Abbas, Sarmad Saeed

Bachelor's thesis

# The Optimal Train Reference

## A Simplified ERTMS Data Analysis

**May 2020**

**NTNU**
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU
Norwegian University of
Science and Technology

Bachelor's thesis

2020

NTNU
Norwegian University of
Science and Technology

Abbas, Sarmad Saeed

# The Optimal Train Reference

A Simplified ERTMS Data Analysis

Bachelor's thesis
May 2020

**NTNU**
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

# Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

| *Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:* | | |
|---|---|---|
| **1.** | **Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.** | ☒ |
| **2.** | **Jeg/vi erklærer videre at denne besvarelsen:**<br>• ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands.<br>• ikke refererer til andres arbeid uten at det er oppgitt.<br>• ikke refererer til eget tidligere arbeid uten at det er oppgitt.<br>• har alle referansene oppgitt i litteraturlisten.<br>• ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse. | ☒ |
| **3.** | **Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§14 og 15.** | ☒ |
| **4.** | **Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver** | ☒ |
| **5.** | **Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens studieforskrift §31** | ☒ |
| **6.** | **Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider** | ☒ |

# BACHELOR THESIS - REPORT

NTNU
Department of ICT and Natural Sciences

| TITLE: |
|---|
| The Optimal Train Reference – A Simplified ERTMS Data Analysis |

| CANDIDATE NUMBER(S): |
|---|
| 10017 |

| DATE: | SUBJECT CODE: | SUBJECT: | DOCUMENT ACCESS: |
|---|---|---|---|
| 20.05.2020 | IE303612 | Bachelor's thesis | Open |

| STUDY: | TOT PAGES/ATTACHMENTS: | LIB. NR: |
|---|---|---|
| Engineering, Computer Science | 75/6 | -- Not in use -- |

| SUPERVISOR(S) : |
|---|
| Main supervisor: Ibrahim A. Hameed<br>Secondary supervisor: Anniken Susanne T. Karlsen |

SUMMARY:

The Optimal Train Reference – A Simplified ERTMS Data Analysis is a bachelor thesis conducted by a student in the field of engineering – computer science. The project was given by Abraham Mosye, a senior engineer at the Norwegian national railway infrastructure management, Bane NOR.
The purpose of the project was to simplify parts of the data analysis process of the new databased railway control and traffic management system, ERTMS. The goal of this simplification was mainly related to the detection of adverse events and deviations as well as errors and weaknesses on train or the system.

As a result of the project, analysis software and algorithms, mainly meant for data comparison, were developed. The final result consisted of two parts. A reference part, which was about finding and storing a reference train for each train-line, which is then used as an "optimal trip" for comparison with other train trips of the same train-line. A visualization part, which was about developing a web-based application for train data visualization and comparison.

This thesis focuses on the planning and the development process of the software and algorithms, and covers different aspects of the project including relevant theoretical basis, methodologies and technologies used. The results, findings and uncertainties of the final developed product are also discussed at the end of the thesis.

*This thesis is an examination answer conducted by students at NTNU.*

**Postadresse**
NRNU i Ålesund
N-6025 Ålesund
Norway

**Besøksadresse**
Larsgårdsvegen 2
**Internett**
www.hials.no

**Telefon**
70 16 12 00
**Epostadresse**
postmottak@hials.no

**Telefax**
70 16 13 00

**Bankkonto**
7694 05 00636
**Foretaksregisteret**
NO 971 572 140

# PREFACE

In the information age, software, IT and digitalization has become a fundamental and an essential part of many industries. Many of the older industries that have existed over many decades are moving towards digitalization and are becoming more dependent on IT systems. In recent years, railway transport and other collective transport services have started moving towards that direction.

This document is a bachelor thesis and a final assignment of the study program engineering – computer science at the Norwegian university of science and technology. The project was assigned by Bane NOR the company responsible for the Norwegian national railway infrastructure, previously known as "Jernbane verket" (JBV).

I would like to thank everyone who assisted during the project in one way or another. I would especially like to thank senior engineer Abraham Mosye for providing me with the opportunity to work on ERMTS and train data and for coming up with the project idea.

I would also like to thank both supervisors, Anniken Susanne T. Karlsen and Ibrahim A. Hameed for guiding me throughout the project, providing help with different aspects of the project and for providing feedback on this document and the final product produced at the end of the project.

# CONTENT

# ABSTRACT

The Optimal Train Reference – A Simplified ERTMS Data Analysis is a bachelor thesis conducted by a student in the field of engineering – computer science. The project was given by Abraham Mosye, a senior engineer at the Norwegian national railway infrastructure management, Bane NOR.

The purpose of the project was to simplify parts of the data analysis process of the new databased railway control and traffic management system, ERTMS. The goal of this simplification was mainly related to the detection of adverse events and deviations as well as errors and weaknesses on train or the system.

As a result of the project, analysis software and algorithms, mainly meant for data comparison, were developed. The final result consisted of two parts. A reference part, which was about finding and storing a reference train for each train-line, which is then, used as an "optimal trip" for comparison with other train trips of the same train-line. A visualization part, which was about developing a web-based application for train data visualization and comparison.

This thesis focuses on the planning and the development process of the software and algorithms, and covers different aspects of the project including relevant theoretical basis, methodologies and technologies used. The results, findings and uncertainties of the final developed product are also discussed at the end of the thesis.

# TERMINOLOGY

## Terms

| | |
|---|---|
| GSM-R | A closed mobile network developed for railways in Europe |
| Big data | Large collections of data |
| Movement authority | Authorization given to train to move to a given position |
| Balise | An electric device (beacon), which lies in between the rails of a railway. When train drives over a it, it transmits data to train telling the train it's position |

## Abbreviations

| | |
|---|---|
| ERTMS | European Rail Traffic Management System |
| ETCS | European Train Control system |
| GSM-R | Global System for Mobile Communication - Railway |
| RBC | Radio Block Center |
| RIU | Radio Infill Unit |
| AE | Absolute Error |
| MAE | Mean Absolute Error |
| SPA | Single Page Application |
| VCS | Version Control System |
| JSX | JavaScript XML |
| UML | Unified Modeling Language |
| HTML | Hypertext Markup Language |
| CSS | Cascading Style Sheets |
| JSON | JavaScript Object Notation |

# FIGURES

# TABLES

# SNIPPETS

# 1 INTRODUCTION

## 1.1 Background

Norwegian railway signaling infrastructure is being digitalized. According to the official website of Bane NOR [1], by year 2034, Norway will become the first country in the world where the entire train traffic is controlled via a data center. A new railway control and traffic management system called ERTMS (European Rail Traffic Management System) is being implemented and will take effect. Already today, there is one line in Norway that is driven by ERTMS, mainly used as a pilot line. The eastern line (Norwegian: "Østre linje") which extends from Ski to Sarpsborg.

Briefly, ERTMS is about creating a common European railway digital signaling and management system by removing all signals and signs from the railway, and replacing them with ERTMS digitalized signaling. The train obtains information about allowed speed, movement authority, restrictions and other information as data sent to it from a data center through a closed network. The train also communicates to the data center and reports information such as current position, speed, train mode, errors if any, and other important information [2].

The continuous communication between the data center and the train throughout the trip produces large collections of data (big data). This data can be used for different proposes such as real-time train monitoring, but is also stored in databases as historical data and used mainly for analysis purposes.

Analyzing these large collections of data can lead to many improvements in the railway and contribute to the avoidance of adverse events and deviation. By analyzing data, deviation along with errors and weaknesses on the train as well as the system can be detected or even prevented before they occur. An example of such errors are speedometer errors and wheel diameter errors on the train. When detected, train can be taken out for calibration or adjustments.

To analyze this data, appropriate analysis tools are a requirement. The current analysis tools used does however not offer every desired functionality or present data in desired way, thus may make the analysis process harder and time consuming. With the help of the existing advanced data technology available today, the analysis process can be improved, simplified or even automated partly or completely. Thus, it is desired to develop software and algorithms for comparison of train data to easily detect deviation and to simplify the total analysis process.

This thesis is about developing such software and algorithms to improve and simplify the analysis process of historical train data.

## 1.2 Research Topic and Objective

The client of this project, Bane NOR, the company responsible for the Norwegian national railway infrastructure, wants to develop additional and better data analysis tools for their big data platform.

For analyzing large collections of data, existing tools and software does not offer every functionality desired and does not present data in desired way, which makes the analysis process harder and time consuming.

Because of this, Bane NOR wants to develop simple-to-use analysis software with desired functionality to illuminate such issues. This software can contribute to the reduction of the analysis time significantly by up to 80% and improve the overall analysis process.

The company has specific requirements for the software to be developed of which a description is provided in the next section.

For this project, the focus is mainly going to be on the Eastern line as it is currently the only line in Norway where ERTMS is in use.

Before presenting the requirement specification for the software, basic understanding of the railway and operations in the Eastern line may be useful. This information is presented in chapter 2.5.4. It may be useful to have a quick read through before reading the requirement specification.

## 1.3 Requirement Specification

The requirement specification provided by the client, Bane NOR, can be divided into two parts.

In the first part, it is required to develop one or multiple algorithms to find a train reference with the most optimal speed for each unique train line. This means for every train line, a reference speed should be found and stored in a database, which then will be used in the second part of this project.

To make this more clear, the client provides a map of planned/allowed speeds on each section of the Eastern line. Using information in the map, the algorithm should find the train that have the closest speed to the allowed speed on each section in the given map.

The algorithm should run continuously, find and update existing references (each unique train-line has own reference) if a new and better one is found later in time.

Thus, a reference can be thought of as “the best version of this train line” in a given period of time, from a given day x, to a given day y. That period of time is from the day the algorithm starts running and on.

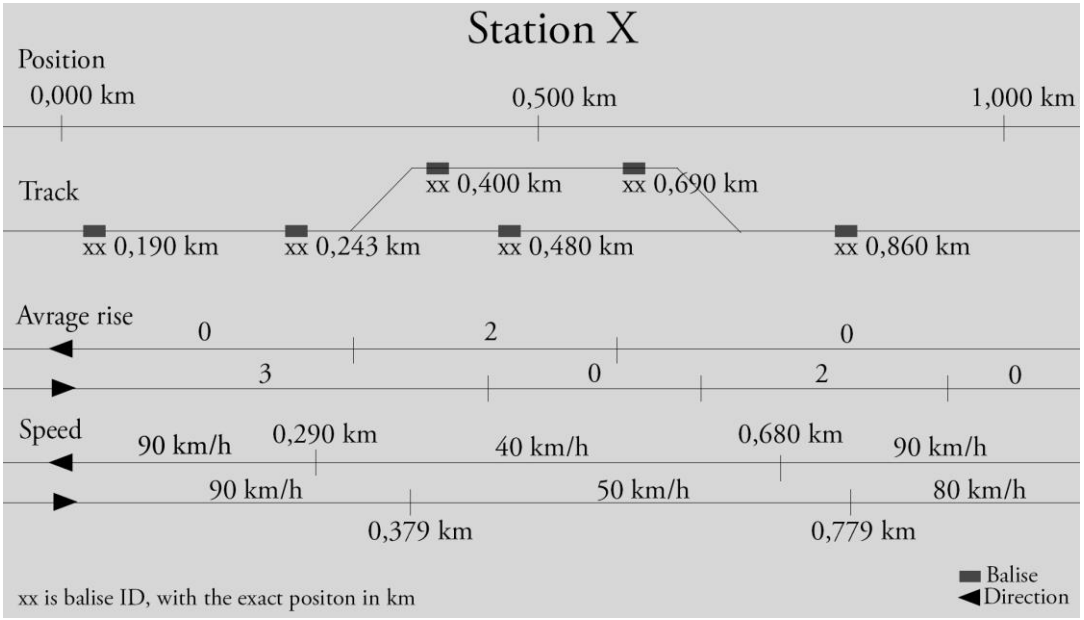



Figure 1.1: Map information of a single station

The provided map contains information about speeds in different sections, tracks and directions of the Eastern line. The map also contains other detailed information about all balises (a unit used for positon, described in 2.5.4) with their exact location placement in kilometer (2.5.4), marker board information, average rise, track information, and other information that may not be need for this task. Figure 1.1 shows a simplified example of such a map for one station. Refer to section 2.5.4 for explanation of position, balise, track, etc.

