Aalto-yliopisto
Insinööritieteiden korkeakoulu

Emma Syrjänen

Web-based Editor for Digital Twin Diagrams

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Technology.

Espoo, 19.07.2021

Supervisor: Professor Jan Blech & Lecturer Seppo Sierla

Advisor: Lecturer Seppo Sierla

Aalto-yliopisto
Aalto-universitetet
Aalto University

Aalto University, P.O. BOX 11000, 00076
AALTO
www.aalto.fi

Abstract of master's thesis

Author Emma Syrjänen

Title of thesis Web-based Editor for Digital Twin Diagrams

Master programme Mechanical Engineering **Code** MEC

Thesis supervisor Professor Jan Blech, Lecturer Seppo Sierla

Thesis advisor(s) Lecturer Seppo Sierla

Date 19.07.2021 **Number of pages** 53 **Language** English

Abstract

Industry projects can involve several parties from different companies, more than one client, and many manufacturers. To minimize the possible errors due to several parties working together, a digital twin concept has been created. A digital twin is a virtual presentation of an object or a process. Through digital twins, facilities performance, production flow, and cost structure can be more accurately understood and managed. Piping diagrams and instrumentation diagrams (PI&Ds) are the most common schematic models used to describe plant processes and functionality. P&ID describes an engineering process flow along with the related process equipment. In this study, requirements for PI&Ds were created by studying the literature.

This thesis provides proof of concept of a web-based editor that can create and handle piping and instrumentation diagrams and can be used as a basis for future software developments. A browser-based software was implemented as part of the thesis. As a result, implementation of the software proves that a piping and instrumentation diagram software can be developed as a web-based and it gives also guidelines for future developments.

Keywords Digital twin, piping and instrumentation diagram, web-based software

Aalto-yliopisto
Aalto-universitetet
Aalto University

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi

Diplomityön tiivistelmä

Tekijä Emma Syrjänen

Työn nimi Selainpohjainen editori digitaalisten kaksosten kaavioille

---

Maisteri ohjelma Konetekniikka **Koodi** MEC

Työn valvoja Professori Jan Blech, Lehtori Seppo Sierla

Työnohjaaja(t) Lehtori Seppo Sierla

Päivämäärä 19.07.2021 **Sivumäärä** 53 **Kieli** Englanti

Tiivistelmä

Teollisuuden projekteissa on mukana useita eri yrityksiä, asiakkaita ja toimittajia. Digitaalinen kaksonen on luotu vähentämään mahdollisia virheitä, jotka aiheutuvat näiden toimijoiden yhteistyöstä samassa projektissa. Digitaalinen kaksoskonsepti on virtuaalinen esitystapa, joka kuvaa objektia tai prosessia. Digitaalisten kaksosten avulla voidaan paremmin ymmärtää ja hallita laitosten prosessien suorituskykyä, tuotantovirtaa ja kustannusrakennetta. PI-kaaviot ovat yleisesti käytettyjä kaaviopiirustuksia, joilla kuvataan tehdasprosesseja ja niiden toimintaa. Näissä kaavioissa kuvataan tuotanto prosessivirtoja sekä niihin liittyviä prosessilaitteita. Tässä tutkimuksessa PI-kaavioille asetetut vaatimukset luotiin tutkimalla kirjallisuutta.

Tässä diplomityössä testattiin ja varmennettiin ohjelman toiminta, jonka avulla pystytään luomaan ja käsittelemään putkisto- ja instrumentointikaavioita ja jota voidaan käyttää mukana tulevissa ohjelmistokehityksissä. Diplomityön yhteydessä toteutettiin selainpohjainen sovellus. Toteutetulla sovelluksella voitiin osoittaa, että putkisto- ja instrumenttikaavio-ohjelmisto voidaan kehittää selainpohjaiseksi. Tämä myös antaa suunnan tulevalle kehitystyölle.

Avainsanat Digitaalinen kaksonen, PI-kaavio, selainpohjainen sovellus

---

Preface

Tämä diplomityö on tehty osaksi Doxie Oy yrityksen kehitystyötä. Työssä luotua sovellusta tullaan käyttämään pohjana Doxie Oy sovellusten kehityksessä.

Olisin halunnut henkilökohtaisesti kiittää erinomaista ohjaajaani Jan Blechia, joka valitettavasti menehtyi diplomityö prosessin aikana. Lämmin kiitos ohjaajalle Seppo Sierlalle, joka pystyi nopeasti jatkamaan Jan Blechin ohjausta.

Haluan kiittää kaikkia minun opiskeluaikanani vaikuttavia ja tukeneita henkilöitä. Iso kiitos perheelleni, joka on aina tukenut opiskeluissani. Kiitos siskoilleni, jotka ovat auttaneet pitkin opiskelu taivalta ja antaneet diplomi-insinööreinä ammatillista esimerkkiä. Kiitos vanhemmilleni, jotka ovat aina kannustaneet ja uskoneet minuun. Iso kiitos isälleni, joka antoi mahdollisuuden vaikeana korona-aikana tehdä tämä diplomityö ja toimi ohjaajana. Hän on myöskin esiintynyt vahvana esikuvana ura valinnassani.

Helsinki 19.07.2021

Emma Syrjänen

Table of Contents

Figures

Abbreviations

AJAX	Asynchronous JavaScript and XML
BFD	Block Flow Diagram
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CGI	Computer-Generated Imagery
CPI	Chemical Process Industries
CSS	Cascading Style Sheets
DEXPI	Data EXchange for the Process Industry
EPC	Engineering, Procurement and Construction
IoT	Internet of Things
JSON	JavaScript Object Notation
ML	Machine Learning
P&ID	Piping and instrumentation diagram
PFD	Process Flow Diagrams
PLM	Product Lifecycle Management
POC	Proof of Concept
PSK	PSK Standards Association
SQL	Structured Query Language
SVG	Scalable Vector Graphics
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	Extensible Markup Language

1 Introduction

1.1 Motivation

In today's highly competitive markets, the ambitions for shortening the time to market and increasing product development performance are required. Most of the chemical and process engineering projects usually involve several parties, for example, different companies, more than one client, and many manufacturers. Because of the constant increase in construction, the project schedules are shortened. These might cause a large number of errors or delays. When talking about large scale industries, these issues can be costly. For example, a dimensional error in the length of a pipe may cause significant problems in a plant project. Many participants have access to the same documents, which may lead the data to change or vanish. All of these possible errors can cause significant cost and problems.

Today's industries face the challenge of delivering more customized products in shorter time frames. Other challenges in plant engineering include globalization, engineering and procurement, construction (EPC) capacity supply and demand balance, environmental stewardship, and demographic challenges. Globalization delivers businesses and global markets to the same competition, which pressures the companies to cope with low costs and quality to maintain their status in the markets. The growing world population has created an increase in the overall demand for new and rebuild facilities. However, the margin of EPC capacity will be small in total facilities. The future challenge for EPC contractors is to be more innovative and cost-efficient. In today's society, environmental stewardship is being emphasized. This awareness puts pressure on industries to be more responsible and protect the environment. In addition, a significant challenge is to secure qualified technical staff. This challenge is the result of the number of essential demographics that are sifting. (ICT Challenges in the Plant Supply Chain, 2003)

The digital twin was found to be the best option to use in the industry for addressing these challenges. Digital twin concepts were first introduced in 2002, and they are increasing in relevance to systems engineering and model-based system engineering (Madni, Madni, and Lucero, 2019). A digital twin is a digital replica of a living or non-living physical entity. It represents a virtual concept of a physical assets, processes, people, places, systems, or devices that can be used for several purposes. A digital Twin enables better understanding and prediction of production machines' performance, production flow, and cost structure. (Lohtander et al., 2018)

The digital twin concept can be applied to engineering, procurement and construction. Also, it can be used in other purposes such as simulation, maintenance, and development. Simulation in a digital twin can reduce time and expenses for development and testing. In case of architectural and construction sector, the digital twin has arrived as an upgrade of building information modeling. (Hinduja, Kekkar, Chourasia and Chakrapani, 2020) Providing a more accurate version of the model, digital twin decreases the possibility for errors and therefore reduces the cost and time used for a project.

Digital twin assembles models from different industries to one whole entity. The most common method of representing plant process and functions using schematic model is piping and instrumentation diagrams (P&IDs). It describes the layout of an engineering process flow along with the related process equipment. (Rahul, Paliwal, Sharma and Vig, 2019)

These diagrams provide essential information for other disciplines and equipment suppliers. The PI&D is used through the designing and operation phases. Since it is used in the plant's entire process, it is developed by several engineering sectors such as process engineering, instrumentation and control, plot plant and piping, mechanical, heat ventilation, and air conditioning. (Toghraei, 2014) P&IDs consist of symbols that represent components, lines that function as a pipeline, tags that contains all of the critical information of the component, and many other sections. These diagrams present the structure of the plant. For this reason, the dimensions, distances, elevations, and layout of the parts are only abstracts.

Several standards define the description of a technical plant used in P&IDs. Standards are essential, especially when more than one party is involved working with a digital twin. Parties may use different software or have other operating systems, which can affect the data differences and interoperability.

Digital twin solutions typically consist of desktop applications. Even though a database can be based on a server or a cloud, most of the authoring tools such as P&ID software are still primarily implemented as desktop software. It is more convenient to develop web-based software instead of desktop software that is downloaded from the Web. Having connected to the internet means that the user has access to the software and is not dependent on the computer being used at that time. This means it is more accessible and secure than the conventional version. With this benefit, it gives a possibility to work from anywhere and anytime. In addition, laptops can be stolen, and not everyone keeps adequate backups of their data. Over the Internet, security patches and updates are always available. All data is assessable worldwide at any time, which is one of the key advantages of web-based applications. When software is based on the Web, it is always compatible with any computer, applicable to all operating systems and most used browsers. This web-based software usually has a login possibility that secures the files and data used in the software. This right also gives the benefits of sharing the software with other people. Since the data is continuously updated, it provides everyone working on the same project access to the latest data. Additionally, some of the users may have different access rights. These access rights are easy to regulate when the software is based on the Web. (Ghulam, Abad, Khadim and Amjad, 2007)

1.2 Scope and Research Methods

This thesis focuses on developing an editor of a web-based software that can create and modify PI&Ds. Thesis illustrates the digital twin and PI&D concept in detail. A prototype of software is presented in the thesis. The thesis gives an overview of the tools used for software development. The thesis also describes the software architecture and implementation used in the created software. The thesis focuses on one type of architecture. Two test cases that are implemented are then explained. In addition, the thesis includes defining the future steps.

The purpose of this thesis is to create a proof of concept of a digital twin solution. Research methods used in this thesis were literature review, an analysis of the competitors, a specification of the requirements, and implementation of a proof-of-concept.

1.3 Structure of the Thesis

This thesis is divided into six main chapters. Chapter 1 displays the introduction, scope and the structure of the thesis. Chapter 2 introduces the essential information to understand the topic of the thesis. The chapter starts with describing the background and general knowledge of the subject, and after that, the techniques required for the development of software are presented. Chapter 3 presents the software architecture and the requirements for the created software. Chapter 4 explains the implementation of the software. Chapter 5 discusses the result of the functionality of the developed software. Lastly, chapter 6 includes the conclusions and presents suggestions for future work.

2 Background Study

This chapter gives essential information about the basic concepts and definitions used throughout this work starting with a brief overview of the history of plant engineering and piping systems. Plant engineering, digital twin, and piping and instrumentation diagrams are presented in chapters 2.2, 2.3, and 2.4. Lastly, the standards used in P&IDs and the reasons to use standards are explained in chapter 2.5.

2.1 Plant Engineering

Industrial company's purpose is to satisfy customers by selling them products made with companies' resources and services needed for their use. These services are after-sale customer service, technical support, and possible reclamations. (Dal Pont and Dal Pont, 2012) Developing a plant is an iterative process that integrates process, mechanical, piping, electrical, automation, and structural engineering (Holi, 2013). Process systems manage equipment, instrumentation, and pipes with electrical units and smaller sections that comprise a plant. (Lee et al., 2017)

In a chemical process industries (CPI) project, chemical and process engineers are responsible for equipment sizing and P&ID development. Engineers are therefore required to have both skills. The previous abilities include knowledge of hydraulic calculations, pump and compressor sizing, vessel and tank sizing, safety valve process sizing, and heat exchanger sizing. Equipment sizing calls for entirely different skill sets, which depend on the trade segment as well as the seniority level. As older technologies retire, there is a constant stream of new technologies being introduced to replace them. P&IDs can vary greatly in content and structure between businesses. (Toghraei, 2014)

One of the main tasks of a project engineer is to coordinate numerous engineering functions in the engineering department. The mechanical engineering department then gives engineers to a specific project in four significant areas of responsibility: systems, components; building services and heating, ventilation and air conditioning; and piping systems. The piping engineer is given the responsibility and authority to manage and coordinate the piping program. These responsibilities include piping engineering, design, layout, pipe stress analysis, pipe support design, pipe rupture restraint, jet shield design, and piping fabrication contract coordination. These responsibilities and authorities are given by the engineering organization. (Smith and Van Laan, 1987)

A flow diagram is a logical basis for piping system design and all piping design drawings. The system engineer prepares the flow diagram and provides the master plan for each plant system. The flow diagram, in this case, P&ID, is usually the initial drawing in the piping design process. This diagram indicates the types of process equipment and interconnecting piping required to perform the function for which the system is intended. (Smith and Van Laan, 1987) It is also a schematic representation that depicts the flow or constitution of a circuit, device, process, or facility (Moreno-García, Elyan and Jayne, 2019).

Prior to developing a detailed P&ID, block flow diagrams (BFD) and process flow diagrams (PFD) must be created to identify the process's key aspects. Primary streams and operational units are identified by the BFD. The PFD adds tanks, pumps, and some instruments to each BFD block. Later, the P&ID provides fuller details. (Toghraei, 2014)

2.2 Digital Twin

Digital transformation has been a growing aspect of technology in society. With engineering applications, it is common to hear concepts such as Industry 4.0 and Big Data (Worden et al., 2000). Industry 4.0 represents the fourth industrial revolution for automation and data exchange in manufacturing. (Rajput and Singh, 2019) Big Data is a result of the data generated at an exponential rate. Based on stakeholders need, big data gives valuable information for the stakeholders. (Rawat and Yadav, 2021)

To understand the behavior of real structures, systems, and processes and to make predictions in previous unseen circumstances, a digital replica has been created called a digital twin (Wagg, Worden, Barthorpe and Gardner, 2020). In practice, digital twins are used to simulate complex systems, where small variations in key factors can result in catastrophic results. The complexity of these systems is characterized by their large number of components, multiple communication channels, and sophisticated information processing. With this many parts involvement, it makes predicting their future states very difficult. (Rodríguez-Aguilar and Marmolejo-Saucedo, 2020)

The digital twin's primary function is to improve predictive capability by augmenting data in computational models. The digital twin creates a virtual prediction tool that can evolve by combining model-based and databased approaches. The combination of these two approaches offers a potential tool to solve both technical and organizational problems. (Wagg, Worden, Barthorpe and Gardner, 2020) A digital twin enables real-time monitoring of systems and processes by integrating physical and virtual worlds. This monitoring provides data on possible issues that might arise. In addition to this, this data helps to schedule preventive maintenance to reduce and prevent downtimes, uncover new business opportunities, and plan for future upgrades and new developments. (Madni, Madni, and Lucero, 2019)

One of the leading roles for a digital twin is to predict critical safety events and finds new issues before they become necessary, or it is too late to take corrective actions. The planning process includes defining the desired characteristics of the system and its desired and unwanted behavior. (Rodríguez-Aguilar and Marmolejo-Saucedo, 2020) Digital twin in the construction industry is used to reduce damage or recommend changes in mission profile to decrease loads. Thus, a digital twin increases both the life span and the probability of completing a task. (Madni, Madni, and Lucero, 2019) Figure 1 presents a diagram where the data and information flow can be seen. Data from the real model is included in the virtual model. After required changes have been made, that information is brought to the real model.

