

Jørgen Haslum Lehne

ICT Architecture for a Key Performance Indicator (KPI) Management Application for Zero Emission Neighborhoods

Master's thesis in Informatics
Supervisor: Sobah Abbas Petersen
July 2021

Jørgen Haslum Lehne

ICT Architecture for a Key Performance Indicator (KPI) Management Application for Zero Emission Neighborhoods

Master's thesis in Informatics
Supervisor: Sobah Abbas Petersen
July 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

ICT Architecture for a Key Performance Indicator (KPI) Management Application for Zero Emission Neighborhoods

Jørgen Haslum Lehne

July 1, 2021

Acknowledgment

This thesis marks the end of my time as a university student. It has been a long and winding journey, through which I have gained a wealth of knowledge and experience, both curricular and extra-curricular.

I want to extend my deepest thanks to my supervisor, Sobah Abbas Petersen, for guiding me through this project and for always being available for help.

I want to thank the people at the ZEN Research Centre for their collaboration and for helping me test the KPI tool, and I especially want to thank Kristin Fjellheim, whom I've had many meetings with to guide in the development of the tool.

I want to thank my parents, for being great parents and for always being supportive of my choices in life.

Finally, I want to thank Arja Pedersen, who has supported me during the writing of this thesis and who has kept me sane through the pandemic, you're great.

Jørgen Haslum Lehne
Trondheim, July 1, 2021

Abstract

This thesis sought to develop an ICT architecture for a Key Performance Indicator (KPI) management application for Zero Emission Neighborhoods (ZENS). This was done by using the pre-existing ZEN KPI tool built in Microsoft Excel which was developed by the ZEN Research Centre as a starting point and investigating how it could be improved. The project was conducted using design science methodology, with iterative evaluation cycles to improve the work. The result was a web application which was built using Flask and React and which connects to the Excel-based KPI tool using Microsoft Graph. This application improves on the previous KPI tool by providing a more user friendly interface and several additional useful features.

Sammendrag

Denne oppgaven hadde som mål å utvikle en IKT-arkitektur for en applikasjon for administrering av nøkkelindikatorer (KPI) for nullutslippsområder (ZEN). Dette ble gjort ved å ta utgangspunkt i det eksisterende ZEN KPI-verktøyet bygget i Microsoft Excel som ble utviklet av ZEN-senteret og undersøke hvordan det kunne forbedres. Prosjektet ble gjennomført ved hjelp av design science metodikk, med iterative evalueringssykluser for å forbedre arbeidet. Resultatet var en nettapplikasjon som er bygget i Flask og React, og som kobles til det Excel-baserte KPI-verktøyet ved hjelp av Microsoft Graph. Denne applikasjonen forbedrer KPI-verktøyet ved å tilby et mer brukervennlig grensesnitt og ytterligere funksjoner.

Contents

Acknowledgment	iii
Abstract	v
Sammendrag	vii
Contents	ix
Figures	xiii
Tables	xv
Code Listings	xvii
Acronyms	xix
1 Introduction	1
1.1 Motivation	1
1.2 Goals & Contribution	2
1.2.1 Research Questions	2
1.3 Development Project	3
1.4 Thesis Structure	3
2 Background	5
2.1 Project Context	5
2.1.1 Zero Emission Neighborhoods	5
2.1.2 ZEN Research Centre	6
2.1.3 Key Performance Indicators	6
2.1.4 The KPI Tool - Initial Concept	6
2.1.5 Front-End Concept	8
2.2 Technology Review	8
2.2.1 Microsoft Excel	8
2.2.2 Excel-based KPI Tools	10
2.2.3 Microsoft Graph	12

2.2.4	Usability of Excel	12
2.3	Revisiting the Research Questions	13
3	Method	15
3.1	Project Process	15
3.1.1	Determining Requirements	16
3.2	Research	17
3.2.1	Research Method	17
4	Design	19
4.1	Requirements	19
4.1.1	Functional Requirements	19
4.1.2	Non-Functional Requirements	19
4.2	Architecture	21
4.2.1	Excel KPI Tool Integration	21
4.2.2	Overview	22
4.2.3	Data Flow	23
5	Implementation	25
5.1	Excel Integration	25
5.1.1	Microsoft Azure	25
5.1.2	Microsoft Graph	26
5.2	Back-end	27
5.2.1	Framework	27
5.2.2	Excel Mapping	28
5.3	Database	29
5.4	Front-end	29
5.4.1	Framework	29
5.4.2	Structure	30
5.4.3	Data Flow	30
5.4.4	Interface	32
5.4.5	Scenario Editor Construction	42
6	Evaluation & Results	43
6.1	Continuous Self-Evaluation	43
6.2	Iterative User Testing	44

6.3 Scalability	47
7 Discussion	49
7.1 Web-Based Implementation of KPI Tool	49
7.2 Excel Integration	50
7.3 Limitations & Issues	52
8 Conclusion & Future Work	53
8.1 Conclusion	53
8.2 Future Work	53
Bibliography	55
A Source Code	59

Figures

2.1	Microsoft Excel layout	9
2.2	Excel KPI tool	11
3.1	The project process.	16
3.2	Deciding upon the area of research.	17
3.3	The three cycle view of design science research.	18
4.1	Overview of the application architecture.	23
4.2	Big picture overview of the application data flow.	23
5.1	Diagram of the components in the React front-end. Components that have child components are marked in yellow.	31
5.2	State management in the scenario editor.	33
5.3	The login screen for the application.	35
5.4	The front page for a scenario in the scenario editor.	36
5.5	Picture of the scenario editor in the KPI tool.	37
5.6	Picture of a KPI subcategory tab.	38
5.7	The application's screen for showing results.	39
5.8	Radar chart visualization of a scenario's calculated results.	40
5.9	Bar chart visualization of a scenario's calculated results.	40
5.10	The application's menu for loading saved scenarios.	41

Tables

2.1 ZEN Key Performance Indicators at time of writing. 7

4.1 The functional requirements for the KPI tool. 20

Code Listings

5.1 Patch request to Microsoft Graph	26
5.2 KPI dictionary excerpt	28

Acronyms

FME Forskningscenter for Miljøvennlig Energi.

GHG greenhouse gas.

ICT information and communications technology.

KPI Key Performance Indicator.

NTNU Norwegian University of Science and Technology.

ZEB Zero Emission Building.

ZEN Zero Emission Neighborhood.

Chapter 1

Introduction

1.1 Motivation

With the threat of climate change becoming ever more apparent, the desire to make different aspects of modern life more environmentally friendly increases. One area with a lot of associated greenhouse gas (GHG) emissions is the construction and operation of buildings, which accounted for 39% of energy-related CO₂ emissions in 2017 [1]. The ZEN Research Centre, also known as FME ZEN [2], is a research center which seeks to address this, by creating solutions which will aid building projects and neighborhoods in eliminating their GHG emissions. One of the most important results of their work is the creation of a set of guidelines and definitions for what constitutes an environmentally friendly building project, quantified in the form of several Key Performance Indicators (KPIs) which measure the environmental impact that such a project will have over its lifetime [3]. To go along with these definitions, researchers at the ZEN Research Centre have also developed a Microsoft Excel-based tool to aid in calculating, organizing, and rating the KPIs according to the specifications laid out by the report which conceptually defines the tool [4].

Microsoft Excel is a program used by millions of people and countless organizations around the world. Excel is ubiquitous due to its powerful functionality, which makes it a good option for a wide range of problems and a myriad of different tools have been developed using it. Such tools can however have certain issues, as users can find Excel complicated and unwieldy to use [5] [6]. Furthermore, over time it may happen that the needs of the users exceed the functionality which can be provided by Excel-based tools, as has turned out to be the case with the ZEN KPI tool. In these cases, one would normally take the knowledge gained from developing and using the Excel tool and apply it to developing a new standalone application. However, using Microsoft Graph it is possible to incorporate the Excel tool as an active part of the new expanded application, thus making it possible to avoid having to re-implement the work which has already gone into

development, as well as making it possible for personnel who is not skilled in programming to continue contributing to the development by working on the Excel tool.

1.2 Goals & Contribution

This thesis seeks to investigate how to make the current Excel-based KPI tool more user friendly, and to develop a conceptual information and communications technology (ICT) architecture and a prototype for an application which can meet the increased needs of a ZEN KPI tool, and lastly to explore the viability of having a pre-existing Excel tool as an active part of such an application. This type of implementation is only possible due to relatively recent developments made by Microsoft to their online services. Due to the recency of these developments, there is limited literature on extending an Excel application into a fuller application through the use of Microsoft Graph. The main contributions of this thesis will therefore be the conceptual ICT architecture, and the research into extending Excel-implemented solutions.

1.2.1 Research Questions

To aid in accomplishing these goals, the following research questions will be answered:

- **RQ1:** *How can the KPI tool be made more accessible to users?*
- **RQ2:** *How can the usability of the current Excel-based ZEN KPI tool be increased?*
- **RQ3:** *Is incorporating an Excel workbook as an active component a possible and viable design choice for a web application?*

RQ1 will be answered by reviewing the technology and the literature. RQ2 and RQ3 will be answered through a combination of a literature review, a technology review, and development and evaluation of a KPI tool prototype. RQ3 is by nature subjective, but will be answered qualitatively by considering scalability, maintainability, and performance.

Note that the term "accessability" usually refers to the ability of people with different capabilities to use a system, for example if color coding in a system prevents a person with color blindness to use it effectively. However, in the case of RQ1, the term "accessible" is used specifically to refer to the ability of a potential user to actually gain access to the tool.

1.3 Development Project

The design and development of an architecture and a prototype of KPI tool forms the majority of the work which this thesis is based on. It resulted in a web application which seeks to offer the functionality required of the ZEN KPI tool as laid out by the specifications in [4], but which also includes additional functionality which has been arrived at through workshops, meetings, and iterative user testing with researchers from the ZEN Research Centre. This additional functionality includes, but is not limited to, the ability for a user to input KPI data in a web portal, to store and retrieve the KPIs for different projects, and to display results from the KPI tool in charts to make the information more accessible to the user.

1.4 Thesis Structure

This thesis is structured into eight chapters, including this introduction.

Chapter 2 details the background of the thesis, such as the stakeholders and concepts which motivated this work, as well as the technologies relevant to the project.

Chapter 3 describes the project process and the research method.

Chapter 4 discusses the design of the application, while chapter 5 documents the actual implementation and the process surrounding it. Chapter 6 describes the way in which the work was evaluated and the results from the evaluation.

Chapter 7 discusses the work done and the obtained results, and finally chapter 8 contains the conclusion for this thesis and outlines possible future work.

Chapter 2

Background

2.1 Project Context

This section details the circumstances that led to the existence of this project, such as the primary stakeholders, the most important related concepts, and previous work leading up to it.

2.1.1 Zero Emission Neighborhoods

The ZEN Research Centre defines a Zero Emission Neighborhood (ZEN) as a neighborhood which seeks to "reduce its direct and indirect greenhouse gas emissions towards zero" over a period of 60 years of service life, where a neighborhood is defined as "a group of interconnected buildings with associated infrastructure" [3, p. 17]. Here infrastructure refers to all the systems relating to the supply, generation, and storage of heat and electricity, as well as systems relating to water, sewage, waste, mobility, and ICT.

The definition of a Zero Emission Neighborhood is based on related earlier work, one example being the concept of a Zero Emission Building (ZEB) [7]. A large portion of the GHG emissions from a building project is from the construction phase, with sources being the transportation of materials, energy use at the building site, and especially emissions embodied in the building materials. To compensate for this, a ZEB has to produce enough green energy during its operational lifetime to replace energy which would otherwise be produced by methods causing GHG emissions to negate the emissions caused by its construction.

This concept then directly leads to Zero Emission Neighborhoods, which follow the concept of embodied and released emissions during the construction phase which are compensated for by green energy production during the operational phase.

2.1.2 ZEN Research Centre

The ZEN Research Centre is a project dedicated to developing solutions for building projects to eliminate their GHG emission. It is a research partner to NTNU and SINTEF, as well as having over 30 other partners in both the public and private sector [8], and the Centre is set to last for eight years from 2017 to 2024 [3]. In 2019, there were nine demonstration projects spread across Norway connected to the Centre [9].

2.1.3 Key Performance Indicators

The Key Performance Indicators are a set of assessment criteria developed by the ZEN Research Centre to enable the measurement of a project's status with regards to achieving its goal of reducing greenhouse gas emissions, and are an integral part of how a Zero Emission Neighborhood is defined [3]. The KPIs are still under development at time of writing, but are currently divided into seven main categories:

- GHG Emissions
- Energy
- Power/Load
- Mobility
- Economy
- Spatial Qualities
- Innovation

These are further detailed in Table 2.1.