In the second part, it is required to develop software that makes it easy to search, view and compare data of a given train to data of the reference that was found in the first part of the project. The software should be web based, simple and easy-to-use.

When it comes to functionality, the user should be able to enter a train number (unique identifier of a train-line), select date, and then compare the data of train provided as input to the data of stored reference of that train-line.

When comparing the two datasets, the speed of both trains should be plotted on a graph, and other information should also be presented on that graph.

Information that should be presented on the graph when comparing the two datasets, is the following:

- Train speed (plotted as a graph)
- Train position in kilometers
- Balises passed during the trip
- Modes train entered during the trap
- Stations train been through during the trip
- Movement authority train received during the trip.

## 1.4 Scope

Since there is limited time to carry out the project, it may not be possible to implement a lot of functionality and advanced features. In such a project, it is hard to estimate how much can be done during this short period of time, so it may not be possible to implement every requirement specified in the requirement specification given by the client. Therefore, the client decided to set some limitations in case time becomes a problem.

There could be many other factors to decide the most optimal reference for a train line, such as when did the train start braking (braking curve), when did it start driving, whether it is late or early and many other factors. Each factor could have an assigned weight. The weigh describes how much we value that factor. However, since there is limited time to carry out this project, things are kept short and simple. It is only required to use one factor, that is the speed of the train. The speed will count as 100% of the weight when finding reference.

The first part, that is finding the reference can be limited to only finding reference for two trains, each running in different direction, instead of all trains that run during the day. This means we only find on reference for one train running in the nominal direction, and one reference for one train running in the reverse direction.

In case time becomes no problem and all feature are implemented early, the client may provide more features to be implemented and integrated in to the software.

## 1.5 Report Structure

This report is divided into six sections, which further are divided into logical subsections to organize the report and to simplify the reading. Below a description of each of these section/chapters is provided.

In this chapter we provided an introduction, introduced the topic/issue, presented the requirement specification and set the scope of the project.

In chapter 2, theoretical basis for this project is provided. That is where theory used in this project or considered is described. This includes software development methodology, software engineering principles, software design principles, software architectures, technologies and frameworks, basic knowledge about ERTMS and railway along with mathematical theories.

In chapter 3, project organization, project planning along with tools, technologies and data used in this project are descried.

In chapter 4, results of this project are presented. The solution, the design, the architecture and other important aspects of the final product are all presented in that section.

In chapter 5, results achieved and solutions implemented along with some uncertainties in these results and solutions are discussed. In addition to that, the development process and further work is discussed in that section.

In chapter 6, final words and a conclusion of this project is drawn.

# 2 THEORY

## 2.1 Software Development

Software is a recent development in human history. In today's age, software, IT and digitalization has become a fundamental part of many industries. Software projects are getting larger and effecting larger parts of organizations.

Software development as described in an article published by IBM [3] is a **process** of creating, designing, publishing and maintaining software applications.

In the following subsections, the process of software development will be discussed, particularly focusing on agile software development and scrum.

### 2.1.1 Agile Software Development

Large software projects often go wrong and has a high rate of failure. According to a study presented by McKinsey & Co., in conjunction with the University of Oxford, 17% of large IT project go so badly that they threaten the existence of the company. In the same study, it is mentioned that half of large IT projects blow their budgets, and on average, large IT projects run 45% over budget and 7% over time while delivering 56% less value than predicted. [4]

Another study by IBM states that only 40% of projects meet schedule, budget, and quality goals. While a study by KPMG New Zealand found that 70% of organizations have suffered at least one project failure in prior twelve months. [5]

It may be hard to attempt to find out the exact reason behind all of that. However, there are many challenges when developing software projects, which may be part of the reason to software projects going badly and may contribute to the failure of a software project.

Software projects traditionally has been developed using the classic development methodology waterfall or similar. In 1969, the classic waterfall model was first used to describe a model for developing large and complex software projects. In 1970, Dr. Winston W. Royce described an unmodified waterfall model which consisted of phases with no iterations. These phases included the following: System requirements, Software Requirements, Analysis, Program Design, Coding, Testing, and Operations [6].

The waterfall model did however not accept changes in plan, and is fixed when it comes to time and scope. One criteria to the success of a software project is continuous adjustments of the project plan, the requirements and the budget [7]. The classic waterfall does not allow for such adjustment.

This along with other challenges such as lack of standards within the software and IT industry, incompatibility between implementation languages and other similar challenges may be the cause of failure of many software projects.

In response to the traditional way of handling software and challenges caused by some development methods such as the classic waterfall, the term **agile** software development was introduced.

Agile software development is a term for a set of practices and frameworks used to develop software. These practices and frameworks are based on principles and values that were introduced in the agile software development manifesto in 2001 and the twelve principles behind that manifesto [8].

The agile software development manifesto [9] states the following:

> ***Individuals and interactions*** *over processes and tools*
>
> ***Working software*** *over comprehensive documentation*
>
> ***Customer collaboration*** *over contract negotiation*
>
> ***Responding to change*** *over following a plan*
>
> *That is, while there is value in the items on the right, we value the items on the left more.*

In other words, both sides of the quote in the agile software development manifesto has value, but the left side of the quote has more value than the right side. Thus, individuals and interactions has more value in agile software development than processes and tools. Agile software development focuses more on developing working software, than writing comprehensive software and writing documentation for the software. It values collaboration with customer more than contract negotiation, accepts and values changes over following a fixed plan.

### 2.1.2 Scrum

Scrum is an agile software methodology and framework for product development management, which was first presented in 1995 by Jeff Sutherland and Ken Schwaber at the Oopsla conference in Austin, Texas (US) [10].

In the official scrum guide by Jeff Sutherland and Ken Schwaber scrum is described as, lightweight, simple to understand but difficult to master. Jeff Sutherland and Ken Schwaber further mention that the scrum framework has been in use since 1990 to manage complex products and work. [11]

#### Team

In scrum, a team consists of tree roles, a product owner, a scrum master, and development team. Each has own responsibilities.

The product owner is usually the client, the one who will receive the final product or who has the authority to it. The product owner is the person responsible for managing the product backlog, that is removing requirements, adding requirements, or changing the priority of requirements and ensuring the development team understands the requirements.

Scrum-master can be thought of as the leader. This role is not exactly about leading, rather, it is about assisting and couching the scrum team ensuring that the team has the understanding of scrum roles, practices and values.

Development team is the team who does the work. That is the development of the product the product owner required. The development team is responsible for releasing increment of "Done" product at the end of each sprint.

**Events**

In scrum, there are multiple events that occur over multiple iterations. These are sprints, daily scrum and sprint reviews.

A sprint as defined in the scrum guide is a time-box of a short period of time (a month or less) during which a release of something "Done" which is part of the product is created. All sprints are consistent in the time duration, and when one sprint ends, another starts after it.

Before starting a sprint, the sprint needs to be planned. This is referred to as sprint planning in scrum. When sprint planning, work to be performed during the sprint is planned and a goal for the sprint is set, this goal is called a sprint goal.

Daily scrum is a short daily meeting (15 minutes) for the development team where the team can discuss the progress. Each member can for example answer some question such as for example the following:

- What did I do yesterday to contribute to the sprint goal?
- What will I do today to contribute to the sprint goal?
- Are there any obstacles in the way that I see can prevent me or the team from achieving the sprint goal?

At the end of each sprint, a sprint review/meeting is held. In this meeting, the product owner, the scrum team, scrum master and stakeholder meet. During the meeting, the work of the team of that sprint will be demonstrated, the product backlog may change with more or less requirements and priority changes may take place.

**Scrum artifact**

Product backlog is a list of requirement that are desired or needed in the final product. This is a list the product owner is responsible for. The product owner can change the content of it by adding or removing requirements. The product owner is also able to prioritize the requirement in the backlog.

Sprint backlog is a log which contain some chosen requirements from the product backlog to be implemented during a sprint.

Everything mentioned above is according to the official scrum guide by creators of scrum, Jeff Sutherland and Ken Schwaber. [11]

## 2.2 Software Engineering and Design Principles

### 2.2.1 Separation of Concerns

Developing software systems may be a hard task, depending on the requirements and the level of complexity. In a research-paper published in ResearchGate by Hafedh Mili and Hamid Mcheick [12], Université de Montréal in Canada, a basic software engineering principle called separation of concerns was discussed. In the paper, they define the principle as the following:

> *"Separation of concerns" is a general problem-solving idiom that enables us to break the complexity of a problem into loosely-coupled, easier to solve, sub problems.*

That is, when developing software systems, concerns should be separated in to smaller problems or parts, which in return will break the complexity and make the task or tasks easier to solve and maintain.

According to [13], a program is called modular when it embodies the separation of concerns principle well, that is a program consisting of multiple separated modules which work together. It is also mentioned that layered design where a system consists of different layers or tiers such as for example a presentation layer, logic layer, data access layer is an "embodiment" of the principle of separation of concerns.

### 2.2.2 User-centered Design

Designing a product with users in mind is an important aspect of software development.

Don Norman, one of the leading thinkers on human-centered design, in his book "The Design of Everyday Things" provides six design principles to designing technology products.

According to an article published on medium [14], Don Norman describes in his book **visibility** as the first principle, that is if an element is more visible, users will more likely know about them and how to use them. And the opposite also applies, that is the less visible an element is, the less likely users will know about it or use it.

The second principle by Don Norman according to the medium article [14] is **feedback**. This principle is about telling the user in one way or another about the actions that have been taken and their results. The important key of this principle is to never leave the user guessing on the action taken and the results of that action.

The third principle described in the article is **constraint**, which is about limiting the actions allowed to provide a cleaner and simpler interface to users, as unlimited possibilities often leave users in confusion.

The fourth design principle is **mapping**. This principle is about showing relationship between controls and their effect.

The medium article also mentions **Consistency** as the fifth design principle by Don Norman. This principle as described in the article is about being consistent in the design, this can be for example having similar style across the application, with similar buttons and theme colors.

The sixth and last design principle by Norman is **affordance**. The Affordance principle is about giving the user a clue. For example, a keyboard key gives a clue that it can be clicked. If something has strong affordance, it means it is very clear how to use that thing.

## 2.3 Software Architecture and Models

Since the formation of the computer science field, developing software encountered many issues related to complexity.

As a concept, software architecture has its origins in the research of two scientests, Edsger Dijkstra in 1968 and David Parnas in the early 1970s. They both underlined that getting the structure of software systems right was an important and critical aspect. Research institutions were an important factor to forwarding software architecture as a discipline in the 1990's.

It was not until 2007 that IEEE 1471-2000, "Recommended Practice for Architecture Description of Software-Intensive Systems" was adapted by ISO as the first formal standard in the area of software architecture. [15]

Today, when developing software systems, there are many architectures a software can be structured by. Many of these architectures has become more important as systems increase in complexity.

### 2.3.1 Client–server Architecture

One of the more popular architectures in software is the so-called client-server architecture. The client-server architecture is a structure for distributed applications where a service provider, that is the server, and a requester, which is the client requesting data or a service, are physically separated and communicate through a network. When the client needs a resource or data, it sends a request to the server, the server then processes the request and sends a response to the requesting client [16].