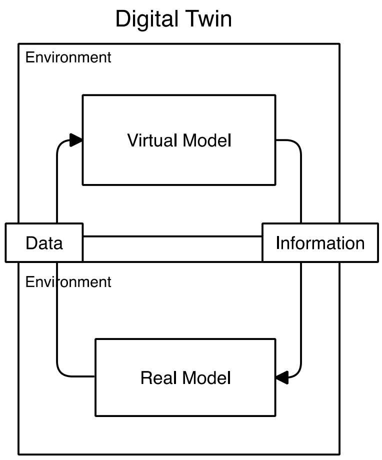


Figure 1. An example of a digital twin.

The digital twin is a relatively new concept in engineering that has received attention across many fields. It is a significant help for engineers who design and manage complex engineering systems (Wagg, Worden, Barthorpe and Gardner, 2020). Many famous companies widely apply the digital twin method, such as Siemens, Aveva, Dassault Systems, Oracle, and ANSYS. Some standards and definitions of digital twin are needed since it is used by several different companies. (Qi, Tao, Zuo and Zhao, 2018)

Building a digital twin relates to producing a copy of a part or a product and then using it to stimulate other instances of the same part or product. This idea belongs to NASA's Apollo program (Schleich, Anwer, Mathieu and Wartzack, 2017), even though the concept was first introduced in 2002 by Michael Grieves at the University of Michigan. Since then, the idea has evolved with the advent of the Internet of Things (IoT). (Madni, Madni, and Lucero, 2019) Glaessegen and Stargel gave a generic and most likely the most common definition of a Digital Twin in 2012: "digital twin is an integrated multi-physics, multi-scale, probabilistic simulation of a complex product and uses the best available physical models, sensor updates, etc., to mirror the life of its corresponding twin." (Lohtander et al., 2018)

Today's widely growing computer technology has increased the requirement of a virtual twin over a physical twin. Virtual twin gives more possibilities for gathering and exchanging data from products. This idea sums up the vision of a digital twin (Schleich, Anwer, Mathieu and Wartzack, 2017).

The planning and design process of the digital twin requires the participation of experts involved in the physical system's operation. Allowing designing the digital model as close as possible to the real object. (Rodríguez-Aguilar and Marmolejo-Saucedo, 2020) A digital twin concept does not cover only a complete virtual model. It is wider set of linked data and simulation models that evolve the product life throughout. Creating a virtual model helps with developing a design that makes predicting expected product behavior possible. The understanding of a digital twin can be observed by Product Lifecycle Management (PLM). (Schleich, Anwer, Mathieu and Wartzack, 2017) The product lifecycle consists of product design, manufacturing, maintenance, and parts where product data are generated. (Huang, Wang, Yan and Fang, 2020)

Madni, Madni and Lucero (2019) have listed how a digital twin differs from the traditional Computer Aided Design (CAD) and Computer Aided Engineering (CAE) model, which have been used widely in the process industry. These factors are following:

a) It contains data about the structure, overall performance, health status, and mission-specific characteristics of the physical twin, such as miles flown, reported malfunctions, and maintenance and repair history.
b) It allows scheduling preventive maintenance based on the system's maintenance history and observed behavior.
c) It provides insight into how the physical twin performs and what can be expected with timely maintenance in the future.
d) The developers can observe performance of the system in order to get a better understanding of, for example, how modifications perform, and how the operational environment is performing.
e) It improves traceability by connecting the different phases of the lifecycle through the digital thread.
f) It allows for the refinement of assumptions using predictive analytics data collected from the physical system and incorporated in the digital model. This data, along with other information sources, can be analyzed to predict future system performance.
g) It allows maintainers to diagnose malfunctioning equipment remotely and perform remote maintenance.
h) It optimizes service and manufacturing processes, as well as identifies needed design improvements (e.g., logistics support, improved performance) by using data collected from the Internet of Things and physical systems.
i) It incorporates historical operational and maintenance data into its simulations and models, thereby determining the age of the physical structure.

One key aspect of digital twin technology is to collect data from the physical system and use it to update the digital twin and predict future states. In this regard, the role of machine learning (ML) algorithms has become essential. (Chakraborty and Adhikari, 2020) However, a digital twin technology offer value while not having to use machine learning.

There is usually a linear relationship between inputs and outputs in simple applications. Real-world systems that face multiple data streams have a lot to learn from machine learning and analytics. Any algorithm used to identify patterns in data could later be exploited in a similar manner using machine learning. For example, machine learning can automate complex analytical tasks. (Madni, Madni, and Lucero, 2019)

An IoT network is a link between the physical and digital worlds, connecting performance, maintenance, and health data from a physical location to a virtual location. Linking digital twins to the IoT brings the required data to know how the physical twin behaves and performs within the operational environment. Combining data from physical and prediction models helps to make beneficial decisions in the design process and production operations and can even create new business models. Notably, IoT and digital twin support an organization to get to know how customers operate a system or a product. Thus, maintenance schedules and resource utilization can be optimized, potential failures can be predicted, and system downtimes can be avoided and reduced. (Madni, Madni, and Lucero, 2019) Industrial

IoT is one of the key pillars for the conclusion of the real vision of Industry 4.0 and one of the significant underlying technology for digital twins. (Kamath, Morga and Ali, 2020)

Engineers can use simulation linked to the digital twin to predict how the physical twin can perform in the real world (Roldán, Gonnet and Leone, 2013). The engineers can use the digital twin to improve the physical system model and to use the analysis with the digital twin to enhance the physical system's operation (Madni, Madni, and Lucero, 2019).

Digital twin virtual simulation models represent the real mappings of existing production lines. The modeling process goes as follows As part of the modeling process, the production line is modeled in four dimensions, including geometry, physics, behavior, and rules. To determine if the model is valid and correct, it's evaluated and validated. Once each dimension model is related, combined, and integrated, it is certain that they will be able to be integrated into a high-fidelity virtual production line model. (Li et al., 2020)

2.3 Piping and Instrumentation Diagrams

P&IDs are widely used in the process and energy industries, where they describe oil and gas facilities' layout functionality, such as oil rigs and plants. This diagram can be drawn based on a PFD. (Rica, Moreno-García, Álvarez and Serratosa, 2020) Over the years, these diagrams were first drawn on paper and then scanned, stored, and generalized as images. (Rahul, Paliwal, Sharma and Vig, 2019) Currently, these diagrams are created by computer-aided design software (CAD) and are essential documents in the process of a project (Schuller and Epple, 2012). These facilities are massive and are composed of thousands of electric, electrical, and mechanical components connected by a network of pipelines (Rica, Moreno-García, Álvarez and Serratosa, 2020). Figure 2 presents one of the procedures that can be created when a plant is designed. However, all three steps are not necessary. When all three diagrams are created, they are done as follows: First, a block flow diagram is created. This diagram shows the basic structure of a plant clearly and understandably. After that, a process flow diagram is formed. PFD shows the overall flow of plant processes and equipment involved in the process with flowing chemical fluids' properties. Lastly, a P&ID is developed. P&ID is more complicated than the other two diagrams and includes much more detailed information on the process and equipment.

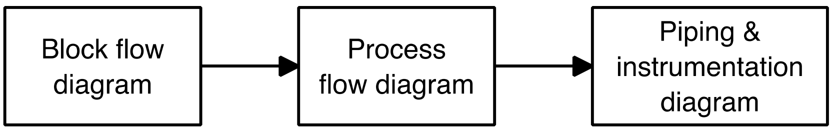
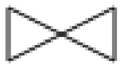
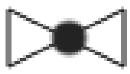
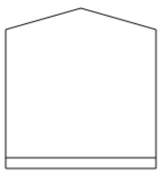
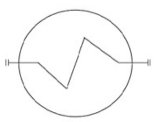
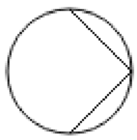


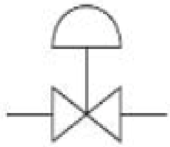
Figure 2. The process between different diagrams before P&ID can be created.

The majority of engineering, procurement, and construction contractors use P&ID for plant design, procurement, and construction. Currently, creating P&IDs is time-consuming and error-prone since it is primarily done manually, and its quality varies depending on the individual worker's skills. (Yu et al., 2019) The flow diagrams are usually representative and not drawn to scale. A unique line number typically identifies each pipeline indicated on the flow diagram. The pipeline number remains constant. Therefore, if the line number changes, it may be expected that line service, sizes such as diameter or wall thickness, material, temperature, pressure, other properties, or any combination thereof, has changed. (Smith and Van Laan, 1987)

As part of P&ID, each object is classified and standardized, and it can be managed on a computer using computerized data management (Yu et al., 2019). P&ID symbols are critical components and are categorized into four types: fitting, instrumentation, equipment, and diagram reference. These symbols are connected by lines that represent pipelines. P&ID also consists of outer borders, title boxes, characters, and tables. Table 1 presents common symbols and their usage. Major components of a diagram encompass symbols carrying field-precise meanings, lines representing inter-symbol connections, and attributes assigned to symbols and lines through texts or characters (Toghraei, 2014). P&IDs can be linked to the project database and can include 3D design models, using a tag ID that is one of the attributes of symbols. (Yu et al., 2019)

Table 1. A few of the most common symbols use in P&IDs are listed in the table.

Name	**Icon**	**Use**
Gate Valve		A gate valve is used in situations when a straight flow of fluid and minimum restriction is desired. Gate valve can be either open or close.
Globe Valve		Globe valve is used as a control valve since it has a complex flow path and precise throttling and control. Globe valves are perfect for low leakage shut-off and regulation.
Storage Tank		A storage tank is a container that stores fuel, water, or oil. Storage tanks are classified based on their pressure and usage.
Heat Exchanger		Heat exchangers are used to transfer heat between fluids and can be used to cool as well as heat.
Pump		A pump is used for raising suction or pressure, compress, or move fluids in and out of other objects.

Reducer		A reducer makes it possible to change the pipe size to meet hydraulic flow requirements or to adapt to existing pipe sizes of various sizes.
Diaphragm		Diaphragm valves can be used to control flow and speed. Diaphragm valves offer advantages in specific low-pressure applications not possible with other types of valves.

2.3.1 Standards

Standardization in piping systems, like in every industry, is used for reducing the cost, inconvenience, and confusion resulting from unnecessary and undesirable differences in equipment, systems, materials, and procedures. In addition to documenting industry practices, standards may also be used to document safety, testing, and installation practices. Within engineers, standardization is employed through the implementation of official standards and procedures. Between organizations, standardization is often accomplished through industry standards. Industry standards are generally published by professional societies, trade organizations, and committees. These standards are prepared by experts in the field and then adopted by industry consensus. (Smith, 1987, p. 17-18)

A valid global standard for data exchange within CAE vendors' CAE systems would reduce the cost of handling engineering data during the planning, construction, and operation of the process. Suppliers and owners also want to discover new value opportunities and added values from the engineering data they currently have available. (DEXPI, 2020)

Symbols used in P&ID are based on technical standards. These symbols are only representing a specific type of component. Symbols can be labeled with words, letters, and numbers, which will later identify and specify the components they represent. It is also important to remember that these diagrams do not always display the physical position of each component. The purpose is only to illustrate the system's process and not serve as a floor plan or map of the system. (Rinker, 2017)

Several standards define the technical plant. The most critical function types of process plan elements are described in standards ISO 10628 and DIN 28004. The general rules for the illustrations of properties, notations, and object references in a diagram are defined in ISO 15519. Other standards, such as ISO3511 and DIN19227, now replaced by IEC62424, are not that commonly used but mentionable. All of these standards discuss the process of practical engineering and the graphical representation of a plant's function. (Schuller and Epple, 2012)

The PSK Standards Association is dedicated to improving standards in industry and industry supporting companies. The idea of the association is to help members in their national and international businesses by standardization and training. There are over 400 standards that have been divided into several sections, such as valves, pipe classes, and flat gaskets. Graphical symbols of flow and P&I diagrams for the process industry are presented in document PSK 3605 (PSK Standardisointiyhdistys, 2016). PSK standards define the symbols used in the P&IDs.

The ISO 15926, “Industrial automation systems and integration—Integration of lifecycle data for process plants including oil and gas production facilities”, is an international standard that represents lifecycle data of process plants, such as oil and gas facilities, during their lifetime. A key objective of the standard is to define an ontology of concepts intended to improve data integration, exchange, and handover across computer systems. (Opdahl, 2010)

Data EXchange for the Process Industry (DEXPI) is an industry initiative to support the ISO 15926 standard for plant transfer. Interoperability between different CAE systems has been difficult for companies. Working with several CAE systems creates challenges in data exchange while working with multiple parties in projects to plan, construct, and operate process plants. Parties typically exchange data between different projects, owner-operators, vendors, and site services and authorities. This data exchange model was created on ISO 15926 to make planning, construction, and use of plants less costly and more efficient. As of now, DEXPI's primary focus is on the exchange of PI&D. DEXPI uses Proteus Schema as its exchange format. (DEXPI, 2020)

2.3.2 Existing Systems

There are plenty of P&ID applications on the market, but the majority are desktop applications. In table 2 are listed some of the commonly used P&ID software and their P&ID versions. Only desktop versions of these programs are available. These software products can all be used remotely, and Autodesk also provides a web-based database. Connections to remote computers are commonly made using remote desktop protocols. A protocol typically consists of two software components: a client and a server. (Bitton et al., 2017)

Table 2. Most common P&IDs software in the markets.

Software	**Version**	**Web-based editor**	**Note**
Aveva	AVEVA P&ID	No	Remote desktop solution
Siemens	COMOS P&ID	No	Remote desktop solution
Intergraph	Smartplant P&ID	No	Remote desktop solution
Autodesk	Autodesk plant 3D	No	Web based database can be used (BIM 360), Remote desktop solution

The software users consist mainly of expert users. Expert users use computer software on a daily basis, and they have at least a basic understanding of the software they use or the subject around it. In regard to this matter, the software can be implemented and designed in a specific type of way. Because the experienced user is presumably familiar with software in a similar area, the implemented software will investigate the same functions. However, one of the demands on a developed software can be that it is simple and easy to use. This specific requirement indicates that even though a significant part of the end-users will be expert users, the software must be accessible for other users as well.