The individual KPIs of a project are found by different methods using a wide variety of tools, with an internal survey performed in 2018 documenting a total of 18 different tools with more or less relevance to KPI calculation [4] [10].

2.1.4 The KPI Tool - Initial Concept

The initial concept for the KPI tool was developed by the ZEN Research Centre and specified a tool which would be able to take as input a building project scenario's different KPI values obtained from the other ICT tools and compare them with reference values and visualize the output to the user [10]. The users in this case are primarily envisioned to be ZEN researchers, so a baseline of technical expertise can be expected and the tool does not need to accommodate users unfamiliar with concepts like KPIs.

An enhanced version of the tool is also proposed. This could include features such as the tool having the ability to suggest improvements to a project which would optimize the project KPIs, the facilitation of users other than ZEN

Category	Assessment criteria and KPIs	Unit
GHG emissions	Total GHG emissions	tCO_{2eq} $kgCO_{2eq}/m^2$ heated floor area (BRA)/yr $kgCO_{2eq}/m^2$ outdoor space (BAU)/yr $kgCO_{2eq}/capita$
	GHG emission reduction	% reduction compared to a base case
Energy	Energy efficiency in buildings: - Net energy need - Gross energy need - Total energy need	kWh/m^2 heated floor area (BRA)/yr
	Per energy carrier: - Energy use - Energy generation - Delivered energy - Exported energy - Self consumption - Self generation - Color coded carpet plot	kWh/yr kWh/yr kWh/yr kWh/yr % % kWh/yr
Power/Load	Power/load performance: - Yearly net load profile - Net load duration curve - Peak load - Peak export - Utilization factor	kW kW kW kW %
	Power/load Flexibility: - Daily net load profile	kW
Mobility	Mode of transport	% share
	Access to public transport	Meters Frequency
Economy	Life cycle cost (LCC)	NOK NOK/m^2 heated floor area (BRA)/yr NOK/m^2 outdoor space (BAU)/yr $NOK/capita$
Spatial Qualities	Demographic needs and consultation plan	qualitative
	Delivery and proximity to amenities	No. of amenities Meters (distance from buildings)
	Public space	qualitative
Innovation	To be determined	

Table 2.1: ZEN Key Performance Indicators at time of writing.

researchers which would help spread awareness of the ZEN definition, or for the tool to have the ability to dynamically access different types of reference data to compare to.

2.1.5 Front-End Concept

In 2019, a group of six students produced a report detailing their work on a project in the subject IT2901 - Informatics Project II [11]. The goal of the project was to implement a system which received, stored, processed, and visualized KPI data from several different sensors in a Zero Emission Neighborhood. However, due to various problems, they were only able to develop the front-end of the project which mainly dealt with the visualization of the KPIs.

The IT2901 Student Project focuses on the creation of a system which monitors and reports recent usage data, while the focus of this master's thesis is on the creation of a tool which is meant to aid in the planning of new construction projects. Despite this difference, both projects deal with the management and visualization of KPI data, so there is enough similarity between them to use the IT2901 Student Project as inspiration.

The project implements the front-end as a website using the popular React library for JavaScript [12]. The end result was a product which was able to meet the requirements that were agreed upon with representatives from the ZEN Research Centre, who were pleased with the outcome. This was taken into consideration when deciding upon the design for the development project of this master's thesis.

2.2 Technology Review

Before beginning development, a technology review must first be conducted in order to obtain an overview of the different technological artifacts which already exists. This will uncover any currently existing products or systems which may meet some or all of the requirements of the proposed KPI tool, or which can offer insight into how to design and develop such a tool from scratch.

The frameworks used to develop the web application stack will not be discussed in this chapter, as they are not important to the nature of the thesis and different ones could have been used in their place to achieve the same results. They will instead be described in detail when discussing the implementation in Chapter 5.

2.2.1 Microsoft Excel

Microsoft Excel is a spreadsheet application made for storing, organizing, and analyzing data [13]. A part of the Microsoft Office software family, Excel was first

launched in 1987. With the large demand for data analysis solutions among organizations and individuals, Excel has gained widespread use due to its powerful data handling capabilities and its relatively low barrier to entry, and also because of Microsoft's historical market dominance and Excel being Microsoft's primary commercial product addressing this need.

Excel is structured around spreadsheets known as worksheets, which contain cells spread out in rows and columns. Worksheets are sized automatically depending on their content but can potentially have over a million rows and over 16 000 columns. An Excel file is known as a workbook, which is a collection of one or more worksheets. This basic layout can be seen in Figure 2.1.

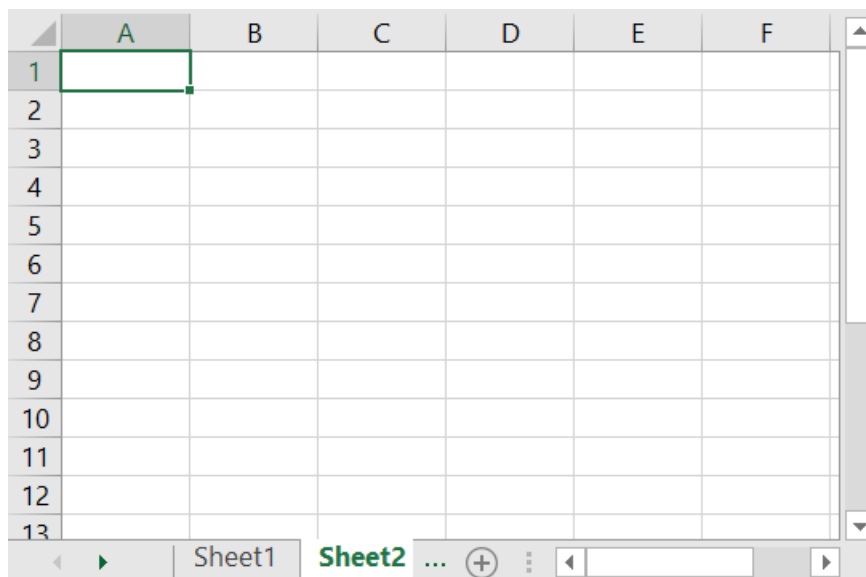


Figure 2.1: Microsoft Excel layout

Cells have great versatility. They can contain raw data, as well as user defined formulas which can calculate results based on data from other cells, even referencing cells in different worksheets. There are also formatting options which make it possible to drastically change the look of the worksheets, which can make a workbook more intuitive and give a better user experience. Taken together, this basic functionality allows users to create intricate and sophisticated workbooks suited for a plethora of different purposes, including accounting, financial analysis, time management, task management, and many more. It has also been used to develop tools for the management of KPIs, as will be seen in the next two sections.

2.2.2 Excel-based KPI Tools

PI-SEC Tool

Planning Instruments for Smart Energy Communities (PI-SEC) is a research project which seeks to deliver planning instruments for the integration of energy issues at the property level in smart neighborhoods, with focus on areas of CO₂-reduction, increased use of renewable energy, increased energy efficiency, increased use of local energy sources, and green mobility [14] [15]. These areas of focus make it apparent that the PI-SEC project has several similarities with the ZEN project, and was in fact an important source of inspiration and knowledge when the ZEN definition and the accompanying KPIs were developed [3] [16].

One of the results of the PI-SEC project is the PI-SEC tool which is a KPI tool that evaluates how different decisions and measures affect different KPIs. The first version of the tool is based on Microsoft Excel, with the suggestion that later versions of the tool can be developed in a different format, like being web-based.

The PI-SEC tool is mentioned by the ZEN Research Centre as an example of a tool which addresses several of the ZEN KPIs [10], however it does not cover all of the necessary KPIs and as such it is an insufficient solution when it comes to addressing the needs of the proposed ZEN KPI tool.

ZEB Tool

The ZEB tool [17] is an Excel-based tool developed as a part of the Zero Emission Building project [7] which can be used to calculate a building's embodied GHG emissions. Embodied emissions are an important part of the ZEN definition, but still only a part, and a ZEN KPI tool will have to be more comprehensive. The design and development of the ZEB tool has been an important source of inspiration for the ZEN KPI tool.

ZEN KPI Tool - Excel Implementation

The KPI tool concept described in section 2.1.4 led to the development of a version of the tool based on Microsoft Excel. This tool is structured by KPI category, and each category is further divided into subcategories with each subcategory having its own dedicated sheet in the Excel workbook with related information and instructions, as well as input fields for KPI values. Currently, the tool allows a user to input and store KPI values and provides them with instructions for how to calculate points based on the values. An example of one of the subcategory sheets is shown in Figure 2.2, though it should be noted that this is a very early version of the tool and that it is still under continuous development, and as such any features depicted are subject to change.

	A	B	C	D	E	F
1	GHG1.1					
2	Mål					
3	Å minimere totale klimagassutslipp fra et område mot null med fokus på materialbruk over et 60 års referanseperiode.					
4	Redusere totale klimagassutslipp fra produksjon- og utskiftningsfaser av materialer (livssyklusmoduler A1-A3, B4) for hver bygning og infrastruktur i nabolaget.					
5	Bygg					
6	Nabolag					
7	Enhet					
8	kgCO ₂ eq/m ² oppvarmet areallår					
9	Metode					
10	IPCC AR5, ISO 14040/44, EN 15978, NS 3720, NS 3451, NS 3940, NS 3457-3, ISO 21930, ISO 14025, EN 15804					
11	Verktøy					
12	ZEB tool, OneClick LCA, Simapro, bLCA4, område LCA, vegLCA, EFPEKT 6.6, Dynamisk stock-flow scenariomodellering, Arda					
13	Tildeling av poeng					
14	Point allocation					
15	Beregning av poeng:					
16	Poenggrensene for NS3720 bygg basis er bestemt gjennom ZEN Case om forslag til klimagassutslippkrav i TEK. Poenggrenser for ulike bygningstyper er publisert i ZEN rapport XX og kan brukes for å lage skreddersydd referanseverdier på område nivå. Systemgrensen skal utvides til NS 3720 avansert for å inkludere tekniske utstyr (bygningdeler 31-69) og infrastruktur (bygningdeler 71-79) når datagrunnlaget foreligger.					
17	3,6	52%				
18	1,5	22%				
19	1,8	28%				
20						
21						
22						
23						
24	Dokumentasjon					
25	Documentation					
26	<input type="checkbox"/>	Følgende er dokumentasjonskrav: Livsløpsanalyse (LCA) rapport for området. LCA av klimagassutslipp for alle bygninger og infrastruktur i nabolaget gjennomført i forhold til NS 3720 basis eller avansert for livssyklusmodulene A1-A3 og B4. LCA rapporten for nabolaget skal inneholde informasjon om bygningstyper og infrastruktur, bygningstyper, bygningstyper, bgningsareal, antall brukere, referanseperioden, systemgrenser, scenario beskrivelser, materialinventar med mengder, utslippsfaktorer og kilder, resultater per bygning og infrastruktur for hver livssyklusmodul (A1-A3 og B4) og bygningstyper samt pålitelighetsvurdering av analysen. Livsløpsanalysen skal gjennomføres i alle prosjektfasene.				
27	Tilleggsinformasjon					
28	Supplementary notes					
29	Systemgrense: Omfanget for NS 3720 basis inkluderer bygningdeler 21, 22, 23, 24, 25, 26, 27, 28, 29 og 49 for livssyklusmodulene A1-A3 og B4. Omfanget for NS 3720 avansert eller bygningdeler 31-69 mens omfanget for NS 3720 infrastruktur dekker bygningdeler 71-79. Eksisterende bygg som har ingen planlagt materialbruk i referanseperioden er betraktet som om de har null klimagassutslipp.					
30	Referanse: ZEN rapport XX : Orienteringsverdier for klimagassutslipp referanseverdier i bygninger					
31	Nøkkelverdier					
32	Key values					
33	ZEN metrics					
34	Klimagassutslipp [kgCO ₂ eq/m ² /yr]					
35	GHG emissions [kgCO ₂ eq/m ² /yr]					
36	Strategisk planleggingsfase					
37	Implementeringsfase					
38	Bruksfase					
39	Use phase					
40	Strategisk planleggingsfase					
41	Implementeringsfase					
42	Bruksfase					
43	Use phase					
44	Tildelt poeng					
45	Points awarded					
46	7					
47	0					
48	0					
49	0					
50	Oversikt					
51	Forklaring					
52	GHG					
53	1.1					
54	1.2					
55	1.3					
56	1.4					
57	1.5					
58	1.6					
59	1.7					
60	EN					

Figure 2.2: Excel KPI tool

The point values the user inputs for each KPI subcategory are combined with a weighting system which results in a final rating for each KPI category which shows how close the project is to reaching its stated ZEN ambition level, i.e. how closely the project adheres to the ZEN definition.