In the web development field, the client is often referred to as the frontend and the server is often referred to as the backend [17].

### 2.3.2 Multitier Architecture

In software engineering, the term multitier architecture, also referred to as n-tier architecture is a software architecture which according to [18] is based on server-client architecture described in 2.3.1 where different tiers or parts of a system are physically separated.

In this form of architecture, data presentation, data processing and data management are separated in different tiers.

One of the most widespread forms of multitier architecture is the three tier architecture. This form of architecture was developed by John J. Donovan in Open Environment Corporation (OEC), which is a tools company he founded in Cambridge, Massachusetts.

The three-tier architecture as described in [18] is an architecture where three tiers exist, that is a presentation tier, a logic tier and a data tier.

Each of the three tiers are responsible for one part of the problem. Presentation tier is responsible for data presentation and providing a user interface, logic tier is responsible for handling requests, processing data, applying business logic and performing calculations, while data tier is responsible for data querying, data management and data storage.

As described in 2.2.1, multitier architectures applies the separation of concerns principle by de-coupling different parts of the system apart from each other. This also means that multitier architecture has the benefit of reusability according to [18]. By segregating an application into tiers, it is possible to modify or add a new tier without having to rework the entire system.

### 2.3.3 Layered Architecture

A software program can be segregated into different parts called layers.

According to *Software Architecture Patterns* by Mark Richards [19] in chapter 1, layered architecture, layers in a layered software architecture are a collection of components where each layer perform a role within the complete application.

In a layered application, there are no specific number of layers defined, according to [19]. Some applications may consist of five layers, others may consist of three layers and so on.

According to an article published on medium [20], the most common type of layered architecture is 3-layered architecture.

The three layers in a 3-layered architecture are:

- **Presentation layer**: a connection to other applications (exposing functionality thought URI's). This layer depends on the application layer.
- **Application layer**: provides services and functionality (business logic). Depends on data access layer.
- **Data access layer**: queries, inserts, deletes and updates data in database (persistence). Does not depend on any of the layers.

Naming may vary depending on the source of information. In oracle documentation of J2EE application layers [21], they refer to the application layer as business logic layer, other sources refer to data access layer as persistence layer [19].

### 2.3.4 Layer vs Tier

There seems to be a lot of confusion when it comes to layers and tiers. These two terms seem to be used interchangeably in many sources. However, these two terms has a clear difference.

According to [18] and [22], a layer is the **logical** segregation of the code in a program or an application, while a tier is the **physical** separation of components that make up a system.

Example of physical separation is distributable application such as a frontend and a backend. Example of logical is the separation for example of code within the frontend application or the backend application.

### 2.3.5 Single Page Application Model

A single page application (SPA) is an application model (usually a web application) where an application consists of only one page. In the context of web, only one HTML file exists, that is index.js. This file changes the content dynamically through JavaScript depending on the logic provided by that script. [23]

## 2.4 Technologies

When developing software applications, there are a set of technologies and frameworks which can be used to simplify the development process, improve system and application design as well as reduce the development time. This section provides theoretical background on technologies and frameworks essential to the development of this project.

### 2.4.1 Spring Framework

Written by Rod Johnson, Spring [24] is an open-source framework as well as an inversion of control container for the Java platform. The first version of the framework was released in 2002 and became a popular replacement or addition to the enterprise JavaBeans. The Spring framework is modular and has extensions for building web applications. However, it also has some core features that can be used in any java application.

### 2.4.2 React Framework

React.js is a JavaScript library/framework maintained by Facebook. It is used to create interactive user interfaces for web application or mobile apps (React Native). React is component based which means it makes it possible to create reusable components to reuse throughout a project. [25]

### 2.4.3 Babel

Babel is JavaScript compiler and transpiler. It produces browser compatible JavaScript which makes a script able to run on any browser, even on old browsers with older JavaScript engines. That way the developer can write the new version of JavaScript, but still be able to run the script on browsers that does not support the newer version and it's features. What Babel exactly does is it converts newer JavaScript version to backward compatible version.

Babel has also other features. For example, if using the React framework, Babel transpiler is responsible for converting JSX to JavaScript. [26]

### 2.4.4 Webpack

Webpack is a static module bundler for JavaScript applications. It bundles all modules included in a JavaScript project and produces a small number of bundles, usually a single file. If using the React framework, Webpack will process React code and bundle the code in to a single JavaScript file. [27]

### 2.4.5 Lucene

Apache Lucene is a high performance Java library, which provides features related to search and indexing along with other features such as spell checking, highlighting of hits, and other more advanced features. [28]

### 2.4.6 ELK-stack

ELK-stack is a term for three open source projects developed by a search company called Elastic. The ELK-stack stands for **E**lasticsarch, **L**ogstash and **K**ibana.

**Elasticsearch**

Elasticsearch is an open source, RESTful search and analytics engine that is based on JSON. It is used as a database where logs, large collections data, structured or unstructured data are indexed and stored. When data is indexed, it is possible to run complex queries against the data, and retrieve complex summaries of the stored data using aggregations.

Elasticsearch works with indices. An index can be thought of or viewed as a database, it is a mechanism to organize and partition data [29]. Each index contains a collection of JSON documents related to each other [30].

The official website of Elastic provides some terminologies related to Elasticsearch, and how they compare to a rational database such as MySQL, these are shown as the following:

Elasticsearch => MySQL

Indices => Databases

Type => Table

Documents with properties => Rows and Columns

In newer versions of Elasticsearch, type was removed due to it being inefficient and adding some limitation to indices [31].

An example of a single document (a row) from a simple index (a table) is shown as the following:

```
1.  {
2.      "_index": "twitter",
3.      "_type": "users",
4.      "_id": "AM-g6ffat-HLl2oRYHKg",
5.      "_score": null,
6.      "_source": {
7.          "@timestamp": "2009-011-17T09:13:42",
8.          "usrType": "standard",
9.          "user_name": "john_smith87",
10.         "user": {
11.           "name ": "John Smith",
12.           "age ": 33,
13.           "email ": "johnsmith@gmail.com",
14.           "addresses ": [
15.            "address 1",
16.            "address 2",
17.            "address 3"
18.           ]
19.         }
20.     }
21. }
```

In comparison to a relational database such as MySQL, the document above would be the equivalent of a selected row and its columns from a table called users in a database called twitter, but the result is represented in JSON instead.

Elasticsearch is RESTful and exposes it's features through easy to use JSON/HTTP interface. This means that it is possible to send request to the Elasticsearch interface to insert data, fetch data, modify data, delete data or perform any other operations using standard HTTP methods such as GET, POST, PUT, DELETE and others.

An example is provided below:

```
1.  //Lists all documents in the twitter index
2.  GET localhost:9200/twitter/_search
3.
4.  //Lists all users with name John Smith
5.  GET localhost:9200/twitter/_search
6.  {
7.      "query": {
8.          "term": { "user.name" : "John Smith" }
9.      }
10. }
11.
12. //Updates name of user with the given id
13. PUT localhost:9200/twitter/users/AM-g6ffat-HLl2oRYHKg
14. {
15.   "user.name" : "Sam Nicholson"
16. }
17.
```

From the code provided above, it's possible to retrieve all documents from an index by sending a GET request to desired index URL. It is also possible to add a request body to provide a query to restrict the search results and the number of documents in the response. Other operations such as updating, adding, deleting are also done the same way using the correct HTTP method.

These requests can be sent using CURL, the Kibana tool (described below), an application such as POSTman or any other application that supports HTTP and its methods.

### Advantages of Elasticsearch

According to the official website of Elastic, Elasticsearh has a big advantage when it comes to speed, it is a fast search engine. The reason for it being fast is that Elasticsearch is built on top of Lucene (2.4.5), which makes it excel at full-text search.

Another advantage of Elasticsearch according to Elastic is that it is near real-time search, which means when a new document is indexed or inserted, it will take almost no time (something near one second) before the document is available and searchable, which makes Elasticsearch suitable for real-time data analysis and infrastructure monitoring.

Elasticsearch provides data loss security. That is because data is copied and redundant copies are distributed across different containers which in the terminology of Elastic are known as shards. That way data loss is prevented in case of failure in hardware.

Distributed data provides scalability, which allows Elasticsearch to scale out to many server and handle enormous amounts of data (could handle petabytes of data).

Other advantages of Elasticsearch are the other Elastic products such as Logstash and Kibana, which makes it easy to parse, transform, filter and insert data, also makes it possible to visualize indexed data.

### Logstash

Logstash is a data processing pipeline or a data collection engine developed by Elastic [32]. Logstash is capable of collecting or receiving data from different sources as input, then filters data and produces output that is then possible to write to a file, send to a service such as a graphing service or insert to a database such as Elasticsearch [33].

**Kibana**

Kibana is a search and visualization tool developed by Elastic which makes it possible to manage, search and visualize indexed and stored data. With kibana it is possible to create complex queries to search for specific data and it also is possible to create charts, tables, graphs, maps and many other visualization types using stored data.

## 2.5 ERTMS and Railway

In order to solve the problem in this project, theoretical understanding of ERTMS and data used in the communication between train and data center is essential. This section provides fundamental theoretical basis required and used to develop and implement a solution.

### 2.5.1 European Rail Traffic Management System

European Rail Traffic Management System (ERTMS) as described in [2] is a common signaling system for railways across Europe. The main principle of ERTMS as a system is removing all optical signals such as light signals, instead, replacing them with data. The train receives important information such as movement authority and allowed speed as data from a data-center referred to as RBC, short for radio block center. RBC communicates to the train wirelessly through a closed network developed for railways in Europe called GSM-R [34]. The train also communicates to the radio block center constantly and reports different types of information such as current position, current speed, running direction, train mode (2.5.3) and other.

ERTMS is illustrated below in Figure 2.1: ERTMS overview.

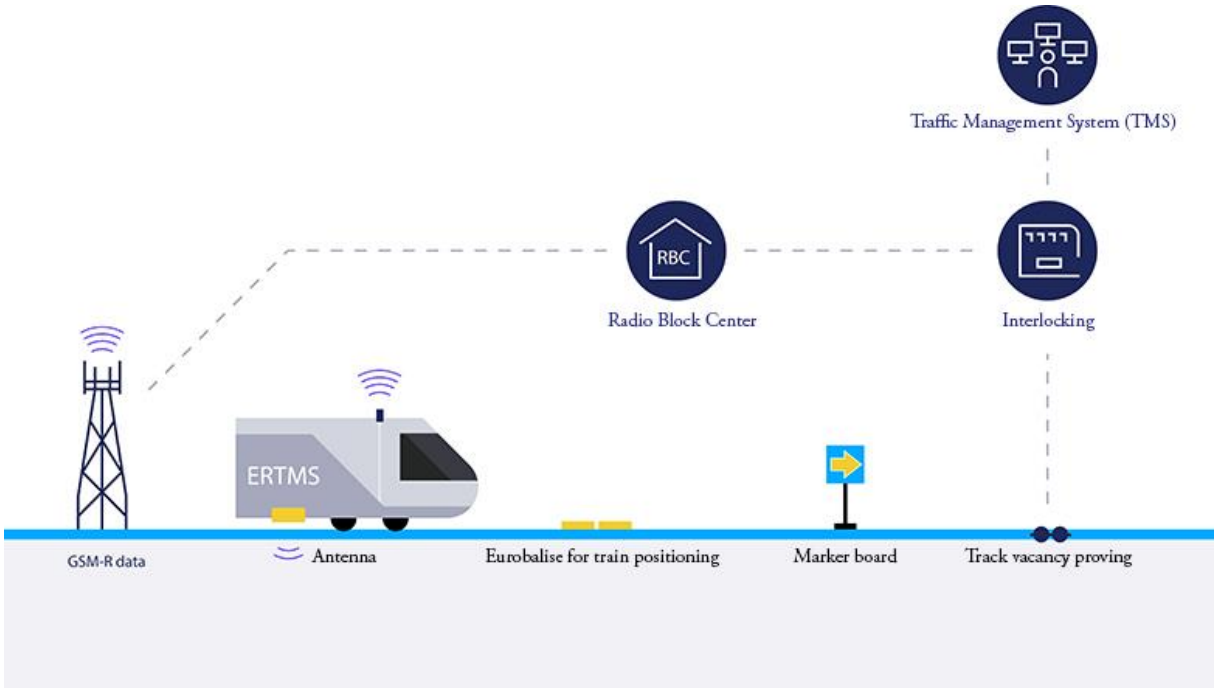



Figure 2.1: ERTMS overview [35]

### 2.5.2 ERTMS Communication and Language

According to [2], and as explained in 2.5.1, RBC and train exchange information and communicate continuously providing useful details to each other in form of data.