2.4 Data Sharing

When sharing data between different parties, the data may change in transit. This change could be due to different parties using different tools, various data formats; standards are missing; there is broken information flowing across lifecycle phases, or the owner of the information is not clarified. (Dietz, Putz and Pernul, 2019) Organizations, governments, and individuals have been exchanging information long before computers and networks existed. Constantly new applications are even new formats entering the markets. This continuous increase with the software makes data sharing practices highly important. Many software have created their own formats, which are not interoperable with other similar software. This approach is a way to get a customer lock with their products. Therefore, data sharing enables competitive supply chains.

Having the ability to share data between organizations and government agencies increases the efficiency and value of services, and enables combining data from different sources, increasing the performance and value of the service. Additionally, data can be reused, making the same procedures redundant. (Support Centre for Data Sharing)

In data sharing, there are issues related to diagram software that can arise due to different data formats, extra functions, and so forth. If the software is not standardized, data sharing may not be possible. In order to minimize errors, some standards have been created. One of the primarily used standards in P&ID software is previously mentioned DEXPI. The DEXPI information model is a conceptual model that describes the objects that appear in a P&ID from an engineering point of view. It is not a model of a P&ID as a graph, but the information model is organized in a way similar to Unified Modeling Language (UML) class diagrams. (DEXPI, 2020)

Sharing data on the Web is not risk-free. Nonetheless, it has advantages such as broad data sharing. If a web database is used, the database encryption has to be created not to affect data sharing. A database contains files, records, fields. In addition, by using separate files, records, and fields as a basic encryption unit, various levels of concepts, different kinds of encryption schemes can be applied to database information. (Schleich, Anwer, Mathieu and Wartzack, 2017)

The most commonly used formats in P&IDs software are Scalable Vector Graphics (SVG) and previously mentioned DEXPI. Both of these formats are written as Extensible Markup Language (XML). XML is clarified more in chapter 2.5. SVG is a standard created for present two-dimensional vector graphics. It enables displaying accurate random pictures that consist of geometrical figures, curve lines, text, and bitmap images. (Nykänen, 2007) Another often used format is DWG, which is defined by Autodesk. It is compatible with most of the used software.

2.5 Software description

Developing a Web front-end starts with understanding three languages with different purposes. Hyper Text Markup Language (HTML) is used for structuring content, Cascading Style Sheets (CSS) to style it, and JavaScript is used to attach particular behavior. (Queiros, 2018) This chapter gives brief information on the concepts used in the implementation, presented in chapter 4.

2.5.1 HTML

HTML is a standard markup language that is used for creating web pages over the World Wide Web. Web pages are composed of elements called tags in HTML. These tags are displayed by using Web browsers. (Raza et al., 2020) HTML is usually stored in plain text files. When the web browser is accessed, the individual HTML file for that exact web page is transferred and decoded into the visual image, which is seen on the screen. (Dorman, 2020)

Cascadian style sheets, also known as CSS, are a standard layout language for the Web; they control typography, colors, and the size and placement of elements and images. CSS defines styling rules for HTML elements. The CSS rules control how the contents of specified elements should be displayed. The rules consist of a selector and declarations. The selector determines which HTML elements to style. The declarations specify the style for the elements. Each declaration is comprised of a property name and value. (Dorman, 2020)

2.5.2 JAVASCRIPT

JavaScript is a programming language used by Web developers for adding functionalities to Web pages. JavaScript scripts can be contained internally in an HTML page, which is the original method of scripting, or reside in an external file. The latter process is the most used nowadays. Despite the name, JavaScript does not have anything to do with the Java program language. JavaScript was introduced when Netscape added some necessary scripting abilities to its Navigation Web browser. Back then, it was called LiveScript. When Netscape revised Navigator to run Java applets, it also renamed LiveScript to JavaScript. (Negrino and Smith, 2007)

JavaScript can create a Web page more interactive and provide a more exciting experience for the user. JavaScript help to create an active user interface while simultaneously giving the user feedback as they navigate the pages. JavaScript also enables creating forms, setting cookies, building HTML pages on the fly, and creating Web-based applications. (Negrino and Smith, 2007)

A key concept in developing Web applications is the distinction between server-side and client-side code. Servers are the places on the Web that serve up websites to the rest of the world, and clients are the computers accessing those websites and receiving those responses. (Dorman, 2020) JavaScript is a client-side language, which means that it is designed to work on one machine and not on the server. Due to this reason, there are some limitations built-in. Limitations can be an example of that JavaScript does not allow to read or write on external files on a client machine or server machine. As another example, the program cannot close a window that is not open or read information from a Web page that was opened from another server. Some of these limitations also have advantages. Since JavaScript cannot write on a file on a client machine, it prevents the Web page from writing viruses onto your disk or manipulating the files on the client computer. (Negrino and Smith, 2007)

In JavaScript, object-oriented programming is supported, enabling developers to define, organize, and access objects with a logical data structure. (Miquel, Fonseca, Gaspar and Vieira, 2020)

2.5.3 JSON

JavaScript Object Notation (JSON) is a data-interchange format that is essentially an associative array or a hash table. In JSON, names and labels are structured with values in an array or comma-delimited list. JSON parsing is supported with JavaScript's eval method. However, this method can be slow or unstable even though it is more straightforward to use with Ajax. JSON syntax is simple and lighter than XML, which can get excessive when there is a lot of data. (Handlock, 2007)

2.5.4 XML

XML is a standard format of content representation and sharing on the Web. It is a custom tag-based structure that is defined by the developer. XML is supported both on the server-side and client-side, which makes it the most flexible solution. XML can be used with multiple languages and transferred back and ford for accessible communication. It enables to create of direct connections from the GUI to a server-side language and database. (Handlock, 2007)

2.5.5 Ajax

Ajax is an abbreviation for Asynchronous JavaScript and XML. It is an emerging presentation layer technology of the Web, which allows dynamic, fast, and flexible Web application procedures to be built (Yan-feng and San-jun, 2013). The term Ajax was first introduced by Jesse Hames Garrett in 2005. Ajax is a small part of JavaScript but is a combination of several technologies such as XHTML, CSS, The DOM (Document Object Model), XML, and XMLHttpRequest.

The benefit of Ajax is that most of the processing for the application takes place within the user's browser, and the server request for data is usually short. The most significant advantage of Ajax is that it updates the web page without refreshing (Yan-feng and San-jun, 2013). Some companies have made considerable investments in Ajax, notably Google, which has built several major Ajax applications, including Gmail, Google Calendar, and Google Maps. (Negrino and Smith, 2007)

Ajax techniques are used to make sites more responsive and attractive. Ajax requires a modern browser to run and JavaScript. JavaScript is needed because it is used with browser functions. These requirements create problems with users who use old browsers or have turned the JavaScript off. Another problem with Ajax is that it may break the expected behavior of the browser's back button. These problems need to be taken to account when using Ajax. (Negrino and Smith, 2007)

2.5.6 Web servers

When a user requests a web page by typing a Uniform Resource Locator (URL) in the address bar, the request is sent to a web server that hosts the website. Web servers are special computers connected to the internet that send web pages to users. Web pages used to be static and done with basic HTML. The complicity and requirements of current web applications have grown in recent years. As a result of this growth, web pages have evolved into dynamic applications on the server, and later, web servers have been made into fully functional single-page web applications using JavaScript on the client. JavaScript allows the server to offload visual logic and functionality to the client, resulting in better scalability via less server usage. (Jannes, Lagaisse and Joosen, 2019)

3 Software Architecture

This chapter outlines the components used in the implementation, which is presented in chapter 4. Figure 3 illustrates the main structure of the software created in the implementation. Structure components are explained further in this chapter.

Software architecture is one of the main functions to create when developing software. The creation of a software architecture is a complex, interactive decision-making process that involves exploration, evaluation, and composition of alternative design approaches. (Roldán, Gonnet and Leone, 2013) Good architecture is essential; otherwise, the software becomes slower, and it will be more expensive to add new capabilities in the future.

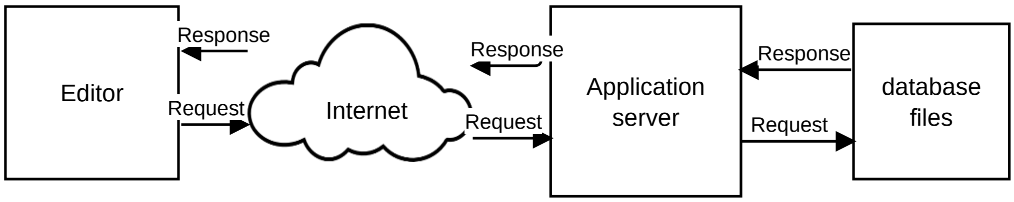


Figure 3. The arrows show how the request and response proceed between the components.

The application created for this thesis is designed using three main components, as shown in figure 3. The three-component architecture was chosen for its accessible modification possibilities and distributed design environment. An editor is used to capture the user data, an application server to run all the applications and process all the data, and a database where all the data is stored. Between these components, REST API validates the incoming request against the entity fields required to save, update, or delete the data in the database.

REST refers to the way web services are designed so they can be consumed by various clients with simple HTTP requests. (Guntupally, Devarakonda and Kehoe, 2018) The type of HTTP request determines how user data is packaged when sent to the server. API is used to get and modify data between all three components. Figure 3 shows the request and response flow between the used components. The most used API calls are GET, POST, PUT, and DELETE. GET call is used when a piece of information is requested from the database. POST request creates new resources and is used when saving data. PUT call modifies data in the database and DELETE call deleted data. Return values are usually in JSON or XML format. The significant advantage of API has increased manageability. The most commonly used mechanism for customizing a server is the Common Gateway Interface. The use of CGI machinery has fairly consequential associated costs. The most obvious is that each CGI request will be handled in a separate process. Another advantage of APIs is that they allow a third-party developer to easily change aspects of the server functionality that are not available via CGI. (Thau, 1996)

In figure 4, the process flow of opening, editing, and saving a file between the software components is displayed. The diagram explains how all three components are used while these actions are performed. The user opens the file using an editor, connecting to the application server that connects to the web server. This connection behaves as a bridge to the database where the file itself is searched. First, it needs to be checked that there is a file under that tag. Then, the application server sends a signal if the file was found or not. If the file was found, the user can edit the file, and the file will be open with an editor. When saving the same file after the user has done desired modifications or created an entirely new file, the steps are relatively the same. Again, the application server connects to the web server. After this is done, the data will be saved to the Structured Query Language (SQL) database. SQL is a standard language for relational and object-oriental databases (Ahadi, Prior, Behbood and Lister, 2016). Next, it needs to be tested that the saving succeeds, and the data can be found from the database. Lastly, an alert message is shown to the user if the save operation was successful or not.

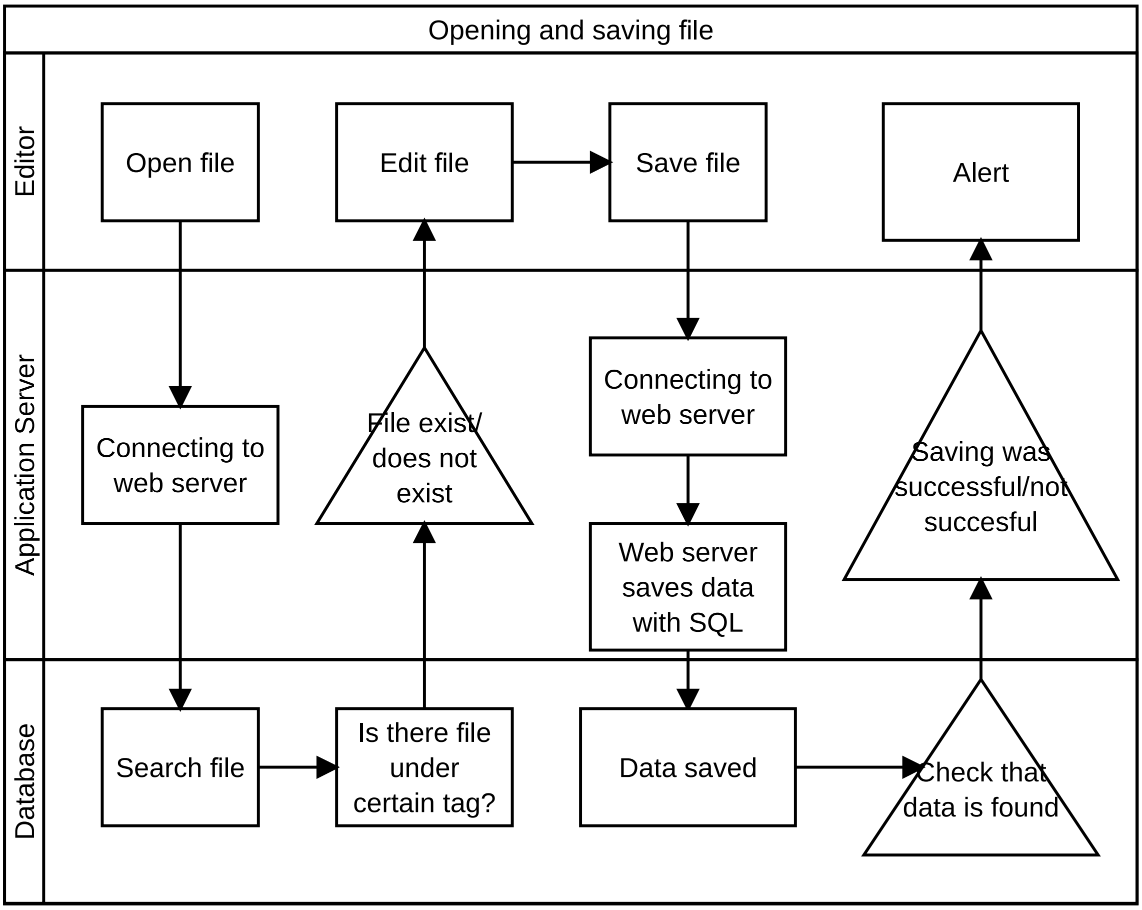


Figure 4. The swim lane diagram presents the signal's route when a file is opened and saved to the database.

With the user interface, the aspiration is to create a pleasant and welcoming outlook where actions are easily accessible and understandable. For this software, the strive is to have a modern outlook used widely in macOS software versions. The idea is that users who are not familiar with similar software can use the implemented software without needing a user manual. However, during this thesis, the main focus is the software functions and not the outlook.

3.1 Requirements

When developing software, defining requirements before starting the development process is needed. There are three kinds of requirements: functional, non-functional, or domain requirements. Functional requirements demand the necessary facilities that the software should offer. Non-functional requirements cover the quality functions of the software, such as security and reliability. Domain requirements are specific to a particular category or area of projects. (Altexsoft, 2018)

Table 2 describes the requirements for the implementation. These requirements are mainly functional. The main requirement is that the software is practical to use and can create P&IDs. Considering that one of the requirements is to create a web application, it can generate challenges for data sharing and processing. Part of the requirements is that the software can process and handle significant amounts of data. These functions are not tested during our project, although these are concerned with the implementation. Testing requirements can be done in a way that a few test cases are generated where P&IDs are created, opened, and modified. These tests are done not only at the end of the project but also during the development. The software testing is presented more broadly in chapter 5 as part of the results.

