er 0 dager', 'Differanse mellom forrige og ny telleverksverdi 0,000 tonn', 'Gjennomsnittsverdi for hver av dagene er 0 tonn', and a button 'Oppdater telleverksverdi'.

Figure 6.9 the manual registration page of "Damp."

The screenshot shows the 'Manuell registrering' (Manual Registration) page for 'VCM overføring TT HIP'. The page is divided into several sections:

- Manuell registrering:** A list of items including Brenngass Yara, Brenngass PVC, Damp Sentralverksted, Fabrikkluft sentralverksted, Hydrogen INEOS Polyolefins, VCM forbruk PVC, VCM overføring TT HIP (selected), and Test.
- Registrert verdi for VP-VCM_HPI-MASS_PERIOD:** A table with columns 'Avlesningstidspunkt' and 'Telleverksverdi'.

Avlesningstidspunkt	Telleverksverdi
01.05.2021 00:00:00	17671,94
- Tidligere kalkulererte verdier VP-VCM_HPI-MASS_PDAY:** A table with columns 'Dato' and 'tonn'.

Dato	tonn
01.05.2021 00:00:00	589,0646
- Form fields:** 'Velg Dato' (05/21/2021) and 'Ny telleverksverdi' (1767.94).
- Text:** 'Differanse mellom datoene er 20 dager', 'Differanse mellom forrige og ny telleverksverdi -15904,000 tonn', 'Gjennomsnittsverdi for hver av dagene er -795,200 tonn', and a button 'Oppdater telleverksverdi'.

Figure 6.10 shows the indication of a value outside of limitation in the registration interface.

6.6 Login

The login page is accessed by clicking the "Innlogging" link on the navigation bar. It is possible to log in and out, but no extra privilege is given regarding available functionality. All functionality has been set available without requiring to log in test it. The reason is to enable the users to test as much as possible. The current username is "qwerty," and the password is "Rafnes2020". Figure 6.11 shows the login in page.

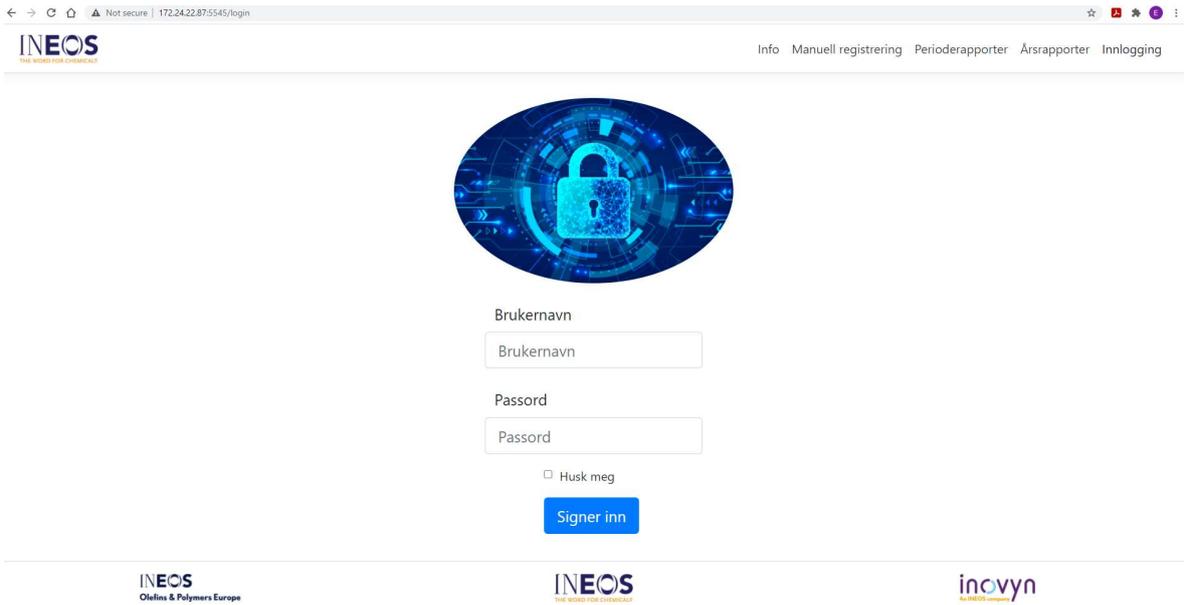


Figure 6.11 the login in page.

7 Frontend

The developed Frontend is C# application with React implemented. Programming languages used in the React application of the client-side are JavaScript, HTML, and CSS. The React application is booted up with C# application on the webserver.

7.1 React

The React part holds only one Html page that is initialized by the C# application. This is the reason why it is called a SPA (Single Page Application). The index page only activates a JavaScript file called index. All the rest of the scripts are nested into the JavaScript index.

All code-written scripts in React are defined as components. A component is either a function or a class. The difference between a class and a function is that the class can hold values. Stored values inside a component are referred to as a state. A component is primarily built up of JavaScript and HTML code. It is possible to use CSS directly into the code of a component or refer to a CSS file to style a component.

The React has a life cycle concept. The code running has a life cycle for performing different operations. The life cycle for React is the following:

- Initialization happens when a component is called for the first time.
- Mounting is when the new content is presented.
- Updating occurs when an event is triggered.
- Unmounting is clearing out the component.

Figure 7.1 shows the overview of the React life cycles and the different functions for each of the life cycles [58]. In mounting and updating, there are two types of operation. Presenting in the browser is referred to ReactDOM methods, and performing tasks in the background are referred to React methods.

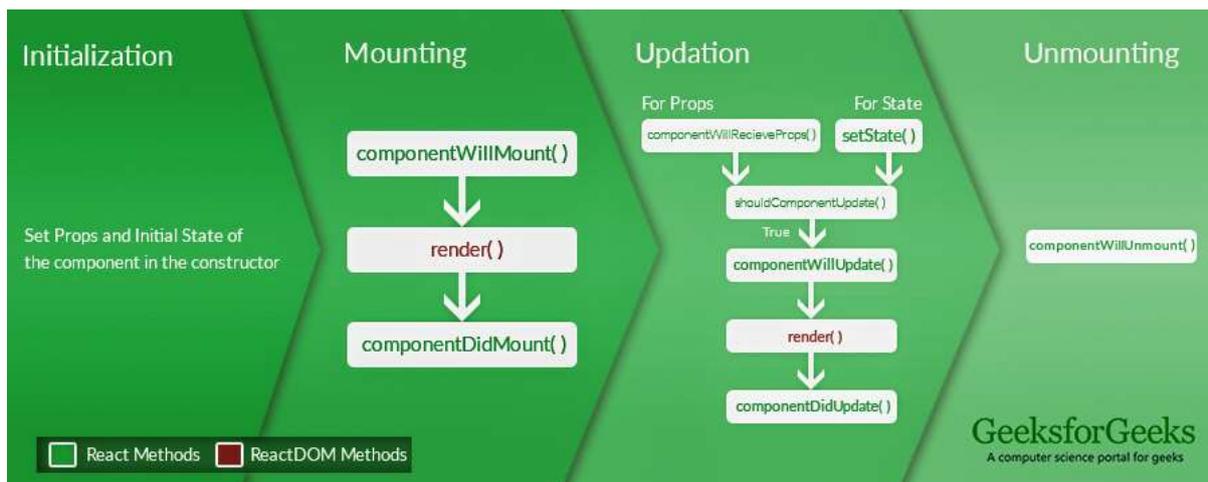


Figure 7.1 Overview of the React framework life cycles [58].

7.2 The structure

The React part has been given a straightforward structure. Figure 7.2 shows the structure from the Html index file and all the components. All the components are constructed as classes, except the Footer, which is a function.

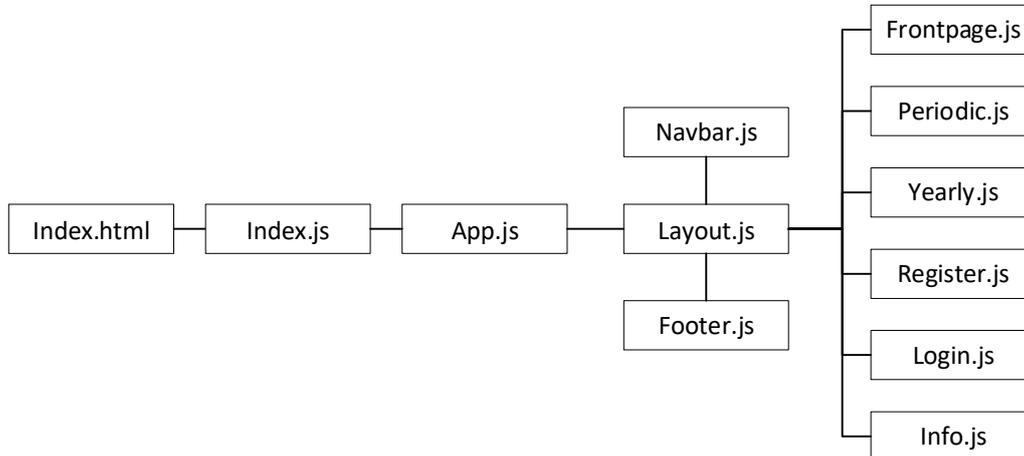


Figure 7.2 the structure from the Html index file and all the components

The layout of the React part can be separated into static and changeable components. The static components are initialized upon startup and unmounted when closing the web page. In contrast, the changeable components can be swapped in and out while the application is running. The changeable components are Frontpage, Periodic, Yearly, Register, Login, and Info. Each of the changeable components has a unique extension added to the URL, so it is possible to bookmark them in a browser. The first time the web page is loaded into a browser, it returns the Frontpage as default. Figure 7.3 shows the nested structure of the Frontpage.

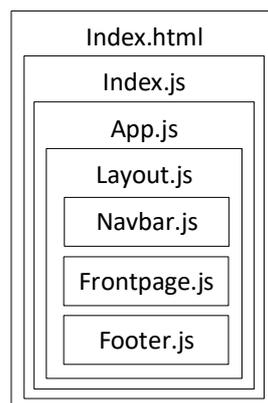


Figure 7.3 the nested structure of the Frontpage.

7.3 Navigation

The navigation between the components is set to component "Navbar.js" which contains the navigation bar. Navigating between the components is done by clicking the names or the icon on the navigation bar. When a name or icon is selected and is not the displayed component,

the current component is swapped out. This process is controlled by the "App.js" and happen in the following order:

1. The navigation bar update the state of which component shall be displayed in App.js
2. App sends dismount to the current component
3. App sends initialize to the new component
4. When the new component renders, the app sends it to the Index.js
5. Index.js updates the index.html

Figure 7.4 Displays the swapping between the components "Frontpage.js" and "info.js."

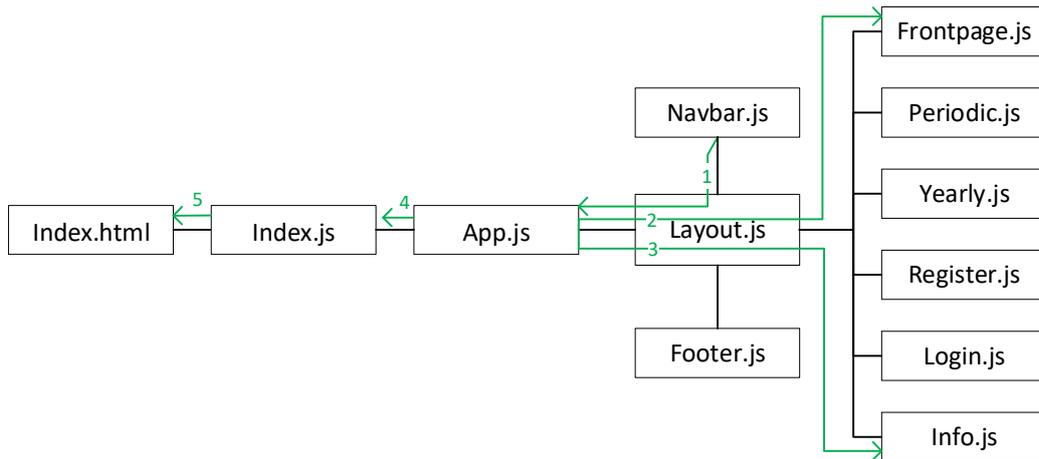


Figure 7.4 Displays the swapping between the components "Frontpage.js" and "info.js."

7.4 Functions

Some of the functions developed are reused in several of the classes. The functions used in more than one component are side menu, fetch, and presenting data in a table.

7.4.1 Side menu

The components yearly, periodic, and register, has a side menu. The setup for the side menu is the same structure. The purpose of the side menu is for the user to select the desired view³. Whenever a user clicks the link for the desired view, the side menu triggers an event to get the content and presents it. Each of the links contains the id for the request view. The side menu is categorized into the different sites responsible for yearly and monthly reports, with a category for each report. The side menu for registration displays all the different possible registration options directly.

7.4.2 Fetch

The communication with the backend is handled through the standard React API library. The protocol is JSON with HTTP. The code either asks for data or sends data, and the command

³ The view can be a report or a manual registration.

for the communication is fetch. The components yearly, periodic, and register, has an implementation of the fetch.

To show how it is implemented into the web application, an example of how it works is Feed Water's monthly report. The user desires to see the monthly report for Feed Water and clicks the "Matevann" link in the side menu. It triggers the fetch and sends out a request to the backend as (web API). The request is <http://172.24.22.87:5540/api/monthreport/9/2021/05/>. Then the backend replays with appropriate information.

Table 7.1 shows how the HTTP request is built up. The frontend sets the ID, year, and month dependent upon the user input, while the rest is set as default. When a user selects a report for the first time, the frontend sets the present time for the HTTP request.

Table 7.1 HTTP request setup

Protocol	IP	Port	Route	Controller	ID	Year	Month
http	172.24.22.87	5540	api	monthreport	9	2021	5

7.4.3 Presenting table with data

The presentation of data in a table from the backend is a function used by the year reports, periodic reports, actual measurement, and manual registration. The data object received from the backend is a JSON object that is constructed as a matrix.

The matrix for both the year reports and periodic reports is designed as shown in equation 7.1. The information in the equation is an array containing the parameters of fix setup and is shown in equation 7.2 with an explanation of abbreviation in Table 7.2. The matrix is divided into three parts. The first part is the information that utilizes the coloring of the columns. The name and unit are set directly into the table as table headers. The last part is all values for the table. The values are flipped from rows to columns before they are inserted into the table.

$$\begin{bmatrix}
 \textit{information} \\
 name_1 & \dots & name_n \\
 unit_1 & \dots & unit_n \\
 value_{11} & \dots & value_{1n} \\
 \vdots & \ddots & \vdots \\
 value_{m1} & \dots & value_{mn}
 \end{bmatrix} \tag{7.1}$$

$$[N_{vv} \quad N_{vc} \quad N_{dc} \quad P_{vc_1} \quad \dots \quad P_{vc_n} \quad N_{dc_1} \quad \dots \quad N_{dc_n} \quad D_v] \tag{7.2}$$

Table 7.2 abbreviation explanation of equation 7.2.

Abbreviation	Description	Type
N _{vv}	Number of verified values	Integer
N _{vc}	Number of verified columns	Integer
N _{dc}	Number of difference columns	Integer
P _{vc}	Position of verified column	Integer

Ndc	Position of difference column	Integer
Dv	Difference value	Integer

The matrix for the actual measurement and the manual registration table is designed as shown in equation 7.3. The matrix is divided into two parts. The name and unit are set directly into the table as table headers. The last part is all values for the table. The values are flipped from rows to columns before they are inserted into the table.

$$\begin{bmatrix} name_1 & \dots & name_n \\ unit_1 & \dots & unit_n \\ value_{11} & \dots & value_{1n} \\ \vdots & \ddots & \vdots \\ value_{m1} & \dots & value_{mn} \end{bmatrix} \quad (7.3)$$

7.5 Components

The components for the use cases have been created with a life cycle based upon the user input. The components of this life cycle are the monthly report, yearly report, manual registration, and log in.

7.5.1 Monthly report

The monthly report is named `Periodic.js` in the application. `Periodic.js` is a class component that handles the two use cases, gets a monthly report, and updates a monthly report. In addition, `Periodic.js` holds the actual measurement of a month. Figure 7.5 shows the flow diagram of how `Periodic.js` handles user input for presentation data to the user. Figure 7.6 shows the flow diagram of handling update values in a monthly report.