However, while this version of the tool serves as a valuable prototype, Excel also carries with it certain limitations which makes it an unsuitable platform for fully satisfying the needs of the KPI tool. For example, as the KPI tool would usually be a file stored locally on a user's computer, there would be no satisfactory way to perform version control and keep all the tools spread out among users up to date. Similarly, saving the KPI data of several different project scenarios would require a user to create several different files, and then to have to manually move all this data to new files if an updated version of the tool is ever released. In addition, using the tool is a complex task. The worksheet displayed in Figure 2.2

is quite complicated with a lot of elements visible on screen, which can make it difficult for a user to grasp how they are supposed to interact with it. The current KPI tool contains 43 of these worksheets, so the complexity can quickly become overwhelming and navigating the tool can be perceived as difficult. It is this Excel-based tool and its limitations which the work in this thesis is primarily based on.

Finally, it should be noted that these limitations do not make Excel a bad tool, but instead merely make it the incorrect tool for the final implementation of the KPI tool. Indeed, Excel is a good option for quickly and efficiently prototyping the KPI tool to gain valuable knowledge on what would be required of future development.

2.2.3 Microsoft Graph

In 2017, Microsoft made several of the products from its Microsoft Office package available as a subscription-based online service in the form of Microsoft 365. Microsoft 365 offers users the ability to work in browser-based versions of the different Office tools instead of having to install dedicated Office software, in addition to giving them access to cloud-based storage to store and access their data.

To facilitate the flow of data in Microsoft 365, Microsoft has created a developer platform known as Microsoft Graph [18]. This platform consists of several components, but of interest to us is the Microsoft Graph API component, a single endpoint which seeks to serve as a common API for all of Microsoft's services. Relevant to this project, it has the option to provide software developers with direct access to Excel workbooks saved in cloud-storage, not only to read data from the workbook, but also to insert new values, and even have Excel actively perform calculations based on these new values and the formulas in the workbook.

This gives many new possibilities. Before Microsoft 365 and Microsoft Graph it was possible to programmatically read and write data to Excel files, but getting Excel to actually perform calculations on this data could not be done without considerable difficulty¹. Now, with the ability of linking a program to an active workbook, many new possibilities appear, with this project aiming to investigate a few of them.

2.2.4 Usability of Excel

Usability is defined by ISO 9241-11 [19] as the "extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". Microsoft Excel has

¹Doing this involves running an instance of the Excel application on the machine with the server and then using the Microsoft Component Object Model (COM) to communicate with the instance. Being only a single instance, it is not scalable beyond a single user and so is not viable for a web application.

some problems when it comes to usability. A 2010 study found that users often had troubles navigating Excel's many complex features, and that they were reluctant to use Excel when they felt they did not have mastery over it [5].

Another study which was published in 2013 surveyed more than 1,000 users on the usability of fourteen everyday products [6], using the Systems Usability Scale (SUS) [20]. One of these products was Microsoft Excel, which got the lowest rating out of all the products with a SUS score of 56.5 out of 100, which lands it outside the "acceptable" range which the study uses (and in fact, it was the only product surveyed not to fall in the "acceptable" range).

It must be noted that the study measures products which are not directly comparable to Excel, consisting of items such as microwaves, iPhones, and Google Search, but since the SUS is not a comparative scale the score stands on its own and the result is clear: people do not rate Excel as having high usability.

As stated by the Technology Acceptance Model (TAM), usability is of great importance to get users to actually use new systems [21]. The ZEN Research Centre focuses heavily on the creation of tools to disseminate their research [10]; if the tools are not being used then the research will not have the desired impact. The consequence of all this is that if usability is to be a focus when designing a KPI tool, then other platforms than Excel should be considered.

2.3 Revisiting the Research Questions

Based on the information in this chapter, it is possible to get a better understanding of how the research questions should be addressed:

- **RQ1:** *How can the KPI tool be made more accessible to users?*

Considering both the issues with trying to distribute an Excel-based tool to different users as well as the promising front-end concept described in Section 2.1.5, it seems that constructing a web-based version of the tool will be the most reasonable approach to address RQ1.

- **RQ2:** *How can the usability of the current Excel-based ZEN KPI tool be increased?*

Taking into account the usability issues with Excel, it appears that the most straightforward way to increase usability would be to use a different platform altogether. Another way to increase usability would be to add more features to the tool, such as the ability to save and load different user-created KPI scenarios, and the ability to easily visualize the results in graphs. Again, the front-end concept described in Section 2.1.5 suggests that a constructing web application is an appropriate way to address RQ2.

- **RQ3:** *Is incorporating an Excel workbook as an active component a possible and viable design choice for a web application?*

Given that an Excel-based version of the KPI tool already exists, it could be advantageous to integrate it into the web application instead of having to recreate all its features. The existence of Microsoft Graph suggests that this should be possible, and developing an application that implements it will show how it actually works out in practice.

Chapter 3

Method

3.1 Project Process

As mentioned in the introduction in Section 1.1, this thesis has its origin in the pursuit of developing a ZEN KPI tool. The following description of a specific development project was defined in the early stages of the overall thesis project:

The project will focus on a conceptual ICT architecture for the FME-ZEN Key Performance Indicator (KPI) application. One of the main products of the FME-ZEN project is the ZEN Guidelines, which is a method for determining how well a building or a neighborhood meets the ZEN criteria, and this is supported by the ZEN KPI tool. The KPI tool must support a variety of data and data formats from a variety of sources. This project will focus on an ICT architecture and application prototype that will support the KPI tool for collecting, storing, providing access to the KPIs. As a part of this project, we will examine the different scenarios of the use of the KPI tool and design and implement the functionality to support the scenarios. A prototype implementation will be tested using data from at least one ZEN pilot. This work will be done in collaboration with relevant ZEN partners from NTNU and SINTEF.

This project description then led to a literature review to understand the context of the problem and the available technologies which could be used to solve it. The review, in combination with input from stakeholders from the ZEN Research Centre, led to a preliminary design for the application architecture. A prototype of the application was developed based on the design, which was then evaluated through the use of heuristics and user tests. Then the process was iterated upon by going back to the earlier stages and refining the work done in these stages, based on needs and issues that were uncovered in these evaluations. An illustration of this process is shown in Figure 3.1.

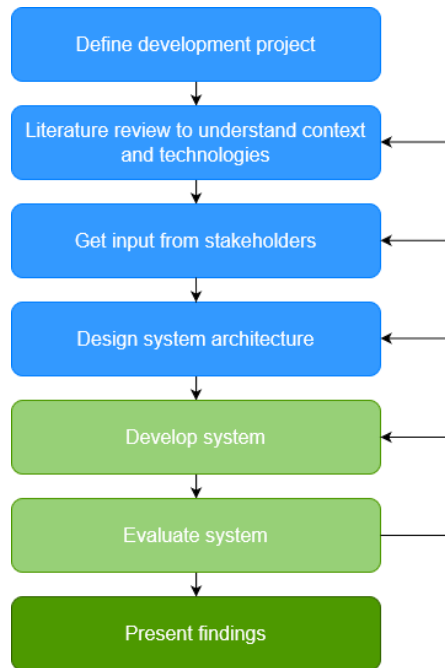


Figure 3.1: The project process.

Throughout the process, meetings with the project supervisor and the contact person from the ZEN Research Centre were conducted every two weeks. In these meetings, project progress and problems which had appeared were discussed, and requirements were workshopped and refined. The bi-weekly meetings with the contact person and the iterative development process are features of agile development, and so the process can be said to loosely follow an agile mindset.

The development process faced certain problems. The project got off to a relatively late start because of difficulties with getting in contact with stakeholders from the ZEN Research Centre, which meant that many things which ideally should have been implemented were not, due to time constraints. Similarly, there were issues with getting hold of people from the target user group (researchers from the Centre) to perform testing due to them having busy schedules.

3.1.1 Determining Requirements

The specific requirements for the application were determined by a variety of methods. These methods include:

- Consulting previous work, such as the KPI tool concept proposed in the report by Petersen [10], and the student report [11] mentioned in section 2.1.5 which describes a similar product.
- Workshops and bi-weekly meetings conducted with researchers from the

- ZEN Research Centre.
- Feedback from users after conducting user testing with researchers from the ZEN Research Centre.
- Necessary functionality requirements discovered during development.

The requirements are listed section 4.1 in the chapter on design.

3.2 Research

3.2.1 Research Method

The development process described in the previous section also naturally influences the way the research method was conducted. The scope of the research was arrived at by considering the development project to be undertaken and the related literature, and then envisioning what new solutions could be created and what new knowledge could be gained. A simplified version of this process can be seen in Figure 3.2.

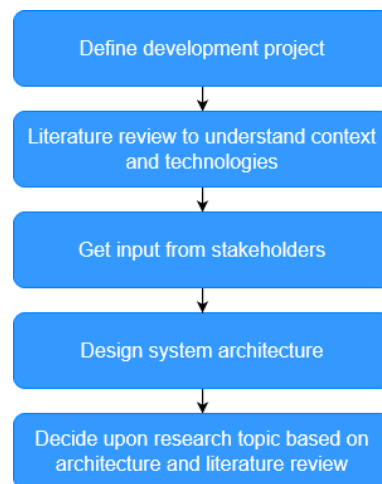


Figure 3.2: Deciding upon the area of research.

The research process follows the methodology of design science research [22], which seeks to gain new knowledge through the development and evaluation of IT artifacts. Design science research specifically focuses on gaining knowledge which will be useful in implementing future solutions, and as such it is of a very pragmatic nature. It can be thought of as a process consisting of three cycles [23]:

- **The relevance cycle:** Design science research seeks to improve some specific environment, which initially provides the requirements for the artifact to be developed, and eventually serves as the test arena to assess the use-

fulness of the artifact.

- **The rigor cycle:** The work done must be grounded in theory, and an overview of the existing knowledge base is necessary to ensure that one is performing original research and not merely implementing traditional solutions. The knowledge gained from the research will then be added to the knowledge base.
- **The design cycle:** The actual development part of the process, a rapid iterative cycle where the artifact is designed, built, and evaluated.

This process is illustrated in Figure 3.3.

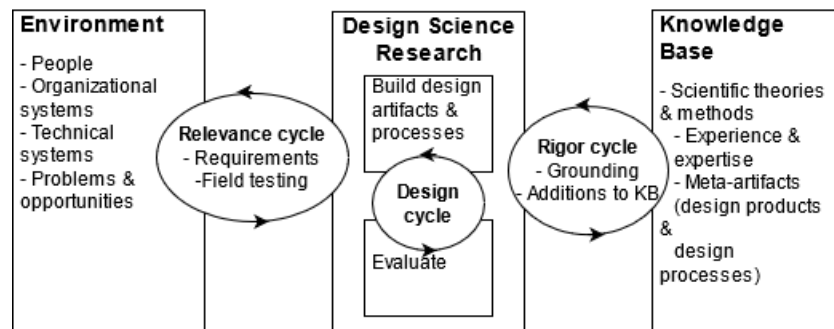


Figure 3.3: The three cycle view of design science research.

The three cycle view describes the way in which this thesis' overall development project has been conducted with its aim of researching and developing a solution for implementing the ZEN KPI tool. The environment consists of, but is not limited to, the ZEN Research Centre, the ZEN project, the Excel version of the KPI tool, etc., and enables the relevance cycle as it serves as both the inspiration for the project as well as providing the assessment criteria. The knowledge base includes the expertise developed by the ZEN Research Centre, the concept for KPI tool, established programming best practices, etc. It is part of the rigor cycle, as the knowledge base provides the theoretical grounding for the project, while the project provides novel research to the knowledge base. Lastly, the design cycle is the actual design and development of the product and the evaluation which includes heuristic evaluation and iterative user tests.

It must be admitted, that initially the project did not set out to follow any specific predefined methodology but rather grew organically in response to the needs and requirements which emerged as a result of developing a product as part of a master's thesis. It was only later realized that the resulting process ended up adhering to the paradigm of design science research.

Chapter 4

Design

This chapter discusses the requirements for the application, the design choices which determined the overall architecture of the application, and a short overview of this architecture.

4.1 Requirements

4.1.1 Functional Requirements

Table 4.1 shows the functional requirements for the application, as determined by the process outlined in 3.1.1.