The software's primary functions are to create and modify P&IDs. These have the importance level 3. This software's one of most essential requirements is that the P&ID design and management is shareable between different parties in different organizations at the same time. For this, a web-based implementation is a suitable solution. Another significant benefit is that the software does not tie the user to specific software but is integrated for other software. These two have the importance level 3. Importance level 2 includes necessary functions, but the software is fully functional without these additions. This level contains requirements: easy to use, data accessing, and standard handling. Only one requirement has been defined as level 1, necessary to view data. This requirement is classified with low priority because it is a fundamental requirement and part of the other requirements and will not need specific production. Most of these requirements are superficial and more or less general. Since the implementation done for the thesis is just a part of the whole software, and most of the testing is not performed, particular requirements were not an obligation.

Table 2. Requirements for the implemented software.

Requirement	**Level of importance**	**Reason for importance**
Create PI-diagrams	3	The main function
Easy to use	2	Fundamental requirement
Easy to share P&ID design and management with different parties in different organizations at the same time	3	One of the most critical advantages
Needs to view data	1	Fundamental requirement
Needs to access data	2	Fundamental requirement
Needs to be able to modify data	3	The main function
Integrated for other software	3	One of the advantages in the competition
Needs to handle standards	2	Important for projects

3.2 Web-based Software

A familiar software type is usually desktop software which is used on a stand-alone computer. These desktop computers are accessible only on a specific computer where the software is installed. This implies that the setup and the configuration of the server are located in the client computer. In typical cases, client-server software can run on several computers connected by a computer network known as a local area network. (Walker & Chapra, 2014) Managing these software and licenses need recourses.

Web-based software is demand when several users work with the same shared data. Therefore, the complex business needs of a large group of users are simplified and make the data sharing more effortless and practical between these parties. This kind of client-server software is divided into three parts: client, server, and database. A client-server software is also called a three-tier Web Architecture. It is designed to provide a greater degree of flexibility and increased security that can be created for each service at each level. The following chapters will explain each component in more detail.

Three-tier architecture is not the only architecture design used in web software. Other architectures are 1-tier architecture which is also called singular tier architecture, and 2-tier architecture. 2-tier architecture has two layers, a presentation layer that runs on a client and a data management layer residing on a server. (Dorofeev and Shestakov, 2018)

Web-based software gives the freedom to use concepts used on the Web, such as links and multimedia. Links can be used to provide additional and related information. Multimedia, such as videos and audio, can be added without obstacles to a web page. These can provide a significant addition when comparing to other software in the same field. (Ward, 2002)

3.3 Editor

An editor consists of the user interface. A graphical user interface that displays information and content in a web browser or Web-based application is usually used. Editors are typically developed using HTML, JavaScript, CSS, or another popular web design framework and communicate with other layers through API calls.

The requirements for the editor are listed in chapter 3.1. Creating a user-friendly interface and a system that is accessible for all types of users is the main focus. The structure of the editor is explained in chapter 4. Essentials for the editor structure are its functionality, flexibility, and speed. The entire software is located on the server. To get access to the server, the user needs to have access rights over what is conserved by the login.

3.4 Application Server

Application servers coordinate the application, process commands, perform logical evaluations and calculations, and make logical decisions. is also the connector between editor and database and transfers data between these components. It contains both the application logic but also presentation logic. Applications servers obscure technical details and provide a simple, structured programming interface with optimal performance for program developers. The application servers are typically developed using Python, Java, Perl, PHP or Ruby, and communicates with the data tier using API calls.

The application server cuts down or expands the components which are needed by the user, making the developing and maintaining process more manageable and enhancing system stability. By separating business logic, user interface, and back end service, the application server provides a flexible way for developers to construct, deploy, and maintain enterprise web applications. (Zhong, Yun and Huang, 2006)

Application server requirements can be divided into four categories. The first group includes connectivity, easy to use, and security. Within the next group is modules that are flexible in terms of extensibility, integrating existing functionality through protocols, and expressing and managing dependencies. The third group of requirements includes specific requirements related to semantic Web, including language support for semantic interoperation, ontology mapping, and modularization, find, access, modify, and store of ontologies, transactions and versioning, monitoring, inferencing, and verification. The last group includes requirements for semantic enhancement of the application server allowing discovery of software modules and APIs, classification of software modules, and facilitating implementation tasks. (Oberle, Staab and Volz, 2004)

3.5 Database

A database stores the data used in the software. Since the software is web-based, the database is a web database that can be accessed from a local network or the internet. In desktop software, the database is usually stored on a desktop or its attached storage. These web databases are hosted on websites, which means that access is provided via a web browser. Data is accessed by the application server via API calls.

Storing data in the cloud benefits the mobility of data within several computers. With a web database, the data can be accessed by multiple users. Mainly, cloud-based databases are saved at the back of secured protection. Security in web databases is fundamental. The basic requirements for security are integrity, confidentiality, consistency, and usability. Integrity signifies the preciseness and compatibility of data in the database. This is guaranteed by various integrity constraints such as physical and logical integrity. Data and information in a database are considered confidential if only authorized users are able to modify or browse them. This helps prevent data leakages. Consistency implies physical integrity, referential integrity, and user-defined integrity. Usability means an easy-to-use interface that respects users' authority. (Huang, 2013)

Since the database is an essential component in the application, its security is a critical part of information system security. It is important to secure the data in the web database using a variety of methods. One of the most prominent security methods is the access control strategy which limits the use of resources. Access control could be achieved by creating exceptional views and storage procedures. (Huang, 2013) It is common to share databases because multiple users may access the database simultaneously. This type of sharing is done primarily to eliminate redundant data. By sharing the database, the number of programs to execute increases. Additionally, the occupation of the disk queue increases the efficiency of resource utilization. The most practical approach to database sharing admits concurrent WRITERS as well as READERS. (King and Collmeyer, 1973)

A two-phase locking protocol is used by most commercial databases to maintain database integrity. In this protocol, many transactions can be run simultaneously, despite the impression that they are all running independently. It uses the concept of a lock as its main primitive component. (King and Collmeyer, 1973) Two-phase locking protocol prevent transactions from executing conflicting operations simultaneously, so if two of them are issued, one is forced to wait. The protocol is required because traditional locks only provide shared and single-lock relationships, and thus transactions are blocked to ensure correctness. (Agrawal and Abbadi, 1995) Database communicates via SQL. SQL statements can be used to update data in a database or retrieve data from a database. (Ahadi, Prior, Behbood and Lister, 2016)

4 Implementation

This thesis aims to develop a browser-based software that can handle and create P&IDs. The whole project is done for the company Doxie Oy. The main purpose is to do proof of concept (POC) of a web-based editor for future development. This chapter discusses the implementation of the developed software.

There is plenty of software in the markets that handle flow diagrams. However, most of these software are limited only to flow diagrams and do not have information about technical specifications. Our target is to create software later linked to Doxie's CAD software and access technical details on piping systems. Since implementing software is a large project, we have defined this thesis's process to limit only to the P&IDs. The implementation is narrowed down to process just the web software editor. We choose not to use any existing libraries to prevent possible copyright risks. This decision determines that everything used in the software is created for the project.

The main idea is to develop software that can be used by several users and which would handle various file formats and thus, facilitate the use of many software and parties in one project. JavaScript was chosen as the primary development tool and HTML to implement the visual interface. The process started with studying the basic concepts of P&ID. When the main idea was internalized, we began the implementation. We were not familiar with HTML, JavaScript, and web servers.

JavaScript is a highly used programming language when developing web pages. Other languages that can be used with HTML are C++, JAVA, Python, and VBScript. The most effortless of these programming languages is JavaScript, which was one of the main reasons why it was used for this project. In addition, it was painless to learn with our previous programming skills. C++ and JAVA require third-party software to function with web pages. VBScript is a form of JavaScript, but it is not supported with every browser. With web page design, JavaScript seems to be a steady choice.

For the drawing itself, an HTML canvas was used. The canvas is easy to adapt to and is reliable to use when a new programming skill is learned. Other options for canvases are Web Graphics Library (WebGL) and SVG. SVG was mentioned earlier in chapter 3.4. SVG has many advantages, such as great scalability and flexibility. However, for the software done for the thesis, it was seen as too detailed. WebGL is a JavaScript API for presenting 2D and 3D graphics. It can be used with any browser and will not need any plug-ins. Although it is accessible to use, it may affect the used browser. The advantages that webGL has were not significant for this project, and therefore it is not used.

Implementation and writing of the background section coincided. The first step required setting up the main page, then the primary drawing mode had to be defined. Several design modifications were needed and, later, other functions such as opening files, saving files, and inserting symbols into a canvas were developed.

During the implementation, we used Google Chrome. Some of the graphics work differently in different browsers. Since we want to use the latest versions and complex implementations, we chose to use Google Chrome. Internet Explorer is no longer supported by ECMAScript and will not most likely operate with our implementation. Since there is still plenty of Internet Explorer users, we have planned to modify the implementation to make it accessible to these browser users. Other browsers that are supported and updated are functioning without additional implementation.

We developed an HTML user interface for the software. However, we were looking for a more Apple MacOS-style graphical user interface for the software. A Mac computer was used to implement the software. However, the software should also be available for other operating systems.

We decided to maintain the features that are familiar to the customers of P&ID software. For this reason, we only used mouse click tracking and not mouse movement tracking, which is typical in a program such as Windows' Paint.

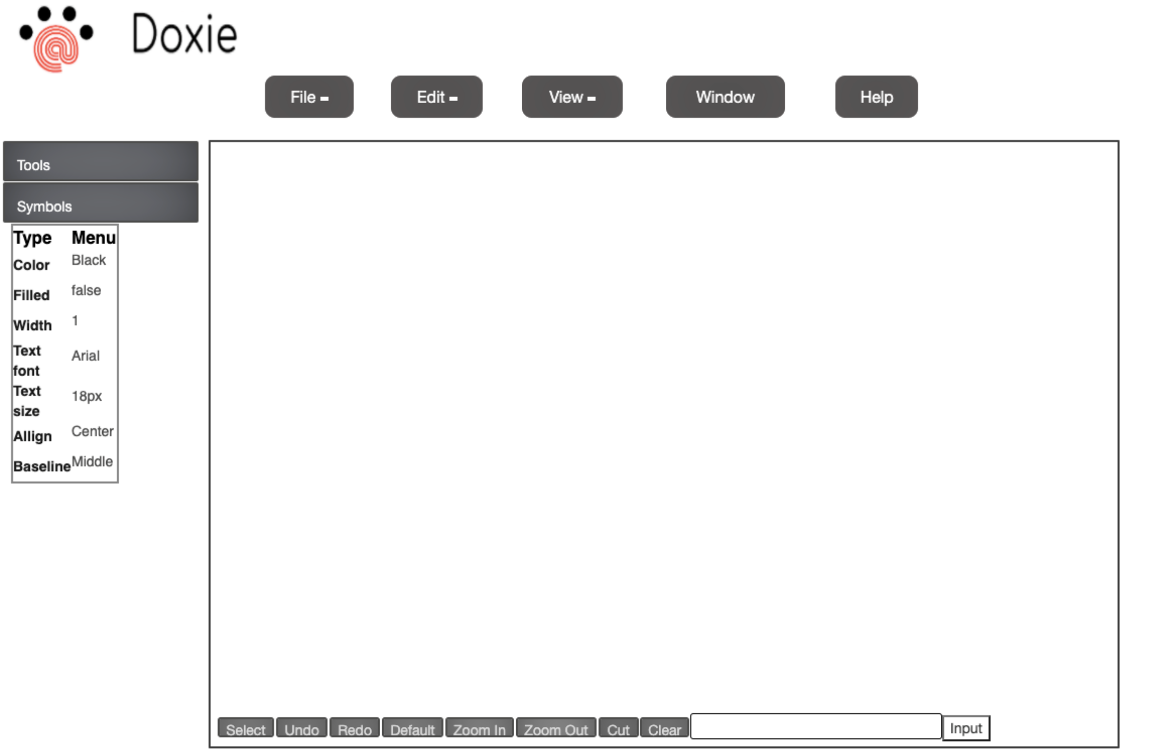


Figure 5. The start page of the software.

Figure 5 shows the start page which is visible when creating a drawing. The page layout is reasonably similar to most creative software. All available tools are shown on the left side of the page, where current settings are listed. Available tools are divided into groups to make the design process effortless. Current settings can be changed from the settings table. At the top of the page are all the essential functions such as save, open, etc. A diagram will be drawn or opened to a canvas that takes most of the space on the page. At the bottom of the page is an input text bar where commands can be entered. The input bar was added because some of the end-users are familiar with working with these functions. Next to the command line, there are various buttons for the most common functions.

```json
{
  "Header":{"AcadVer":"A1501","HandSeed":"61"
,"Extents":["-0.02086462 -0.12682654 -0.00152582","2520.19795015 1967.60596194 0.00000003"]
,"Limits":["-250 -250 0","2692.28307255 2162.44665718 0"]}
,"Blocks":{}
,"Tables":{"Layer":{}
  ,"Linetype":{"13":{"Name":"CONTINUOUS"},"11":{"Name":"BYBLOCK"},"12":{"Name":"BYLAYER"}}
  ,"Style":{"F":{"Name":"STANDARD"}}
  ,"DimStyle":{}
  ,"View":{}
  ,"VPort":{"60":{"Name":"*ACTIVE"}}
  ,"AppId":{"10":{"Name":"ACAD"}}
  ,"Ucs":{}}
,"Entities":{"1": {"Type":"circle","Layer":"1E","Linetype":"13","Centerpoint":["0 0 0"], "Radius": "30"},
           "2": {"Type":"line","Layer":"1E","Linetype":"13","Color":"256","Points":["0 30 0","30 0 0"]},
           "3": {"Type":"line","Layer":"1E","Linetype":"13","Color":"256","Points":["0 -30 0","30 0 0"]}}
}
```

Figure 6. The structure of the JSON-file of a pump.

All drawn items are stored in a drawing database object. The data is stored in JSON format to practice quicker development. After every modification done in the canvas, it is changed to the object. When a new file is created, the drawing database object named *currentDatabase* is created. This object includes several keys that contain information of the drawing. When new entities are created, these are then added to the entities section in the *currentDatabase*. Creating and modifying files in this way facilitates their saving to the database. Figure 6 presents the basic structure used in JSON files. This particular file includes the drawing information of pump block.

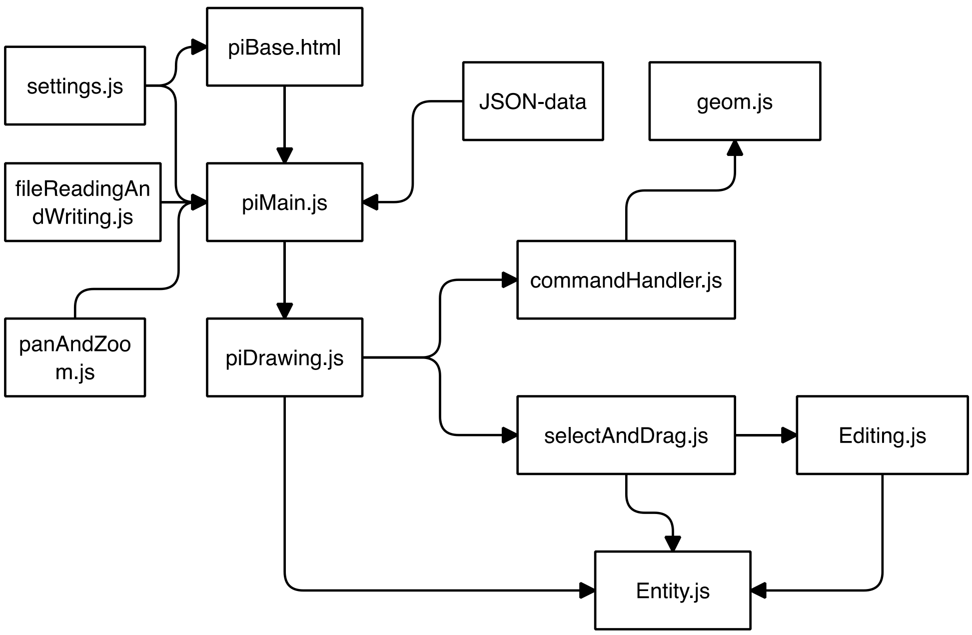


Figure 7. The structure of the files used in the implementation.