In order for different parties to understand each other, a common language and standard is used throughout the communication session. This language is referred to as "The ERTMS/ETCS Language" and is based on variables, packets, messages and telegrams as described in the official documentation of the *European Union Agency for Railways*, *subset 26* [36].

#### Variables

The definition of a variable in the ERTMS/ETCS Language according to [36] is an encoded single data value which cannot be split in minor units and has one meaning and type.

There are many different types of variables in the ERTMS/ETCS Language, each representing a single property and is a container of a single value, such as speed value, train direction, train mode, and so on.

In the ERTMS/ETCS Language, each variable is assigned a prefix, where each prefix has own meaning. Below is a selection of prefixes obtained from the official documentation [36].

| Prefix | Meaning/Definition |
|---|---|
| D_ | Distance |
| NID_ | Identity number |
| T_ | Time |
| V_ | Speed |

Table 2.1: A selection of variable prefixes in the ERTMS/ETCS Language

According to the official documentation, train speed value can for example be obtained by reading variable V_TRAIN, where V_ as listed in Table 2.1 stands for speed, thus V_TRAIN means "train speed". Same logic applies to all other variables in the language.

#### Packets

According to [36], a packet is a container for a group of variables with a defined internal structure. In other words, a packet is a form of data structure that holds a defined set of variables where each variable holds a data value.

There are different packet types used in the ERTMS/ETCS communication Language. Each packet contains a header, which consists of three/four variables:

- A unique identifying number
- Packet length in bits
- Distance scale (any distancing related information in the packet uses this scale)
- Orientation information (only contained in packets transmitted by train)

Along with the header, the packet contains a set of variables (a body) which the content of is specific to each individual unique packet.

Further on, the official documentation by the European Union Agency for Railways, subset 26 [36] provides a complete list of all packets that can be transmitted by train, and a complete list of all packets that can be transmitted to train.

**Messages**

In the ERTMS/ETCS Language, information is transmitted in form of messages.

A message as described in [37] is a container of information. It is composed of a header, a predefined set of coherent variables, a predefined set of packets, and some optional packets depending on the need of application.

To put things in a simple way and avoid confusion:

- Messages contains packets
- Packets contain variables
- Variables contain values

A message still contains variables outside of its packets, such as header variables, and other coherent once.

In the official documentation of ERMTS/ETCS, subset 26 [37], two important terms are introduced:

- Track to train.
- Train to track.

These terms are used to describe the direction of a communication message.

Being track to train indicates a message transmission to train, while train to track indicates a message transmission from train. In other words, these describe whether a message was sent by train or to train.

There are different types of messages according to the official documentation, each of these has own unique identifier. The only difference between different messages is the packets contained in each message and whether it is transmitted from train or to train, it will contain some extra variables. This is illustrated in Figure 2.2: Standard format of a radio message from train to track.

| Field No. | VARIABLE | Remarks |
|---|---|---|
| 1 | NID_MESSAGE | Message Identification Number |
| 2 | L_MESSAGE | Message length including everything (from field 1 to padding inclusive). |
| 3 | T_TRAIN | Time Stamp from Train (see chapter 3 – Data Consistency). |
| 4 | NID_ENGINE | Identity of the train. |
| 5 | variables as required by NID_MESSAGE | If needed for this message. Used when sending variables which are not included in a packet. |
| 6 | Packet 0 or 1 | Train-to-track packet type 0 – Position report, or packet type 1 - Position report based on two balise groups. Not included in messages 146, 154, 155, 156 and 159. |
| 7 | Other Packets as required by NID_MESSAGE | |
| 8 | Optional packets | |
| | Padding | If required. |

Figure 2.2: Standard format of a radio message from train to track [37]

### 2.5.3 Modes in ERTMS

In ERTMS, different train modes [2] are used, each of these modes has own features and purpose. Below, the most common modes in ERTMS are described:

- Full supervision mode (FS): movement authority at the allowed speed of the route
- On - Sight (OS): movement authority license with speed limitation
- Shunting (SH): movement authority for shifting with speed limitation
- Staff responsible (SR): movement authority with speed limitation when movement authority in FS / OS cannot be granted due to failures in train or infrastructure.

Other modes than the once described above also exist, but these are not described in [2], ERTMS for dummies 1, as they are not that common according to the document.

### 2.5.4 Operations and Railway

This section provides basic information about ERTMS and railway with the main focus on Eastern line in particular. This information is useful to understand and is used throughout this project to develop a solution.

#### Train line

Everyday there are many train trips driving though the eastern line, each of these trips are referred to as a **train-line**. Each train-line has a unique number, such as line 1808, or line 1801. Each of these unique lines drive the same route every day, meaning each has the same geographical starting point and drives to same geographical destination,

through the same stations daily. The start and end time of each of these lines is also the same every day.

The figure below illustrates this in a simple way where there is a line, in this case 1801, starting at 07:25 driving thought station 1 to station n, and ends at 08:10 every day.

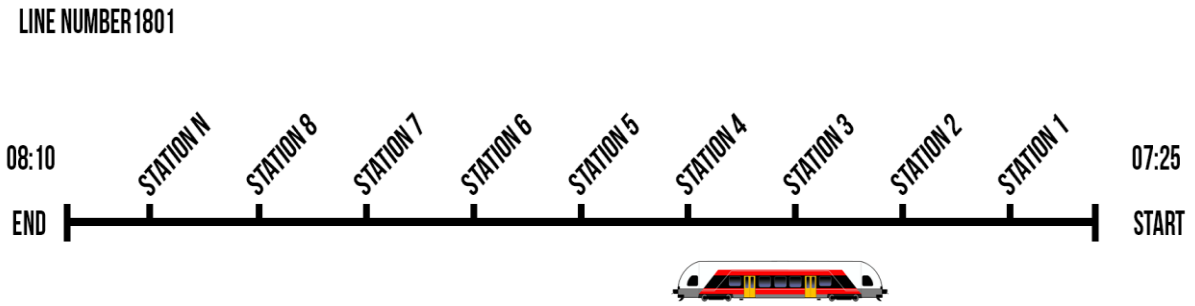



Figure 2.3: Example of a train line

**Train number and ID**

Each train has a unique ID referred to as engine ID, and each train line has a number as mentioned in section above. This is the Train's physical number and can be used to operate different routes. A train route and a train running number have the same ID. For example 1906 is the route running from x station to y station every day at a given time. The train used for this route will have the same train running number regardless of its physical ID. [36]

A single line or train trip can consist of multiple trains connected together as illustrated in the figure below.

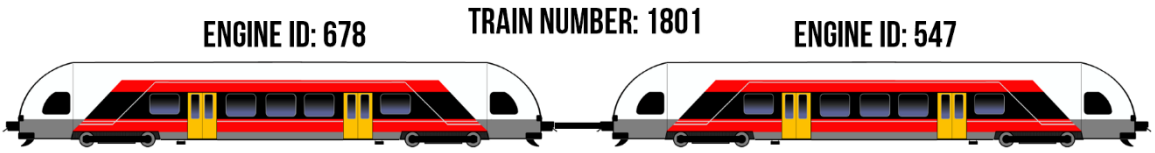



Figure 2.4: Trip consisting of multiple trains

When multiple trains are connected, they will have the same train number representing the line, but each train still has own unique ID.

A train trip can also consist of a single train, in that case it will have a train number to represent the line and one engine ID representing the physical train itself. This is illustrated below.

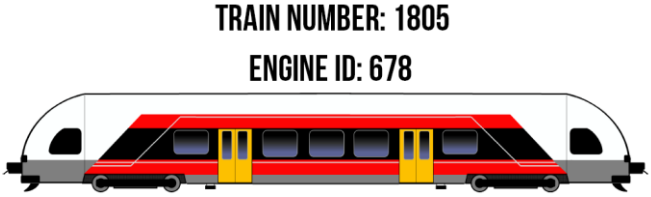



Figure 2.5: Trip consisting of a single train

Engine ID is a unique identifier which represents the physical train itself, and should not change, it can be thought of as "the name of the train" that is attached to the it. [38]

### Balise

A balise is an electric device (a beacon), which lies in between the rails of a railway. When train drives over a balise, balise transmits data to train telling the train the position [39]. Balises are illustrated in Figure 2.6, each marked with xx as an id.

### Position

In the Eastern line, an incremental position number measured in a unit such as kilometers is used. At the very start of the line (ski station), the position is 0 km, and further the line this number is increasing. This is illustrated in the simplified figure below.

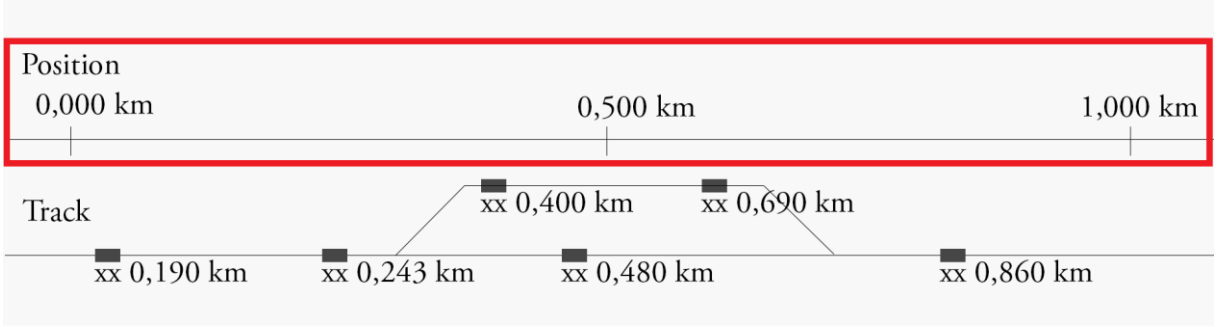



Figure 2.6: Line of incremental positon in km

There is no coordinate system or similar used to track position, instead train reports last read balise. Since each balise is mapped to a position, train location is known. However, train also report estimated frontend position. This is an estimate of how far train is from last read balise. This number can be added to last read balise positon (or subtracted if train is driving in reverse direction) to get an estimated exact position in kilometers.