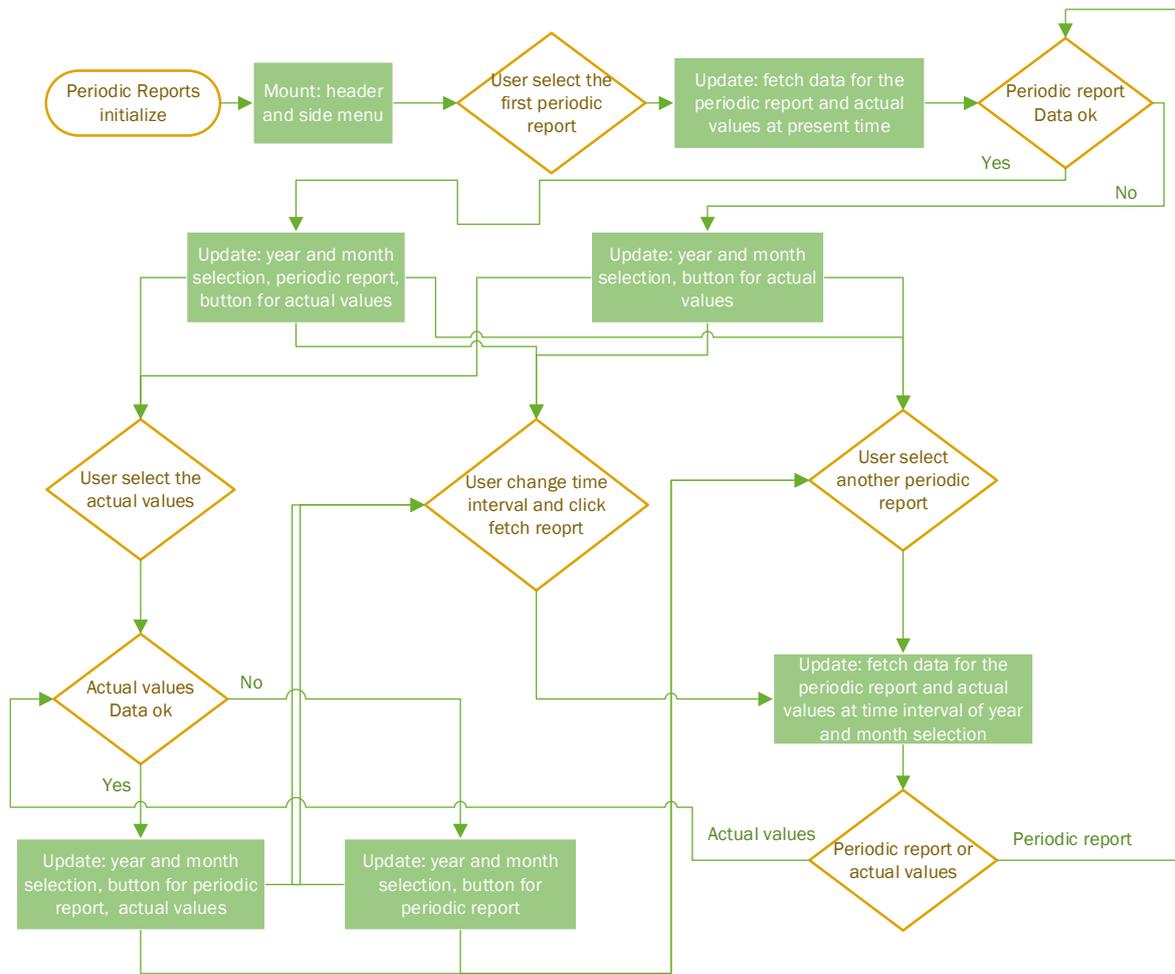


Figure 7.5 the flow diagram of how Periodic.js handles user input for presentation data to the user.

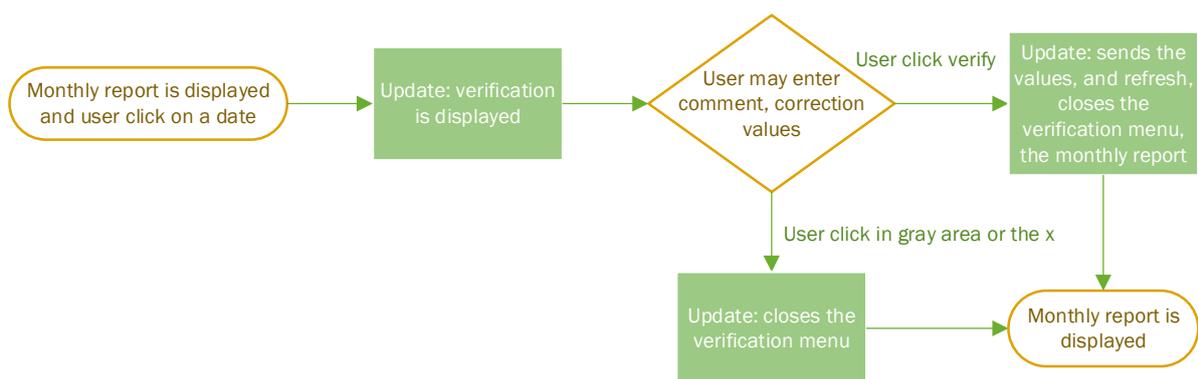


Figure 7.6 the flow diagram of handling update values a monthly report.

7.5.2 Yearly report

The yearly report is named Yearly.js in the application. Yearly.js is a class component that handles the use cases get a yearly report. Figure 7.7 shows the flow diagram of getting a yearly report.

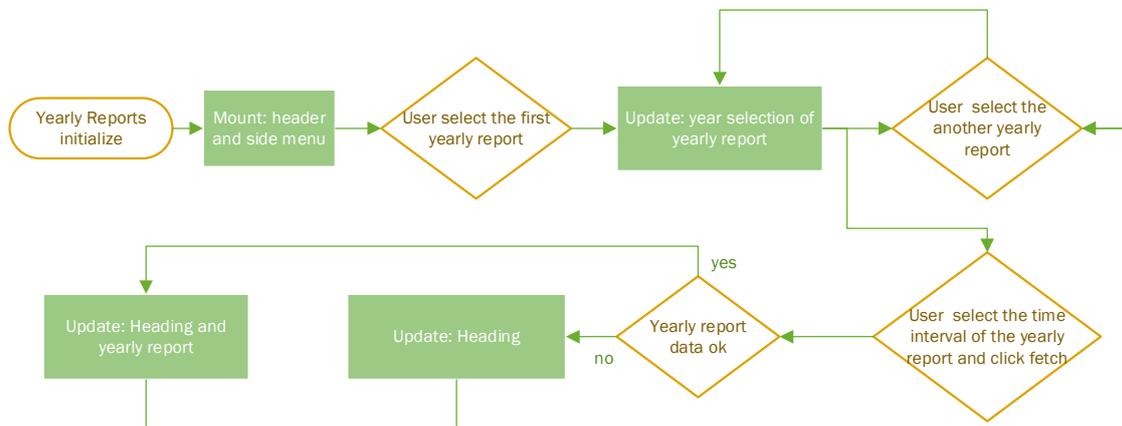


Figure 7.7 the flow diagram of the getting a yearly report.

7.5.3 Manual registration

The manual registration is named Registry.js in the application. Registry.js is a class component that handles the use cases manual registration. Figure 7.7 shows the flow diagram of the manual registration.

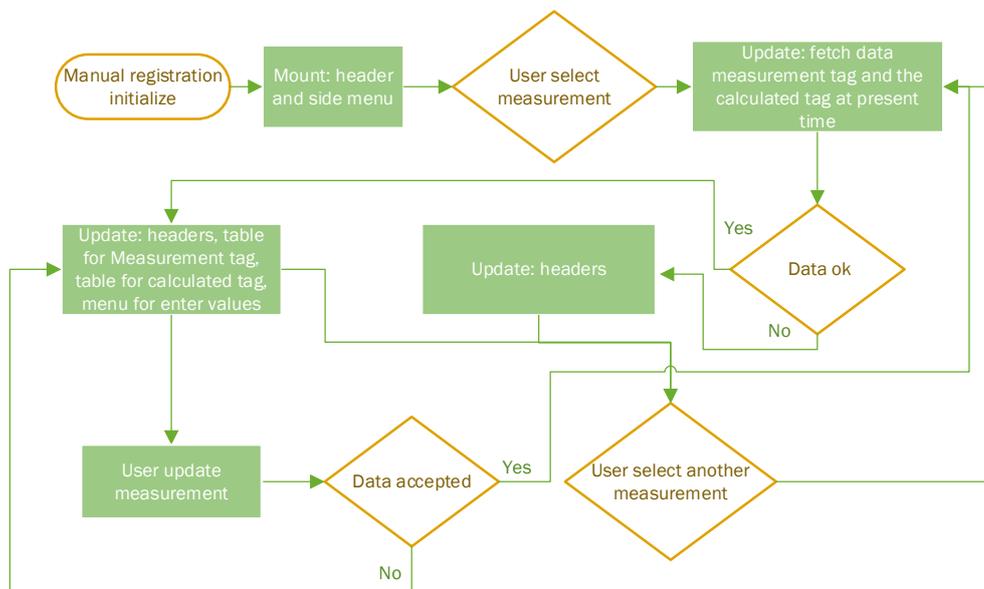


Figure 7.8 the flow diagram of the manual registration.

7.5.4 Login

The Login is named `login.js` in the application. `Login.js` is a class component that handles the use cases get a login. Figure 7.7 shows the flow diagram of the login.

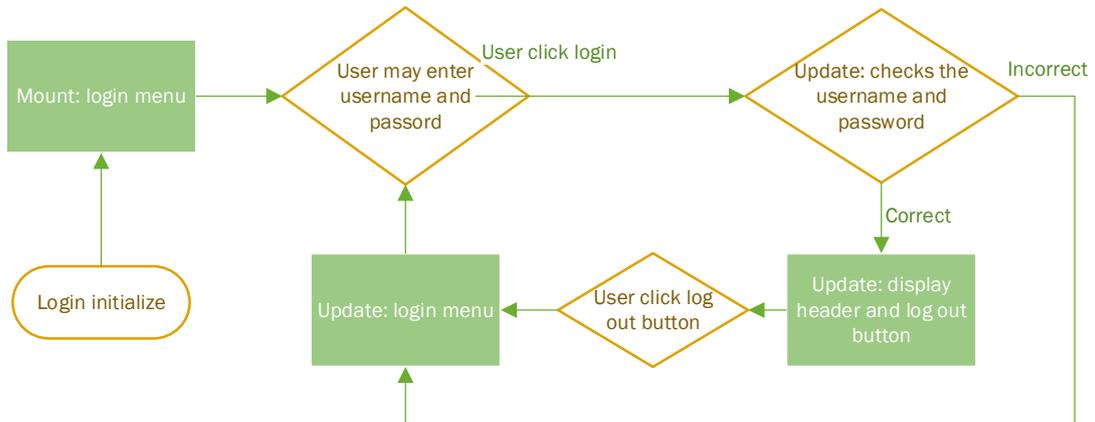


Figure 7.9 the flow diagram of the login.

8 Developed Backend

The backend is a Web API application integrated with the UPHD. The application tasks handle interaction with UPHD, perform the use cases, and send the data on request. The given application name is API_PHD_NET_Framework.

8.1 Structure

The application is structured into a folder containing subfolders as the layers. Figure 8.1 Show a diagram of the folder containing subfolders of the application and some of the files. In addition, there is a line between some of the folders and files to indicate the connection between them. Most of the presented files are classes inside the application. The folder represents a part of the application and are:

- Program and startup are the roots of the application
- The controller is the Web API service layer.
- Model is the business logic layer.
- PHD is the data access layer for UPHD.
- The parameter is the predefined values of the application.
- The library is the storage of needed DLL.
- IP21 is the data access layer for IP21.

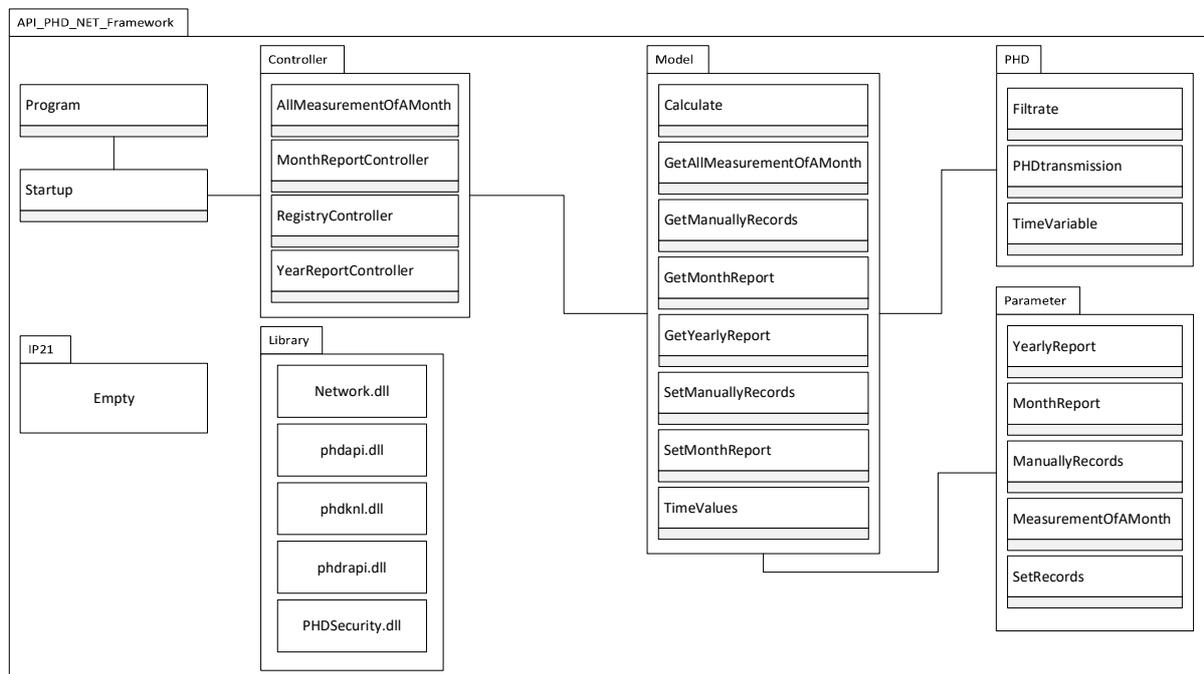


Figure 8.1 diagram of the folder containing subfolders of the application and some of the files.

Program and startup

8 Developed Backend

The program is a class and the root of the application that launches the application. The program initializes the startup class. The startup class sets the application settings and enables the Web API services layer to be activated for requests.

Web API service layer

The Web API service layer is the application interface for other parties, where HTTP request is sent to the application. The Web API service layer is located inside the folder controller, where four classes represent the start for five of the use cases. Each of the classes is created with the same design patterns. The design patterns are Controller, Creator, and Low Coupling. The classes have either one or two functions, an HTTP get and an HTTP post function.

Business logic layer

The Business logic layer is where the logic for converting data either from UPHD to report or decoding incoming data to be set in UPHD. The business logic layer is located inside the folder model, which is the logic for five use cases. There are six classes responsible for handling the use cases, while the Calculation and Time Values are support classes. The design patterns for the six classes performing the use cases are High Cohesion, Controller, Strategy. The Calculations class has the design patterns Polymorphism and Singleton, and the Time Value has Singleton.

Data access layer

The Data access layer is the interface against databases. Currently, there are two folders, one for UPHD and IP21. The folder for IP21 is empty at the moment. The folder PHD contains three classes. PHD transmissions, Filter, and Time Variables. The PHD transmissions class is created with the design patterns Facade, High Cohesion, and Information Expert. The filter and Time Variables are design patterns Low Coupling and Singleton.

Parameter

The parameter folder contains static classes with the specific information of each report. Each of the classes is based on Low Coupling, Singleton, and Information Expert.

Library

The library folder contains all the needed DLL for the application. This folder holds all the DLL that is used in one place. When the application is published, these DLLs need to be copied into the published application.

8.2 Operations

The principal design idea has been kept and extended for performing the operations for the use cases and a new function. The operations of the application are divided into these:

- Get a yearly report
- Get a monthly report
- Set update values
- Get all measurement of a month
- Get registered and calculated records
- Set registered and calculate records

Instead of explaining each of them, a block diagram is representing each of them. The number indicates the order of the loop between each of the elements for performing its task. The block UI represents a request, and PHD represents both UPHD.