The structure of the editor is presented in figure 7. The web server runs the piBase.html file, which then launches the main program, located in the file *piMain.js*. The imported and exported data from the software is structured in JSON format. This file is read in the piMain.js file. The reading itself is done in *fileReadingAndWriting.js*. The file is structured as follows: header, tables, line type, blocks, and entities. The header contains all the document's primary information, such as name, version, limits, and extents. The tables section covers all the current events, such as layers and used blocks. Blocks sections include all the available block information used when a block is selected and drawn. The entities section contains all the drawn objects, and this section will be modified when new objects are created on the canvas. Every object has an id number that makes the data modification possible.

Zooming and panning the canvas is defined in a file named *panAndZoom.js*. This file is connected to the main file *piMain.js*. All of the mouse movements are implemented in the file *piDrawing.js*. This file also stores the keyboard shortcuts and functions, making the drawing more effortless and more understandable. *CommandHandler.js* contains all the possible commands and the main functions related to these commands. This file has a link to *Entity.js* and *piDrawing.js*. *SelectAndDrag.js* file contains selecting an item in the canvas and dragging them inside the canvas with keeping track of the connections between the items. The drawing functions are created as canvas-based.

Current tool settings are presented in the table left side of the browser window. These settings are controlled in a file named *settings.js*. *Settings.js* file is also connected straight to *piBase.html*, that the settings can be presented in the table.

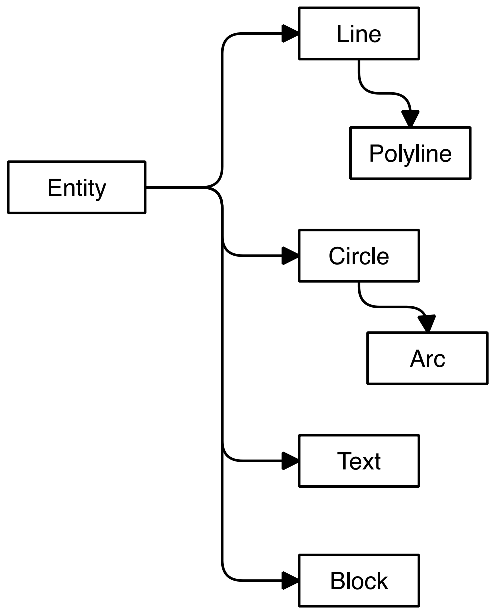


Figure 8. The inheritance of the objects from subclass Entity.

The object structure and the functions related to the objects are specified in the *Entity.js* file. Every drawn item is saved to the *currentDatabase* object. Figure 8 shows the inheritance connections. An entity is an upper class where the main functions are defined such as *drawEntity* and *drawGripBox*. Functions like these are used by every entity. All of the objects are inherited from this class. Other objects are line, circle, arc, text, and block. Every lower class has its functions. This structure was chosen to make the code easy to understand

and make it easier to modify. It also reduces the used code since all of the object classes can use the entity functions. Most of the entities operate similarly. When one function needs to be modified, the modification will affect all of them. This practice reduces time and human error, which might occur when copying the modification to several places. Editing objects is controlled by the *Editing.js* file. It contains functions that enable objects to be moved, connected, et cetera.

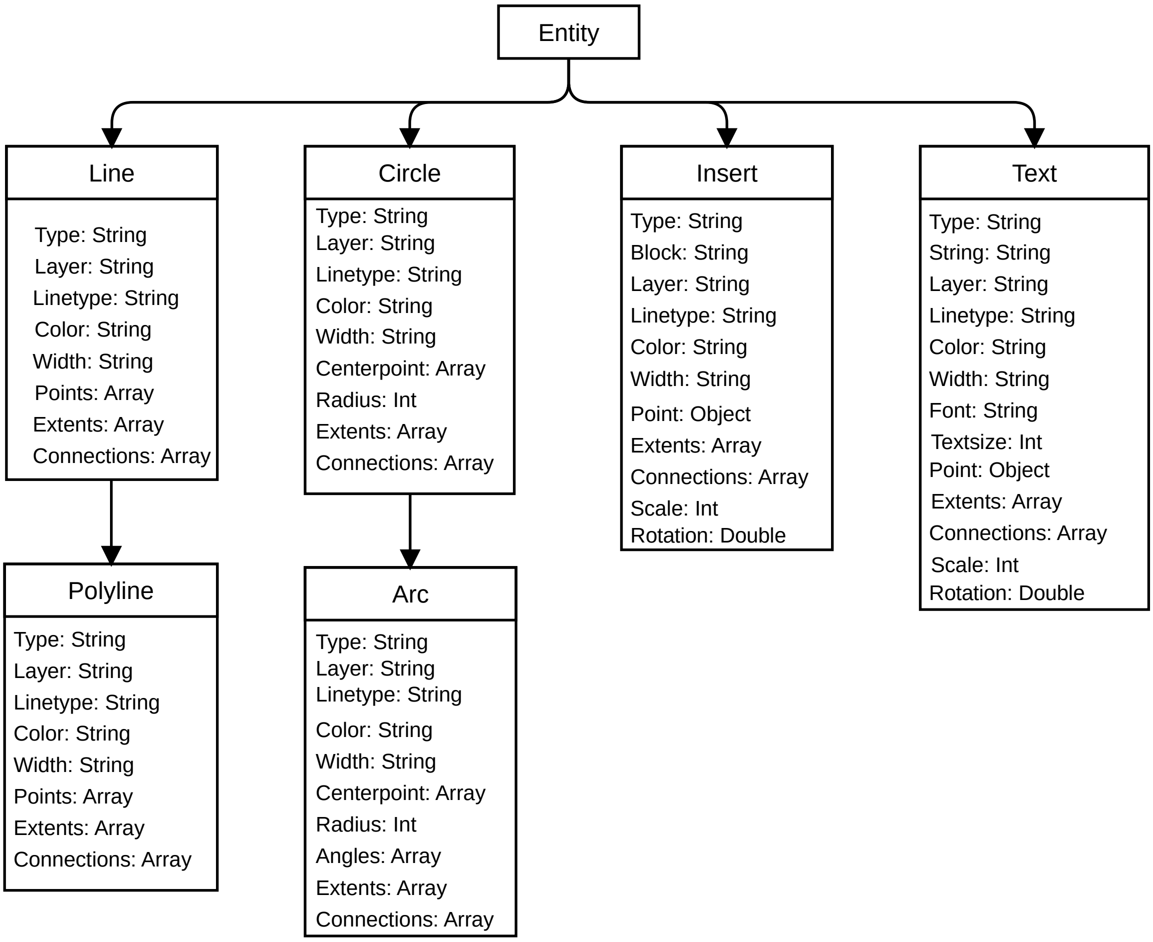


Figure 9. Entities are saved to the current database with object properties.

Every entity has its properties. Figure 9 shows properties used in which entity and what is the type of the variable. Type, layer, linetype, color, and array are defined as strings. In line function, circle and arc points are defined in an array with two points. Polyline points are listed in an array. Angles used in the arc are defined in an array. In insert and text, only the center point is needed and defined as an object. Rotation defines the angle the text or insert objects are inserted to the canvas. Rotation property type is double. Extents and connections are defined as an array in every entity. Every entity extends array has four points. Connections include classes of each connection point. A connection class contains the connection point, name of the entity, and other needed information.

Every block is named after PSK standards. Blocks in the drawing are created as follows: if the selected block has not yet been used in the drawing, the block's entities are fetched from the database. The database contains JSON files of all of the most common blocks. Additionally, a user can create own blocks and store them in the database. Used JSON files are structured in the same way as the JSON files containing the whole drawing. Every time a new block is used, all entity information is copied to the Blocks-section of the drawings *currentDatabase*. This block is also added to the entities section with a type insert. Entities of the blocks are created as listed in the JSON file. The entities section of a valve block is presented in the figure 10. This method speeds up the use of blocks since the block data is not fetched from the database whenever used.

```
,"Entities":{"1": {"Type":"line","Layer":"1E","Linetype":"13","Points":["-20 10 0","20 -10 0"]},
           "2": {"Type":"line","Layer":"1E","Linetype":"13","Color":"256","Points":["20 10 0","-20 -10 0"]},
           "3": {"Type":"line","Layer":"1E","Linetype":"13","Color":"256","Points":["20 10 0","20 -10 0"]},
           "4": {"Type":"line","Layer":"1E","Linetype":"13","Color":"256","Points":["-20 10 0","-20 -10 0"]}}
```

Figure 10. The valve block consists of four line objects.

Text function works as any other entity. It is listed in the database with the same information as the insert entity. The user input comes from the keyboard event listener. Text can be modified, rotated, or the text's style can be changed after being stored in the database. The current text style is displayed in the settings table.

When a user wants to save a drawing, the *currentDatabase* will be sent to the database. If the drawing already exists and the file was only modified, the old file will be deleted and the new one saved. The saving process, in general, will be implemented in the future. During this thesis process, opening and saving a file is created to function with the local database.

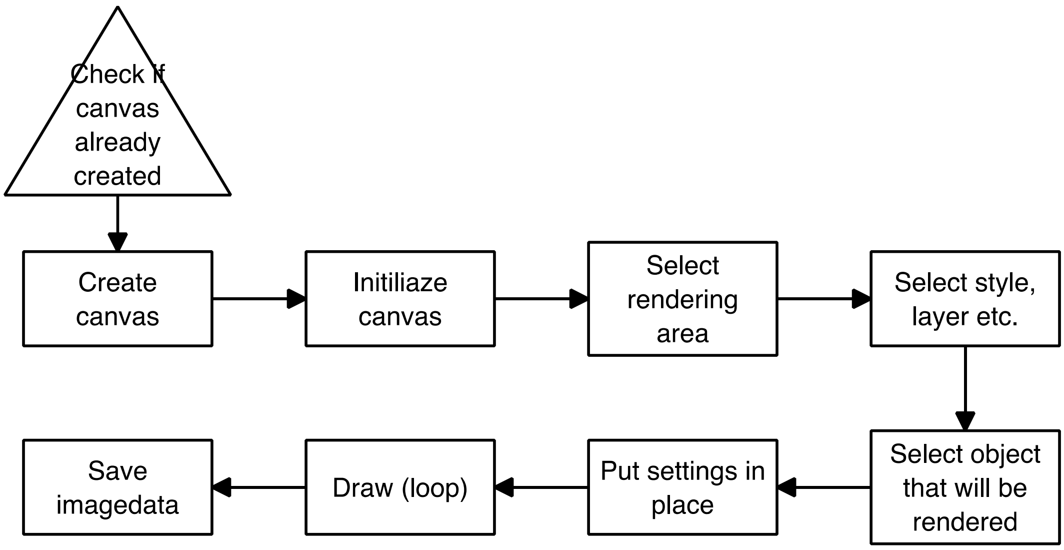


Figure 11. The rendering process is explained.

Rendering a drawing is shown in figure 11. First, a canvas is created. When creating a new canvas, it must first be tested if there is an existing canvas already created. The canvas is initialized, and the rendering area is selected. Next, the used style and layer that is rendered are selected. Then, the rendered object will be chosen, and the setting is set in place. The drawing uses a drawing function that is defined in the entity class. Every object has its draw function. Drawing takes place in a loop. After the drawing is finished, the image data is saved to the database. Figure 12 present the flow of editing an object. First, the edited object is selected, then the wanted implementation type. This implementation can be, for example of dragging a grip box or deleting the whole object. The editing function will modify the *currentDatabase* by adding the new points or deleting the object. After, the changes are done to the *currentDatabase* which will later be saved to the database. Lastly, the graphics are rendered. Figure 13 presents the event of creating an object. First, the user selects the wanted object type. This is done by clicking the button on the screen or a command in the command box. Next, the drawing is presented with the mouse tracking functions. Current settings are set to the object, and the object is drawn to the canvas. Lastly, the entity is added to the currentDatabase in JSON format.

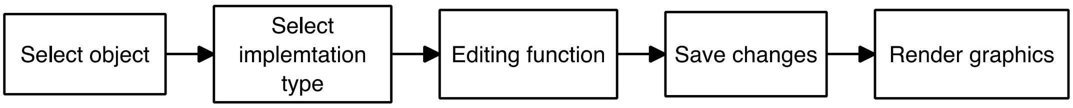


Figure 12. Diagram of modifying an object.

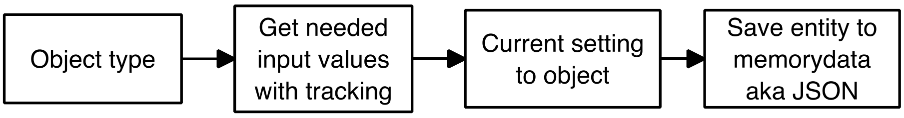


Figure 13. Diagram of creating an object.

Drawing and modifying geometrical figures such as circle and arc require several calculations to operate. Also, detecting the drawn figures from the canvas need calculations. Figure 14 below presents a function that calculates the distance between a point that is clicked by the mouse and a line on the canvas. This function is used when selecting a line on the canvas and modified later on. The function in figure 15 returns a list of the circle's center point and radius. These numbers are calculated by using three points.

```
function pointDistToLineSeq(p1, p2, x, y){
  var XKJ = p1[0] - x;
  var YKJ = p1[1] - y;
  var XLK = p2[0] -p1[0];
  var YLK = p2[1] -p1[1];
  var denom = XLK * XLK + YLK * YLK;
  if (denom < 0.1){
    return Math.sqrt(XKJ*XKJ + YKJ*YKJ);
  }
  var T = -(XKJ*XLK + YKJ*YLK)/denom;
  T = Math.min(Math.max(T, 0.0), 1.0);
  var XFAC = XKJ + T*XLK;
  var YFAC = YKJ + T*YLK;
  return Math.sqrt(XFAC*XFAC + YFAC*YFAC);
}
```

Figure 14. Function *pointDistToLineSeq* calculates the distance from a clicked point to a line.

```
function circleBy3Points(L, K, M){
  var XLK = L.x - K.x;
  var YLK = L.y - K.y;
  var XMK = M.x - K.x;
  var YMK = M.y - K.y;
  var X = null;
  var Y = null;
  var RCCSQ = null;
  var DET = XLK*YMK - XMK*YLK;
  if(Math.abs(DET) < 10E-5){
    return null;
  }
  else{
    var DETINV = 0.5/DET;
    var RLKSQ = XLK*XLK + YLK*YLK;
    var RMKSQ = XMK*XMK + YMK*YMK;
    var XCC = DETINV*(RLKSQ*YMK - RMKSQ*YLK);
    var YCC = DETINV*(XLK*RMKSQ - XMK*RLKSQ);
    RCCSQ = XCC*XCC + YCC*YCC;
    X = XCC + K.x;
    Y = YCC + K.y;
    return [X, Y, RCCSQ];
  }
}
```

Figure 15. The function returns a list of the circle's center point and radius after calculating it with three points.