Regrettably, due to time constraints, only FR1 - FR6 were able to be completed, however this nonetheless served to increase usability compared to the Excel-based KPI tool.

4.1.2 Non-Functional Requirements

The non-functional requirements are all the requirements of the application that do not define any required functionality, but which still describe specific needs or constraints the application has. These consist of the following:

Modifiability

It is expected that the tool being developed by this project is not going to be put into use as-is, but will rather serve as a prototype and a basis for further work on developing a KPI tool. As such, it is necessary that the architecture is designed with modifiability in mind so that future functional requirements can be easily implemented, and future developers will have an easier time learning and

FR1	It should be possible for a user to create a user account in the application.
FR2	It should be possible for a user to input all KPI values relating to a project scenario.
FR3	It should be possible for a user to save and load different project scenarios.
FR4	A user should initially only have access to their own scenarios.
FR5	A user should have the possibility to visualize a scenario's KPI data in some different ways.
FR6	Information/instructions should be available to explain all aspects of the tool.
FR7	The user should be able to change the display language.
FR8	Researchers from ZEN should be able to access all the saved scenarios.
FR9	A user should have the ability to share a scenario with other users.
FR10	It should be possible for a user to create a printable report with a scenario's KPI values.

Table 4.1: The functional requirements for the KPI tool.

understanding the code.

Usability

One of the motivations for moving from an Excel-based tool is the fact that Excel is often unpleasant to use, with examples being that spreadsheets can be difficult to navigate, or that the Excel application itself can be slow to respond. With this in mind, a new version of the KPI tool will hopefully be able to provide a more smooth user experience.

Upgradability

Upgradability is a characteristic which describes a system's ability to have parts of it upgraded without having to replace the whole system. In that regard, it is closely linked to modifiability, but it also has special implications for the KPI tool. As the KPI tool is currently under development, it is important to be able to upgrade the tool and have these upgrades be available to the users. A traditional Excel-based tool has low upgradability, as there is no easy way to upgrade the different versions of the tool stored locally on different systems, and the entire Excel file has to be replaced by the new version.

4.2 Architecture

4.2.1 Excel KPI Tool Integration

This section describes the different possible design solutions that were initially considered for translating the work done on the Excel KPI tool to a more full-fledged application. Each solution's advantages and disadvantages will be discussed, with regards to the previously listed requirements.

Manual Implementation

This is the most traditional approach. It entails taking the knowledge gained from the Excel KPI tool and manually implementing its model in a web application. In this case there would be no active connection between an Excel workbook and the application, the latter would be completely standalone.

- **Advantages:**
 - Allows the most amount of flexibility when it comes to adding new functionality, as all aspects of the application would be customizable.
- **Disadvantages:**
 - Requires re-implementing all the work already done in the Excel tool.
 - Requires researchers to be sufficiently proficient in programming to continue development of the tool.

Microsoft 365

This solution entails hosting the Excel KPI tool on Microsoft's cloud and developing a web application to communicate with it using Microsoft Graph.

- **Advantages:**
 - Limits duplication of work by directly connecting to the Excel tool instead of re-implementing it as a web application.
 - Allows a great deal of flexibility when it comes to adding new functionality.
 - Allows ZEN researchers without programming knowledge to continue development on the tool in the form of the Excel workbook.
 - There is no need to manually keep track of minor changes made to the Excel workbook to implement these in the web application, these changes will just be automatically fetched.
- **Disadvantages:**
 - The web application must make certain assumptions about the struc-

- ture of the Excel workbook, so major changes to the workbook will likely necessitate extensive maintenance of the web application.
- Introduces additional complexity to the architecture, by virtue of there being more connected parts.

Dedicated Excel Parser

This solution entails developing a program which would be able to parse an Excel workbook and create an internal copy of its data model. Developing such a program would presumably be very difficult and require a lot of effort, and likely end with a system that is over-engineered for the specific purpose of enhancing the KPI tool. It is probable that it would be more time efficient and more relevant to the task at hand to do the manual implementation. This option was only considered before the Microsoft 365 solution was fully researched, because it turns out that that solution also has the capability to fully interact with an Excel workbook's data model. Essentially this means there are no specific advantages to choosing this solution over the Microsoft 365 one, rendering this one obsolete.

Online Excel Dashboard

This solution entails taking the Excel KPI tool and embedding it in a website hosted by the ZEN Research Centre. This online solution would ensure that the tool is easily available to users, and the centralized nature of the solution would allow the developers to update the tool. On the other hand, it would not allow the user to save KPI data from different building project scenarios, thus it is not a viable solution.

Of these four possible options, the Microsoft 365 solution was chosen as the one to implement. The fact that the researchers currently working on the KPI tool are not proficient in web development was the most important factor, but also playing a part was the fact that this is an unorthodox solution to this type of problem, which fueled a desire to research the viability of it.

4.2.2 Overview

As stated in Section 2.3, it was decided that the KPI tool should be implemented as a browser-based web application. The application architecture follows the pattern of a traditional full-stack web application, with a front-end component, back-end component, and database component. This application is then connected to the Excel KPI tool through Microsoft Graph. An illustration is given in Figure 4.1.

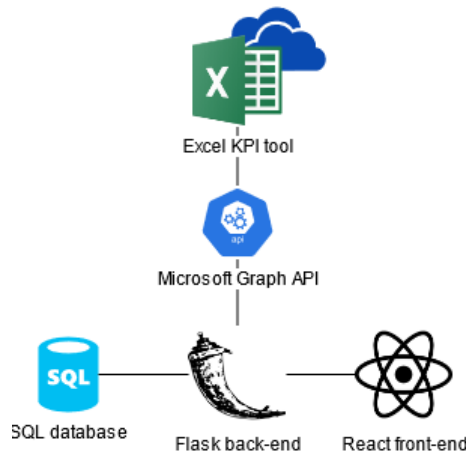


Figure 4.1: Overview of the application architecture.

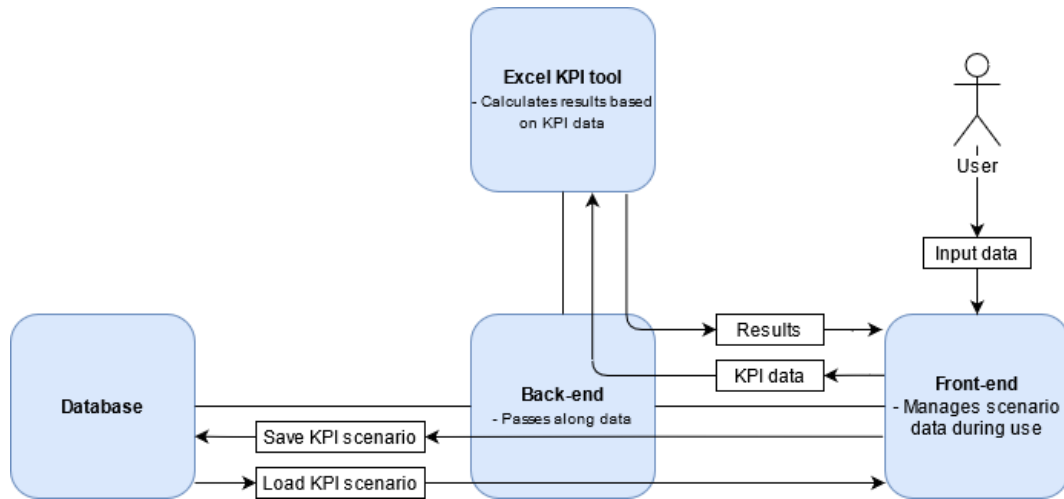


Figure 4.2: Big picture overview of the application data flow.

4.2.3 Data Flow

The big picture data flow of the application is illustrated in Figure 4.2. The user inputs KPI data into the front-end, which holds the data during use. When the user has entered scenario data, they can request results to be calculated which sends the KPI data from the front-end to the Excel workbook which calculates the results based on its own internal logic. The results are then sent back to the front-end and displayed to the user. The user can save and load scenarios in the SQL database. The transfer of data between components is facilitated by the back-end.

The data flow inside the front-end will be described in detail in Section 5.4.3 where the implementation of the front-end is described.

Chapter 5

Implementation

This chapter goes into detail about the implementation of the specific parts of the application. These parts are Excel integration part, back-end, front-end, and database.

5.1 Excel Integration

As stated earlier, the application can be viewed as an extended interface built on top of the already existing Excel-based KPI tool. This is made possible by Microsoft Graph as described in Section 2.2.3. This section will go into detail on how this was implemented.

5.1.1 Microsoft Azure

Before an application can fully access Microsoft Graph, it needs to be registered with Microsoft Azure, Microsoft's cloud computing service for developers to create and manage applications. This is a multi-step process:

1. One must register an organization with Microsoft. This will create an organization account which has the necessary authorization to register an application.
2. The admin of the organization must register a new application in Azure Active Directory. Different types of applications can be registered, one option being an application where users log in with an account type recognized by Microsoft (e.g., Hotmail) to authenticate. The application developed in this project only has to authenticate on behalf of itself, so it will be a registered as a daemon-type application. After registering and setting up permissions, Azure provides a set of credentials which the application can use to connect and authenticate with Microsoft's services.

3. The organizations needs to set up an active subscription to Microsoft 365 for Business. This is required for the application to be able to connect to the Excel KPI tool which is hosted on the cloud.

Once this is done, the application can use the credentials to generate access tokens which it needs to use Microsoft Graph.

5.1.2 Microsoft Graph

The Excel-version of the KPI tool is hosted on Microsoft's cloud storage platform OneDrive by the same admin account which registered the application on Azure. When users perform actions which require interaction with the Excel KPI tool, requests are sent from the front-end to the back-end which handles all matters related to Microsoft Graph, as illustrated in Figure 4.2.

As mentioned, the application handles authentication by using the credentials generated upon app registration to request an access token from Microsoft Graph, which it then uses in all subsequent requests to certify that the requests are legitimate. There are some problems with this process, the implementation does not always properly refresh the access token, which leads to the application not always being able to access Microsoft Graph. One possible solution is to replace the current implementation with the Microsoft Authentication Library (MSAL) for Python, though this has not been done due to time constraints.

Get and patch requests can be made to the Graph API to read or update values in the Excel KPI tool. Listing 5.1 shows an example of a patch request which updates the value of a cell in a given row, column, and sheet.

Code listing 5.1: Patch request to Microsoft Graph

```

WORKBOOK_URL = "https://graph.microsoft.com/v1.0/users/
                jorgenhl@jhltestfirm.onmicrosoft.com/drive/
                root:/Book.xlsx:/workbook/worksheets"
NON_PERSISTENT_HEADER = {
    'authorization': ga.get_access_token(),
    'content-type': "Application/Json",
    'workbook-session-id': ga.get_nonpersistent_session_id()
}
r = requests.patch(
    url = WORKBOOK_URL + "/{}/cell(row={}, column={})".format(sheet,row,column),
    headers = NON_PERSISTENT_HEADER,
    json = {'values': value}
)

```

One issue with the Graph API is that a request can patch a single cell or a range of cells in the same row and/or column, but not cells spread out over different rows, columns, and sheets. The relevant cells in the Excel KPI tool are spread out exactly like that, so patching them requires several patch requests. Currently there are 30 sheets with 3 relevant cells each, which in theory makes for a total of 90 simultaneous patch requests each time results need to be calculated

(in practice, this number is smaller, but only because not all sections of the Excel KPI tool have been fully developed yet). Luckily, this is expected to only happen once per user session, but it does take several seconds to complete and has a negative impact on user experience.

One of the most important aspects of this implementation is the existence of the non-persistent session, which makes the whole design possible. Normal patching of the Excel workbook would permanently change the values stored in it, which means that if several users were to try to use the web application simultaneously, their actions would probably interfere with each other, overwriting values that others put in. A non-persistent session is a temporary personal copy of the Excel workbook which is generated upon request and can deal with all the required calculations, and is then discarded once it is no longer needed and the session expires. Simultaneous users will in this case get different non-persistent sessions which means there is no interference.

5.2 Back-end

5.2.1 Framework

The back-end of the application is built using the Python web framework Flask [24]. Flask is regarded as a "microframework", which means that it seeks to keep its core module as lightweight as possible, with the option of using its many extensions to include additional functionality if necessary. This contrasts with for example Django [25], another Python web framework, which includes an extensive suite of functionality which makes it appropriate for handling the full stack of a web application.