Get a yearly report

The use case of returning a yearly report is an 18 steps loop and starts with an HTTP get request with the last part in order controller name, identification number, and year. Figure 8.2 shows a block diagram of the loop for returning a yearly report upon a valid HTTP request. The returned yearly report is an object in JSON format containing the data.

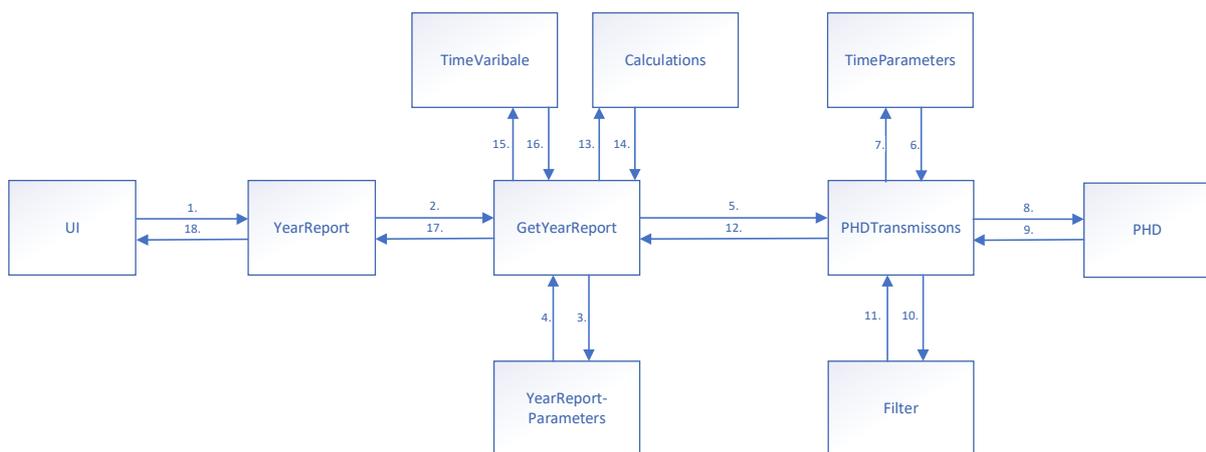


Figure 8.2 block diagram of the loop for returning a yearly report upon a valid HTTP request.

Get a monthly report

The use case of returning a monthly report is an 18 steps loop and starts with an HTTP get request with the last part in order controller name, identification number, year, and month. Figure 8.2 shows a block diagram of the loop for returning a monthly report upon a valid HTTP request. The returned monthly report is an object in JSON format containing the data.

8 Developed Backend

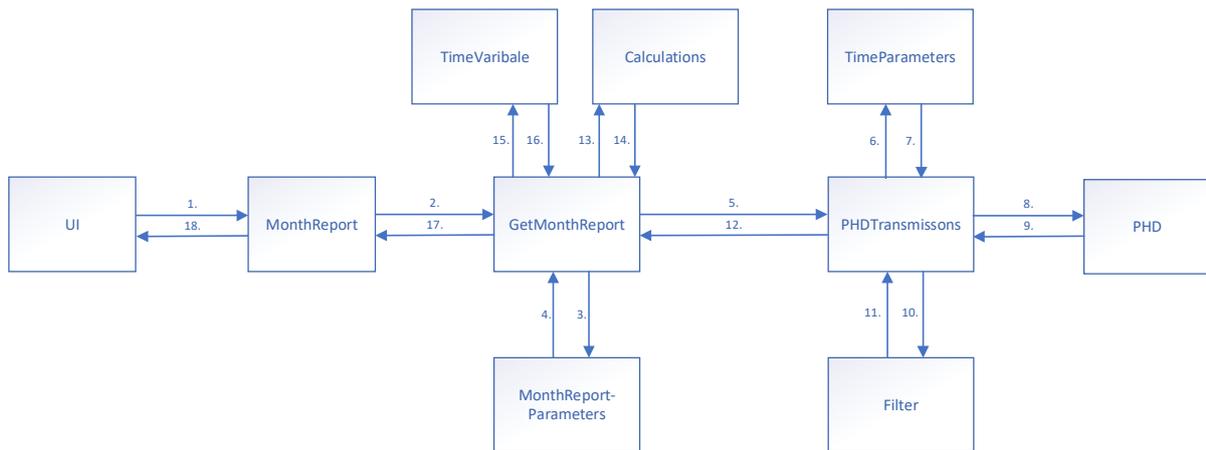


Figure 8.3 block diagram of the loop for returning a monthly report upon a valid HTTP request.

Set update values

The use case of set update values is an 11 steps loop and starts with an HTTP put request with the last part in order controller name and an objected of data in JSON format. Figure 8.4 shows a block diagram of the loop for set update values upon a valid HTTP request. The returned value is OK or BAD.

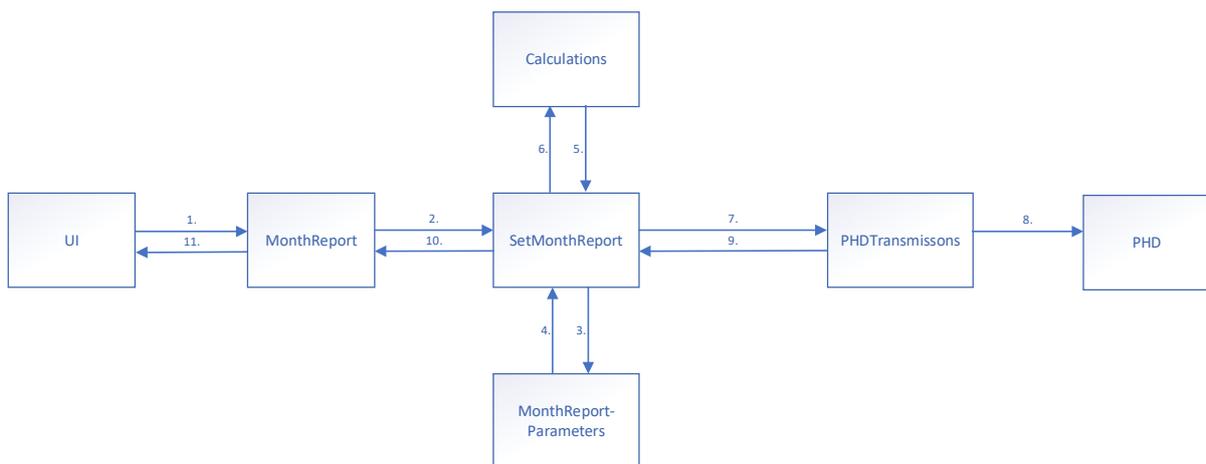


Figure 8.4 block diagram of the loop for set update values upon a valid HTTP request.

Get all measurements of a month

The new feature of getting all measurements of a month is a 14 steps loop and starts with an HTTP get request with the last part in order controller name, identification number, year, and month. Figure 8.5 shows a block diagram of the loop for getting all measurements of a month upon a valid HTTP request. It returns all measurements of a month as an object in JSON format containing the data.

8 Developed Backend

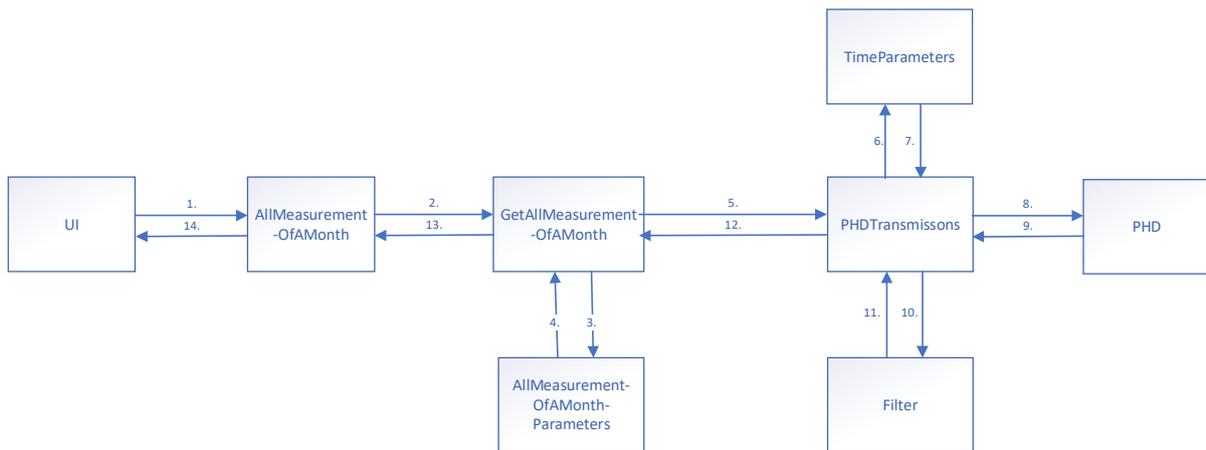


Figure 8.5 block diagram of the loop for getting all measurements of a month upon a valid HTTP request.

Get registered and calculated records

The use case manual registration is divided into two parts. The first part is getting registered and calculated records and is a 14 steps loop and starts with an HTTP get request with the last part in order controller name, identification number. Figure 8.6 shows a block diagram of the loop for returning either registered or calculated records upon a valid HTTP request. The returned registered and calculated records is an object in JSON format containing the data.

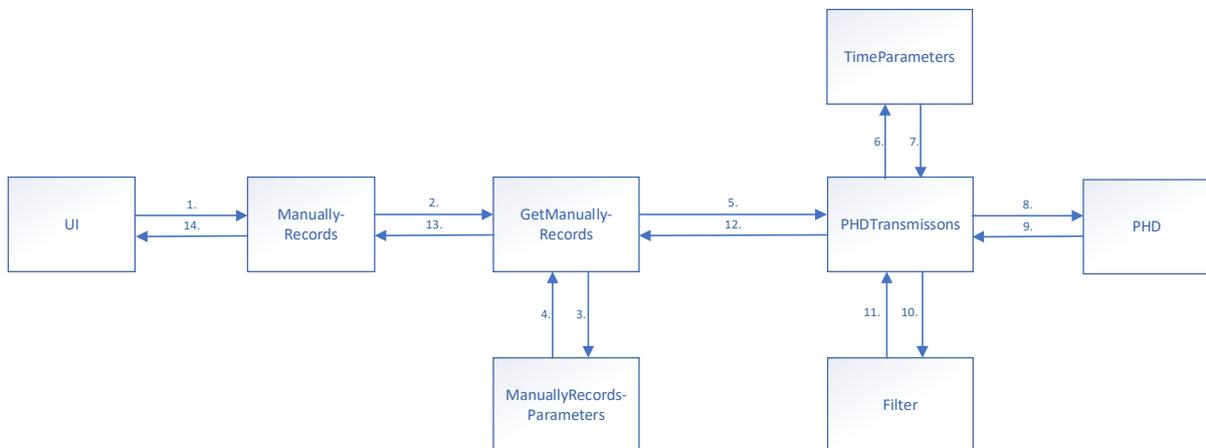


Figure 8.6 block diagram of the loop for returning either registered or calculated records upon a valid HTTP request.

Set registered and calculate records

The second part of the use case manual registration is a ten steps loop and starts with an HTTP put request with the last part in order controller name, and an objected of data in JSON format. Figure 8.7 shows a block diagram of the loop for setting either a registered or calculated record upon a valid HTTP request. The returned value is OK or BAD.

8 Developed Backend

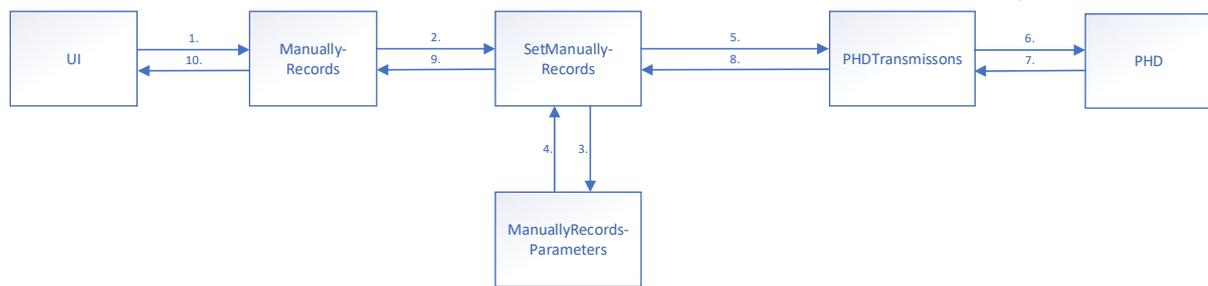


Figure 8.7 block diagram of the loop for setting either a registered or a calculated record upon a valid HTTP request

8.3 Functionality descriptions

The application's functionality is to have one unique entry point for each task to be performed. From this entry point, an object is created for handling the operation. Each type of operation has a unique class that fits the operation for all equal action. This class aims to perform the use case and controls the order of tasks and flow of data between different objects and static classes, with one exception. The exception is that the class that interacts with UPHD has its static classes for filtering and time intervals purposes. Upon successful or failed execution, the application returns a replay.

8.3.1 Access Layer

As mentioned earlier, each class holds either one or two functions and is now referred to as operations. The Month Report Controller and Registry Controller have two operations, while the others have one. The operation all classes have is getting a report

The operation for getting a report is built up correspondingly for each of them inside the access layer. When the operation is called upon, it creates an object and starts a function inside the object for getting an object list from the corresponding get-class in the business logic layer. On receiving the object list from the function, the operation sends it out.

The operation for posting is unique for each Month Report Controller and Registry Controller. The difference between them is that Registry Controller sets variables directly, while the Month Report Controller sets arrays directly to its corresponding set-class. Then operation activates the function to insert in the set-class. The operation either sends an HTTP status of 200 (OK) on success or an HTTP status of 400 (BAD) on failure. . An example of the function code for the Registry Controller is given in Appendix E.

8.3.2 Business layer

The business layer can be divided into four parts: get-classes, set-classes, calculations, and Time Values. The get-classes and set-classes performing the use cases are built upon the same principle. The principle is that each of them requires identification and has one public function that can be enabled. The Calculation and Time Value classes are static and have reusable functions for the get- and set-classes.

Get-classes

The get-classes are created on the same idea, but they vary from each other due to the task they shall perform. Figure 8.8 Shows a simplified diagram for the process to get a monthly report or yearly report. Figure 8.9 Shows a simplified diagram for the process to get all measurements of a month. Figure 8.10 Shows a simplified diagram for the process to the get registered records. An example of the primary function code for getting a monthly report is given in Appendix E.

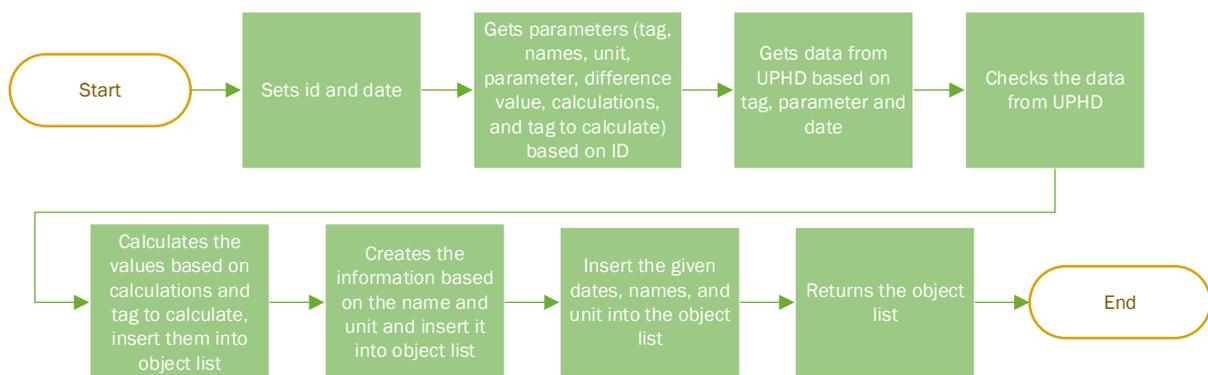


Figure 8.8 a simplified diagram for the process to get a monthly report or yearly report.



Figure 8.9 a simplified diagram for the process to get all measurements of a month.