Setting up the server was not part of the scope. For the implementation and testing, a local server was used. This server is used as a testing environment and will be changed in the future. When opening and saving a file, an HTTP request is used. In figure 16, both PUT and GET functions are shown. While using a local server, these are connected to developing computers.

```
function readFile(filename){
    var xmlHttp = new XMLHttpRequest();
    xmlHttp.open( "GET", filename, false); // false for synchronous request
    xmlHttp.send( null );
    var data = xmlHttp.responseText;
    return data;
}

function writeFile(filename, data){
    var url = "/Library/WebServer/Documents/Dippa";
    var json = JSON.stringify(data);
    console.log(json);
    var xmlHttp = new XMLHttpRequest();
    xmlHttp.open( "PUT", url+'/'+filename, true); // false for synchronous request
    xmlHttp.setRequestHeader('Content-type','application/json; charset=utf-8');
    xmlHttp.send(json);
    var data = xmlHttp.responseText;
    return data;
}
```

Figure 16. The figure presents a code that makes an HTTP request for GET and PUT.

A significant implementation challenge was to understand the most practical way to structure the database and operations. We were not familiar with the used programming language, JavaScript, or even web-based software techniques, HTML and CSS. That, by all means, caused deceleration with the implementation. The segment created for the thesis is only a section of the full software.

5 Results

In this thesis a proof of concept of a web-based P&ID software was implemented. The requirements for the software were listed in chapter 3.1 and the structure of the implementation in chapter 4. The purpose of this chapter is to explain how testing should be done to gather feedback from the software's functionality. Then, two test cases are described, followed by a conclusion of the results.

Testing the software is part of the implementation process. Testing is carried out by creating performance tests. The performance tests' sections are speed, reliability, capacity, safety, maintenance, security, and usability. On this occasion, these factors are difficult to test. Software speed specifies the time available to complete a specific task. Capacity describes the volume the software needs to be able to handle. Safety is one of the most critical requirements for the software. Testing that quality can lead to some difficulties since different countries might have their standards; therefore, the product's standards must meet the need to be specified precisely. There might be special requirements for maintenance. For example, the software must be possible to be maintained by someone who has not been part of the development process. When using web-based software, security is an essential factor to be tested. It is crucial to specify who has authorized access to the software, under what circumstances the access is granted, and which parts of the product can be accessed. The last test is the usability of the software. This testing type describes how easy it is to use the software and if there needs to be some external training. Usability also includes how easy the diagrams created with the software are to fit with other projects. The main usability requirement is defined in chapter 4.

The software's performance can be tested by creating several test cases where previously listed elements focus. To test this feature is to perform functions repeatedly and compare the results to what was expected. Maintenance tests can be achieved by letting another programmer try out the software and make some programming code changes. Usability tests can be done by giving access to the software to customers and receiving feedback on how efficient the software was to use. Since the software is not entirely functional, these tests cannot be run before complimenting the programming.

The testing done to test the functionality of the implemented software included creating two test cases. In the first test, a new P&ID is made. This test shows that the tools can be used in the software, and these functions work correctly. The second test is to open the existing drawing and generate changes to the drawing. After that, the diagram is saved. The purpose of testing to open a file is to see how the software reads data and can access the data stored in the database.

The first test starts with the user creating a new file from the top menu. This action will create the drawing canvas. Figure 17 displays the top menu. The top menu includes saving, exporting, and editing, including undo, redo, cut, paste and other functions that are essential.

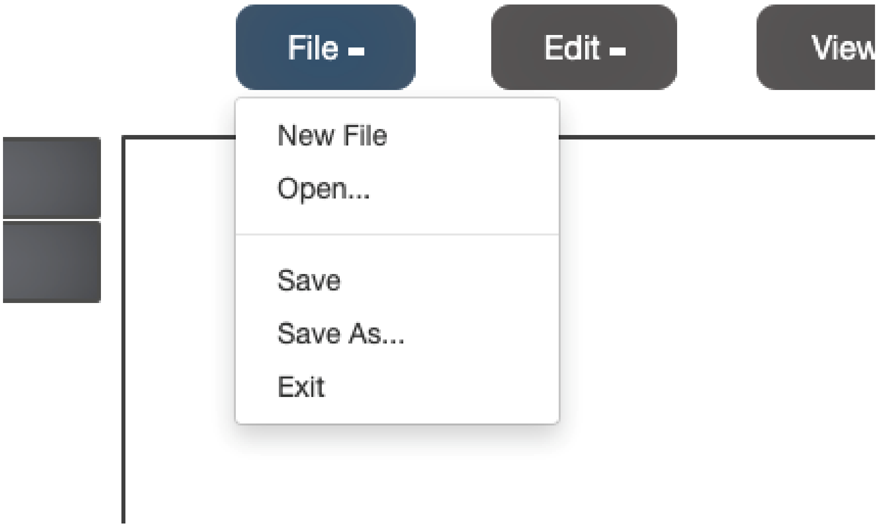


Figure 17. Top menu used for general file operations.

Type	Menu
Color	Black
Filled	false
Width	1
Text font	Arial
Text size	18px
Allign	Center
Baseline	Middle

Figure 18. Current drawing setting for creation and modification for objects.

The current drawing settings are presented on the left side of the browser window (figure 18). These settings can be changed by clicking the value under the menu column. Clicking the value opens a modal window (figure 19). The changed value can be typed to its text area. *Filled* selections describe if the object is filled or not. False is presented as the default since it is not filled. Text settings consist of *text font, text size, align and baseline*. The user only needs to modify the desired setting; all other settings remain the same. After clicking the save button, the settings window closes, and the current settings table is updated, and settings put in place.

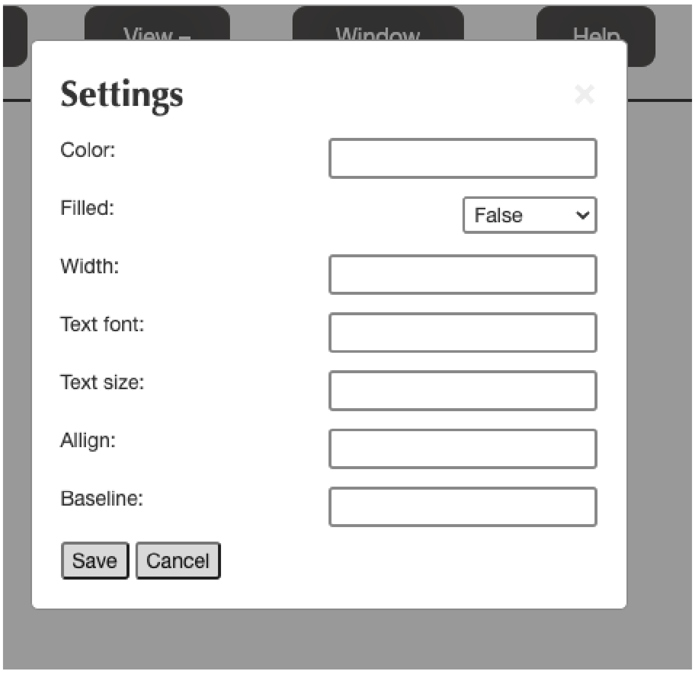


Figure 19. The modal window of the settings.

All of the possible tools and symbols are located in the menus on the left side of the window. These menus are dropdown windows and can be opened and hide by clicking the title of the menu (figure 20). The tools section includes line, circle, arrow, arc, polyline, and text. There have only been a few symbols created for the tests. The rest will be created in the future. Users will also have the option of creating their own symbols in the future.

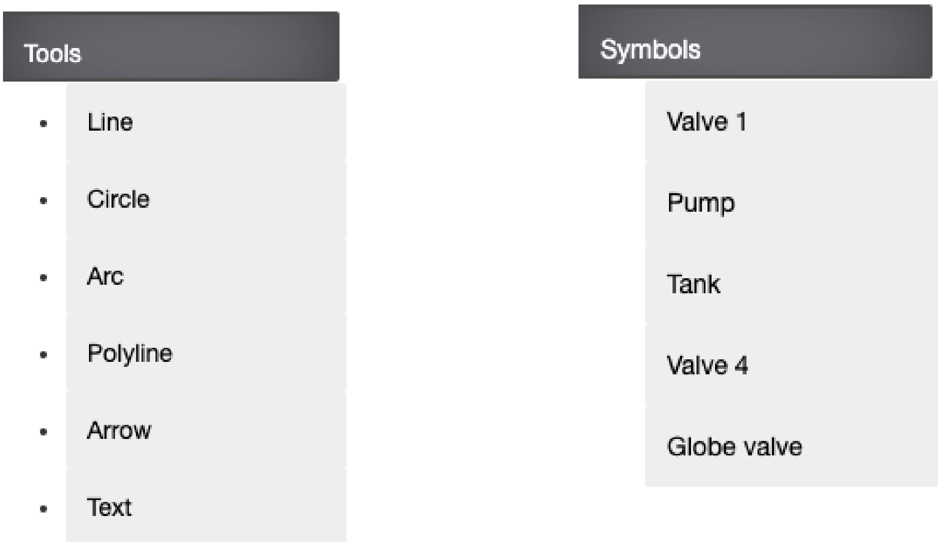


Figure 20. Tool and symbol menu windows.

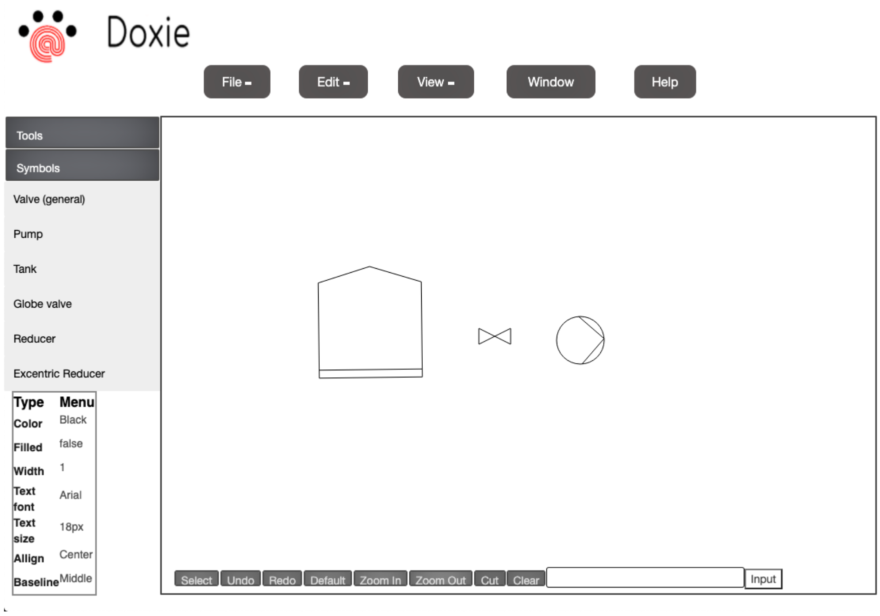


Figure 21. All symbols are placed.

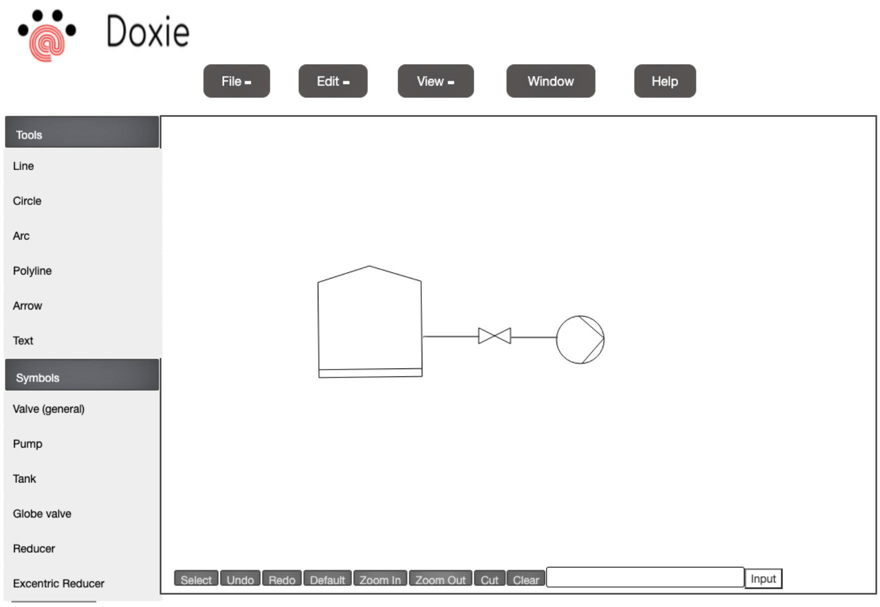


Figure 22. Final drawing on the canvas.

The drawing of this test started with placing the symbols on the canvas (figure 21). First, the desired symbol is selected from the symbol menu. The first mouse click defines the symbol's center point. The second mouse click represents the point from which the rotation vector can be calculated. To help the user determine the rotation of the symbol, the rotation vector is shown. Then these symbols are connected with pipelines that are presented as lines and arrows in the drawing. The ortho mode allows lines to be drawn in 90 degrees. Symbols that are horizontally or vertically oriented can also be drawn with orthomode. The final drawing is shown in figure 22.

The second test is to open an existing drawing and create some modifications, and then save it to the computer database.

Figure 23. File opening modal window.