Flask and Django are two of the most popular Python web frameworks available. Flask was chosen over Django for this project because it is more lightweight and the framework only needed to handle the back-end. As the project progressed, it became clear that more functionality than initially anticipated was required, mainly related to user login systems and advanced database handling. Django has this functionality built-in, while Flask has it available as extensions which had to be set up and configured. This makes Django more of a suitable choice than it appeared initially, though Flask is also still appropriate. In the final analysis, either framework would have been able to perform the job to a satisfactory degree.

Another option was to use a JavaScript-based framework. The advantages of this is that the back-end and front-end (which is made in React) would be written in the same language which could potentially make it easier for a developer to switch between them during development, as well as the fact that some research exists which points to JavaScript web frameworks being more performant than Python ones [26]. On the other hand, Python is generally regarded as better for data manipulation and analysis, a factor which was initially considered to be rele-

vant due to the data-centered nature of the application and that it was speculated that the back-end would have to handle this data. In the end, this turned out to not be the case as the final implementation merely stores and passes data along to the Excel tool through MS Graph.

As with the choice between Flask and Django, both Python frameworks and JavaScript frameworks are fully able to satisfactorily function as a back-end for this kind of web application. The thing which nudged it in favor of a Python framework over a JavaScript framework was the fact that I, as the developer, am far more comfortable working with Python than I am with JavaScript, and in the interest of saving time and effort it was deemed better to choose the programming language I am more familiar with.

The back-end was developed using Python version 3.9.1 and Flask version 1.1.2. The newest version of Flask is version 2.0, but it was released near the end of the development process and so this project does not use it.

5.2.2 Excel Mapping

For the web application to be able to interact with the Excel version of the KPI tool, it needs an accurate map of where the relevant cells in the Excel workbook are located, which is accomplished by having a large list—more specifically, a Python dictionary object—in the back-end that contains all the important cells and their row and column locations. An excerpt from this dictionary is shown in Listing 5.2.

Code listing 5.2: KPI dictionary excerpt

```
kpi_definitions = {
  "categories": {
    1: {
      "name": "GHG",
      "subcategories": {
        1: {
          "id": "1.1",
          "instructions": {
            "startLoc": { "row": 8, "column": 1},
            "endLoc": { "row": 22, "column": 5}
          },
          ...
          "points": {
            "pointsAllocatedLoc": { "row": 32, "column": 1 },
            "strategicPlanningPhaseLoc": { "row": 32, "column": 2 },
            "implementationPhaseLoc": { "row": 32, "column": 3 },
            "usePhaseLoc": { "row": 32, "column": 4 }
          }
        },
        ...
      }
    },
    ...
  }
}
```

This is not an ideal solution since the dictionary has to be manually maintained. Any changes that are made during development of the Excel workbook which adds, deletes, or changes the location of any important cells must be carefully tracked and corresponding updates must be made to the dictionary. The file containing the dictionary is 650 lines long, its large size may increase the risk of errors when updating it. Furthermore, there may be no immediate indication that something is wrong if one accesses the incorrect cell, since all the cells in the workbook look largely similar to the application. This means that errors in the dictionary can lead to bugs which are difficult to detect in practice. A possible solution to this problem would be to implement automatic testing for the Excel integration, which would be able to easily detect if a predetermined set of inputs gave the expected outputs, though this test suite would also have to be maintained and update in line with the workbook and the dictionary.

5.3 Database

The database is not implemented as any specific database, instead the application uses the SQLAlchemy toolkit [27] (specifically the Flask-SQLAlchemy extension [28]), which is an Object-Relational-Mapper that allows the application to be developed without regard for the database type, and then later connect a suitable SQL database to it. In development an SQLite database was used for testing purposes.

A lot of the data which is stored in the database exists in the application in the form of JSON data. Because of this, a NoSQL database might have been a better choice in hindsight, since these are often specifically designed to deal with JSON data.

5.4 Front-end

5.4.1 Framework

The front-end of the application is built using the popular JavaScript library React [12] with support from the Material-UI framework [29], which offers many ready-made UI components.

React was chosen for several reasons, the most important being that it does a good job at creating responsive interfaces with a great deal of interactivity, for example in the form of interactive dynamic data charts, which were deemed to be of importance for the KPI tool. In addition, React is ubiquitous in web development, and is used by companies such as Facebook, Netflix, BBC, Instagram, and many more. Its widespread use means there is a large online developer community which can be relied upon for support.

React was also used for the student project described in Section 2.1.5, which shows that it is a suitable framework for building an application that deals with KPI management and visualization. Lastly, React was partly chosen because another master student who was working concurrently on developing a related web tool for the ZEN Research Centre had chosen React/Material-UI, and developing them using the same libraries and frameworks could potentially help give them similar look and feel if there ever were to be a desire to integrate these two tools more closely in the future.

A JavaScript framework was chosen because it is the only practical way to get the sort of user experience that a modern website requires, with the need for interactive data charts being one clear example of something requiring JavaScript. It is possible to create a user interface using only the back-end framework Flask, but it would not have been able to fulfill the necessary requirements of the application.

There are many different JavaScript frameworks available for front-end development, such as Angular and Vue.js, to name just two others. However, there are no huge advantages or disadvantages between these frameworks and React, and based on the reasons already discussed in this section, React ended up being the eventual choice. There is also the fact that, like with the choice of back-end framework, React is the front-end framework that I personally am most familiar with, which means saving time due to not having to learn a new framework.

The front-end was developed using React version 17.0.1.

5.4.2 Structure

React applications are traditionally built up of parts known as components. These components are code blocks which are largely self-contained and can be inserted almost anywhere into the code structure. The component structure of the front-end can be seen in Figure 5.1. Of note is the component "ScenarioEditor" which is the part of the application where users enter, process, and visualize the KPI data for different scenarios. In other words, it is the part of the application where the majority of the functionality lies, and it is therefore unsurprising that the scenario editor appears to be the most complex part of the front-end.

5.4.3 Data Flow

Many developers will choose to use a state manager like Redux or MobX to store state data when developing React applications. These state managers can be called from anywhere in the application to set or read the state. It is, however, not mandatory to use a state manager, as React is perfectly able to manage state on its own. The way React does this is by storing state in a component, and then passing the state up or down the component hierarchy to other components who can then

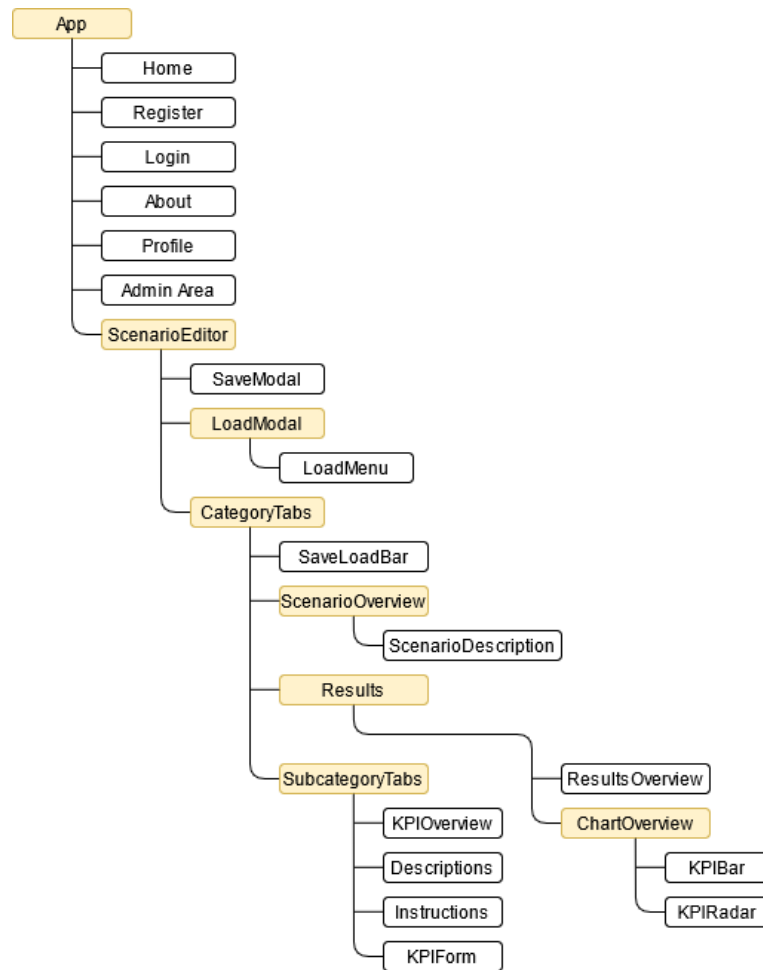


Figure 5.1: Diagram of the components in the React front-end. Components that have child components are marked in yellow.

read or update the state.

There are advantages and disadvantages to both approaches. Using a state manager means that state can easily be accessed from the state manager by any component which requires it, without having to pass the state through the component hierarchy. This means that the components are less interconnected, which in theory increases modifiability as it should be easier to restructure the code by moving components around. Conversely, using React's built-in state management makes it much easier for a developer to follow the state through the component hierarchy and gives a clearer overview of where states originate from and where they are used, which should theoretically also have a positive impact on the modifiability. Using the built-in system for state management also means that there is one fewer external dependency to manage.

Prioritizing a clear view of the data flow in the code, it was decided that the built-in state management system should be used. The state management is illustrated in Figure 5.2, which shows in which components specific states are stored, as well as showing in cursive which other components access these states. This figure only shows state that is directly relevant to the scenario data; other minor parts of the state which is used to ensure correct behavior of the interface are not included. As can be seen, the data is stored in the high-level component "ScenarioEditor". This ensures that state is not reset as the user navigates around the editor, for example preserving the results that have been calculated, and the charts. The data will be lost if the user navigates away from the scenario editor to another part of the application without saving, but this can be changed by storing the state one level up in the "App" component, at the top of the component hierarchy.

5.4.4 Interface

This section will go into detail in describing the user interface of the application. First will be a short introduction, followed by detailed descriptions of the different parts.

The interface design is inspired by the product developed in the student project described in Section 2.1.5. The interface contains simple functionality for registering, logging in, and logging out, and gives access to the scenario editor. The scenario editor is, as mentioned previously, the main part of the application and is the part that deals with everything related to KPI data such as entry, storage, processing, and visualization. A KPI scenario is divided into several KPI categories, which themselves are divided into several KPI subcategories, for a total of 30 subcategories in a scenario. In an effort to make this manageable, the scenario editor is set up in a nested tab structure, with an outer vertical tab bar along the left side and an inner horizontal tab bar along the top which manages access to the subcategories. This tab system will be further explained later. The scenario editor also

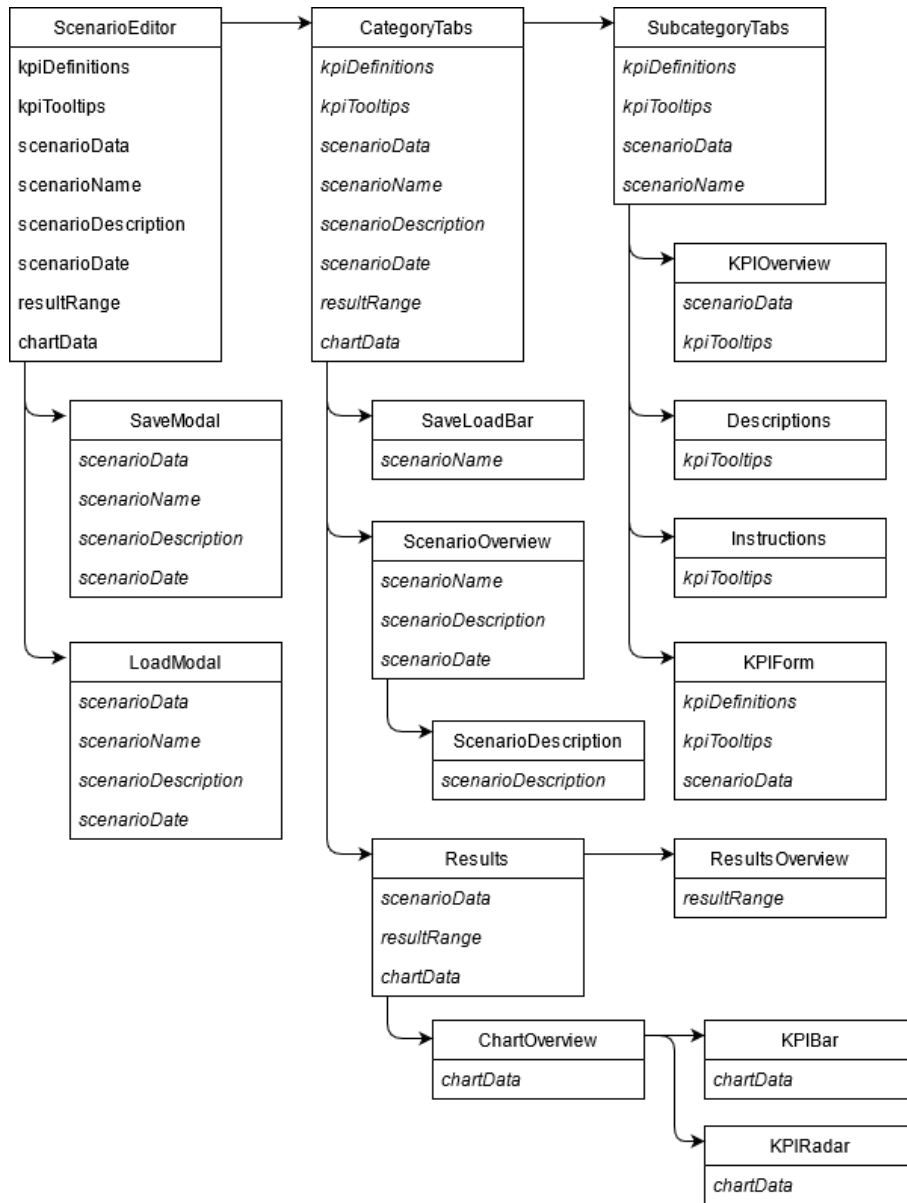


Figure 5.2: State management in the scenario editor.

contains an option for viewing results relating the KPIs and for saving and loading scenarios. Following this section is a series of figures describing the different parts of the interface.