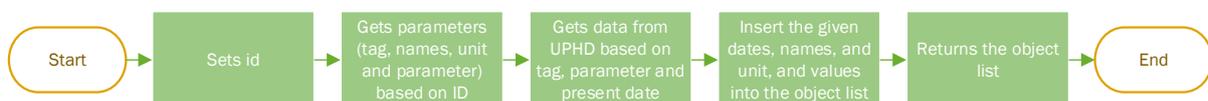


Figure 8.10 simplified diagram for the process to get registered records.

set-classes

The two set-classes are based on the same idea, but they vary from each other due to the task it shall perform. It is currently only the setting either a registered or calculated record that is developed, while the update values for a month are not finished. Figure 8.11 shows a simplified diagram for the process of setting either a registered or a calculated record. Figure 8.12 shows a simplified diagram for the process to the update values for a month. An

8 Developed Backend

example of the primary function of setting either a registered or a calculated record in code is Appendix C.

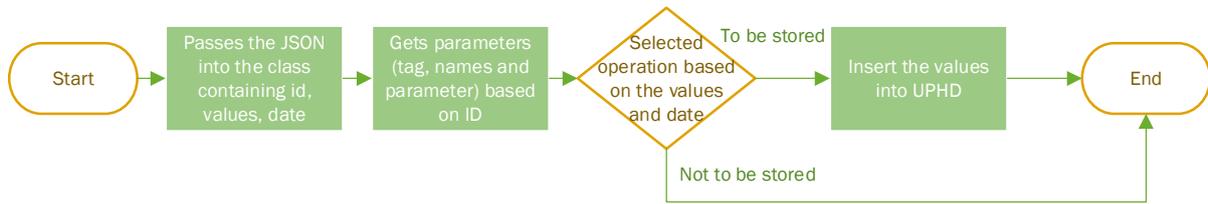


Figure 8.11 a simplified diagram for the process to set setting either a registered or calculated record.

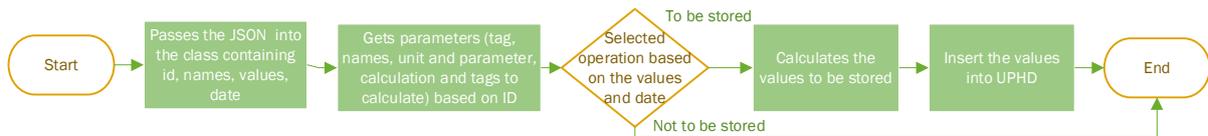


Figure 8.12 a simplified diagram for the process to the update values for a month.

Time Value

The static class Time Value is used for converting time values into names or dates for getting Monthly Report and Get Yearly Report.

Calculation

The static calculation class holds all the calculations of all reports. The calculation types are converting entity, mass balance regarding ratio, sum, subtraction, difference, and percentage. The total variation of the unique calculation is currently 56. An example of sum and mass balance regarding ratio as formula and code is given in Appendix A.

All calculations have the condition that all values must be stored inside the database. This condition enables the calculation to be reusable in other reports. In addition, the usage of a calculation inside another calculation has tried not to occur. This strategy is for making the code more readable to others.

8.3.3 Data access layer

The data access layer is responsible for connecting the UPHD to set or get values from it. All three classes of the data access layer are only in use if the operation is getting data from UPHD. In contrast, for setting data to UPHD, only the class PHD Transmissions is used. Figure 8.13 shows a principal sketch of getting a dataset from UPHD. Figure 8.14 shows a principal sketch of setting a dataset to UPHD.

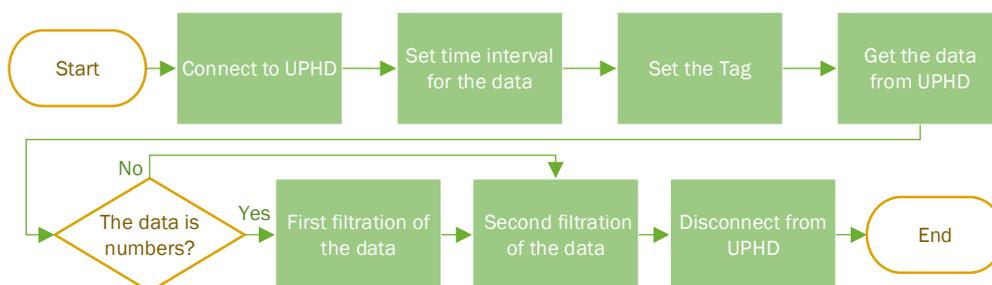


Figure 8.13 principal sketch of getting a dataset from UPHD



Figure 8.14 principal sketch of setting a dataset to UPHD

8.3.4 Parameters

All the needed information for getting and setting values into UPHD is located inside different static parameter classes. The static classes are named up after the type of use case. The design of these classes is based upon a database structure for being pushed into a database later.

The information is stored inside a function, and the name of the function is the same as the report. Figure 8.15 shows the static classes with the variable and functions for each of the types of reports⁴.

MonthReport	YearlyReport	ManuallyRecords	MeasurementOfAMonth	SetRecords
+String[] name	+String[] name	+String[] name	+String[] name	+String[] name
+String[] unit	+String[] unit	+String[] unit	+String[] unit	+String[] tag
+String[] tag	+String[] tag	+String[] tag	+String[] tag	+byte[] parameter
+byte[] parameter	+byte[] parameter	+byte[] parameter	+byte[] parameter	-----
+byte[] calcType	+byte[] calcType	+SetEmpty	+SetEmpty	+SetEmpty
+byte[] calcTag	+byte[] calcTag	+SetReport	+SetReport	+SetReport
+double perdiff	+SetEmpty			
-----	+SetReport			
+SetEmpty				
+SetReport				

Figure 8.15 the static classes with the variable and functions for each of the types of reports.

8.4 Communication with UPHD

Communication with the different UPHD is enabled through the Phdapinet. The data access layer uses the functions of the Phdapinet to perform its tasks.

8.4.1 Posting data

From the documentation from Honeywell Inc., the function put is the most suited for inserting data into the UPHD. The put function is an overloaded function⁵ and requires a tag name and value as a minimum for setting the value to UPHD. The used put function is the

⁴ Parameter is which UPHD the tag is stored and if the tag is number or text. Set report is short for all the types reports

⁵ An Overloaded function is different function having the same name, where the values sent to the function determine which one of them is used.

one requiring a tag name, a value, and a timestamp. This function enables the storage of a value at a chosen timestamp inside UPHD.

8.4.2 Extraction of data

From the documentation from Honeywell Inc., the function fetch is the most suited for getting data from the UPHD. Efficiently using the fetch function to get data from the UPHD, a discriminant analysis was performed to extract data. The optimal extraction method between time and data is connecting, setting the time interval, setting the tag, getting the tag's data, and then disconnecting from the UPHD. To switch between tags and keep it efficient, setting the tag and getting data is repeated numerous times before disconnecting.

8.4.3 Data storage

The storage of values inside the UPHD is essential to understand before the filtration can be explained. All data of a tag stored inside UPHD uses the timestamp as the index. The UPHD has existed for many years and has been modified. Hence the storage of data is not uniformly done into UPHD and differs between the sites.

The current metering report system store most values with the timestamp at 03:00:00 the day after. The manually registered values are stored either at 00:00:00 or 07:00:00. In addition, there is a couple of inconsistency for a few tags back in time, where the timestamp is somewhere between 21:00:00-03:00:00.

The raw values used for the monthly reports have different possibilities of being stored. The primary method is to store data at midnight around 23:10 and 00:10 a clock the day after. The system setting the timestamp in the UPHD is the metering server, implying summer- and wintertime can give some problems with the timestamp setting.

The metering server holding the values is a redundant system meaning there are two metering servers, so if one fails or reboots, the other server takes over. Whenever this occurs, the metering server that stops sends the values to the UPHD before the other takes over. Once the other server takes over, it also sends the values it holds to the UPHD.

The UPHD is also in charge of storing some raw values and using the parameter Scan Frequency of the tag. The Scan Frequency is how often the tag shall be stored. Commonly used Scan Frequencies are 3600 sec (every hour) or 86400 sec (every day). There are tags with a Scan Frequency lower than every hour. In addition, there exist tags that are stored at midday every day. The raw values stored in the midday are applying only to the current measurement.

Inovyn has some unloading and filling stations where products are from and to boats and cars. When a product has been loaded, the consignment note from the event is stored in UPHD. However, the same consignment note can be stored once or many times in UPHD.

8.4.4 Filtration

The data extracted from each tag need to be filtered since it is not given that the correct data was gotten. There exist several filters dependent on the desired output.

8 Developed Backend

The first filtration is only viable for numbers and not for text. This filtration is done right after the values of a tag are received from UPHD. The task for the filtration is to get viable numbers and removes all data that is not viable. Figure 8.16 shows the flow diagram of the first filtration.

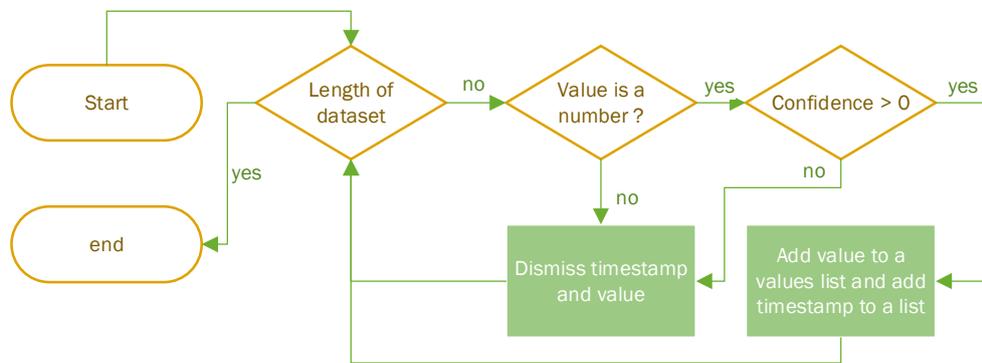


Figure 8.16 the flow diagram of the first filtration.

The second filtration happens in a static filter class that contains different filters. There are currently three filters in use for text, monthly numbers and yearly numbers. There is not a second filter for either manual registration or actual measurement for a month.

The text filter reduces the number of text strings containing space in ASCII Encoding (%20) and gets the correct comments for the monthly report. The reason is that the current system sends space inside the comments to be stored in UPHD.

The yearly filter is to get all the numbers inside the desired year and add zero if there is a missing value on a date. The range for the yearly filter is from the first of January to the second of January next year. The timespan it gets values out of is from 21:00 to 04:00 the next day.

The monthly filter is to get all the numbers inside the desired month and add zero if there is a missing value on a date. The current monthly filter handles most of the different stored raw values. Figure 8.17 shows the flow diagram of the current monthly filter.

8 Developed Backend

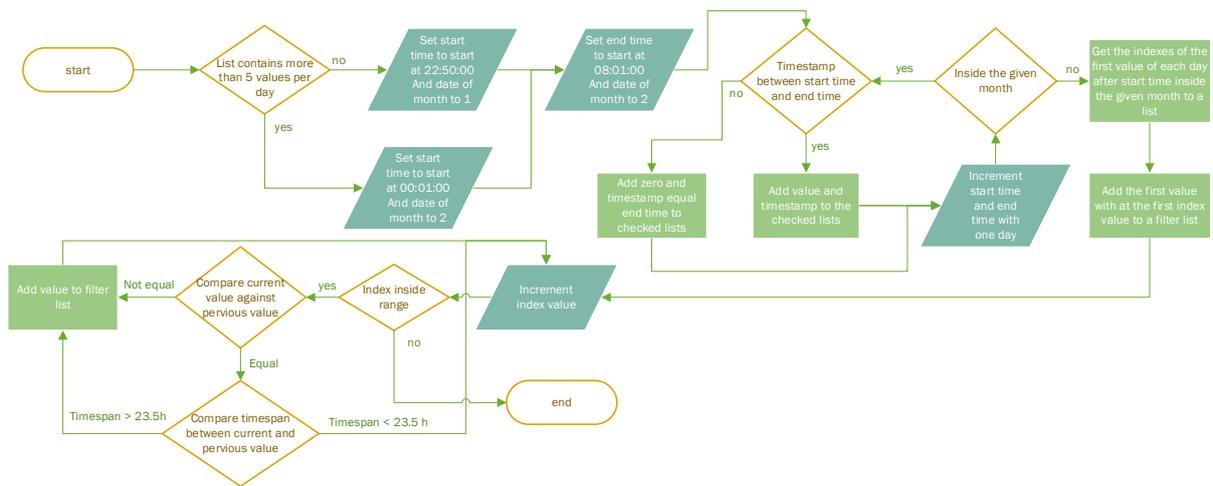


Figure 8.17 the flow diagram of the current monthly filter.

The filter for a consignment note is currently in development. The reason is that there is inconsistency in the stored raw values.

9 Testing

It has been conducted tested at all stages of the development of the prototype.

9.1 UI

Testing of the code written for the UI has been tested in Google Chrome. The expansion "React Develop Tool" has been used in Google Chrome. It gives the benefit of seeing which element and component are rendered in order.

As the UI is booted from a C# application, it is possible to run the code with Visual Studio 2019 while editing the code with Visual Code. The benefit is the possibility to alter the code of the user interface while it is running.

The test strategy has been to write the code while running in debug mode in the C# application and test it as a new line of code has been implemented.

9.2 web API

The C# web API application has had a form for unit testing as the test strategy. The reason for calling it a form for unit testing is because the application has been developed, compiled, and tested in steps.

The first part has been developed in steps in the same order as operations in chapter 8.2. It has mainly been focused on the logic of each module, and one test tag has been used for developing and testing when needed. The second part has been implementing the information and calculation if required for one of either a report or manual registration and then testing it.

9.3 Version test

A version test was performed before publishing and deploying the two applications to the webserver. The version test consists of an integration test between the two applications and tests all possible user input scenarios. The integration test consisted of testing the communication work as planned between the two applications and checking the web application's performance.

9.4 User test

The user test has been to let the user try out the prototype freely. It was added an information page so the user could read what was implemented in the current version.

10 Maintenance

The new application aims to reduce the required maintenance to a minimum. There will always be some maintenance required for a software application.

10.1 Version updating

Most software is constantly under development to keep up, being enhanced, or adapt to new technology. As this application has been developed with programming languages constantly under development, it will require to be updated. The update rate dependent upon how often a new version is released of the specific programming language. Since it has been used several versions of frameworks and libraries for the applications, it is affected by its update rate.

Frontend

As per the time of writing this report, a new version of React library has already been released. The current version of the React library is 17.0.2 [59], while the application is currently 16.14.0. The typical rate of a new version of the React library is between one month to six months. It has been announced that version 18.0.0 will come some time into the future, but not given a release date.

In contrast to the React library, Microsoft has a release plan and support overview of the current versions and the ones to come. Figure 10.1 shows the timeline of the version and release time for the .Net Core and .Net. .Net Core and .Net reference to the ASP.NET Core, Entity Framework Core, and more. LTS stands for lifetime support and is defined by Microsoft to be three years.

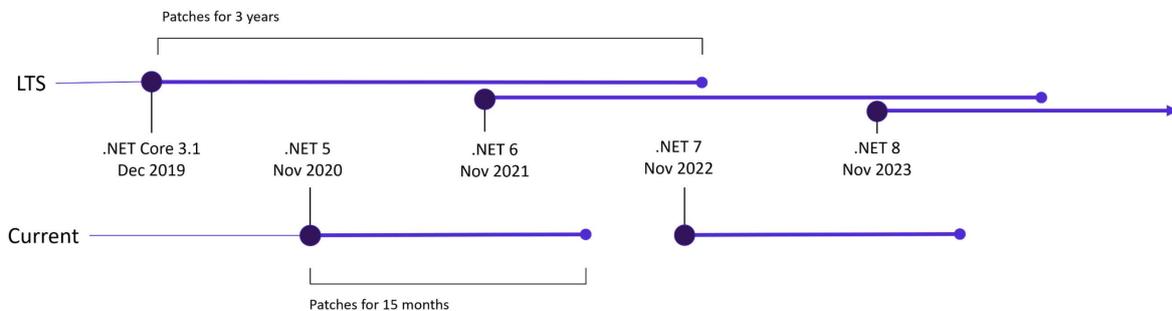


Figure 10.1 The release schedule for the .Net Core and .Net [60].