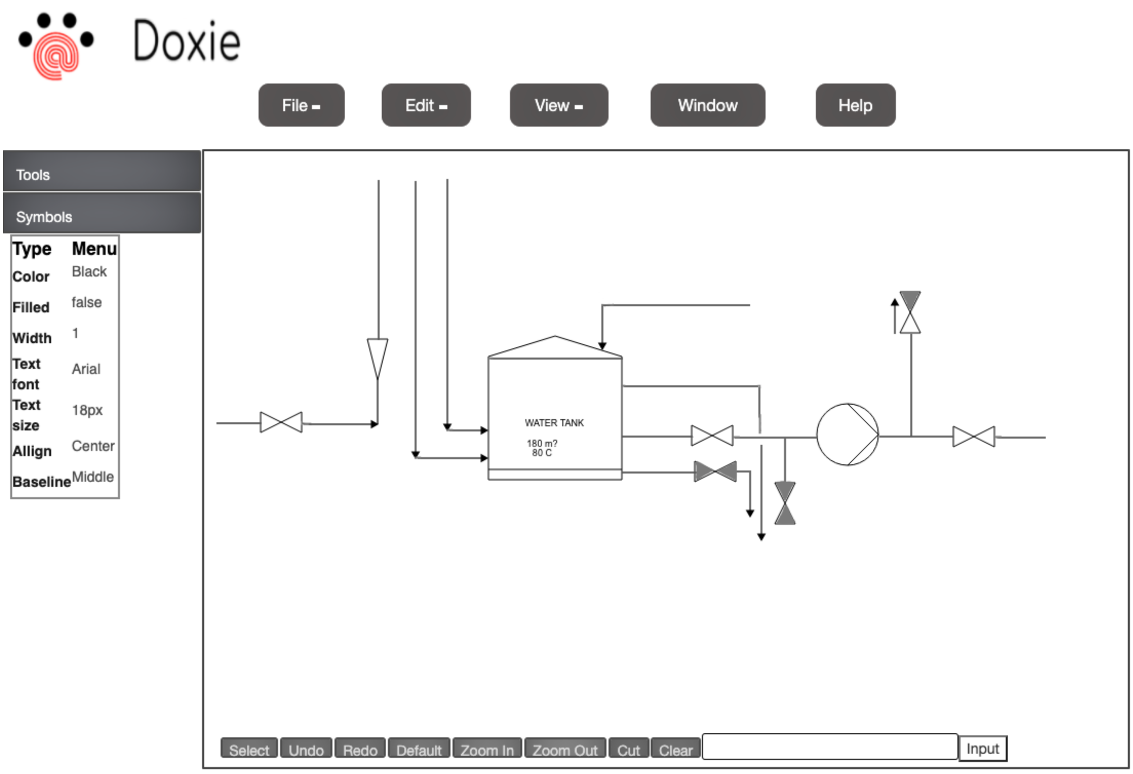


Figure 24. Opened JSON-file is read and is shown on the canvas.

Upon opening a file from the top menu window (figure 23), the software makes an HTTP request to the database — in this instance, to its local location on the developer's computer — with the file name. The fetched JSON-file is read and drawn on the canvas and to the *currentDatabase* to make the drawing easily editable. Also, this speeds up the modifying process and the previously saved drawing can still be restored. Figure 24 shows the drawing opened and drawn on the canvas.

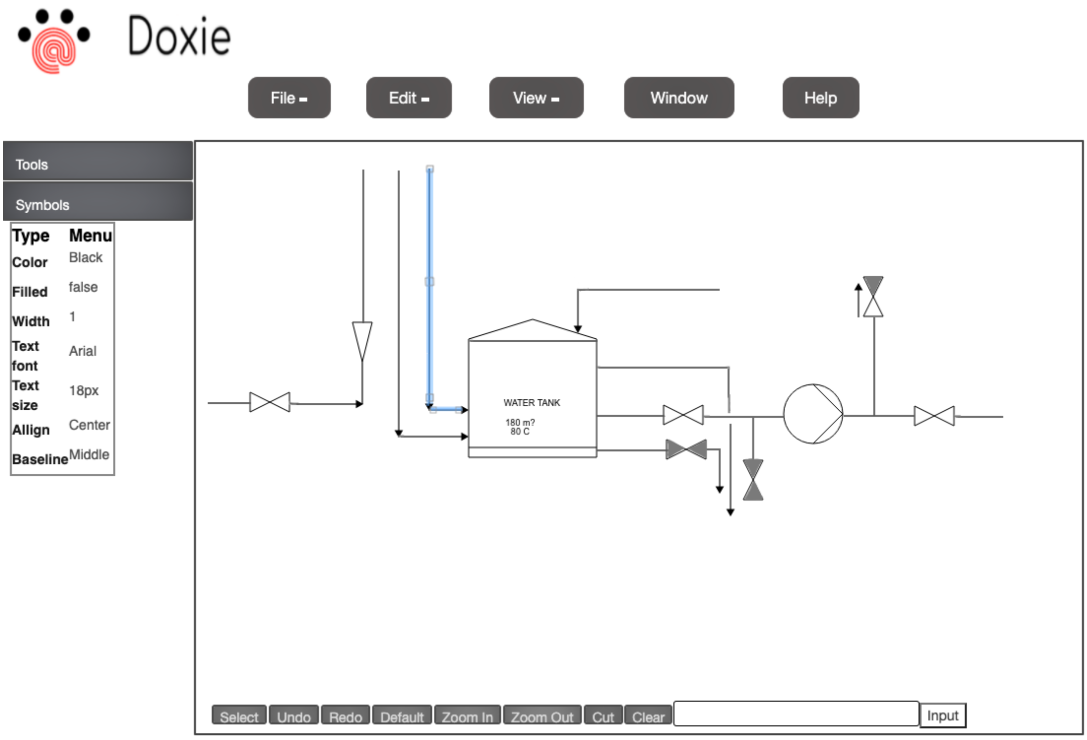


Figure 25. The figure presents how an item is shown when it is selected and can be modified.

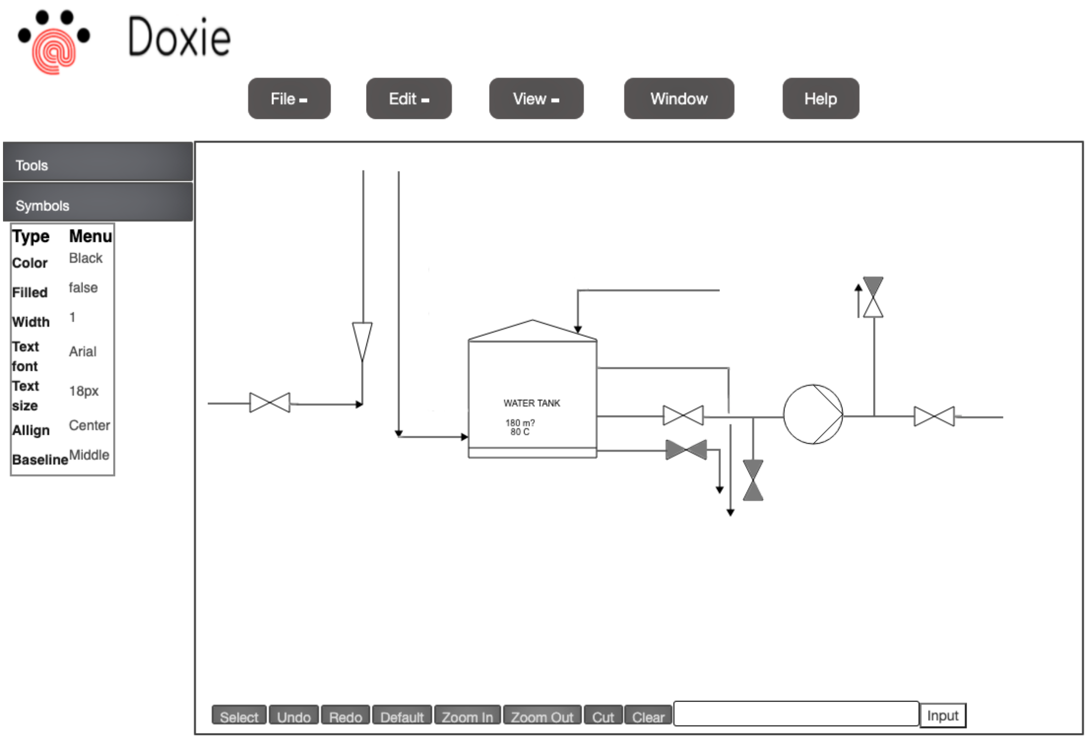


Figure 26. Selected arrow deleted from the drawing.

An object that is selected is highlighted with grip boxes (figure 25). Straight line objects contain three grip boxes. Objects can be moved by these boxes. Selected objects can also be removed with the delete command or by pressing delete (figure 26).

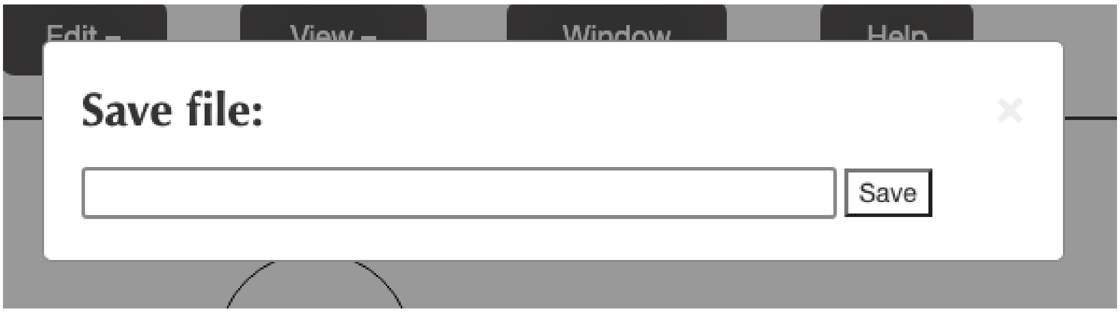


Figure 27. Saving modal window.

No commands can be on when a user wants to modify an object. Commands can be turned off by pressing ESC on a keyboard. When an object is selected, it is presented with a blue highlight, and it has grip boxes places where the object can be dragged. Grip boxes are drawn in the sides and center of the objects. The selected item can be deleted with a delete button. When an object is selected for the first time, object extends will be calculated. For selecting and moving an object, the distance from the mouse click point and object is calculated. If the mouse click is close to the given value, it is selected ad added to a *selectedItems* list. *SelectedItems* list keeps track of the items that are selected at the moment. Figure 14 presents the calculations used when a line is selected. After the user has made the desired changes to the drawing, it will be saved to the database. Saving is performed by simply making an HTTP request. Figure 27 shows the Saving window. Drawings are saved as JSON files.

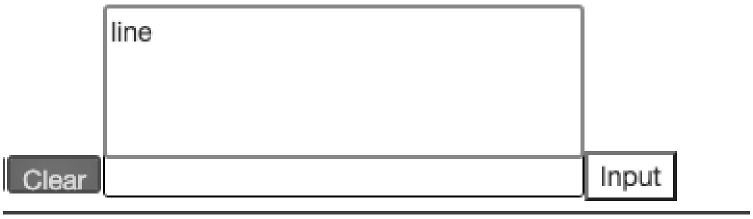


Figure 28. Command bar.

P&ID software are mainly used with engineers that are used to work with the command bar. We wanted to provide this function with our software too. At the bottom of the browser window is a text area where the user can write commands. For example, when the user wants to use line-tool, they can type the word line. The used commands are shown in the upper text area that can be hidden. This is presented in figure 28. All available commands can be seen when typed ‘help’ on the command bar.

Two test cases performed for the thesis showed that the software functions correctly. More testing is needed to find possible errors. As a result, the software is only useful if it is used correctly. Testing has not been done with wrong commands, and error situations are not taken into account. The software reads data only when it is precisely in a particular style.

Working with web-based software, the developer has a limited amount of equipment. JavaScript and HTML are simple enough to get to know with basic programming skills and are commonly used to create web-based software. Storing data in JSON format was a valuable solution since it is easy to read and can be opened with several different software, which was one of the requirements.

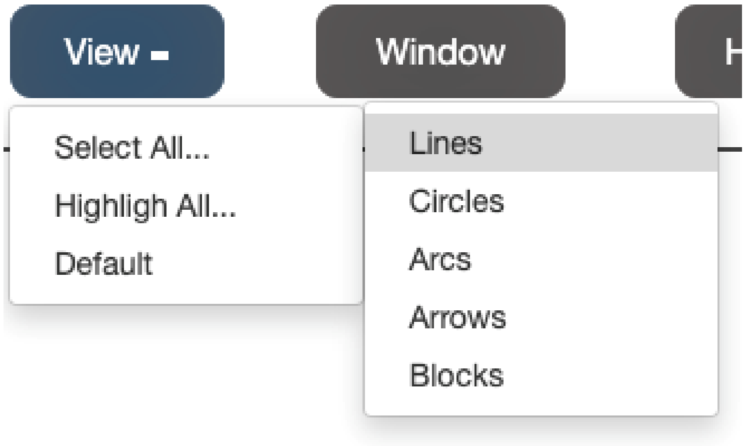


Figure 29. View option to select all the lines in the drawing.

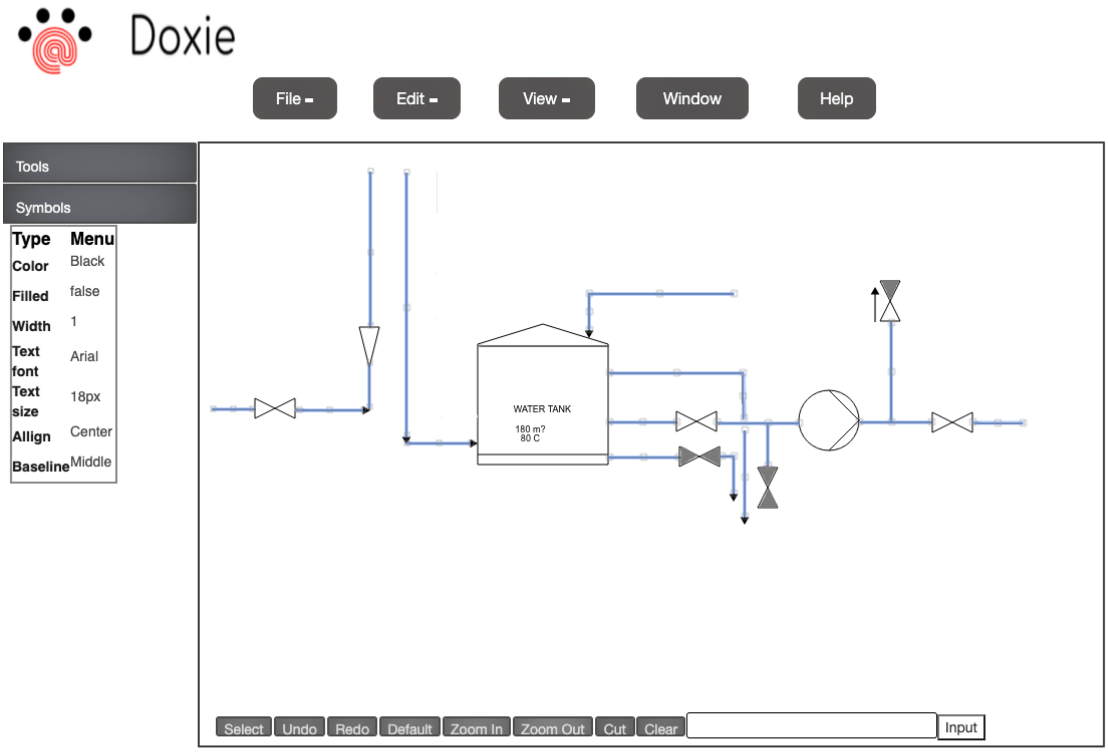


Figure 30. Drawing when all the lines are selected.

Additional features have been created in the program to facilitate the work of the engineer such as a feature that allows the user to select all certain types of objects from a drawing. View tab shows the available features (figure 29). *Select All* and *Highlight All* open a new window listing the objects to select. *Default* sets all the settings and selections to default mode. Figure 30 shows the drawing when *Select All Lines* has been selected.

6 Discussion

This chapter concludes the thesis topic with a conclusion chapter 6.1 and then discusses the future work that should be followed in chapter 6.2.