### Position reports

Every 6 seconds train sends an update message called train position report, which contains useful information about the state of the train. This message contains some variables and packets which are described in 3.6.2, but the most important variables for this projects that exist in one of it's packets are current mode, last read balise, current speed, estimated frontend position from last read balise along with few others. [37]

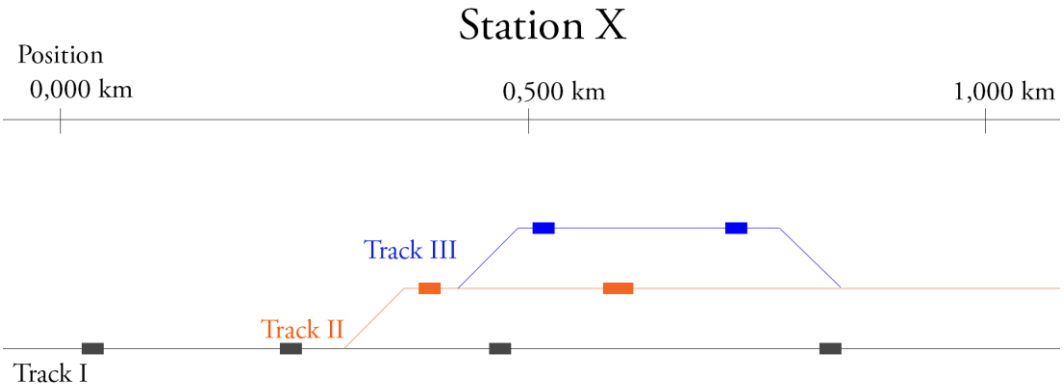



Figure 2.7: Tracks in a railway

**Track**

On the railway, there may be more than one track. Each track may have different allowed speed, and trains can drive both directions in same track. Each direction may also have different speed restrictions. This is illustrated in Figure 2.7.

## 2.6 Mathematical Methods

When comparing two different registered values to determine the one closest to a given value, multiple mathematical methods can be used. Some of these and the once used or taken in to consideration in this project are presented in the following subsection.

### 2.6.1 Mean Absolute Error (MAE)

The mean absolute error (MAE) can be used to find or measure the amount of difference between two data sets.

The mean absolute error (MAE) is the average of all absolute errors (AE) in a set of data without direction in consideration.

The absolute error is the amount of error in a measurement, which is basically the absolute value of the difference between the measured value and the true value. The formula representing the absolute error is expressed as follows:

$$(\Delta x) = |x_i - \mathrm{x}| \tag{2.1}$$

Where $x$ is the true value, and $x_i$ is the measured value.

Having a multiple sets of values, like for example two sets containing speed values, where one set contains the measured speed values, and another contains the real speed values, we can find the mean absolute error (MAE) by summing all the absolute errors $\Delta x$ in equation (2.1) and dividing the sum by the number of data points. Thus, we have the MAE formula defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - \mathrm{x}| \tag{2.2}$$

Where $n$ is the number of data points or the length of the data set. [40]

# 3 METHOD

## 3.1 Project Organization

### 3.1.1 Project Group

The group for this project consisted of a single student, that is student with candidate number 10017. For this project, some previews experience or prior knowledge of ERTMS was desired, though not required. This is to keep the focus during the project on the software and computer science side of things rather than on ERTMS related matters.

To keep things easy and simple, the student carried out the project alone, though it would have been an advantage to work in a team of two or more if prior experience or knowledge existed among these students.

### 3.1.2 The Client

The client for this thesis was Bane NOR which is a state owned company that is responsible for the Norwegian national railway infrastructure. Bane NOR is mainly responsible for developing, administrating as well as maintaining the Norwegian railway. The company consist of over 4500 employees and has offices in different locations in the country, with the main office located in Oslo.

Abraham Mosye is a senior engineer at Bane NOR who is responsible for this project. He had some ideas for a software project related to data analysis. He had requirement specification for that software, which then became this project.

### 3.1.3 Supervisors

For this project, two supervisors where allocated. The main supervisor for the project was Ibrahim A. Hameed who is a professor, deputy head of research and innovation and a program manager for masters in simulation and visualization. The secondary supervisor was Anniken Susanne T. Karlsen who is head of department of ICT and Science at NTNU.

## 3.2 Project Planning

The project was initially planned by writing a preliminary report. In this report, many details were described and a project schedule was set up.

After the preliminary report, some research took place. The research was mainly focused on technologies and finding out different existing technologies to be used. During this research different libraries including component libraries and charting libraries were looked into before choosing the technologies and libraries that are well fit for this project.

After the research, system planning took place. During this period, multiple UML diagrams, wireframes and flowcharts where created to plan the overall design of the system and the solutions before implementing them.

## 3.3 Methodology

For the development of this project, the scrum methodology described in 2.1.2 was used. One limitation to using scrum when developing this project was the team size consisting of a single student. Because of this, not all aspect of scrum were followed strictly. As an example, a scrum master did not exist and daily scrum was not possible to hold.

However, other beneficial and important aspects of scrum were followed in a great way. That is, the development of the project was done in short iterations (sprints), sprints were planned before starting them, product backlog was consistently updating after each sprint (with done requirements or new requirements and updates), and progress was tracked with a professional scrum tool (Jira).

Each sprint was set to a length of 14 days, followed by a short meeting with the product owner, and another meeting with supervisors to update on the progress and receive feedback and determine what was considered to be "Done" was actually "Done" according to the product owner expectations.

## 3.4 Development Process and Procedure

**The first step**

Initially, at the start of the development faze (after ending the planning faze) all software and plugins described in the tools section 3.5.1 had to be installed. After the installation, a Git repository was initialized locally and on Github. To track the progress and to make use of the scum methodology, a JIRA board was created for this project. After that, multiple software modules had to be created locally, and required frameworks were added to these modules. Other libraries were also installed, but not all of them at once, libraries were only installed when they were needed.

The module created had to be pushed to the Git repository created on Github, after that, the development process started. The modules and their development are described below.

**Development of different modules**

The development of this project was divided into several modules. A client module, which is called the frontend, a server which is called the backend.

The development of frontend module and the backend module started at the same time and was run in parallel through out the project. The main reason for this is to have a visual product to show to the client on each sprint meeting. Since the client is not a developer, we thought that showing nonvisual results, such as data and algorithms might lead us to the wrong path, that is, it might lead us to developing the wrong functionality and product. Rather the choice was to represent the results in a visual way from the start. That way it becomes easier for the client to understand the solutions and the product being developed, and was able to trach the progress and give good feedback.

## 3.5 Technologies

To develop this project, a variety of technologies and tools were used. In the subsequent sections, a detailed description of all tools and technologies used such as different programming languages, frameworks, libraries, version control systems (VCS), IDE's, cloud solution and websites are described.

### 3.5.1 Tools

Under the development of this project, multiple tools has been used. Some are required to reproduce this project, others are optional such as browser plugins. Many of the tools have multiple options to choose from such as an IDE where developers are not required to use the exact same IDE or a version control system (VCS) where other options and choices exist.

**Downloads and Installs:**

| Name | Version |
|---|---|
| Java Development Kit (JDK) | 1.8 |
| IntelliJ IDEA | 2019.3.4 |
| Node.js | 10.4.1 |
| Maven | 3.6.1 |
| Git | 2.16.2 |
| Elasticsearch | 5.5.3 |
| Kibana | 5.5.3 |
| Postman | 7.20.1 |

Table 3.1: List of downloads and installs

**Core frameworks and libraries:**

- Spring framework
- React framework
- FusionCharts
- Material UI

Note that the list of frameworks and libraries above only contains the "Core" once. Complete list of dependencies and libraries is described under 3.5.2 - Programming Languages and Frameworks. Specific versions are listed in the project source code. For backend, versions are specified in the pom.xml file, while for the frontend, versions are specified in the package.json file.

**Tools, platforms and services:**

- Github
- JIRA

- Draw.io

**Browser plugins:**

- React developer tools

**Other:**

- PC, running a windows operating system.

### 3.5.2 Programming Languages and Frameworks

Programming languages, frameworks, libraries and dependencies used to develop this project are separated into two different parts, a backend part, and a frontend part.

**Backend**

The backend part of the software is written in Java. The specific version of Java is 1.8. Since this application was web based, the spring framework described in section 2.4.1 was used. The spring framework was added as a dependency in the application, this dependency is called "spring-boot-starter-web", which included all dependencies related to web development.

The dependency include or in other words depends on the following dependencies:

- `org.springframework.boot:spring-boot-starter`
- `org.springframework.boot:spring-boot-starter-tomcat`
- `org.springframework.boot:spring-boot-starter-validation`
- `com.fasterxml.jackson.core:jackson-databind`
- `org.springframework:spring-web`
- `org.springframework:spring-webmvc`

Since all dependencies listed above were included in the dependency "spring-boot-starter-web", there was no need to add these as separate dependencies when getting started with the implementation of the backend part of the project. However, other dependencies were also needed during the development of the application, such as dependencies for the database, dependency for working with JSON objects and arrays and others to simplify the development process, produce boilerplate code and make the code more readable and easier to understand such as Lombok.

Below, is an example showing Lombok in use:

```java
import lombok.Getter;
import lombok.Setter;
import lombok.experimental.Accessors;

@Getter
@Setter
@Accessors(chain = true) //Builder pattern
public class TrainTrip {

    private Train train;

    private Long startTime; //in ms

    private Long endTime; //in ms
}
```

Snippet 3.1: Lombok example

Below is a list of all dependencies that were used in the backend part of the project:

- `org.elasticsearch:elasticsearch`
- `org.elasticsearch.client:transport`
- `org.json:json`
- `org.projectlombok:lombok`
- `org.springframework.boot:spring-boot-starter-test`

These dependencies can also be found in the `pom.xml` file provided in the source code of the backend application.

### Frontend

For the frontend part of the project, standard web technologies were used. That is HTML, CSS and JavaScript. However, the frontend application was not written in plain JavaScript, the React framework described in section 2.4.2 was used for the development of the frontend part.

For creating the react application (file structure, and all react required libraries), a library called "create-react-app" was used. This library initialized all needed files, and setup all configuration. It also takes use of Webpack (2.4.4), babel (2.4.3) and sets up the configuration for those as well

Along with React as the "core" framework for the frontend part of the project, other third-party libraries were also used in this project. These are listed below:

- Material UI (Google's UI component library. Contains assets and components such as buttons, Input fields, date-pickers  and other UI elements)
- Fusion charts (charting and data visualization library, used to plot graphs, highlight time-events and visualize data)

Fusion charts is not an open-source library, but using it for free does not have any time restriction of restrictions on functionality. However, the chart displays a link to official website of Fusion charts on the chart. This link is marked as "free trial" and cannot be removed unless framework is purchased [41].

Complete list of specific libraries used in the frontend part of the project and their specific version are listed in the package.json file provided in the source code.

## 3.6 Data

Data utilized in this project is a result of continuous communication between a train and a data center (RBC) as described in 2.5.1.

In this section, communication message and packets that are essential for the development of this project are described, along with data storage and retrieval.

### 3.6.1 Communication Messages

As mentioned in 2.5.2, there are two different types of communication messages, namely train to track and track to train.

Track to train messages are neglected, as they have no use in the project. However, messages reported by train, that is train to track messages, are heavily used in this project and are described in this section.

#### Train to track messages and types used

There are different types of train to track messages as described in 2.5.2.

Many types of train to track messages are not of use in this project either, therefore will be neglected and not presented. However, some train to track messages are very important, and essential to the development of this project. These are presented in the table below.

| Message ID | Message Name | Transmitted to |
|---|---|---|
| 129 | Validated Train Data | RBC |
| 136 | Train Position Report | RBC, RIU |
| 155 | Initiation of a communication session | RBC, RIU |
| 156 | Termination of a communication session | RBC, RIU |

Table 3.2 - Essential Project-specific communication messages

Other messages containing packet number zero (train position report packet, see 3.6.2) are also used, but are not important to mention.