Backend

The .Net Framework 4.6.1 has not been given an exact date from Windows when going out of support. As the .Net Core 2.1 has been used with the .Net framework, it will hold the support to the .Net Framework 4.6.1 [61]. Currently, the .Net Framework 4.6.1 follows the life cycles to the following operating system to Microsoft: Windows 7 SP1, Windows Server 2008 R2 SP1, Windows 8.1-updating, Windows Server 2012, Windows Server 2012 R2, and Windows 10. [62]

Furthermore, the backend depends on the DLL from Honeywell Inc. So, to update the version of the code. A new version of the DLL from Honeywell is needed, or a new type of solution has to be developed with another structure for handling the DLL from Honeywell Inc.

Summarize

Each of the applications will have different update rates. The frontend may have an update rate of six to nine months, dependent upon each of the software's new releases. While the backend can handle standstill until a new version or method for communication is available for the UPHD by Honeywell Inc.

10.2 Modifications

Modifying the applications with new features and functions is dependent on the skills and availability of knowledgeable resources. There are many possibilities for enhancing and expanding the application, but it will depend on the requirements and the priority.

10.3 Change of reports

The setup for altering any of the reports has been developed as a straightforward procedure. For altering a report, some information is needed of how it shall be changed. Under the development of the prototype, a change in a report happened. Appendix B shows the information that is needed for changing a report for the prototype.

11 Comparison

The main discussion is that the prototype a sustainable alternative compared to the current system.

11.1 Technology

The selected framework, library, and programming language are all more sustainable than the current system holds, and OS for the webserver is a newer version than the current system. Since the framework, library, and programming language are more popular and commonly used, it is probable to be known by other developers. It gives current and maybe new employees the benefit of both Inoes Rafnes and Inovyn may program it.

11.2 Design

To compare the prototype against the current system, the contrast between them is the design. The current system is an MPA containing the logic of each report inside each page. In comparison, the prototype is a SPA containing all the logic on the server. The primary benefit is that it reduces the load on the webserver since most of the content is sent at the first request.

11.3 Performance

The performance value of time is an essential component to compare between the two systems. A comparison test between the two systems has been performed. The comparison test is based upon different operations that are equivalent for both systems, with changes in time intervals if possible. It has been chosen a monthly report and a yearly report containing a large amount numbers and is the monthly report of propane and yearly report of ethylene. In addition, the presentation of manual registration of steam and air tested, and the new function of the prototype of getting all raw values of a month. Table 11.1 shows the comparison test between the two systems.

Table 11.1 the comparison test between the two systems.

Test	Operation	Response time for the Current system	Response time for The Prototype
1	Propane monthly report of May 2021	460 ms	573 ms
2	Propane monthly report of May 2020	389 ms	648 ms
3	Propane monthly report of May 2015	476 ms	587 ms
4	Propane monthly report of May 2010	339 ms	586 ms
5	Ethylene yearly report of 2021	526 ms	562 ms

11 Comparison

6	Ethylene yearly report of 2020	826 ms	537 ms
7	Ethylene yearly report of 2015	922 ms	587 ms
8	Ethylene yearly report of 2010	925 ms	572 ms
9	Presentation of manual registration for Steam	329 ms	600 ms
10	Presentation of manual registration for Air	346 ms	586 ms
11	All raw values for the month of propane in May 2021	Not possible	964ms
12	All raw values for the month of propane in May 2020	Not possible	1,04s
13	All raw values for the month of propane in May 2015	Not possible	1.13s
14	All raw values for the month of propane in May 2010	Not possible	893ms

The table shows the prototype is slower than the current system, apart from the yearly report back in time.

11.4 Maintenance

The most significant advantage of the prototype is its flexibility in modifying reports. The current version of the prototype is only a tiny piece of the code that needs to be modified. In contrast, the current system needs a new page with the changes implemented and a new code for switching between the different pages depended on the time interval.

11.5 Functionality

The functionality of the manual registration has been enhanced in comparison to the current system. The prototype is not time-sensitive on the registered date-time value since the time value is fixed to the last registered date-time value. The current function can change a registered value up to a year back in time. This part has not been transferred to the prototype for the safeguarding of verified reports. Instead, the latest registered value can be reentered to update the calculated values, and the previously calculated and registered values are displayed for the current month.

The presentation of a monthly report is a shorter process in clicks and tabs than the current system. Selecting a monthly report in the prototype displays the report for the present month, and the time interval can be changed in the same view. In the current system, when a monthly report is selected, a new tab is opened, and the time interval has to be selected before the report is presented. To change the time interval, the user has to either go back to the page or switch to the main page and select the monthly report again.

The new function of enabling the user to see all stored data for a month is enhancing the prototype. In the current system, the user cannot see if the correct values are displayed. Instead, the user has to check another application for the specific tag if the value is correct.

11 Comparison

The monthly report and all raw values for the month are performed simultaneously in parallel by the prototype. This parallel function gives the illusion to the user that one of them is there instantly when switching between them the first time. In addition, if the user is disconnected from the intranet and has loaded the monthly report or all raw values for the month, it is still possible to switch between the two pages.

Updating a monthly report is equal in terms of how it shall be developed compared to the current system, with one exception. The exception is the function verify and unverified of daily measurement back in time. The reason is to take away the possibility of a report gets unverified when a report is approved. Furthermore, it has been considered only to approve the daily measurement after another approved daily measurement to enforce the daily measurement in the proper order.

The presentation of the yearly report is equal between the prototype and the current system. The current system holds one more function than the prototype, and that is that each monthly value in the yearly report is linked to the monthly report. Hence of the MPA design.

11.6 Status

The new custom-made solutions for a new Metering Report system must be evaluated if it is viable to replace the current system. The state of the prototype compared to the current system is the following:

- Monthly reports
 - o 22 of 29 is implemented and tested.
 - o 7 of 29 is in development (Lye and HCL reports).
- Yearly report
 - o 19 of 19 yearly reports need to be changed structure (Logic).
- Manually registration
 - o 7 of 7 manual registration is ready to be implemented.
- Login
 - o is not implemented currently.
- Update values
 - o UI is finished for sending verification, comment, and values for a day.
 - o Calculations of correction and verified values are incomplete before storing them in UPHD.
 - o Not developed a function for monthly correction values.

The development of the prototype is not finished and needs some more time to be completed.

11.7 Security

The security of the UI (C# application with React) is the same as the current system. Meaning it is only possible to interact through a web browser on the intranet.

The Web API application security is open for anyone connected to the intranet. It allows anyone to send a request with or without data to the application. The reason is for testing purposes, allowing to test it while it is running on the server. If the Web API application goes

into operation, it will be bound to the frontend's address. Meaning the Web API application will only be visible to the frontend on the intranet.

11.8 Future work

The development of the prototype requires some more work to be finished. The work that needs to be completed is the following:

- Lye and HCL reports need a special filter for the consignment notes.
- Yearly reports need to change to the same structure as the monthly report. Meaning the tags need to be moved.
- Develop the login feature incorporation with IT.
- Develop the calculation function for monthly reports for inserting values into UPHD.
- Decide where the monthly correction shall be implemented and develop into the application.

A time frame of how much time it will take is not easy to give, but a simplified estimation can be given. If the estimation is only coding and a final test, it can be estimated around one month with one person. The uncertainties of the estimation are the availability of the resource performing the task and the IT department for integrating the logging feature.

12 Discussion

12.1 Study for selection

The mixed study gave a perspective of the advancement of programming language frameworks and libraries. The quantitative data revealed that a programming language does not need to be new to being trendy. In contrast, frameworks and libraries seem to be the opposite, where the newer it is, the better option. The Qualitative data is based mainly on statements from different developers. The reason is that developers have a varying view on essential for a framework, library, or programming language. In addition, they tend to compare the type to others that are equivalent. The study gave insight into the essential argument why to choose or not to choose the options.

The study on programming languages has reduced a vast amount of options to be selected. Evaluating the selected options will endure into the future is ambitious to answer. The reason is that no one knows what the future holds. By applying the most widely used and popular parameters, it is possible to say it will endure sometime into the future. In addition, using an OS, platforms, frameworks, libraries, or programming language supported by a large corporation ensures someone is maintaining it even if it loses its popularity. In total, the selected parts will be sustainable at the moment and for some time into the future.

In consideration of the choices made, most of them are developed and maintained by Microsoft Corporation. The exceptions are JavaScript and React. The European Computer Manufacturers Association maintains JavaScript, and it has become standardized. Facebook maintains the React library. The sum gives the benefit of support, development, and version updates.

It is one exception. Microsoft ISS may need to be swapped with Apache or Nginx if Microsoft Corporation decides not to continue supporting and developing IIS. The reason is the newly published report from Netcraft. Figure 12.1 shows the market share of all sites of web hosting services from 29th March 2021 [63]. Microsoft ISS has lost a massive market share since 2019, and the percentage is down to 5,96%. In terms of number, it means that 70,826,342 websites are hosted with Microsoft ISS. market share of all sites of web hosting services from 29th March 2021

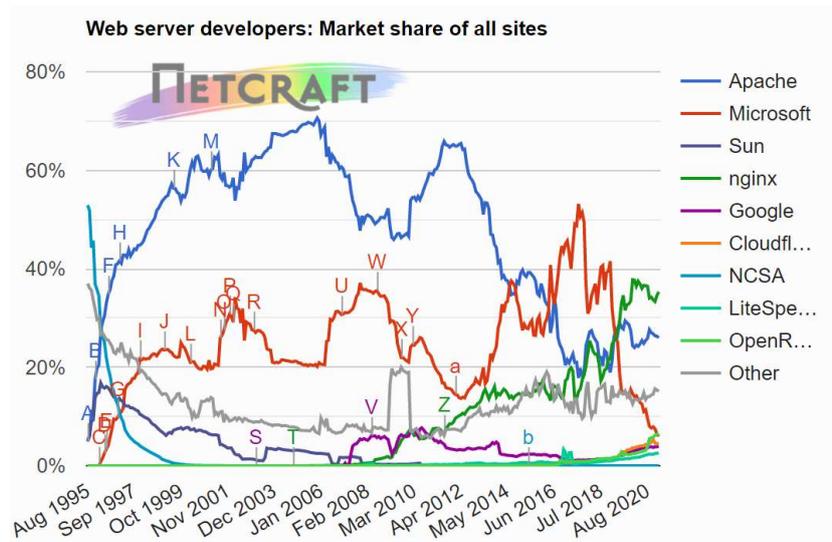


Figure 12.1 the market share of all sites of web hosting services from 29th March 2021 [63].

12.2 Development process

The schedule was based on reaching a new version at a given time. The tasks to be implemented at each version have been focused on being a use case. As input from the user has been given, the focus has been shifted to accommodate the suggested solution. The use cases had a time frame to be developed before being cut off and moved on to the next use case for showing progress.

The schedule was created to consider what was believed to be the most challenging part and presented to the users. It has been reconned that the development process began in the wrong order. The development process should have started with all aspects of the UPHD instead of the UI.

Scrum has not been utilized in the sense of documenting progress and updating plan in digital format. The project started with a digital format but was discharged because of the time usage that went into it. The development process has relied on Scrum. The tasks to implement have been written on paper and crossed out on completion. The downside of not logging digital is that there are no good records of the progress. The sprints were often prolonged for trying to complete the tasks.

The scrum method is more suited for teamwork to track each team member and discuss the integration between parts of the application. In addition, ideas can be discussed, and the responsibility of parts can be appointed amongst team members. As this development has been a one-person show, all parts fall on one person. The solution has been to try seeing the whole application before a new function was to be implemented. This has led to each use case treated as independent parts of the application, and similar processes between each use case may have been overlooked.

12.3 Frontend development

The creation of the web application started with a steep learning curve. It has been put more time into the development of the UI than assumed. The reason for the time consumption is because JavaScript and React were not known when the project started. It may seem to be a poor decision to select JavaScript with the React library for the frontend. In an optimistic view, this meant that many concepts had to be inspected and many attempts to get it right. The upside is that the code gets tested quite heavily to ensure it works properly. While the downside is that some code development has taken more time, and it may not contain the optimal solution for solving part of the code.

In addition, Elm was tried out to see if it could be a good option. The reason for trying out Elm was the performance, but the documentation and community were poor and trim. It was put aside after a week of learning the structure of the programming language. The time spent reviewing and testing Elm was a waste of time, but it gave some insight into how it can be utilized. It also backs up the arguments for not selecting it for the UI.

The current version of the web application holds an example of the downside. As React uses components, it is possible to create every function to enable reuse and easier maintenance of the code. Instead, each class has been treated independently, holding all the functions needed to perform its tasks. It is not an optimal solution for maintaining the code since this gives a large script inside each class. However, it is a solution, and it is possible to rewrite the script. Even though the class script is large, it is scripted with the functions in a sequence on the anticipated events, making the structure clear to the creator but not for others.

The presentation of the table is a function that is possible to recreate as one or more components. The function itself is considered optimal because it enables one presentation method to fit them all instead of having many individual functions for each report. Hence, it gives the benefit of not having specific variables in the code.

The current version of the UI holds the names and index of all the different reports, meaning they are constant in the code. A goal has been to have a minimum amount of constants in the UI to make it more dynamic to change. It may be possible to integrate them into a database later to increase the UI's flexibility.

12.4 Backend development

The code of getting and setting data to UPHD has been altered a couple of times. The altering depends on how the frontend desired the data, storage of data inside UPHD, and keeping it flexible. It has been challenging to develop a two parted system if one part changes, the other has to follow, and the backend has been suffering from it.

The major problem has been the extraction of data from the two UPHD. The goal was to create a universal way of getting the data for all reports back in time. The patterns of stored data have been tried to be handled by a universal filter. The strategy of one universal filter can be seen as a poor decision because it is time-consuming to expand the filter each time a new pattern unfolds. It may have been better to create unique filters for reducing the time of development. The strategy of keeping the universal filter was to enable the flexibility of changing the reports. In addition, it gives the benefit of making the system robust against new raw values that are added to reports.

12 Discussion

The implementation of the parameter classes has been one of the most time-consuming parts after the filter. All tags, names, units, parameters, type of calculation, tags to be calculated, and percentage difference values have been written into the application. The calculation for a specific report has been implementing alongside the implementation of the report.

In light of the dependency on .Net Framework for the DLL Phdapinet, the web application was split into two parts. The decision can be review as both good and bad to make. The positive is that the Web API can be integrated with other applications, and their maintenance is separate from each other. The downside is that two web applications run on the same server, and the communication between them takes a bit longer time.

12.5 Undeveloped function

The function that has not been developed is the login and adjustment of the total monthly values. The function of adjustment of the total monthly values was down prioritized since it is a rarely used function. It will have to be implemented at a later stage.

The login function stopped being developed after discussing with the users and IT to use Active Directory Domain Service (AD DS) for in logging. The short answer for stopping the development is the lack of knowledge to utilizing AD DS in the application with the system of the IT department. However, it is a better and safer solution to give authorization to the user through the windows login. The benefit is that the authorized personnel do not need to log in, and the application can be view by others safely.

12.6 Testing

The prototype was tested as it was developed. The strategy for testing may have been better. Instead of partly test it whenever a new change had been added, it could have been implemented all parts at once, which means not implemented one and one report and test it.

When the testing against both UPHD started, for the first yearly report, the challenges started. The first issue was seeing the different patterns of the stored data, which has been the main challenge. As more and more report was implemented, new patterns of how the data was stored were revealed.

Whenever a new pattern had been implemented to be handled by the filter, all previously implemented reports must be retested. As a result of all this testing, the filter is split into three filters to reduce testing time. The three filters are for text values, yearly numbers, and monthly numbers. The reason for splitting the filter this way is that the yearly number does not rely on monthly numbers because yearly numbers are approved and stored values.

Testing with the users was very beneficial for the prototype, especially for the UI. The users that tested the prototype gave input on desired functionality, features, and layout. It could have been more considerable interest by the users because three users have given input. Of the three users, only one of them works directly with the current system daily.

It has not created a test report of the prototype because of it is not fully developed. The prototype has mainly been tested for an extension or modification, which has not had a specific test document.

12.7 Documentation

Currently, the documentation is an excel sheet containing all the tags and a word document containing the report set for some of the reports. It has been focused on developing the product rather than documenting it. In light of the development process, it should have been considered to document more.

The excel sheet contains all 566 tags in use of the current system. Each tag is associated with the product, site owner, virtual or raw value, usage in a report, written to, Source Tag Type⁶, Datatype⁷, and Scan Frequency. The idea of the excel sheet is to transfer the useable information into a document for redevelopment purposes.