6.1 Conclusion

The aim of this thesis was to create software for a solution for a problem in the process industry projects where several different parties work on the same project. Process industries are constantly faced with issues where project data has been affected by interoperability issues in data sharing. For preventing these issues and solve both technical and organizational problems, a digital twin solution can be used. The main idea of the digital twin is to create a digital replica of a living or non-living physical entity. Digital twin consists of a large set of data from different fields. In process, instrumentation and piping engineering P&I diagram has large role. P&ID consist of symbols, lines, tags and other entities that include important information for designing a plant.

The methods used for this thesis were a literature review, an analysis of the competitors, a specification of the requirements, and implementation of a proof-of-concept.

In this thesis, part of the web-based P&ID software was developed. The software implementation focused on the editor. The software architecture used in the implementation was a three-tier architecture that consists of three components: editor, application server, and database. The software was implemented as web-based based on the data sharing demands on digital twin.

The software requirements include supporting many formats and it can be used together with other applications. Two test cases were performed to test the functionality of the software. The first test was to create a new drawing, and the second test was to open an existing drawing, make some modifications and then save it back to the database. Both tests performed successfully. The scope of the thesis included part of the editor; therefore, future development is needed to complete the whole digital twin solution.

This thesis implements proof of concept of P&ID solution in web-based environment in scope of digital twin. This solution can be used as good basis for future developments.

6.2 Future Work

The scope for the software implementation was focused on the editor. At the end of the thesis work, the software was implemented to function functionally only with the limited number of environments. All of the used resources were located in the testing environment, and for the test server a localhost was used. Large-scale future work includes setting up a functional webserver and database. Then the used recourses can be transferred to the web database, which can be located in the cloud. In the test cases limited number of P&ID symbols were used. Better handling of the symbols accessed and modified by end-user must be created.

As defined, the software is based on industry standards such as DEXPI, which has not yet been integrated. Future work will be focused on improving the connection to DEXPI.

DEXPI uses Proteus XML as mentioned in chapter 2.3.1. It will be possible to use this format with the developed software as well.

One of the main requirements were that the software is easy to use. To fulfill this requirement changes to make the software more practical should be done. The main point of the project was not to focus too much on the outlook or the similarities to other software in the same field. This matter will be part of future work. Attention should be made to development that it will function with all current and future browsers and will not be affected by browser features.

When implementing the software, the primary programming rules were used. However, the code should be cleaned up and make easier to modify and understand by other developers. Also, commenting needs to be added. In the HTML- file, CSS and all of the elements are maintained in the same file. When the software extends, for easy navigation, every section should be placed on its files—for example, separate files should be created for HTML and CSS.

The drawing was implemented on an HTML canvas element. This technique may be switched in the future to SVG or webGL. The reasons not to use these options during this thesis are mentioned in chapter 4. A drawing mode is automatically started when the software is opened at the moment, but in the future, there will be a login page that will protect the software. Once logged in, all of the software's functions will be available. The login action is also linked to the web database, where both symbol files and code files will be stored. The login action is part of the web database setting process.

Bibliography

Agrawal, D. & Abbadi, A., 1995. Ordered sharing: A new lock primitive for database systems. Information Systems. Vol 20(5), pp. 361-392. DOI: 10.1016/0306-4379(95)00020-5

Ahadi, A., Prior, J., Behbood, V. & Lister, R., 2016. Students' Semantic Mistakes in Writing Seven Different Types of SQL Queries. In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '16). Association for Computing Machinery, New York, NY, USA, 272–277. DOI: 10.1145/2899415.2899464

Bitton, R., Feher, C., Shabtai, A., Shugol, G., Tikochinski, R. & Kur, S., 2017. A Proxy-Based Solution for Securiting Remote Desktop Connections in Mission-Critical Systems. IEEE 18th International Symposium on High Assurance Systems Engineering (HASE), 2017, pp. 153-156. DOI: 10.1109/HASE.2017.38

Chakraborty, S. & Adhikari, S., 2020. Machine Learning-Based Digital Twin for Dynamical Systems with Multiple Time-Scales. A Preprint[online], Vol. 243, pp. 2. DOI: 10.1016/j.compstruc.2020.106410

Dal Pont, J., 2012. Process Engineering and Industrial Management, John Wiley & Sons, Incorporated, Somerset. Available from: ProQuest Ebook Central. Available from: http://site.ebrary.com/id/10671554

Data Exchange in the Process Industry (DEXPI), 2020. Available from: http://www.dexpi.org/

Dietz, M., Putz, B. & Pernul, G., 2019. A Distributed Ledger Approach to Digital Twin Secure Data Sharing. Data and Applications Security and Privacy XXXIII. Springer International Publishing. pp. 281-300. DOI: 10.1007/978-3-030-22479-0_15

Dorman, M., 2020. Introduction to Web Mapping. Available from: http://132.72.155.230:3838/js/html.html [viewed:10.10.2020]

Dorofeev, D. & Shestakov, S., 2018. 2-tier vs. 3-tier Architectures for Data Processing Software. In Proceedings of the 3rd International Conference on Applications in Information Technology (ICAIT'2018). Association for Computing Machinery, New York, NY, USA, 63–68. DOI: 10.1145/3274856.3274869

Functional and Nonfunctional Requirements: Specification and Types, 2018. Altexsoft Software R&D Engineering. [online] Available from: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/

Ghulam M., Abad A. S., Khadim H. A. & Amjad A., 2007. A Strategy for Testing of Web Based Software. Information Technology Journal. Vol. 6, pp 74-81. DOI: 10.3923/itj.2007.74.81

Guntupally, K., Devarakonda, R. & Kehoe, K., 2018. Spring Boot based REST API to Improve Data Quality Report Generation for Big Scientific Data: ARM Data Center

Example. IEEE International Conference on Big Data. DOI: 10.1109/BigData.2018.8621924

Handlock, K., 2007. Ajax for Web Application Developers. United States of America: Sams Publishing.

Hinduja, H., Kekkar, S., Chourasia, S. & Chakrapani, H., 2020. Industry 4.0: Digital Twin and its Industrial Applications. RIET-IJSET International Journal of Science Engineering and Technology. 8. Available from: https://www.researchgate.net/publication/343713676_Industry_40_Digital_Twin_and_its_Industrial_Applications

Holi, H., 2013. Handbook of Paper and Plane. John Wiley & Sons, Incorporated, Weinheim. pp. 1371-1373. DOI: 10.1002/9783527652495

Huang, L.N., 2013. Research and Application of Web Database Security Technology. Applied Mechanics and Materials, Vol 380-384, pp. 2306. DOI: 10.4028/www.scientific.net/AMM.380-384.2306

Huang, S., Wang, G., Yan, Y. & Fang, X., 2020. Blockchain-based data management for digital twin of product. Journal of Manufacturing Systems, Vol 54, pp. 361-371. DOI: 10.1016/j.jmsy.2020.01.009

ICT Challenges in the Plant Supply Chain, 2003. Netherlands Ministry of Economic Affairs

Jannes, K., Lagaisse, B. & Joosen, W., 2019. The Web Browser as Distributed Application Server: Towards Decentralized Web Applications in the Edge. EdgeSys. DOI: 10.1145/3301418.3313938

Kamath, V., Morga, J. & Ali M., 2020. Industrial IoT and Digital Twins for a Smart Factory An open-source toolkit for application design and benchmarking. Global Internet of Things Summit (GIoTS). pp. 1-6. DOI: 10.1109/GIOTS49054.2020.9119497

King, P. F. & Collmeyer, A. J., 1973. Database sharing-An efficient mechanism for supporting concurrent processes. National Computer Conference. pp. 271-275. DOI: 10.1109/AFIPS.1973.72

Lee, T., Cha, J., Kim, J., Shin, J., Kim, J. & Yeom, C., 2017. Plant Modeling Based on SysML Domain Specific Language. IEEE International Systems Engineering Symposium (ISSE). Vienna. pp. 1-5. DOI: 10.1109/SysEng.2017.8088289

Li X., Du J., Wang X., Yang D., Yang B., 2020. Research on Digital Twin Technology for Production Line Design and Simulation. In: Xhafa F., Patnaik S., Tavana M. (eds) Advances in Intelligent Systems and Interactive Applications. IISA 2019. Advances in Intelligent Systems and Computing, vol 1084. Springer, Cham. DOI: 10.1007/978-3-030-34387-3_64

Lohtander, M., Ahonen, N., Lanz, M., Ratava, J. & Kaakkunen, J., 2018. Micro Manufacturing Unit and the Corresponding 3D-Model for the Digital Twin. Procedia Manufacturing. Vol 25, pp. 55-61. DOI: 10.1016/j.promfg.2018.06.057

Madni, A., Madni, C., & Lucero, S., 2019. Leveraging Digital Twin Technology in Model-Based Systems Engineering. Systems. 7. 7. DOI: 10.3390/systems7010007

Miquel, S., Fonseca, I., Gaspar, H. & Vieira, D., 2020. An Open-source Library for Hydrodynamic Simulation of Marine Structures. Marine Systems & Ocean Technology. Vol 15, pp. 160–174. DOI: 10.1007/s40868-020-00083-3

Moreno-García, C.F., Elyan, E. & Jayne, C., 2019. New Trends on Digitisation of Complex Engineering Drawings. Neural Comput & Applic. Vol 31, pp. 1695–1712. DOI: 10.1007/s00521-018-3583-1

Negrino, T. & Smith, D., 2007. JavaScript & Ajax. 6th ed. The United States of America. Peachpit Press.

Nykänen, O., 2007. SVG Skaalautuva Vektorigrafiikka. Jyväskylä: WSOY

Oberle, D., Staab, S. & Volz, R., 2004. An Application Server for the Semantic Web. WWW2004, May 17–22, New York, New York, USA. pp. 220-221. DOI: 10.1145/1013367.1013405

PSK Standardisointiyhdistys ry., 2016. 3605. Graphical symbols of flow and P&I diagrams for process industry. [viewed: 19.10.2020] Available from: https://psk-standardisointi.fi/

Qi, Q., Tao, F., Zuo, Y. & Zhao, D., 2018. Digital Twin Service towards Smart Manufacturing. Procedia CIRP. Vol 71, pp. 237-242. DOI: 10.1016/j.procir.2018.03.103

A. L. Opdahl, A. L., 2010. A Platform for Interoperable Domain-Specific Enterprise Modelling Based on ISO 15926. 14th IEEE International Enterprise Distributed Object Computing Conference Workshops. pp. 301-310. DOI: 10.1109/EDOCW.2010.29

Queiros, R., 2018. CSS Preprocessing: Tools and Automation Techniques. Information. Vol 9. DOI: 10.3390/info9010017

Rahul, R., Paliwal, S., Sharma, M. & Vig, L., 2019. Automatic Information Extraction from Piping and Instrumentation Diagrams. In Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods, pp. 163-172. DOI: 10.5220/0007376401630172

Rajput, S. & Singh, S. P., 2019. Industry 4.0 − challenges to implement circular economy. Benchmarking an International Journal. DOI: 10.1108/BIJ-12-2018-0430

Rawat, R. & Yadav, R., 2021. Big Data: Big Data Analysis, Issues and Challenges and Technologies. IOP Conf. Series: Materials Science and Engineering. Vol 1022. DOI: 10.1088/1757-899x/1022/1/012014

Raza, M., Jabeen, T., Raza, B., Raza, S. & Abbas, M., 2020. Using Combined List Hierarchy and Headings of HTML Documents for Learning Domain-Specific Ontology. International Journal of Advanced Computer Science and Applications. Vol 11(4). Available from: https://thesai.org/Downloads/Volume11No4/Paper_31-Using_Combined_List_Hierarchy_and_Headings_of_HTML.pdf

Rica, E., Moreno-García, C., Álvarez, S. & Serratosa, F., 2020. Reducing Human Effort in Engineering Drawing Validation. Computers Industry. Vol 117. DOI: 10.1016/j.compind.2020.103198

Rinker, J., 2017. P&IDs (Piping & Instrumentation Diagrams) and P&ID Valve Symbol Library. Assured Automation. Available from: https://assuredautomation.com/news-and-training/pids-piping-instrumentation-diagrams-and-pid-valve-symbol-library/ [viewed: 10.10.2020]

Rodríguez-Aguilar, R. & Marmolejo-Saucedo, J., 2020. Conceptual framework of Digital Health Public Emergency System: digital twins and multiparadigm simulation. DOI: 10.4108/eai.13-7-2018.164261

Roldán, M.L., Gonnet, S. & Leone, H., 2013. Knowledge representation of the software architecture design process based on situation calculus. Expert Systems, 30: 34-53. DOI: 10.1111/j.1468-0394.2012.00620.x

Schleich, B., Anwer, N., Mathieu, L. & Wartzack, S., 2017. Shaping the Digital Twin for Design and Production Engineering. CIRP Annals. Vol 66, pp. 141-144. DOI: 10.1016/j.cirp.2017.04.040

Schuller, A. & Epple, U., 2012. PandIX – Exchanging P&I Diagram Model Data. Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation. pp. 1-8. DOI: 10.1109/ETFA.2012.6489537

Smith, P.R & Van Laan, TJ, 1987. Piping and Pipe Support Systems. The United States of America: McGraw- Hill.

Thau, R., 1996. Design considerations for the Apache Server API. Computer Networks and ISDN Systems. Vol 28(7–11). pp. 1113-1122. DOI: 10.1016/0169-7552(96)00048-7

Toghraei, M., 2014. Principles of P&ID Development. Chemical Engineering. Vol 121(4), pp. 62-71. Available from: https://www.scribd.com/document/438039313/Principles-of-P-ID-Development

Wagg, D., Worden, K., Barthorpe, R. & Gardner, P., 2020. Digital Twins: State-of-the-Art and Future Directions for Modeling and Simulation in Engineering Dynamics Applications. Risk Uncertainty Part B. DOI: 10.1115/1.4046739

Walker, J. & Chapra, S., 2014. A client-side web application for interactive environmental simulation modeling. Environmental Modelling & Software. 55. pp. 49–60. DOI: 10.1016/j.envsoft.2014.01.023

Ward, M., 2002. A template for CALL programs for endangered languages. Doctoral dissertation, Dublin City University. School of Computing

Worden, K., Cross, E. J., Barthorpe, R. J., Wagg, D. J. & Gardner, P., 2000. On Digital Twins, Mirrors, and Virtualizations: Frameworks for Model Verification and Validation. DOI: 10.1115/1.4046740

Zhong, K., Yun, Z. & Huang, J., 2006. A Model of Geography Information Service Based on Application Server. IEEE International Symposium on Geoscience and Remote Sensing. pp. 2852-2856, DOI: 10.1109/IGARSS.2006.733

Yan-feng, Z. & San-jun, L., 2013. Research and design of web application framework based on AJAX. Proc. SPIE 8768, International Conference on Graphic and Image Processing (ICGIP 2012), 87684B. DOI: 10.1117/12.2005860

Yu, E., Cha, J., Lee, T., Kim, J. & Mun D., 2019. Features Recognition from Piping and Instrumentation Diagrams in Image Format Using a Deep Learning Network. Energies. Vol 12. DOI: 10.3390/en12234425

Yu, E., Cha, J., Lee, T., Kim, J. & Mun, D., 2019. Features Recognition from Piping and Instrumentation Diagrams in Image Format Using a Deep Learning Network. Energies. 12(23), pp. 4425. DOI: 10.3390/en12234425