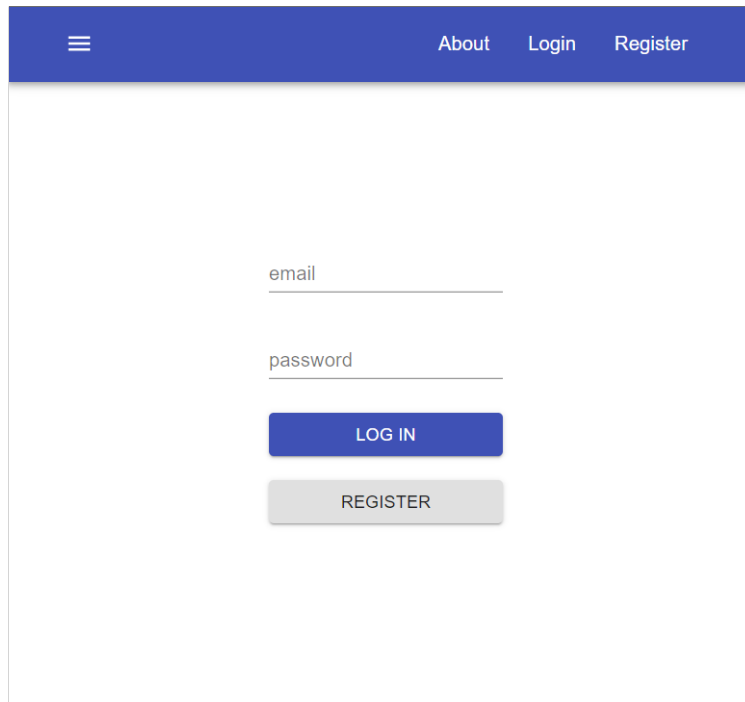
The image shows a web application's login screen. At the top, there is a dark blue navigation bar with a white hamburger menu icon on the left and the text 'About', 'Login', and 'Register' on the right. The main content area is white and contains two text input fields: the first is labeled 'email' and the second is labeled 'password'. Below these fields are two buttons: a blue button with the text 'LOG IN' and a light gray button with the text 'REGISTER'.

Figure 5.3: The login screen for the application.

Registration and Login

Figure 5.3 shows the login screen, with the "log in" button and the "register" button which takes the user to the registration screen. The registration screen looks very similar, with the same field for email and password, but with buttons for registering a new account and for going to the login screen. If the user tries to register or log in with incorrect credentials (log in with wrong password or register an email that is already registered), their attempt will fail and an error message will appear telling the user why the action they are trying to perform is failing.

Currently, the user's email is not used for anything, but Flask does have an extension for interacting with emails, so in the future it would be possible to include features that sends the user emails if that is desirable.

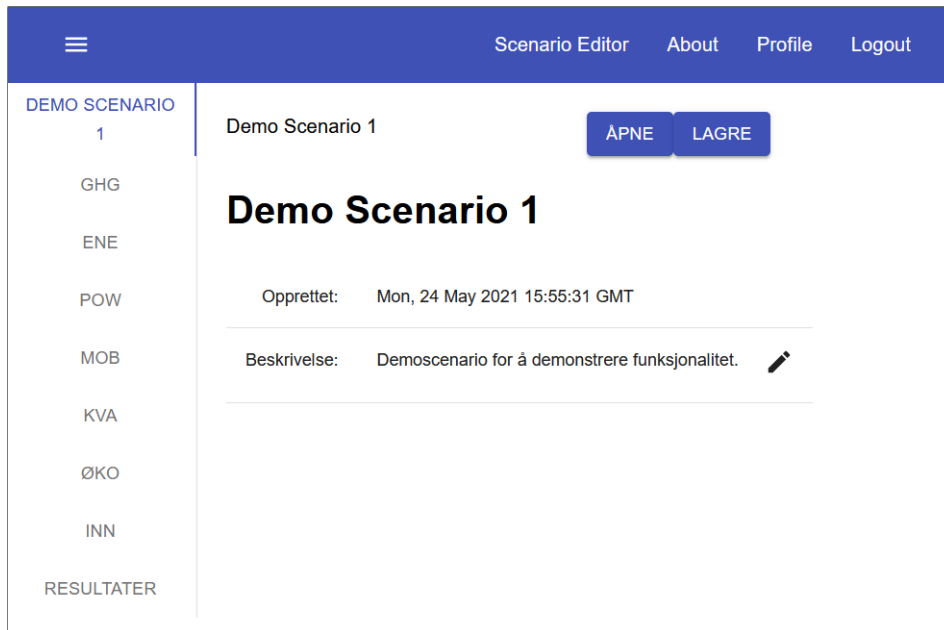


Figure 5.4: The front page for a scenario in the scenario editor.

Scenario Editor - Front Page

Figure 5.4 shows a scenario's front page. This is the screen which contains the main information for the scenario, such as the name, the date the scenario was created, and a short user-created description for the scenario. The front page also contains the buttons for saving and loading a scenario.

On the left side is outer tab bar of the nested tab structure. This contains tabs for accessing the scenario front page, the different KPI categories, and the scenario results. In the figure, the scenario front page tab is currently the tab that is active.

KPI	Navn	Strategisk planleggingsfase	Implementeringsfase	Bruksfase
2.1	Energibehov	0 / 5	0 / 5	0 / 5
2.2	Lvert og eksportert energi	0 / 7	0 / 7	0 / 7
2.3	Egenforbruk og egenproduksjon	0 / 3	0 / 3	0 / 3

Figure 5.5: Picture of the scenario editor in the KPI tool.

Scenario Editor - KPI Overview

Figure 5.5 shows the overview page of a KPI category, which lists all the KPI subcategories in that category, their names, and how many points the user has achieved for each phase of each subcategory. Along the top is the inner tab bar of the nested tab structure, which contains the tab for the KPI overview and the tabs for the KPI subcategories.

The screenshot shows a web application interface for a KPI Management Application. The top navigation bar includes 'Scenario Editor', 'About', 'Profile', and 'Logout'. The main content area is titled 'DEMO SCENARIO' and shows a list of KPI categories on the left: 1, GHG, ENE, POW, MOB, KVA, ØKO, INN, and RESULTATER. The 'Energibehov' subcategory is selected, and the 'OVERSIKT' tab is active. The subcategory details include a description and a calculation method. The calculation method is 'Specific net energy demand [kWh/m2year]'. The user can enter a 'Reference value' and a 'Strategic planning phase' for both 'Specific net energy demand' and 'Sum årlig energibruk energy use yearly [kWh/år]'. The 'Implementation phase' and 'Use phase' are also visible. The 'Poeng (maks 5)' section shows three dropdown menus for 'Strategic planning phase', 'Implementation phase', and 'Use phase'.

Figure 5.6: Picture of a KPI subcategory tab.

Scenario Editor - KPI Subcategory

Figure 5.6 shows the screen for a KPI subcategory. Near the top are two accordion elements which contain a more detailed description of the subcategory, and instructions for how to calculate points based on the KPI values the user provides. In the middle are fields for storing the KPI values. At the bottom are the fields where the user enters the points they have calculated for the subcategory. The lack of automation in the calculation of the points is an obvious issue which came up during user testing and which will be discussed further in Chapter 7.

One notable thing shown on this screen is how the names of the KPIs are in both Norwegian and English. This is a result of how these names are obtained, as they are datamined directly from the Excel version of the tool. The Excel workbook contains both the Norwegian and English name stored in the same cell, and so both are returned when querying for the name.

Kategori Category	Ambisjonsnivå Ambition level	Strategisk planleggingsfase Strategic planning phase	Implementeringsfase Implementation phase	Bruksfase Use phase	Poeng tilgjengelig Points available	Poengsum (%) Score (%) Strategisk planleggingsfase Strategic planning phase
GHG	50	7	7	7	33	21
ENE	50	12	12	12	21	56
POW	50	4	4	4	17	24
MOB	70	6	6	6	19	32
KVA	60	8	8	8	15	53
ØKO	70	10	10	10	15	66
INN	50	2	2	2	14	15
SUM	58	42	42	42	100	100

Figure 5.7: The application's screen for showing results.

Results - Overview

Figure 5.7 shows the overview screen for a scenario's calculated results. Near the top is the button for calculating the results. This takes all the point values that the user has entered and sends them to the Excel KPI tool, which calculates the results which are then sent back to the application and displayed to the user.

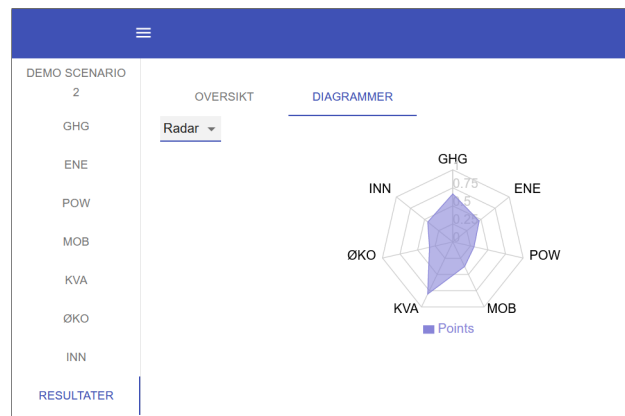


Figure 5.8: Radar chart visualization of a scenario's calculated results.

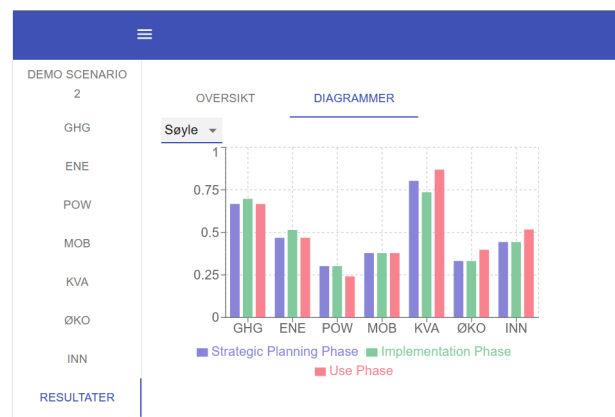


Figure 5.9: Bar chart visualization of a scenario's calculated results.

Results - Charts

In addition to an overview of the results, the results section also contains an option for visualizing the results. Figure 5.8 and Figure 5.9 show two possible chart visualizations.