The word document is going to be a part of the final documentation for the users. The idea of this document is to be used for changing reports. Implying it will give an overview of the structure of all the reports and calculations. The word document holds all monthly reports in the structure shown in appendix B, all the calculations, manual registration, update values, measurement of a month, and yearly reports.

⁶ Source Tag Type is some letter that stands for add functionality of the datatype in the UPHD, eks. I4 = Interger datatype, L8 = Long Interger datatype, F4 floating point data type, etc.

⁷ Datatype is what type of variable the tag is in UPHD, eks. float, integer, char16, datetime, etc.

13 Enhancement

The prototype still has some incompleteness that will be needed to complete before moving on to enhancing it.

13.1 Database

The convenient enhancement is to remove the parameter classes and add them into a database. This improvement will only make it easier for someone to alter the data without republishing the application. Figure 13.1 gives an idea of implementing the information into a database. The enhancing part would be creating a page for altering the reports in the UI, giving authorized personnel the possibility to alter and create new reports. In addition, it will be needed to increase the number of calculations to give more options to be used.

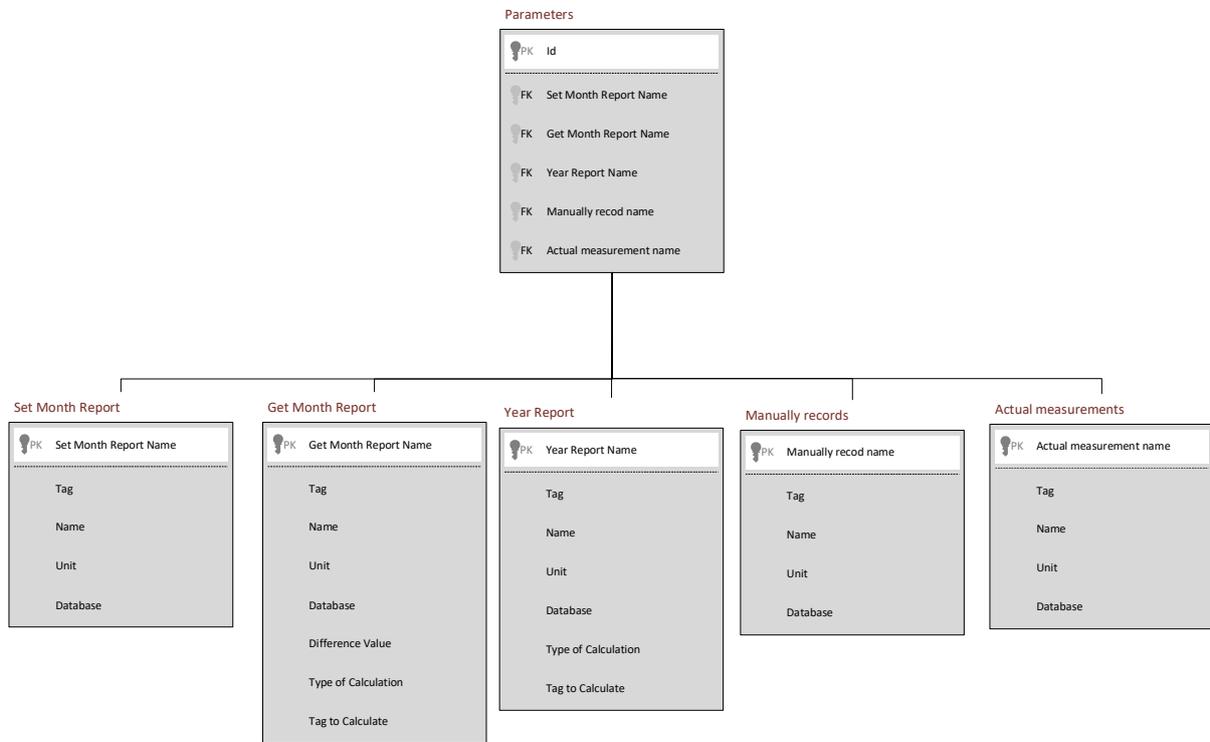


Figure 13.1 an idea of implementing the information into a database.

13.2 Correction calculation

The correction calculation is based on input from a user. The problem occurs when a sensor is out for calibration. At the same time, the flow of the product is still being transferred to the

13 Enhancement

customer. Currently, it is done a manual estimation of the amount of product that was transferred. The estimation is based on the flow of the product before and after the calibration. Furthermore, compared to the other measurement on the same flow.

The idea is to implement the estimation calculation into the application. The reason for implementing it in the application is that the estimated value is a correction value for daily measurement of a report. In addition, it would reduce the time of doing the calculation.

13.3 Automated verification

The automated verification lets the application verify by itself and notify the authorized personnel in charge when an abnormal event occurs. The abnormal events and what is reconned as approved must be specified.

A criterion can be to evaluate the previous values if the new value is reasonable. If the report contains a difference calculation, it is possible to evaluate the result if the new values can be reasonable.

The process of determining if a daily measurement is viable can be set to a specific time. The compelling time would be around after eight and before ten in the morning. Hence, the report needs to be verified before eleven.

14 Conclusion

The selected programming languages for the prototype are sustainable for the present time. It is not possible to say how long into the future they will endure. However, all the been selected programming languages with frameworks are widely used and popular amongst developers and backed by large Corporations. Hence, the prototype will have the endurance to chosen programming languages.

The prototype is two applications cooperating for presenting the metering reports and handle user input. Either of the application can be swapped out and integrated with another application. They are resulting in simplifying the process of replacing one of them if needed.

The development process has shown the complexities of developing a new application integrated with two existing UPHDs and adjusting the results to another application with another design.

The result of the development process is a flexible design. The flexibility is arranged in the different operations by having one process for performing desired result dependent upon the information. Resulting in the maintenance and modifications are a more straightforward procedure.

Enabling the users to test the prototype as it has been developed has proven useful for both parties. The testing with the users has yielded a new function of displaying the stored data in UPHDs. The new function enables the users to see if the correct value is displayed in the metering report or if it has been stored different values.

The prototype is a custom-made solution that can replace the current system when it is completed. In addition, the prototype has the potential of being a better system in terms of functionality than the current system.

References

- [1] Microsoft, "A break from the past, part 2: Saying goodbye to ActiveX, VBScript, attachEvent...", Microsoft Corporation, [Online]. Available: <https://blogs.windows.com/msedgedev/2015/05/06/a-break-from-the-past-part-2-saying-goodbye-to-activex-vbscript-attachevent/>. [Accessed 11 11 2020].
- [2] Microsoft Corporation, "Out-of-date ActiveX control blocking," [Online]. Available: <https://docs.microsoft.com/en-us/internet-explorer/ie11-deploy-guide/out-of-date-activex-control-blocking>. [Accessed 11 11 2020].
- [3] Microsoft, "Using VBScript," Microsoft Corporation, 31 05 2018. [Online]. Available: <https://docs.microsoft.com/en-us/windows/win32/lwef/using-vbscript>. [Accessed 10 02 2021].
- [4] Microsoft, "ASP Overview," Microsoft Corporation, 16 06 2017. [Online]. Available: [https://docs.microsoft.com/en-us/previous-versions/iis/6.0-sdk/ms524929\(v=vs.90\)](https://docs.microsoft.com/en-us/previous-versions/iis/6.0-sdk/ms524929(v=vs.90)). [Accessed 20 04 2021].
- [5] WHATWG, W3C, "HTML Living Standard," WHATWG, W3C, 20 04 2021. [Online]. Available: <https://html.spec.whatwg.org/>. [Accessed 21 04 2021].
- [6] Microsoft, "JScript (ECMAScript3)," Microsoft Corporation, 24 11 2011. [Online]. Available: [https://docs.microsoft.com/en-us/previous-versions//hbxc2t98\(v=vs.85\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions//hbxc2t98(v=vs.85)?redirectedfrom=MSDN). [Accessed 20 04 2021].
- [7] Microsoft, "What Is VBScript?," Microsoft Corporation, 19 04 2011. [Online]. Available: [https://docs.microsoft.com/en-us/previous-versions//1kw29xwf\(v=vs.85\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions//1kw29xwf(v=vs.85)?redirectedfrom=MSDN). [Accessed 20 04 2021].
- [8] MDN Web Docs (previously known as MDN — the Mozilla Developer Network), "CSS: Cascading Style Sheets," MDN Web Docs, 14 04 2021. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/CSS>. [Accessed 21 04 2021].
- [9] S. M. S. Kabir, Basic Guidelines for Research: An Introductory Approach for All Disciplines, Chittagong-4203, Bangladesh: Book Zone, 2016, pp. 202-204.
- [10] N.-O. Skeie, *Software Engineering Object-oriented Analysis, Design, and Programming using UML and C#*, Porsgrunn: USN, 2018.
- [11] P. Christensson, "Frontend," Sharpened Productions, [Online]. Available: <https://techterms.com/definition/frontend>. [Accessed 10 05 2021].
- [12] P. Christensson, "Backend," Sharpened Productions, [Online]. Available: <https://techterms.com/definition/backend>. [Accessed 10 05 2021].

References

- [13] P. Christensson, "Website," Sharpened Productions , [Online]. Available: <https://techterms.com/definition/website>. [Accessed 10 05 2021].
- [14] P. Christensson, "Web Page," Sharpened Productions , [Online]. Available: <https://techterms.com/definition/webpage>. [Accessed 10 05 2021].
- [15] M. Uchiha, "What are the differences between server-side and client-side programming?," Stack Exchange Inc, 07 08 2018. [Online]. Available: <https://softwareengineering.stackexchange.com/questions/171203/what-are-the-differences-between-server-side-and-client-side-programming>. [Accessed 10 05 2021].
- [16] Computer Hope, "Programming language," Computer Hope, 13 03 2021. [Online]. Available: <https://www.computerhope.com/jargon/p/programming-language.htm>. [Accessed 10 05 2021].
- [17] Eleven Fifty Academy, "What is a Framework in Programming Language?," Eleven Fifty Academy, 22 10 2020. [Online]. Available: <https://elevenfifty.org/blog/what-is-a-framework-in-programming-language/>. [Accessed 10 05 2021].
- [18] A. DEVERO, "Programming languages, libraries and frameworks," ALEX DEVERO BLOG, [Online]. Available: <https://blog.alexdevero.com/programming-languages-libraries-and-frameworks/>. [Accessed 10 05 2021].
- [19] Awio Web Services LLC, "Web Browser Usage Trends," Awio Web Services LLC, 01 04 2021. [Online]. Available: <https://www.w3counter.com/trends>. [Accessed 23 04 2021].
- [20] Paessler AG, "IT Explained: Server," Paessler AG, [Online]. Available: <https://www.paessler.com/it-explained/server>. [Accessed 22 04 2021].
- [21] What Is My IP Address, "What is a Web Server?," What Is My IP Address, [Online]. Available: <https://whatismyipaddress.com/web-server>. [Accessed 22 04 2021].
- [22] Netcraft Ltd, "March 2019 Web Server Survey," Netcraft Ltd 28th February, 2019, 2019 02 2019. [Online]. Available: <https://news.netcraft.com/archives/2019/02/28/february-2019-web-server-survey.html>. [Accessed 25 02 2021].
- [23] UpGuard Team, "IIS vs Apache: Which is the Best Web Server?," UpGuard, Inc., 15 02 2021. [Online]. Available: <https://www.upguard.com/blog/iis-apache>. [Accessed 22 04 2021].
- [24] TIOBE Software BV, "TIOBE Index for March 2021," TIOBE Software BV, 10 03 2021. [Online]. Available: <https://www.tiobe.com/tiobe-index/>. [Accessed 10 03 2021].

References

- [25] Stack Exchange Inc, "stackoverflow," Stack Exchange Inc, 21 04 2021. [Online]. Available: <https://stackoverflow.com/>. [Accessed 21 04 2021].
- [26] GitHub Inc., "GitHub," GitHub Inc., 21 04 2021. [Online]. Available: <https://github.com/>. [Accessed 21 04 2021].
- [27] S. O'Grady, "The RedMonk Programming Language Rankings: June 2020," RedMonk, 27 07 2020. [Online]. Available: <https://redmonk.com/sogradey/2020/07/27/language-rankings-6-20/>. [Accessed 21 04 2021].
- [28] Stack Exchange Inc, "2020 Developer Survey," Stack Exchange Inc, 28 02 2020. [Online]. Available: <https://insights.stackoverflow.com/survey/2020#technology-how-technologies-are-connected>. [Accessed 21 04 2021].
- [29] L. K. Cox, "Web Design 101: How HTML, CSS, and JavaScript Work," HubSpot, Inc., 26 10 2020. [Online]. Available: <https://blog.hubspot.com/marketing/web-design-html-css-javascript#:~:text=HTML%20and%20CSS%20are%20actually,every%20web%20page%20and%20application>. [Accessed 10 05 2021].
- [30] Q-Success, "Usage statistics of JavaScript as client-side programming language on websites," Q-Success, [Online]. Available: <https://w3techs.com/technologies/details/cp-javascript>. [Accessed 10 05 2021].
- [31] S. Varaksina, "Single-Page Applications vs Multi-Page Applications: The Battle of the Web Apps," Mind Studios, 30 09 2020. [Online]. Available: <https://themindstudios.com/blog/spa-vs-mpa/>. [Accessed 22 04 2021].
- [32] S. Valuy, "A Comparison of Single-Page and Multi-Page Applications," 17 06 2020. [Online]. Available: <https://dzone.com/articles/the-comparison-of-single-page-and-multi-page-appli>. [Accessed 22 04 2020].
- [33] P. Skólski, "Single-page application vs. multiple-page application," Neoteric Sp. z o.o., 01 12 2016. [Online]. Available: https://neoteric.eu/blog/single-page-application-vs-multiple-page-application/?utm_source=medium.com&utm_medium=social&utm_content=neo&utm_campaign=blog. [Accessed 22 04 2021].
- [34] J. M. J. W. D. W. A. W. R. A. J. A. S. J. K. I. E. Y. k. KirstenS, "Cross Site Scripting (XSS)," OWASP Foundation, Inc., [Online]. Available: <https://owasp.org/www-community/attacks/xss/>. [Accessed 21 04 2021].
- [35] E. Czaplicki, "A delightful language for reliable web applications.," [Online]. Available: <https://elm-lang.org/>. [Accessed 05 16 2021].
- [36] AltexSoft, "The Good and the Bad of Vue.js Framework Programming," AltexSoft, 11 09 2019. [Online]. Available:

References

- <https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/>. [Accessed 16 05 2021].
- [37] E. Bojanowska, "Pros and Cons of the Vue.js Framework," *Naturally Sp. z o.o.*, 30 11 2018. [Online]. Available: <https://naturally.com/blog/pros-cons-vue-js>. [Accessed 16 05 2021].
- [38] DDI DEVELOPMENT, "Pros and Cons of Vue.js Framework Programming," DDI DEVELOPMENT, 12 2020. [Online]. Available: <https://ddi-dev.com/blog/programming/the-good-and-the-bad-of-vue-js-framework-programming/>. [Accessed 16 05 2021].
- [39] R. Patel, "Pros and Cons of the Vue.js Framework," *Medium*, 17 09 2019. [Online]. Available: <https://ronakataglowid.medium.com/pros-and-cons-of-the-vue-js-framework-8015dcbc05ef>. [Accessed 16 05 2021].
- [40] E. Czaplicki, "Elm: Concurrent FRP for Functional GUIs," 03 30 2012. [Online]. Available: <https://elm-lang.org/assets/papers/concurrent-frp.pdf>. [Accessed 16 05 2021].
- [41] J. Warden, "React Redux Thunk vs Elm," *WordPress*, 5 10 2019. [Online]. Available: <https://jessewarden.com/2019/10/react-redux-thunk-vs-elm.html>. [Accessed 16 05 2021].
- [42] T. Assus, "10 reasons why you should give Elm a try," *Medium*, 22 08 2016. [Online]. Available: <https://medium.com/elmlightments/10-reasons-why-you-should-give-elm-a-try-62b56d305643>. [Accessed 16 05 2021].
- [43] C. Gregori, "Elm and why it's not quite ready yet," *Medium*, 29 05 2019. [Online]. Available: <https://blog.bitsrc.io/elm-and-why-its-not-quite-ready-yet-2c516a81e252>. [Accessed 16 05 2021].
- [44] Codementor, "Study of Programming Languages Not to Learn in 2019," Codementor, 16 04 2019. [Online]. Available: <https://www.codementor.io/blog/worst-languages-2019-6mvbfg3w9x#:~:text=As%20one%20of%20the%20languages,by%20Objective%2DC%20and%20CoffeeScript..> [Accessed 16 05 2021].
- [45] N. Pandit, "What And Why React.js," *C# Corner*, 10 02 2021. [Online]. Available: <https://www.c-sharpcorner.com/article/what-and-why-reactjs/>. [Accessed 22 04 2021].
- [46] T. GRABSKI, "React JS Pros and Cons in 2020," *Pagepro*, 10 08 2020. [Online]. Available: <https://pagepro.co/blog/react-js-pros-and-cons-in-2020/>. [Accessed 22 04 2021].
- [47] D. Kumar, "What are the Pros and Cons of React," *KnowledgeHut*, 24 03 2021. [Online]. Available: <https://www.knowledgehut.com/blog/web-development/pros-and-cons-of-react>. [Accessed 22 04 2021].