Each of these messages has own usage. When the train wants to report its position, it sends message 136, train position report. When it connects to a session, it will send message 155, initiation of a communication session, when it terminates a session it will send a termination of communication session message, and when train changes number and drives a new train-line it sends validated train data message after it initiates a communication session. [37]

All of these messages has in common all variables illustrated in Figure 2.2: Standard format of a radio message from train to track, in addition to required and optional

packets. These packets may vary from message to the other as mentioned in 2.5.2. Some messages does not contain any packets at all. Below is the content of each of the messages listed in Table 3.2 - Essential Project-specific communication messages. [37]

| Field No. | VARIABLE/ PACKET | Remarks |
|---|---|---|
| 1 | NID_MESSAGE | |
| 2 | L_MESSAGE | |
| 3 | T_TRAIN | |
| 4 | NID_ENGINE | |
| 5 | Packet 0 or 1 | |
| 6 | Train data | Train - track packet type 11. |

Figure 3.1: Validated Train Data message structure [37]

| Field No. | VARIABLE/ PACKET | Remarks |
|---|---|---|
| 1 | NID_MESSAGE | |
| 2 | L_MESSAGE | |
| 3 | T_TRAIN | |
| 4 | NID_ENGINE | |
| 5 | Packet 0 or 1 | |
| 6 | Optional packets | |

Figure 3.2: Train Position Report message structure [37]

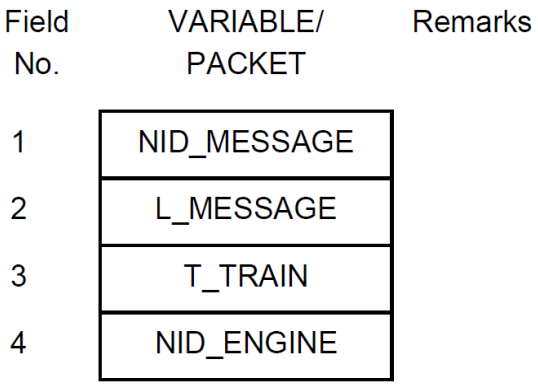



Figure 3.3: Initiation of a communication session message structure [37]

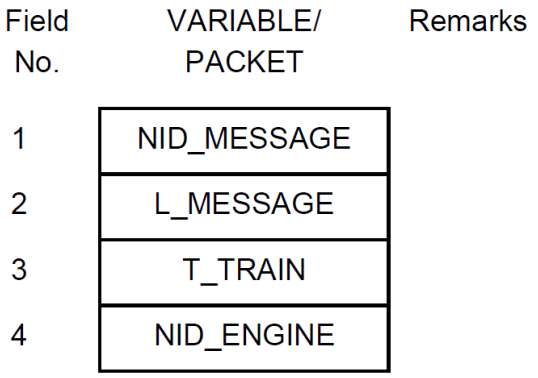



Figure 3.4: Termination of a communication session message structure [37]

### 3.6.2 Packets

As mentioned in section 2.5.2, each communication message may contain some optional or required packets and there are many different types of packets. However, not all packets are important to this project. In the table below, all packets that are essential to the development of this project are listed.

| Packet number | Packet Name | Contained in Message ID |
|---|---|---|
| 0 | Train position report | All except: 146, 154, 155, 156, 159 |
| 11 | Validated train data | 129 |

Table 3.3 - Essential Project-specific packets

The content of each of these packets is shown in Figure 3.5 and Figure 3.6.

| ***Description*** | This packet is used to report the train position and speed as well as some additional information (e.g. mode, level, etc.) | | |
|---|---|---|---|
| ***Transmitted to*** | RBC, RIU | | |
| ***Content*** | **Variable** | **Length** | **Comment** |
| | NID_PACKET | 8 | |
| | L_PACKET | 13 | |
| | Q_SCALE | 2 | |
| | NID_LRBG | 10 + 14 | |
| | D_LRBG | 15 | |
| | Q_DIRLRBG | 2 | |
| | Q_DLRBG | 2 | |
| | L_DOUBTOVER | 15 | |
| | L_DOUBTUNDER | 15 | |
| | Q_LENGTH | 2 | |
| | L_TRAININT | 15 | If Q_LENGTH = "Train integrity confirmed by integrity monitoring device" or "Train integrity confirmed by driver" |
| | V_TRAIN | 7 | |
| | Q_DIRTRAIN | 2 | |
| | M_MODE | 4 | |
| | M_LEVEL | 3 | |
| | NID_NTC | 8 | If M_LEVEL = NTC |

Figure 3.5: Content of Packet number 0 [36]

From the figure above, this packet contains many variables, such as packet number/ID, last read balise group, train speed, direction, orientation, mode, and others. Therefore, this is the most important packet for this application as it contains valuable information.

| | | | |
|---|---|---|---|
| ***Description*** | Validated train data. | | |
| ***Transmitted to*** | RBC | | |
| ***Content*** | **Variable** | **Length** | **Comment** |
| | NID_PACKET | 8 | |
| | L_PACKET | 13 | |
| | NC_CDTRAIN | 4 | |
| | NC_TRAIN | 15 | |
| | L_TRAIN | 12 | |
| | V_MAXTRAIN | 7 | |
| | M_LOADINGGAUGE | 8 | |
| | M_AXLELOADCAT | 7 | |
| | M_AIRTIGHT | 2 | |
| | N_AXLE | 10 | |
| | N_ITER | 5 | |
| | M_VOLTAGE(k) | 4 | Identity of the traction system |
| | NID_CTRACTION(k) | 10 | NID_CTRACTION(k) given only if M_VOLTAGE(k) ≠ 0 |
| | N_ITER | 5 | |

Figure 3.6: Content of packet number 11 [36]

### 3.6.3 Elasticsearch and Data Storage

This project is based on big data. All data used in this project described in the two previews sections is stored as historical data in a database and new data was being downloaded, parsed and stored for all new train trips continuously. Because of the large amounts of data and the different data structures and attributes of message and packets, data was not stored in a relational database such as MYSQL, rather the JSON based search engine, Elasticsearch described in section 2.4.6 was used. For data management and visualization, the Kibana tool was used. This tool as described in 2.4.6 makes it possible to create complex queries to search in database and manage data.

Data downloading and parsing is out of the scope of this project, since its already been done by other developers few years ago. Thus, no details about data parsing or similar is required presented here.

## 3.7 Deployment and Testing

To test the application developed in this project, an installation and testing guide is included under appendix 1. This manual describes in detail how to set up the environment for testing exactly as it was set up under the development of this project.

# 4 RESULTS

## 4.1 System Overview

Before presenting the results in detail, an overview of different parts is presented to give a general idea of the complete system and the final product. This is shown in Figure 4.1.

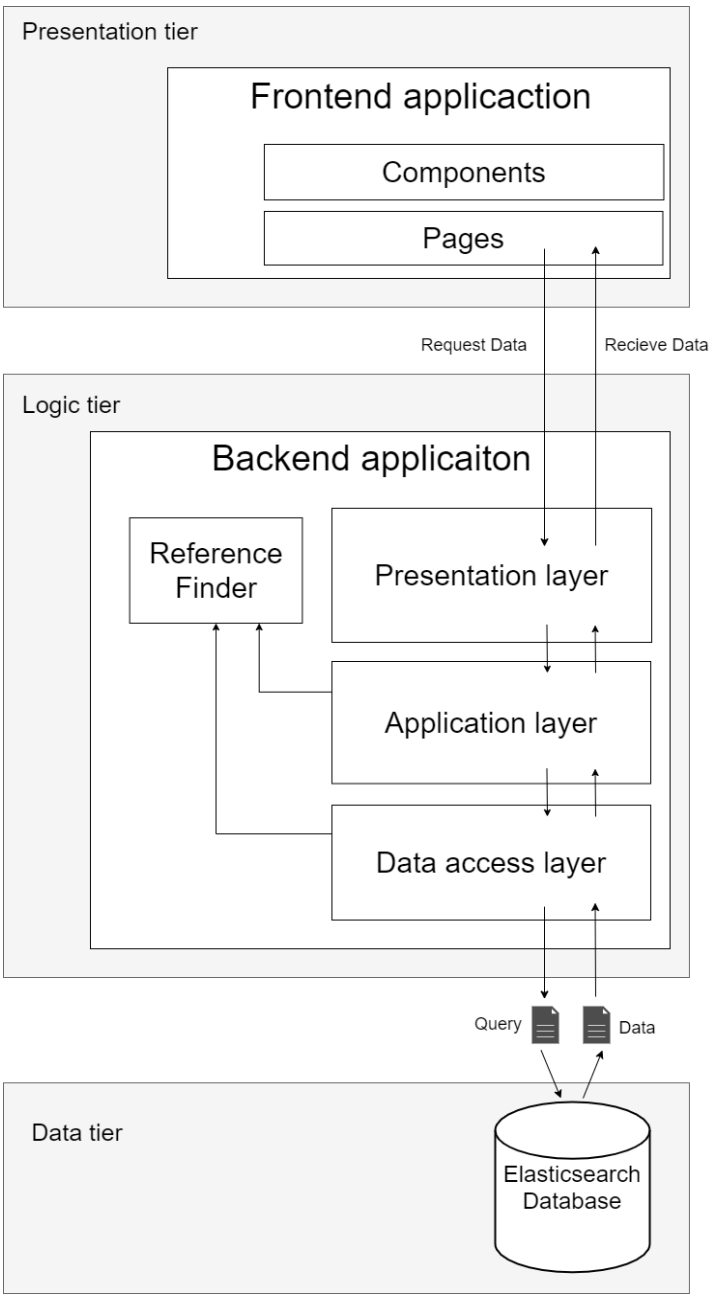



Figure 4.1: System overview

Figure above does not follow any standards, it is only meant to show main components that make up the system, and the main parts that make up each application.

## 4.2 Architecture

The architecture that was chosen for this software project was based on a multitier architecture described in 2.3.2 .In this case, the solution consists of three parts, a backend application and a frontend application and a database each representing a tier and is physically separated.

The frontend application represents tier one, namely the presentation tier in this case. It only is responsible for providing the user with an intractable interface, as well as presenting data it receives from the backend application.

The backend application represents tier two, namely the logic tier and provides tier one with data. This application consists of multiple layers (multi-layered application) as shown in Figure 4.1 in the previews section.

The Elasticsearch database (2.4.6) represents tier three of the complete system, namely the data tier and is responsible for data management, data storing and providing data.

Thus, the complete system consists of the following three tiers:

- Presentation tier
- Logic tier
- Data tier

Tier architecture is illustrated in the Figure 4.2: Tiers of the complete system and how they work together.

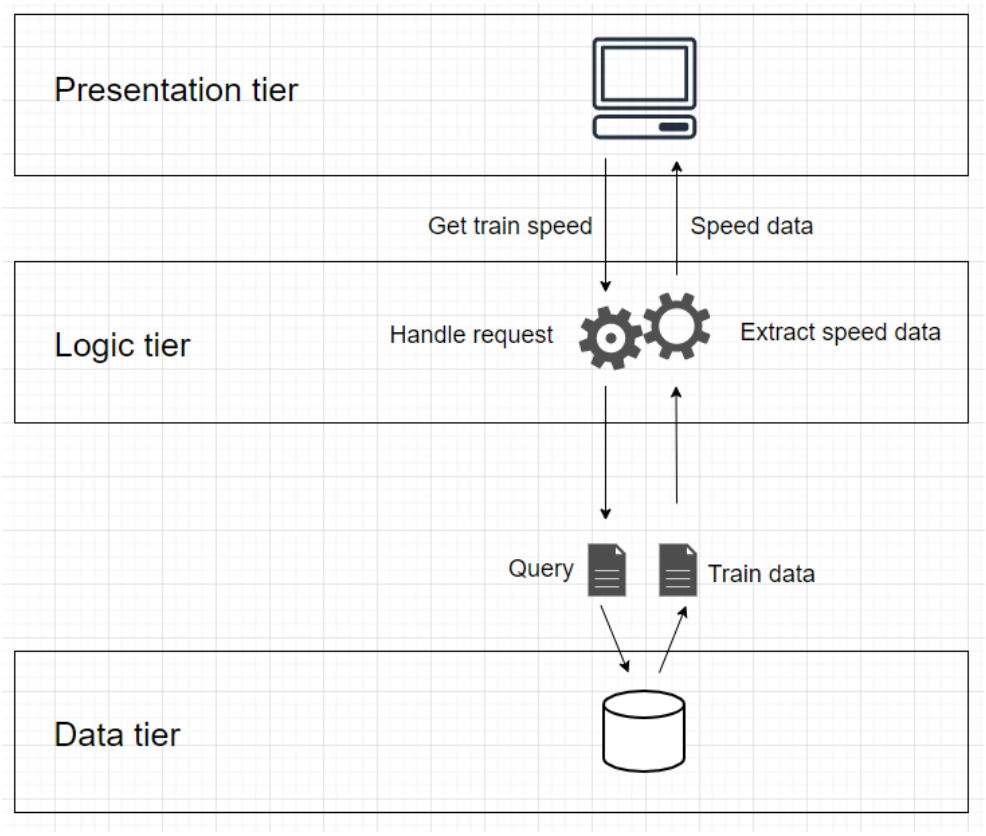



Figure 4.2: Tiers of the complete system and how they work together

### 4.2.1 Client-Server

As mentioned in 2.3.2, multitier architecture is based on the client-server architecture described in section 2.3.1, which means the communication between different parties/tiers of the system happens through a network.