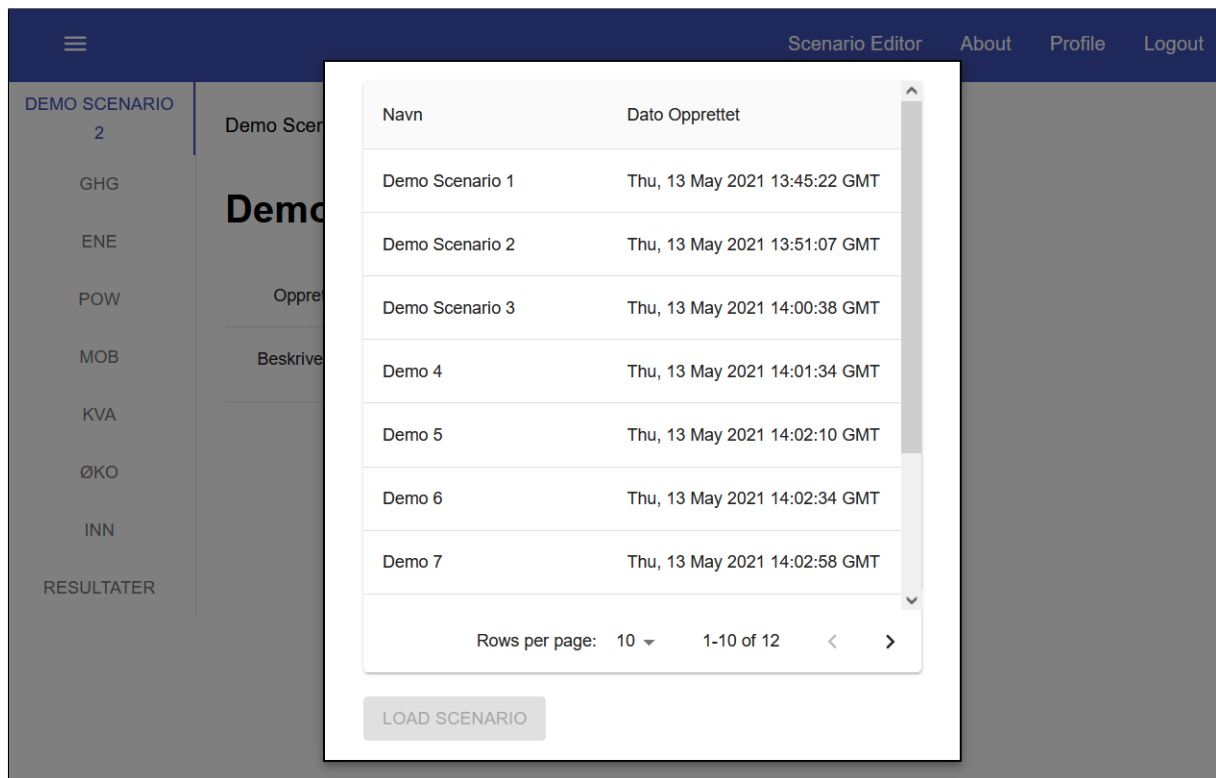


Figure 5.10: The application's menu for loading saved scenarios.

Saving and Loading

On the scenario front page are buttons for saving and loading a scenario. Clicking these buttons will bring up a window with either save or load functionality, Figure 5.10 for example shows the menu for loading a previously stored scenario, with the scenario front page visible in the background.

5.4.5 Scenario Editor Construction

The KPI categories and subcategories shown in the scenario editor are not hard-coded into the application, but are instead dynamically constructed when the application loads, based on the Excel mapping described in Section 5.2.2. This makes it easy for a developer to add or remove categories and subcategories, as it just requires adding a new reference to the Excel KPI tool in the mapping.

Much of the textual information in the scenario editor comes directly from the Excel KPI tool. The location of this information is specified in the Excel mapping. The application contains an option to initiate a scan of the Excel tool, which reads the relevant information and stores it in the application database. Reading from the database is much quicker than reading from the Excel tool, so the caching of the information is an obvious choice for performance reasons. This scanning feature has the advantage that it is not necessary to keep track of any changes to the textual information in the Excel tool to implement them in the web application, instead it is simply a matter of initiating a new scan and the information will be automatically updated in the database.

One issue with this approach is that automatic scanning of the Excel tool was not something the developers had in mind when creating it, and so certain problems appear. One example which was mentioned previously is shown in Figure 5.6 where the names of the KPIs are shown in both Norwegian and English. In the Excel tool, these names are stored in the same cell so when the tool is scanned, both language versions are included.

Chapter 6

Evaluation & Results

This chapter describes the different efforts that were made to evaluate the application developed in this project, and the results of those evaluations.

6.1 Continuous Self-Evaluation

During development, the different aspects of the application were continuously evaluated by myself as a way to ensure that the usability would be as high as possible. The features were evaluated against Nielsen's Usability Heuristics [30], which are ten points one should keep in mind that help create user friendly systems:

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

This method of evaluation was not intended to be a method of documenting the quality of the application, but rather served as a way to quickly and easily detected areas that needed improvement. Self-evaluation by heuristics cannot replace evaluation by means of user testing as user testing is vital to cover blind spots the developer may not be aware that they have, but its simplicity means that it nicely complements user testing.

6.2 Iterative User Testing

User testing was done with researchers from the ZEN Research Centre, who are the expected user group for the KPI tool. The tests were done in an iterative manner: first, a user test would be performed which would highlight issues with the application. These issues were then addressed in the next development cycle, after which a new user test was performed to help determine if the issues were fixed and to highlight new issues. This iterative process can be described as a type of agile alpha testing, as per Sommerville [31, p. 249]. Ideally, alpha testing should be followed up by more structured testing once the product reaches a more finalized stage. This was not done, partly due to the time constraints described in Section 3.1, and partly because the Excel version of the tool is itself not near completion, which means that performing tests meant for a completed product would be premature.

Three user tests were done. The first was an informal test session with the primary contact person from the ZEN Research Centre to gain feedback on the status and direction, while the two others were more structured tests with researchers who were familiar with the KPI definitions. The latter two tests had the following format:

1. A think-out-loud session where the user is instructed to use the tool based on likely usage scenarios and describe their thoughts and actions while doing so.
2. Initial feedback based on their immediate thoughts after using the tool.
3. A survey for the user to fill out based on the Systems Usability Scale [20].
4. A longer semi-structured interview based on the answers given in the survey.

All three tests were done through digital video calls due to pandemic precautions. The users' actions were observed (through screen-sharing) while they performed different tasks in the tool, and notes were taken whenever a user ran into an issue, made a valuable suggestion, or anything else that was noteworthy happened. These tests were a type of expert evaluation, since all the users were researchers with expertise in areas directly relating to the KPI tool.

It would have been preferable to run tests with more users, but this was not possible due to scheduling conflicts, as mentioned earlier. It might not be necessary to test with very many more users though, Nielsen [32] argues that you will discover the majority of problems with only five users, and after that you will just keep rediscovering the same problems. Others have argued that five users is not actually enough [33], though in that case the systems being tested were relatively complex e-commerce websites. Based on this, it is safe to say that getting at least a few more users to test the application would have been ideal.

First User Test

This test was an informal test with the contact person from the ZEN Research Centre, who is also one of the people responsible for developing the Excel KPI tool. Several issues of varying size were discovered, and different desirable features were discussed and planned. Initial feedback was positive, with the user expressing satisfaction with using a web-based interface, compared to the more complex and unwieldy Excel interface.

Key issues that were discovered during the test:

- Lack of explanatory text, like names of sections and how to calculate points.
- There should be an option to change the display language.
- It should be possible to share a scenario with other users.
- Visualization of the results in graphs must be implemented.
- There should be some kind of overview page for each KPI category.

After this test, several changes were made. More explanatory text was added, graph visualization was implemented, an overview page was added to each KPI category which kept track of how many points the user had awarded to each KPI subcategory, in addition to other minor changes. A system for language localization has been partially implemented but not completed. A system for sharing scenarios with other users does not yet exist, though theoretically there should be no problems with implementing it (other than the time and effort required).

Second User Test

This was the first of the structured tests described previously. The user was another researcher from the ZEN Research Centre who had also taken part in developing the Excel KPI tool, which meant that the feedback they gave were rooted in an understanding in how the KPI tool should function. As with the previous test, several issues were discovered and more features were suggested. The user positively commented on the experience of using the application, noting that the nested tab structure made it easy to navigate the many sections of the tool and that entering KPI data in the web application felt more user friendly than in the Excel workbook. The user also commented on how they felt that few people used the Excel KPI tool and that having it available as a web application would increase its usage, which again underlines the possible benefits of this work.

Key issues that were discovered during the test:

- Still more explanatory text is needed. Little things like including names in addition to numbers on overviews, showing maximum number of points in addition to current number of points, clarifying what units the KPIs are, etc.
- There is an expectation that the tool should calculate points automatically for each KPI subcategory after the user has put in the KPI values, instead of

having to calculate them manually.

- There should be a main overview page for the entire scenario.
- There was confusion on whether you had to save each separate subcategory, since the save and load buttons for the scenario were constantly visible.

Again, changes were made after the test. Text describing each KPI subcategory was added in expandable accordion elements. A main overview page for the scenario with the name, time of creation, and user-created description was added, and the save and load buttons were moved to this main overview page. By limiting access to the save and load functionality to only this page, the user should no longer have to worry about whether they have to constantly save.

Nothing was done about the issue of having to manually calculate the points, as it is an issue deeply connected to the overall design choice of the application. This will be discussed more in detail in Chapter 7.

Third User Test

This was the second of the structured tests described previously. This time the user was a researcher who was involved with developing the ZEN definitions, but not involved with the development of the Excel KPI tool, which gave important insight into how a user with no pre-existing mental model of the tool perceived the application.

The lack of mental model was apparent, as this user had more difficulty navigating the tool and utilizing it successfully than the previous users had had. Nevertheless, their proficiency increased through the duration of the test, and the user was confident that they would be able to quickly learn how the application worked and would have little problem using it in the future. They also professed approval of having the tool available as a web application instead of as an Excel workbook.

Key issues that were discovered during the test:

- Again there was an expectation that the tool should automatically calculate the points awarded for each KPI subcategory on behalf of the user.
- There should exist a tutorial on how to use the tool that user can choose to go through.
- There should be some export function so the user can export their KPI data for use in external reports.
- ZEN researchers should have access to the KPI data which users store in the system, so that they can include it in their research.

The last item on the list raises some privacy concerns, though it should not be an issue that ZEN researchers have access to the data if those same researchers are also the only users of the tool, as is currently the intention.

This last test was completed relatively late in the project and as such there was not enough time to make improvements based on it, but it still provided valuable feedback which can be used in future work.

6.3 Scalability

Scalability is not one of the main focuses of the application, so it naturally follows that it has not been a focus during evaluation either. Still, it can be useful to know what the theoretical prospects for scalability are, for future development and usage purposes.

Both the Flask back-end and the React front-end are well-established and widely used web frameworks, their scalability can be assumed to be more than satisfactory for this application and will not be examined further. Of much greater interest is the scalability of the system that deals with integrating the Excel KPI tool. The documentation for Microsoft Graph claims that there is a limit of 1500 requests to an Excel workbook per 10 seconds [34]. As noted in Section 5.1.2, nearly one hundred requests have to be made when the user wishes to perform the calculations for their scenario. This means that, in theory, the application can only handle fifteen users requesting calculations in the same ten seconds. Most likely, this will not be a problem since users are not expected to calculate the results often and the application is not expected to get a lot of simultaneous traffic, but it could prove to be a very real bottleneck if the scope of the application's use is ever expanded, and is therefore something which is necessary to be aware of.

Chapter 7

Discussion

This chapter will consider the work done in this project and how it serves to answer the research questions posed in the beginning of the thesis. It will also discuss the limitations of the project, and reflect on whether anything should have been done differently.

7.1 Web-Based Implementation of KPI Tool

Since RQ1 and RQ2 are somewhat similar in theme, this section will address them both and consider to what extent the work done in the thesis has managed to answer them.

- **RQ1:** *How can the KPI tool be made more accessible to users?*

Answering RQ1 is fairly straightforward, a good way to make the KPI tool more accessible is to make it into a web application. A web application stands in stark contrast to an Excel-based application when it comes to ease of access. The latter has to be distributed (and care must be taken to ensure that it is the correct version which is being distributed) to the users, who themselves have to have a system which is capable of running an Excel workbook. On the other hand, the former is centrally managed and can be accessed by anyone with an internet connection and a web browser, which makes it far easier to access.

This answer is supported by the previously developed front-end concept described in Section 2.1.5, which sufficiently demonstrates that a web application is a viable way to implement the KPI tool. The actual web application developed in this project further supports this conclusion, with the fact that the application was able to deliver all the same features of the Excel tool and the positive feedback from the ZEN researchers during evaluation being clear points in its favor.

- **RQ2:** *How can the usability of the current Excel-based ZEN KPI tool be in-*

creased?

Answering RQ2 is slightly more complicated than RQ1. Based on the issues with the Excel tool identified in Section 2.2.2 and the usability problems with Excel as a whole discussed in Section 2.2.4, it was clear that using a different platform would be one way in which the usability could be increased. Again, the front-end concept described in Section 2.1.5, as well as the different options for integrating the pre-existing Excel-based KPI tool considered in Section 4.2.1, led to the conclusion that a viable alternative was to build a web application using React.