References

- [48] A. Insignares, "React Pros and Cons: What are the Advantages and Disadvantages of ReactJS?," Koombea, Inc., 10 03 2021. [Online]. Available: <https://www.koombea.com/blog/react-pros-and-cons-what-are-the-advantages-and-disadvantages-of-reactjs/>. [Accessed 22 04 2021].
- [49] Learneroo, "What Programming Language Should You Learn?," Learneroo, [Online]. Available: <https://www.learneroo.com/modules/9/nodes/620>. [Accessed 16 05 2021].
- [50] T. Fairy, "Java VS Python VS C# detailed comparison, which language to learn first?," Tech Fairy, [Online]. Available: <https://tech-fairy.com/java-vs-python-vs-c-detailed-comparison-which-language-to-learn-first/>. [Accessed 16 05 2021].
- [51] A. Goel, "Best Programming Languages to Learn in 2021 (for Job & Future)," hackr.io, 11 05 2021. [Online]. Available: <https://hackr.io/blog/best-programming-languages-to-learn-2021-jobs-future>. [Accessed 16 05 2021].
- [52] M. Chand, ".NET Core vs .NET," C# Corner, 05 07 2020. [Online]. Available: <https://www.c-sharpcorner.com/article/difference-between-net-framework-and-net-core/>. [Accessed 16 05 2021].
- [53] Codecademy, "What Is an IDE?," Codecademy, [Online]. Available: <https://www.codecademy.com/articles/what-is-an-ide>. [Accessed 23 04 2021].
- [54] Microsoft Corporation, "Visual Studio 2019," Microsoft Corporation, [Online]. Available: <https://visualstudio.microsoft.com/vs/>. [Accessed 23 04 2021].
- [55] Microsoft Corporation, "Redefined.," Microsoft Corporation, [Online]. Available: <https://code.visualstudio.com/>. [Accessed 23 04 2021].
- [56] Postman, Inc., "The Collaboration Platform for API Development," Postman, Inc., [Online]. Available: <https://www.postman.com/>. [Accessed 23 04 2021].
- [57] JetBrains s.r.o., "Free .NET Decompiler and Assembly Browser," JetBrains s.r.o., [Online]. Available: <https://www.jetbrains.com/decompiler/>. [Accessed 23 04 2021].
- [58] P. M. Shubham Yadav, "ReactJS | Lifecycle of Components," geeksforgeeks, 22 01 2021. [Online]. Available: <https://www.geeksforgeeks.org/reactjs-lifecycle-components/>. [Accessed 02 05 2021].
- [59] Facebook Inc, "React Versions," Facebook Inc, [Online]. Available: <https://reactjs.org/versions/>. [Accessed 03 05 2021].
- [60] Microsoft Corporation, ".NET Support Policy," Microsoft Corporation, [Online]. Available: <https://dotnet.microsoft.com/platform/support/policy>. [Accessed 03 05 2021].

References

- [61] Jamshed, ".NET Core 2.1 will reach End of Support on August 21, 2021," Microsoft Corporation, 04 03 2021. [Online]. Available: <https://devblogs.microsoft.com/dotnet/net-core-2-1-will-reach-end-of-support-on-august-21-2021/#:~:text=The%20%2Dyear%20lifecycle%20for,version%20will%20continue%20to%20run.&text=NET%20Core%202.1%20starting%20September,issue%20these%20security%20updates%20.> [Accessed 03 05 2021].
- [62] Microsoft Corporation, "Hva er retningslinjene for livssyklus på ulike versjoner av .NET Framework?," Microsoft Corporation, 09 07 2020. [Online]. Available: <https://docs.microsoft.com/nb-no/lifecycle/faq/dotnet-framework>. [Accessed 03 05 2021].
- [63] Netcraft Ltd, "March 2021 Web Server Survey," Netcraft Ltd, 29 03 2021. [Online]. Available: <https://news.netcraft.com/archives/2021/03/29/march-2021-web-server-survey.html>. [Accessed 12 05 2021].
- [64] Encyclopædia Britannica, Inc, "Encyclopædia Britannica," 09 02 2021. [Online]. Available: <https://www.britannica.com/technology/computer>.
- [65] C. Ostlund, "A GUIDE TO THE IT LIFECYCLE: WHAT IT IS & WHAT YOU NEED TO KNOW," MARCO TECHNOLOGIES, LLC, 27 08 2019. [Online]. Available: <https://www.marconet.com/blog/a-guide-to-the-it-lifecycle-what-it-is-what-you-need-to-know#:~:text=What%20is%20the%20IT%20Life,technology%20enters%20its%20usable%20stage..> [Accessed 2021 09 02].
- [66] S. Hardmeier, "The History of Internet Explorer," Microsoft Corporation, 25 08 2005. [Online]. Available: <https://web.archive.org/web/20051001113951/http://www.microsoft.com/windows/ie/community/columns/historyofie.mspix>. [Accessed 10 02 2021].
- [67] MDN Web Docs (previously known as MDN — the Mozilla Developer Network), "developer.mozilla.org," MDN Web Docs, 18 12 2020. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Microsoft_Internet_Explorer. [Accessed 19 04 2021].
- [68] I. Codesido, "What is front-end development?," Guardian News & Media Limited, 28 09 2009. [Online]. Available: <https://www.theguardian.com/help/insideguardian/2009/sep/28/blogpost>. [Accessed 21 04 2021].
- [69] U. ROUZ, "IS THERE A VIABLE ALTERNATIVE TO JAVASCRIPT?," webdesignerdepot, 21 11 2019. [Online]. Available: <https://www.webdesignerdepot.com/2019/11/is-there-a-viable-alternative-to-javascript/>. [Accessed 22 04 2021].
- [70] D. R. C. X. Han Liang, "What is a DLL," Microsoft Corporation, 22 09 2020. [Online]. Available: <https://docs.microsoft.com/en-us/troubleshoot/windows->

References

- client/deployment/dynamic-link-library#the-net-framework-assembly. [Accessed 26 04 2021].
- [71] H. L. R. C. Simonx Xu, "What is a DLL," Microsoft Corporation, 22 09 2020. [Online]. Available: <https://docs.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library#the-net-framework-assembly>. [Accessed 26 04 2021].
- [72] Microsoft Corporation, "Microsoft .NET Framework," Microsoft Corporation, [Online]. Available: <https://docs.microsoft.com/nb-no/lifecycle/products/microsoft-net-framework>. [Accessed 03 05 2021].

Appendices

Appendix A, examples of calculations in formula and code

$$Sum = a + b + c + d \tag{B.1}$$

```
1 reference
public static List<double> sumFourList(
    List<double> _list1, List<double> _list2,
    List<double> _list3, List<double> _list4)
{
    List<double> sumedList = new List<double>();
    double sum;

    for (int i = 0; i < _list1.Count; i++)
    {
        sum = 0;
        sum = _list1[i] + _list2[i] +
            _list3[i] + _list4[i];
        sumedList.Add(Math.Round(sum, 3));
    }

    return sumedList;
}
```

$$\begin{aligned}
 & a_{Balance\ Tonn} \\
 &= \left(a_{raw\ Tonn} + \frac{a_{raw\ Tonn} * a_{corr\ m^3}}{a_{raw\ m^3}} \right) \\
 &+ \frac{\left(a + \frac{a_{raw\ Tonn} * a_{corr\ m^3}}{a_{raw\ m^3}} \right) * \left(b + \frac{b_{raw\ Tonn} * b_{corr\ m^3}}{b_{raw\ m^3}} + c + \frac{c_{raw\ Tonn} * c_{corr\ m^3}}{c_{raw\ m^3}} \right)}{\left(a + \frac{a_{raw\ Tonn} * a_{corr\ m^3}}{a_{raw\ m^3}} + d + \frac{d_{raw\ Tonn} * d_{corr\ m^3}}{d_{raw\ m^3}} + e + \frac{e_{raw\ Tonn} * e_{corr\ m^3}}{e_{raw\ m^3}} \right)}
 \end{aligned} \tag{B.2}$$

```
public static List<double>
1 reference
ratiosTonnCalcCoolingWater(
    List<double> targetRawTonn,
    List<double> corrTargetM3,
    List<double> rawTonnDenominator1,
    List<double> corrTonnDenominator1,
    List<double> raw3Denominator1,
    List<double> corr3Denominator1,
    List<double> rawTonnDenominator2,
    List<double> corrTonnDenominator2,
    List<double> raw3Denominator2,
    List<double> corr3Denominator2,
    List<double> rawTonnRatio1,
    List<double> corr3Ratio1,
    List<double> corr3Ratio1,
    List<double> rawTonnRatio2,
    List<double> raw3Ratio2,
    List<double> corr3Ratio2
)
{
    //converting to tonn
    corrTargetTonn = 0;
    if (rawTargetM3[i] != 0)
    {
        corrTargetTonn =
            (targetRawTonn[i] * corrTargetM3[i]) / rawTargetM3[i];
    }
    corrTonnDenominator1 = 0;
    if (raw3Denominator1[i] != 0)
    {
        corrTonnDenominator1 =
            (rawTonnDenominator1[i] * corr3Denominator1[i]) / raw3Denominator1[i];
    }
    corrTonnDenominator2 = 0;
    if (raw3Denominator2[i] != 0)
    {
        corrTonnDenominator2 =
            (rawTonnDenominator2[i] * corr3Denominator2[i]) / raw3Denominator2[i];
    }
    corrTonnRatio1 = 0;
    if (raw3Ratio1[i] != 0)
    {
        corrTonnRatio1 =
            (rawTonnRatio1[i] * corr3Ratio1[i]) / raw3Ratio1[i];
    }
    corrTonnRatio2 = 0;
    if (raw3Ratio2[i] != 0)
    {
        corrTonnRatio2 =
            (rawTonnRatio2[i] * corr3Ratio2[i]) / raw3Ratio2[i];
    }
    //massbalance calculation
    sum = 0;
    part = 0;
    denominator = 0;
    denominator = targetRawTonn[i] + corrTargetTonn + rawTonnDenominator1[i]
        + corrTonnDenominator1 + rawTonnDenominator2[i] + corrTonnDenominator2;
    if (denominator != 0)
    {
        part = ((targetRawTonn[i] + corrTargetTonn) * (rawTonnRatio1[i] +
            corrTonnRatio1 + rawTonnRatio2[i] + corrTonnRatio2));
        sum = (targetRawTonn[i] + corrTargetTonn) + (part / denominator);
    }
    calculated.Add(Math.Round(sum, 3));
}
```

Appendix B, change of a Month Reports

Month report, Process Fuel Gas Ineos Bamble pre April 2021

Tag to get data from PHD:

Index	PHD tag	PHD
0	E-FIQ8407.GVOL_PDAY_TOT	1
1	E-FIQ8407.MASS_PDAY_TOT	1
2	E-FIQ8407.ENER_PDAY_TOT	1
3	EP-FIQ8407.MASS_PDAY_KORR	1
4	EP-FIQ8407.ENER_PDAY_KORR	1
5	E-FIQ8411.GVOL_PDAY_TOT	1
6	E-FIQ8411.MASS_PDAY_TOT	1
7	E-FIQ8411.ENER_PDAY_TOT	1
8	EP-FIQ8411.MASS_PDAY_KORR	1
9	EP-FIQ8411.ENER_PDAY_KORR	1
10	EP-BRENNGASS_POLY_VER	1
11	EP-BRENNGASS_POLY_TEKST	3

Setup of the report in terms of information:

Index	Name	Unit	Calculation	Tag for calculation
0	,	dato,		
1	FIQ8407 Råverdier	m ³	1	0
2	FIQ8407 Råverdier	tonn	1	1
3	FIQ8407 Råverdier	GJ	1	2
4	FIQ8407 Korreksjon	tonn	1	3
5	FIQ8407 Korreksjon	GJ	1	4
6	FIQ8407 Verifisert	tonn	2	1,3
7	FIQ8407 Verifisert	GJ	2	2,4
8	FIQ8411 Råverdier	m ³	1	5
9	FIQ8411 Råverdier	tonn	1	6
10	FIQ8411 Råverdier	GJ	1	7
11	FIQ8411 Korreksjon	tonn	1	8
12	FIQ8411 Korreksjon	GJ	1	9
13	FIQ8411 Verifisert	tonn	2	6,8
14	FIQ8411 Verifisert	GJ	2	7,9
15	FIQ8407-FIQ8411 Relativt avvik (1%)	vol %	4	0,5
16	FIQ8407-FIQ8411 Relativt avvik (1%)	tonn	3	1,6
17	FIQ8407-FIQ8411 Relativt avvik (1%)	masse %	4	1,6
18	FIQ8407-FIQ8411 Relativt avvik (1%)	GJ	3	2,7
19	Sluttverdi	tonn	7	2,4,7,9
20	Sluttverdi	GJ	7	1,3,6,8
21	Kommentar	Tekst		

Presentations of the report in the application:

Råverdier for Brenngass Ineos Bamble 2021

dato	FIQ8407 Råverdier		FIQ8407 Råverdier		FIQ8407 Korreksjon		FIQ8407 Verifisert		FIQ8411 Råverdier		FIQ8411 Korreksjon		FIQ8411 Verifisert		FIQ8407 Relativt avvik (%)		FIQ8411 Relativt avvik (%)		FIQ8407 Relativt avvik (%)		FIQ8411 Relativt avvik (%)		Sluttverdi		Sluttverdi		Kommentar				
	m ³	tonn	GJ	tonn	GJ	tonn	GJ	m ³	tonn	GJ	tonn	GJ	tonn	GJ	vol %	tonn	masse %	GJ	tonn	GJ	tonn	GJ	tonn	GJ	tonn	GJ	tonn	GJ	Tekst		
01.03.2021	7887.656	9.09	583.585	0	0	9.09	583.585	7884.126	9.352	600.398	0	0	9.352	600.398	0.04	-0.262	-2.88	-16.813	583.555	9.09	583.555	9.09	583.555	9.09	583.555	9.09	583.555	9.09	583.555	9.09	583.555
02.03.2021	7974.809	9.143	586.963	0	0	9.143	586.963	7975.019	9.419	604.713	0	0	9.419	604.713	0	-0.276	-3.02	-17.75	586.963	9.143	586.963	9.143	586.963	9.143	586.963	9.143	586.963	9.143	586.963	9.143	586.963
03.03.2021	8080.554	9.484	608.837	0	0	9.484	608.837	8079.056	9.769	627.132	0	0	9.769	627.132	0.02	-0.283	-3.01	-18.295	608.837	9.484	608.837	9.484	608.837	9.484	608.837	9.484	608.837	9.484	608.837	9.484	608.837
04.03.2021	8303.731	9.938	638.01	0	0	9.938	638.01	8307.196	10.217	655.943	0	0	10.217	655.943	-0.02	-0.279	-2.81	-17.933	638.01	9.938	638.01	9.938	638.01	9.938	638.01	9.938	638.01	9.938	638.01	9.938	638.01
05.03.2021	8064.005	9.932	637.617	0	0	9.932	637.617	8055.612	10.207	655.25	0	0	10.207	655.25	0.1	-0.275	-2.77	-17.653	637.617	9.932	637.617	9.932	637.617	9.932	637.617	9.932	637.617	9.932	637.617	9.932	637.617
06.03.2021	7765.006	9.729	624.582	0	0	9.729	624.582	7753.618	9.989	641.247	0	0	9.989	641.247	0.19	-0.28	-2.67	-16.665	624.582	9.729	624.582	9.729	624.582	9.729	624.582	9.729	624.582	9.729	624.582	9.729	624.582
07.03.2021	8001.245	9.887	634.757	0	0	9.887	634.757	7995.585	10.156	652.019	0	0	10.156	652.019	0.07	-0.269	-2.72	-17.262	634.757	9.887	634.757	9.887	634.757	9.887	634.757	9.887	634.757	9.887	634.757	9.887	634.757
08.03.2021	8039.555	10.146	651.388	0	0	10.146	651.388	8026.736	10.429	669.525	0	0	10.429	669.525	0.16	-0.283	-2.79	-18.137	651.388	10.146	651.388	10.146	651.388	10.146	651.388	10.146	651.388	10.146	651.388	10.146	651.388
09.03.2021	7860.699	9.857	632.823	0	0	9.857	632.823	7850.688	10.118	649.573	0	0	10.118	649.573	0.13	-0.261	-2.65	-16.75	632.823	9.857	632.823	9.857	632.823	9.857	632.823	9.857	632.823	9.857	632.823	9.857	632.823
10.03.2021	8306.854	10.146	651.34	0	0	10.146	651.34	8293.017	10.433	669.805	0	0	10.433	669.805	0.17	-0.287	-2.83	-18.465	651.34	10.146	651.34	10.146	651.34	10.146	651.34	10.146	651.34	10.146	651.34	10.146	651.34
11.03.2021	8641.298	10.857	696.978	0	0	10.857	696.978	8622.217	11.16	716.454	0	0	11.16	716.454	0.22	-0.303	-2.79	-19.476	696.978	10.857	696.978	10.857	696.978	10.857	696.978	10.857	696.978	10.857	696.978	10.857	696.978
12.03.2021	8190.496	10.316	662.301	0	0	10.316	662.301	8169.335	10.591	679.901	0	0	10.591	679.901	0.26	-0.275	-2.67	-17.6	662.301	10.316	662.301	10.316	662.301	10.316	662.301	10.316	662.301	10.316	662.301	10.316	662.301
13.03.2021	8751.705	10.474	672.404	0	0	10.474	672.404	8733.898	10.768	691.267	0	0	10.768	691.267	0.2	-0.294	-2.81	-18.863	672.404	10.474	672.404	10.474	672.404	10.474	672.404	10.474	672.404	10.474	672.404	10.474	672.404
14.03.2021	8478.294	10.094	648.049	0	0	10.094	648.049	8462.752	10.379	666.331	0	0	10.379	666.331	0.18	-0.285	-2.82	-18.282	648.049	10.094	648.049	10.094	648.049	10.094	648.049	10.094	648.049	10.094	648.049	10.094	648.049
15.03.2021	8168.364	9.682	621.561	0	0	9.682	621.561	8166.614	9.943	638.318	0	0	9.943	638.318	0.02	-0.261	-2.7	-16.757	621.561	9.682	621.561	9.682	621.561	9.682	621.561	9.682	621.561	9.682	621.561	9.682	621.561
16.03.2021	8061.665	9.547	612.934	0	0	9.547	612.934	8054.268	9.809	629.753	0	0	9.809	629.753	0.09	-0.262	-2.74	-16.819	612.934	9.547	612.934	9.547	612.934	9.547	612.934	9.547	612.934	9.547	612.934	9.547	612.934
17.03.2021	8212.128	9.846	632.114	0	0	9.846	632.114	8203.286	10.115	649.361	0	0	10.115	649.361	0.11	-0.269	-2.73	-17.247	632.114	9.846	632.114	9.846	632.114	9.846	632.114	9.846	632.114	9.846	632.114	9.846	632.114
18.03.2021	8224.261	9.977	640.529	0	0	9.977	640.529	8213.541	10.244	657.634	0	0	10.244	657.634	0.13	-0.267	-2.68	-17.105	640.529	9.977	640.529	9.977	640.529	9.977	640.529	9.977	640.529	9.977	640.529	9.977	640.529
19.03.2021	8356.667	10.069	646.424	0	0	10.069	646.424	8345.447	10.348	664.315	0	0	10.348	664.315	0.13	-0.279	-2.77	-17.891	646.424	10.069	646.424	10.069	646.424	10.069	646.424	10.069	646.424	10.069	646.424	10.069	646.424
20.03.2021	8299.016	9.423	604.933	0	0	9.423	604.933	8292.797	9.698	622.625	0	0	9.698	622.625	0.07	-0.275	-2.92	-17.692	604.933	9.423	604.933	9.423	604.933	9.423	604.933	9.423	604.933	9.423	604.933	9.423	604.933
21.03.2021	8798.51	10.104	648.663	0	0	10.104	648.663	8793.832	10.395	667.314	0	0	10.395	667.314	0.05	-0.291	-2.88	-18.651	648.663	10.104	648.663	10.104	648.663	10.104	648.663	10.104	648.663	10.104	648.663	10.104	648.663
22.03.2021	8090.156	9.239	593.149	0	0	9.239	593.149	8091.050	9.501	609.966	0	0	9.501	609.966	-0.01	-0.262	-2.84	-16.817	593.149	9.239	593.149	9.239	593.149	9.239	593.149	9.239	593.149	9.239	593.149	9.239	593.149
23.03.2021	8410.665	9.487	609.067	0	0	9.487	609.067	8416.488	9.769	627.169	0	0	9.769	627.169	-0.07	-0.282	-2.97	-18.102	609.067	9.487	609.067	9.487	609.067	9.487	609.067	9.487	609.067	9.487	609.067	9.487	609.067
24.03.2021	8679.292	9.887	634.718	0	0	9.887	634.718	8686.944	10.184	653.77	0	0	10.184	653.77	-0.09	-0.297	-3	-19.052	634.718	9.887	634.718	9.887	634.718	9.887	634.718	9.887	634.718	9.887	634.718	9.887	634.718
25.03.2021	7922.664	9.562	613.84	0	0	9.562	613.84	7915.795	9.824	630.716	0	0	9.824	630.716	0.09	-0.262	-2.74	-16.876	613.84	9.562	613.84	9.562	613.84	9.562	613.84	9.562	613.84	9.562	613.84	9.562	613.84
26.03.2021	7976.772	9.675	621.103	0	0	9.675	621.103	7964.024	9.94	638.144	0	0	9.94	638.144	0.16	-0.265	-2.74	-17.041	621.103	9.675	621.103	9.675	621.103	9.675	621.103	9.675	621.103	9.675	621.103	9.675	621.103
27.03.2021	7603.671	9.306	597.446	0	0	9.306	597.446	7594.815	9.557	613.569	0	0	9.557	613.569	0.12	-0.251	-2.7	-16.123	597.446	9.306	597.446	9.306	597.446	9.306	597.446	9.306	597.446	9.306	597.446	9.306	597.446
28.03.2021	7194.646	9.067	582.096	0	0	9.067	582.096	7177.317	9.307	597.481	0	0	9.307	597.481	0.24	-0.24	-2.65	-15.385	582.096	9.067	582.096	9.067	582.096	9.067	582.096	9.067	582.096	9.067	582.096	9.067	582.096
29.03.2021	7397.866	9.07	582.279	0	0	9.07	582.279	7398.303	9.31	597.691	0	0	9.31	597.691	-0.01	-0.24	-2.65	-15.412	582.279	9.07	582.279	9.07	582.279	9.07	582.279	9.07	582.279	9.07	582.279	9.07	582.279
30.03.2021	7465.115	8.814	565.848	0	0	8.814	565.848	7478.629	9.052	581.097	0	0	9.052	581.097	-0.18	-0.238	-2.7	-15.249	565.848	8.814	565.848	8.814	565.848	8.814	565.848	8.814	565.848	8.814	565.848	8.814	565.848
31.03.2021	7730.178	9.072	582.431	0	0	9.072	582.431	7734.407	9.332	599.103	0	0	9.332	599.103	-0.05	-0.26	-2.87	-16.672	582.431	9.072	582.431	9.072	582.431	9.072	582.431	9.072	582.431	9.072	582.431	9.072	582.431
Sum:	250942.543	300.92	19318.769	0	0	300.92	19318.769	250736.42	309.315	19857.584	0	0	309.315	19857.584	0.08	-0.271	-2.79	-17.381	19318.769	300.92	19318.769	300.92	19318.769	300.92	19318.769	300.92	19318.769	300.92	19318.769	300.92	19318.769

Month report, Process Fuel Gas Ineos Bamble after April 2021

Tag to get data from PHD:

Index	PHD tag	PHD
0	E-FIQ8407.MASS_PDAY_TOT	1
1	E-FIQ8407.ENER_PDAY_TOT	1
2	EP-FIQ8407.MASS_PDAY_KORR	1
3	EP-FIQ8407.ENER_PDAY_KORR	1
4	E-FIQ8411.MASS_PDAY_TOT	1
5	E-FIQ8411.ENER_PDAY_TOT	1
6	EP-FIQ8411.MASS_PDAY_KORR	1
7	EP-FIQ8411.ENER_PDAY_KORR	1
8	EP-BRENNGASS_POLY_VER	1
9	EP-BRENNGASS_POLY_TEKST	3

Setup of the report in terms of information:

Appendices

Index	Name	Unit	Calculation	Tag for calculation
0		dato		
1	FIQ8407 Råverdi	tonn	1	0
2	FIQ8407 Råverdi	GJ	1	1
3	FIQ8407 Korreksjon	tonn	1	2
4	FIQ8407 Korreksjon	GJ	1	3
5	FIQ8407 Verifisert	tonn	2	0,2,
6	FIQ8407 Verifisert	GJ	2	1,3,
7	FIQ8411 Råverdi	tonn	1	4
8	FIQ8411 Råverdi	GJ	1	5
9	FIQ8411 Korreksjon	tonn	1	6
10	FIQ8411 Korreksjon	GJ	1	7
11	FIQ8411 Verifisert	tonn	2	4,6,
12	FIQ8411 Verifisert	GJ	2	5,7
13	FIQ8407-FIQ8411 Relativt avvik (1%)	tonn	3	0,4,
14	FIQ8407-FIQ8411 Relativt avvik (1%)	masse %	4	0,4,
15	FIQ8407-FIQ8411 Relativt avvik (1%)	GJ	3	1,5
16	Sluttverdi	tonn	7	1,3,5,7,
17	Sluttverdi	GJ	7	0,2,4,6
18	Kommentar	Tekst		

Presentations of the report in the application:

Råverdi for Brenngass Ineos Bamble 2021

dato	FIQ8407 Råverdi	FIQ8407 Råverdi	FIQ8407 Korreksjon	FIQ8407 Korreksjon	FIQ8407 Verifisert	FIQ8407 Verifisert	FIQ8411 Råverdi	FIQ8411 Råverdi	FIQ8411 Korreksjon	FIQ8411 Korreksjon	FIQ8411 Verifisert	FIQ8411 Verifisert	FIQ8407-FIQ8411 Relativt avvik (1%)	FIQ8407-FIQ8411 Relativt avvik (1%)	FIQ8407-FIQ8411 Relativt avvik (1%)	Sluttverdi/Sluttverdi	Sluttverdi/Sluttverdi	Kommentar
	tonn	GJ	tonn	GJ	tonn	GJ	tonn	GJ	tonn	GJ	tonn	GJ	tonn	masse %	GJ	tonn	GJ	Tekst
01.05.2021	0	0	9.17	517.483	9.17	517.483	9.17	517.483	0	0	9.17	517.483	-9.17	0	-517.483	517.483	9.17	korrigering%20
02.05.2021	0	0	8.837	498.691	8.837	498.691	8.837	498.691	0	0	8.837	498.691	-8.837	0	-498.691	498.691	8.837	korrigering%20
03.05.2021	0	0	10.366	584.943	10.366	584.943	10.366	584.943	0	0	10.366	584.943	-10.366	0	-584.943	584.943	10.366	korrigering%20
04.05.2021	0	0	12.036	679.193	12.036	679.193	12.036	679.193	0	0	12.036	679.193	-12.036	0	-679.193	679.193	12.036	korrigering%20
05.05.2021	0	0	17.126	966.408	17.126	966.408	17.126	966.408	0	0	17.126	966.408	-17.126	0	-966.408	966.408	17.126	korrigering%20
06.05.2021	0	0	13.767	776.907	13.767	776.907	13.767	776.907	0	0	13.767	776.907	-13.767	0	-776.907	776.907	13.767	korrigering%20
07.05.2021	0	0	11.836	667.907	11.836	667.907	11.836	667.907	0	0	11.836	667.907	-11.836	0	-667.907	667.907	11.836	korrigering%20
08.05.2021	0	0	10.26	578.97	10.26	578.97	10.26	578.97	0	0	10.26	578.97	-10.26	0	-578.97	578.97	10.26	korrigering%20
09.05.2021	0	0	9.266	522.876	9.266	522.876	9.266	522.876	0	0	9.266	522.876	-9.266	0	-522.876	522.876	9.266	korrigering%20
10.05.2021	0	0	9.557	539.313	9.557	539.313	9.557	539.313	0	0	9.557	539.313	-9.557	0	-539.313	539.313	9.557	korrigering%20
11.05.2021	0	0	9.747	550.037	9.747	550.037	9.747	550.037	0	0	9.747	550.037	-9.747	0	-550.037	550.037	9.747	korrigering%20
12.05.2021	0	0	0	0	0	0	10.567	596.318	0	0	10.567	596.318	-10.567	0	-596.318	0	0	
13.05.2021	0	0	0	0	0	0	9.287	524.099	0	0	9.287	524.099	-9.287	0	-524.099	0	0	
14.05.2021	0	0	0	0	0	0	10.708	576.034	0	0	10.708	576.034	-10.708	0	-576.034	0	0	
15.05.2021	0	0	0	0	0	0	9.608	542.207	0	0	9.608	542.207	-9.608	0	-542.207	0	0	
16.05.2021	0	0	0	0	0	0	9.889	558.039	0	0	9.889	558.039	-9.889	0	-558.039	0	0	
17.05.2021	0	0	0	0	0	0	9.178	517.943	0	0	9.178	517.943	-9.178	0	-517.943	0	0	
18.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
19.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
21.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
22.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
23.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
24.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
25.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
26.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
27.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
28.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
29.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
30.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
31.05.2021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Sum:	0	0	121.968	6882.728	121.968	6882.728	180.705	10197.368	0	0	180.705	10197.368	-5.829	0	-328.947	6882.728	121.968	

Appendix C

```
1 reference
public void insertManuallyRecords()
{
    selectOperation();

    if (op>0)
    {
        run = false;

        phd = new PHDtransmission();

        setParameters(idChangeableTag);
        manyValues = false;

        if (run)
        {
            if (parameter[0]==1)
            {
                setValuesToUPSRAF2();
            }
            if (parameter[0] == 2)
            {
                setValuesToUPSRAF();
            }
        }

        setParameters(idCalculatedTag);
        if (dates.Count>1)
        {
            manyValues = true;
        }

        if (run)
        {
            if (parameter[0] == 1)
            {
                setValuesToUPSRAF2();
            }
            if (parameter[0] == 2)
            {
                setValuesToUPSRAF();
            }
        }
    }
}
```

Appendix D,

```
1 reference
public List<object> ReturnMonthReport(int id, int _year, int _month)
{
    year = _year;
    month = _month;

    phd = new PHDtransmission();
    objectList = new List<object>();
    information = new List<int>();

    run = true;

    setParameters(id);

    if (run)
    {
        setIndexForLists();

        getValuesFromUPSRAF2();

        getValuesFromUPSRAF();

        if (id==14 || id==20)
        {
        }
        else
        {
            checkValues();
        }

        calculatedMonthReport();
    }

    setInformation();

    setMonthReportToObjectList();

    return objectList;
}
```

Appendix E, code of Manual Registration

```
// GET api/<RegistryController>/5
[HttpGet("{id}")]
0 references
public ActionResult<IEnumerable<object>> Get(int id)
{
    gmr = new GetManuallyRecords();
    try
    {
        return gmr.ReturnManuallyRecords(id);
    }
    catch (Exception)
    {
        throw;
    }
}

// POST api/<RegistryController>
[HttpPost]
0 references
public void Post([FromBody] SetManuallyRecords setManuallyRecords)
{
    setManuallyRecords.insertManuallyRecords();
}
}
```