Below, in Figure 4.3, a deployment diagram is included to illustrate the client (frontend) and the server (backend) that make up the system and how they are connected and communicate together. This also illustrates the actual deployment of the system and the end-result of this project.

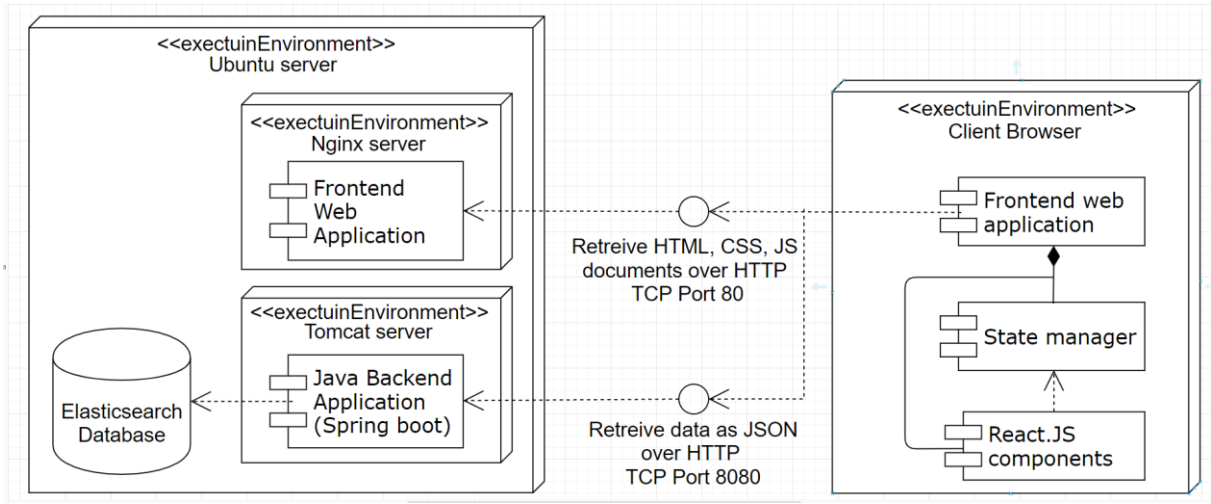



Figure 4.3: Deployment diagram of the functioning system

From Figure 4.3, the frontend and the backend applications are both hosted in the same server machine along with Elasticsearch database, though it is sill possible to host them on different physical machines as distributed applications as they are based on multitier architecture. Each of these applications runs in own container/server.

The backend application runs on a tomcat server port 8080, while the frontend application's static files are served by Nginx server which is listening to port 80 as shown in the deployment diagram.

On the right side of the figure, the client browser is shown. The client first retrieves the frontend application files by contacting the Nginx server on port 80. When the files are received and the application is rendered into the browser, the client is able to request data from the backend Java application though port 8080. When the backend application receives a request from the client, it processes the request, fetches data from database, applies logic, processes data, and sends response back in JSON format. When the client application receives the response and data, it will parse it, transform it, and then update the state of the frontend application.

When the state is updated, the frontend application will render or re-render components effected by the state change. Rendering and re-rendering of components is handled by the React framework (2.4.2), and beyond the scope of this report.

Everything described above, is illustrated in the deployment diagram.

### 4.2.2 Multi-tier as an Architecture

As mentioned in section 2.2.1, when developing software systems, an important principle is "separation of concerns".

Main reason for choosing multi-tier as an architecture for this system is because it applies the separation of concerns principle. By choosing a multitier architecture for this software system, different parts of the system are loosely-coupled which broke the complexity of the problem into easier to solve, sub problems. This also made different parts of the application reusable as mentioned in 2.3.2.

If the client decides in the future to develop a mobile versions of this system, only a new client mobile app (another presentation tier) needs to be developed without touching the logic tier or the data tier or the current developed presentation iter. In other words, the same backend and database can be used without a single change if a mobile version is desired in the future.

This way it is possible to create different presentation layers, for mobile, tablet and desktop without having to touch other layers.

Same logic applies to modifying the business logic or algorithms implemented in the application. Only the logic tier will need to be modified or changed, without touching any of the other tier.

Thus, for this system, the principle of separation of concerns was applied through multitier and server-client architecture which contributed to an overall better system design and reusable, distributable software.

## 4.3 Backend

In section 4.2, it was mentioned that the backend application consists of multiple layers. Each of these layers has own responsibility and depends on another layer.

In this section, the general design of backend application along with different layers, and the reference finder algorithm will be described.

### 4.3.1 The Design

The overall design of the backend application is layered as mentioned earlier and follows good practices and known principles and design roles.

As presented in 2.3.3, layered architecture is about segregating the software into different parts (layers). For the backend application, the number of layers that chosen was three. This is because it is the most popular number of layers to have on a RESTFUL web app like this one as described in 2.3.3. Not only that, but also having many layers in this application seemed to overcomplicate things and would be an "overkill". Therefore, the choice was to go with 3-layered architecture, which consisted of a presentation layer, application layer and data access layer.

The reason for choosing the layered architecture in the first place was that it applies an important principle descried in 2.2.1, separation of concerns. Each part has own concerns and these "concerns" are separated.

The presentation layer does not retrieve any data or perform any database related operations. Neither does it process data, transform data or apply logic. It has only one responsibility and one "concern", that is receiving a request, forwarding it to the

application layer, and then returning processed data to the requesting client. Data retrieval, data processing, data transformation and logic application are operations specific to other layers and are the responsibility of other part of the application. Data processing and logic application is the concern of another part and is handled by a separate layer, in this case the application layer. Data retrieval and persistence is the concern of the data access layer, thus is handled by that layer.

This way "concerns" within the backend application are logically separated. The way they are separated is by following layered architecture principles.

Below, a class diagram of all classes the backend application consists of and how they are connected to each other is shown in Figure 4.4: Classes that make up the backend application of the system.

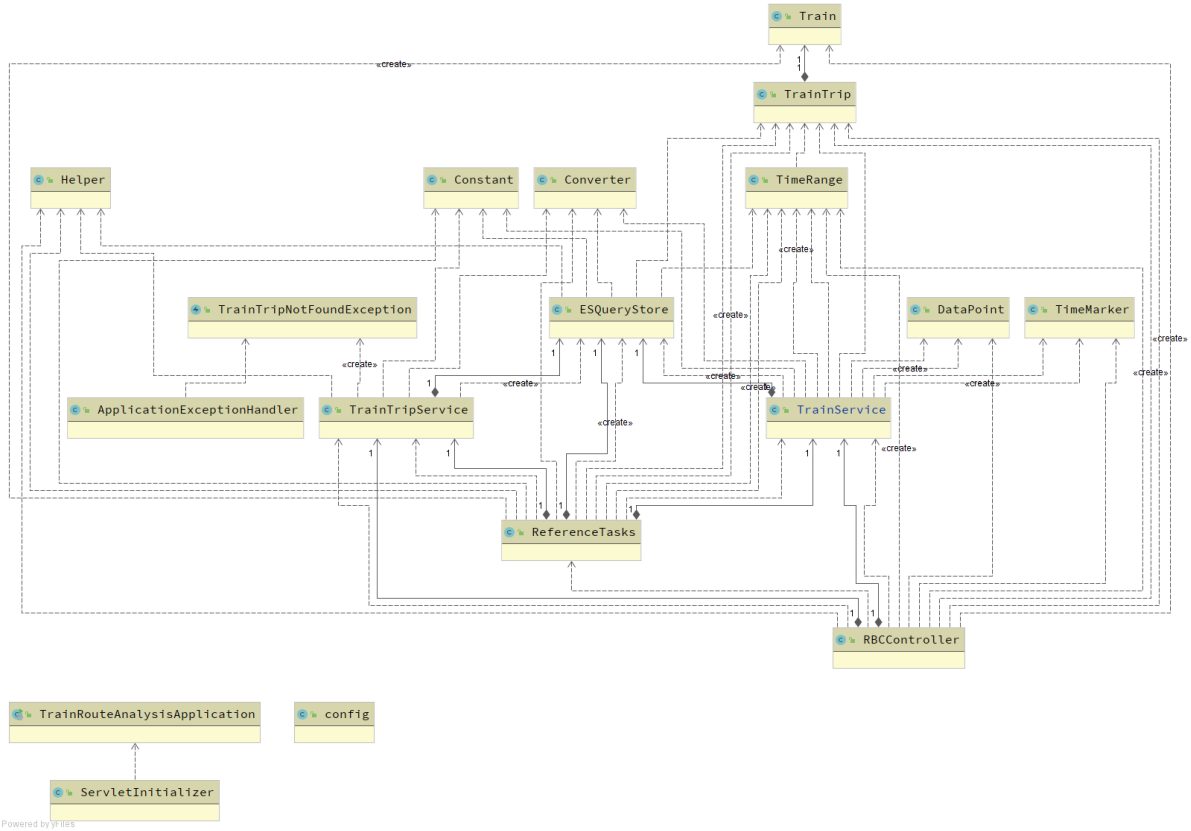



Figure 4.4: Classes that make up the backend application of the system

To start, we can zoom into the three layers of the application which provide the core functionality, that is the presentation layer, the application, and the data access layer. The classes that make up these three layers are shown in figure below.