Another way to increase usability is to increase the functionality of the tool to cover more of the user's needs. Table 4.1 shows the proposed new features, though only the first six were successfully implemented. Most prominent of these are the options to save and load several different KPI scenarios, and to easily visualize the data in charts.

There are certain disadvantages to using a web application over an Excel tool. Developing an Excel tool has a lower barrier to entry than developing a web application, and likely requires less work. A centralized web application also requires maintenance and upkeep to keep running, whereas an Excel tool is just a file that can be passed around to different users. These increases in development work and maintenance are the prices one pay for the increase in usability.

The evaluation of the system went very well. All three users who participated in the user testing were positive to the usability improvements that the new application offered. This feedback is especially promising given that all the users were ZEN researchers directly involved with projects relating to the KPIs, and so their opinions can be expected to be particularly significant regarding this matter.

Based on this, RQ2 can be answered by concluding that an architecture consisting of a full web-stack with a React front-end and a connection to the Excel KPI tool through Microsoft Graph, with possibilities to develop broader functionality based on user needs, is a viable way to increase the usability of the KPI tool. There are some issues with the integration of the Excel KPI tool though, which will be discussed in the next section.

7.2 Excel Integration

This section focuses on the specific aspect of the project which dealt with integrating the Excel KPI tool into the web application, and what experience was gained as a result.

- **RQ3:** *Is incorporating an Excel workbook as an active component a possible and viable design choice for a web application?*

First of all, it is clearly possible to incorporate an Excel workbook as an active component of a web application, as this work has shown. It can also be said to be viable due to the successful implementation and the positive feedback during evaluation, but there are both advantages and disadvantages that must be considered.

There are several advantages. Firstly, it enables a developer of a web-application to start building directly "on top" of the already existing Excel-based tool, instead of having to re-implement it all from scratch. Secondly, if the people responsible for developing the Excel-based tool do not have sufficient web-development experience to change over to developing a web application (as was the case in this scenario), then this approach can enable them to continue development on the Excel-based tool. Thirdly, in the case of development of the Excel-based tool continuing in parallel with the web-based application, the integration means that it is not necessary to track every little change in the former to manually implement them in the latter, instead an automatic scan can take care of it.

On the other hand, there are several disadvantages. In order for the web application to successfully connect to the Excel-based tool, a strict mapping has to be defined so that the web application knows where all the relevant cells are located, and this mapping has to be manually kept up to date. If the layout of the Excel-based tool changes, it has a chance of ruining the connection with the web application. This can be alleviated by being aware that the Excel-based tool is going to be connected to a web application from the beginning of the development process, and thereby adhering to a set of guidelines which determines a uniform and predictable layout of the Excel-based tool.

Another issue is the matter of scalability as described in Section 6.3. Due to the request throttling, an application where user actions generate such requests is simply not scalable. This design can certainly work for applications where user traffic is expected to remain low, or in cases where the number of requests does not scale with the number of users, but not otherwise.

Lastly, there is the matter of the web-based tool not automatically calculating points after the user has entered the KPI values. This was commented on by two of the three users who tested the application. It is understandable for users to expect that the tool should be able to handle this, and it would certainly be possible to implement such a feature. The issue is that the points are calculated based on different criteria for each KPI subcategory, so automation would have to be implemented individually for each subcategory, instead of devising a general approach. At this point, the Excel integration starts losing its advantages and may start acting as a constraint, and the work needed to work around that constraint may exceed the amount of work needed to just re-implement the tool as a web application instead.

Thus, the answer to RQ3 is that yes, it is possible, and it can be viable, depending on the web application in question, but there are certainly disadvantages

one must be aware of. In many cases, it will not be worth it to bother with the integration and instead it will be better just to completely re-implement the features in the web application. In this specific case I would argue that the advantages outweigh the disadvantages.

7.3 Limitations & Issues

There were several limitations and issues with this project which are worth talking about. One major issue which affected every other aspect of the project is that it was relatively slow to get started. This was largely due to difficulties with getting in touch with the relevant people from the ZEN Research Centre, as the person I was supposed to meet with was unavailable due to personal reasons. It took several months for a replacement to be found, at which point the actual work on the project could start. This delay of course meant that there was less time available for the project than I would have preferred.

Having more time would also have made it possible to implement more features, specifically functional requirement 7 through 10 in Table 4.1. It would have been interesting to see how these features could have increased the usability and usefulness of the tool.

As mentioned in Chapter 6, more user tests would have been desirable to gain a more well-founded understanding of the issues that users may encounter when using the application. Getting qualified users to test with was a problem because of the criteria that the users should be ZEN researchers, who all appeared to be very busy during the time period when the tests were being done. The fact that the project was already suffering from time issues didn't help.

One last issue is that I, the author of this work, do not consider myself very good at front-end design and development. It was not clear at the beginning of the project that it would have such a large focus on front-end development, but that is the way it turned out. If nothing else, it has certainly been a valuable learning experience.

Chapter 8

Conclusion & Future Work

8.1 Conclusion

This thesis has examined how to create an ICT architecture for a Key Performance Indicator management application. This has been done by developing a concept and a prototype of such an application, based on the already existing Excel-based KPI tool developed by the ZEN Research Centre.

It was shown that the usability of the KPI tool could be increased and that it could be made more accessible to users by developing a full-stack web application, built using Flask and React, and connecting this web application and the Excel-based KPI tool using Microsoft Graph. This approach lets the web application function as a user interface for the Excel version of the KPI tool, thus providing a more user friendly experience. The web application also has the possibility of including additional features which an Excel tool is unable to provide.

Further, the advantages and disadvantages of connecting a web application to an Excel tool through Microsoft Graph were investigated. It was found that this was possible and also viable for certain types of applications, but that there are issues with scalability and that this type of Excel-integration can also be constraining.

8.2 Future Work

With regards to the application developed as part of this project, there is certainly more to be done. There are still more features listed in Table 4.1 which should be implemented, and there are other issues which need fixing, such as the authentication problems mentioned in Section 5.1.2. More user tests should also be performed, as these will likely uncover more issues with the application that need to be improved, and might also lead to new feature suggestions that the

application can benefit from.

With regards to research into Microsoft Graph functionality, it would be interesting to see a project which attempts to build a web application connected to an Excel tool, but in which the Excel tool was designed with this specific purpose in mind. This would give a better understanding of the use cases for this type of architecture.

Bibliography

- [1] T. Abergel, B. Dean, J. Dulac, and I. Hamilton, “2018 global status report: Towards a zero-emission, efficient, and resilient buildings and construction sector,” *Global Alliance for Buildings and Construction*. [https://www.worldgbc.org/sites/default/files/2018% 20GlobalABC% 20Global% 20Status% 20Report. pdf](https://www.worldgbc.org/sites/default/files/2018%20GlobalABC%20Global%20Status%20Report.pdf), 2018.
- [2] (2021). FME ZEN, [Online]. Available: <https://fmezen.no/>. (accessed: 09.05.2021).
- [3] M. K. Wiik, S. Fufa, J. Krogstie, D. Ahlers, A. Wyckmans, P. Driscoll, H. Brattebø, and A. Gustavsen, “Zero emission neighbourhoods in smart cities: Definition, key performance indicators and assessment criteria,” Tech. rep., Research Center on ZEN in Smart Cities, Tech. Rep., 2018.
- [4] M. K. Wiik, S. M. Fufa, I. Andresen, H. Brattebø, and A. Gustavsen, “A norwegian zero emission neighbourhood (ZEN) definition and a ZEN key performance indicator (KPI) tool,” in *IOP Conference Series: Earth and Environmental Science*, IOP Publishing, vol. 352, 2019, p. 012 030.
- [5] C. Chambers and C. Scaffidi, “Struggling to excel: A field study of challenges faced by spreadsheet users,” in *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, IEEE, 2010, pp. 187–194.
- [6] P. T. Kortum and A. Bangor, “Usability ratings for everyday products measured with the system usability scale,” *International Journal of Human-Computer Interaction*, vol. 29, no. 2, pp. 67–76, 2013.
- [7] S. M. Fufa, R. D. Schlanbusch, K. Sørnes, M. R. Inman, and I. Andresen, *A Norwegian ZEB definition guideline*. SINTEF Academic Press, 2016.
- [8] (2021). Our partners, [Online]. Available: <https://fmezen.no/partners/>. (accessed: 06.06.2021).
- [9] A. Bremvåg, A. G. Hestnes, and A. Gustavsen, “Annual report 2019,” 2020.
- [10] S. A. Petersen and J. Krogstie, “Intermediate ICT coordination and harmonization guidelines,” Research Center on ZEN in Smart Cities, Tech. Rep., 2019, (Unpublished).
- [11] M. Sundnes, J. B. Giske, K. H. Hveding, N. C. Danielsen, A. B. Johnsen, and B. M. V. Iversen, *Zero-Emission Neighborhoods*. NTNU, 2019.

- [12] (2021). React, [Online]. Available: <https://reactjs.org/>. (accessed: 08.06.2021).
- [13] *Excel*, eng, 2018.
- [14] B. F. Nielsen, E. Juhasz-Nagy, C. Lindkvist, A. Wyckmans, I. Andresen, and D. Baer, "Planning instruments for smart energy communities," *PI-SEC Report Nr. XXXX. Trondheim*, 2017.
- [15] H. T. Walnum, K. Sørnes, M. Mysen, Å. L. Sørensen, and A.-J. Almås, "Preliminary toolkit for goals and kpis," 2017.
- [16] A. A. M. H. Wiberg and D. Baer, "Zen toolbox: First concept for the zen toolbox for use in the development of zero emission neighbourhoods," 2019.
- [17] M. Wiik, R. Schlanbusch, A. Wiberg, and T. Kristjansdottir, "Zeb tool manual," *User Guide. Version 1. External memo. The Research Centre for Zero Emission Buildings*, 2017.
- [18] (2021). Overview of Microsoft Graph, [Online]. Available: <https://docs.microsoft.com/en-us/graph/overview>. (accessed: 12.06.2021).
- [19] "Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts," International Organization for Standardization, Standard, Mar. 2018.
- [20] J. Brooke, "Sus: A "quick and dirty" usability scale," *Usability evaluation in industry*, vol. 189, 1996.
- [21] P. Legris, J. Ingham, and P. Collette, "Why do people use information technology? a critical review of the technology acceptance model," *Information & management*, vol. 40, no. 3, pp. 191–204, 2003.
- [22] A. Hevner and S. Chatterjee, "Design science research in information systems," in *Design research in information systems*, Springer, 2010, pp. 9–22.
- [23] A. R. Hevner, "A three cycle view of design science research," *Scandinavian journal of information systems*, vol. 19, no. 2, p. 4, 2007.
- [24] (2021). Flask, [Online]. Available: <https://flask.palletsprojects.com/en/2.0.x/>. (accessed: 06.06.2021).
- [25] (2021). Django, [Online]. Available: <https://www.djangoproject.com/>. (accessed: 07.06.2021).
- [26] K. Lei, Y. Ma, and Z. Tan, "Performance comparison and evaluation of web development technologies in php, python, and node.js," in *2014 IEEE 17th international conference on computational science and engineering*, IEEE, 2014, pp. 661–668.
- [27] (2021). SQLAlchemy, [Online]. Available: <https://www.sqlalchemy.org/>. (accessed: 07.06.2021).
- [28] (2021). Flask-SQLAlchemy, [Online]. Available: <https://flask-sqlalchemy.palletsprojects.com/en/2.x/>. (accessed: 07.06.2021).

- [29] (2021). Material-UI, [Online]. Available: <https://material-ui.com/>. (accessed: 08.06.2021).
- [30] J. Nielsen. (1994). 10 usability heuristics for user interface design, [Online]. Available: <https://www.nngroup.com/articles/ten-usability-heuristics/>.
- [31] I. Sommerville, *Software engineering*, 2016.
- [32] J. Nielsen. (2000). Why you only need to test with 5 users, [Online]. Available: <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>. (accessed: 21.06.2021).
- [33] J. Spool and W. Schroeder, "Testing web sites: Five users is nowhere near enough," in *CHI'01 extended abstracts on Human factors in computing systems*, 2001, pp. 285–286.
- [34] (2021). Microsoft Graph throttling guidance, [Online]. Available: <https://docs.microsoft.com/en-us/graph/throttling>. (accessed: 09.06.2021).

Appendix A

Source Code

The source code for the application developed in this project can be found online at <https://github.com/jorgenhlehne/kpi-tool>