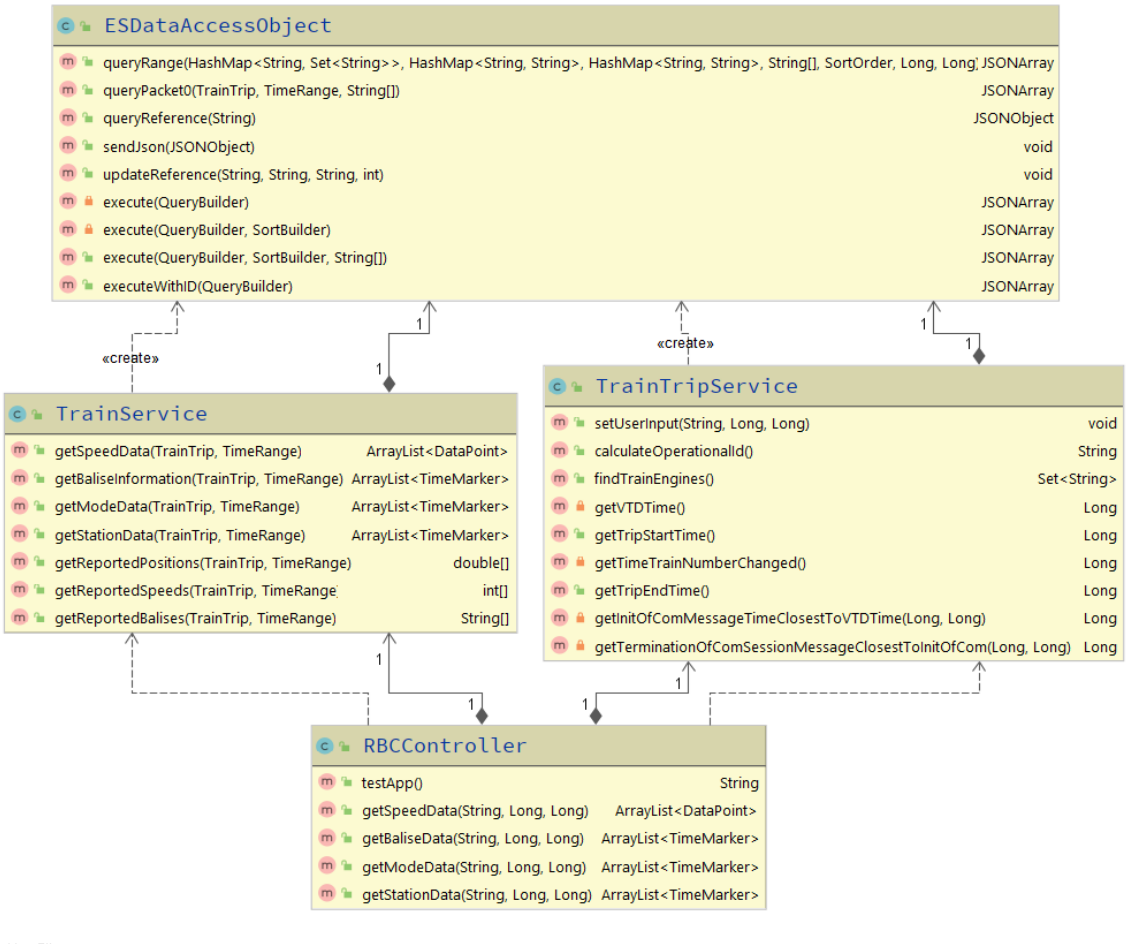



Figure 4.5: Classes that make up the layers of the backend and their connection

Each of these three layers as mentioned has a single responsibility, and depends on the layer above it. From the figure, it is clear that the presentation layer does not know about the data access layer, it uses the application layer for that. Same logic applies on the other layers, application layer does not know about presentation layer, the data access layer does not know about any of the other layers below it.

Each of the layers are described in detail in the following sections.

### 4.3.2 Layer 1: Presentation Layer

The presentation layer of the backend application consist of only one controller class. It has a single responsibility, which is exposing some functionality to other third party applications as described in multilayered architecture section 2.3.3. In this case it exposes functionality to the client frontend application (the presentation tier).

The controller depends on the application layer to function. Dependency injection is handled by the spring framework (2.4.1) and is beyond the scope of this report.

The controller class provides a set of methods which external applications (frontend application) can invoke.

Looking at the figure above, there are methods for getting speed data of train, balises, modes and stations reported by train, which later on are used in the frontend application to plot graphs and display events for data analysis.

In the table below, all API's this controller class exposes are listed.

| API | Method | Required parameters | Response |
|---|---|---|---|
| api/v1/rbc/speed | GET | trainNumber: String<br>from: Long<br>to: Long | List of speeds and exact time reported by train (reference and actual) |
| api/v1/rbc/balise | GET | trainNumber: String<br>from: Long<br>to: Long | List of balise and exact time reported by train |
| api/v1/rbc/mode | GET | trainNumber: String<br>from: Long<br>to: Long | List of modes and time range the train was in that mode |
| api/v1/rbc/station | GET | trainNumber: String<br>from: Long<br>to: Long | List of stations and time range the train was in that station |

Table 4.1: List of API's

When the controller class receives a request, it first passes the request parameters to the application layer, more specifically to "TrainTripService.java" to calculate and find out some values required to perform a search in the database.

These values are:

- Operational ID (to obtain train number/line-number mentioned in section 2.5.4)
- Train ID or IDs if trip consists of multiple connected trains (physical Train ID, see section 2.5.4)
- Actual start and end time of the trip

After calculating/finding out these values, the controller constructs a TrainTrip object and puts all of these values in that object. As these are required further to search in the database, there will be no need to calculate them again, until a new request for another trip is received. Everything described above is shown in snippet code below.

```
trainTripService.setUserInput(trainNumber, from, to);

TrainTrip trip = new TrainTrip()
        .setTrain(new Train()
                .setNumber(trainNumber)

.setOperationalID(trainTripService.calculateOperationalId())
                .setEngines(trainTripService.findTrainEngines())
        )
        .setStartTime(trainTripService.getTripStartTime())
        .setEndTime(trainTripService.getTripEndTime());

TimeRange timeRange = new TimeRange(trip, from, to);
return this.trainService.getSpeedData(trip, timeRange);
```

Snippet 4.1: Constructing a TrainTrip object

A TrainTrip object consists of a Train object and a start and end time values. Thus, a train object also needs to be constructed and stored within the TrainTrip object. A Train object consists of a train number, operational ID, and a set of engines. Structure of both objects is shown in the snippets below.

```
@Getter
@Setter
@Accessors(chain = true) //Builder pattern
public class TrainTrip {

    private Train train;

    private Long startTime; //in ms

    private Long endTime; //in ms
}
```

Snippet 4.2: TrainTrip object

```
@Getter
@Setter
@Accessors(chain = true) //Builder pattern
public class Train {

    private String number;

    private String operationalID;

    private Set<String> engines;
}
```

Snippet 4.3: Train object

Now that a TrainTrip object is constructed and contains all required information to apply logic and performs search in the database, the object is passed to the application layer to do the rest of the job.

### 4.3.3 Layer 2: Application Layer

From Figure 4.5, it is clear that the application layer consists of two classes, namely "TrainService.java" and "TrainTripService.java". These two classes as shown in the figure depend on the data access layer and will not function correctly without it. They also use/depend on other helper classes, exception classes, entity classes and so on.

The application layer contains all logic of the backend, and provides presentation layer with needed processed data, which further will be transferred to the client application.

Each of the two classes in this layer has own responsibility. "TrainTripService.java" is responsible for the trip, it contains some logic and searching algorithms to find information about the trip which the user is searching for. This information cannot be obtained from the user input passed as parameters, unless the user does the job manually by finding that information in the database, therefore, it is important to have this class to automate that part of the problem.

The information that is mainly found by this class is which train/trains were used for that trip (engine ID/ID's, described in 2.5.4) along with the actual start and end time of that trip.

In the requirement specification it is mentioned that the user should be able to search for a trip by train number (line-number) and selecting a data/time, therefore user cannot be asked to provide other input such as which trains used for that trip (engine ID/ID's) or similar. This will cost the user time, as he/she will have to search manually in database to find such information.

The other class "TrainService.java" is responsible for data processing. Using the information found by "TrainTripService.java" it will get required data, apply some logic, transform data and forward data to the presentation layer, which will then send data further the client (presentation tier).

Logic applied in this class is mostly related to time events. For example finding out when a station was entered and passed, when train-mode started and changed, along with some logic related to balises and speed. This information is later used by the frontend application to mark time events and display them to user for analysis.

```
if(!balise.equals(lastBalise)){
    String baliseId = lrbgToBG(balise); //Convert LRBG to balise ID
    Double position = BALISE_KM_MAP.get(baliseId); //Get KM locaiton

    TimeMarker timeMarker = new TimeMarker()
            .setLabel("Balise: " + baliseId +
                    ", Position: " + position + " km")
            .setStart(time);
    baliseData.add(timeMarker);
    lastBalise = balise;
}
```

Snippet 4.4: Example of processing balise data in TrainService

Transformation in this class is related to decoding values, converting time zone, along with getting the mapping of different values. This class does that by using other helper classes to convert, decode or map those values. For example train-mode is not reported as text, rather an ID representing a mode is reported. A hash map mapping each mode ID to its text is stored in a constant file. Thus, different mode-values are transformed using the existing hash map of key-value pairs.

The application layer uses custom exceptions. Currently only one custom exception exist, that is "TrainTripNotFoundException.java". If for example the user is searching for a train and that train does not exist in database, or did not drive that day or similar, this exception will be thrown with a custom message to be displayed to the user. Exception handling is described in section 4.3.6. Figure 4.6 shows "TrainTripService.java" using custom exception "TrainTripNotFoundException.java".

There is much more details that goes in this, but only the most important information is presented as it will be hard to explain every single detail in the system.

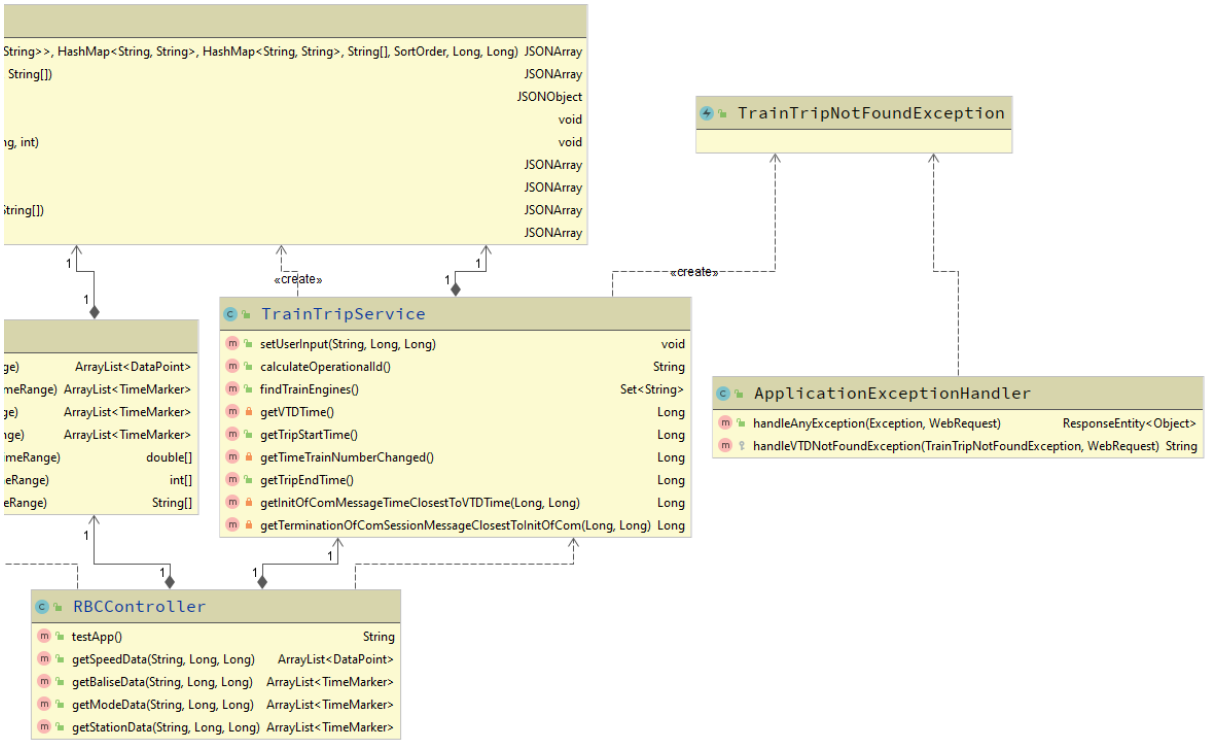



Figure 4.6: TrainTirpService uses TrainTripNotFoundException

### 4.3.4 Layer 3: Data Access Layer

The third and the final layer the backend application consists of is the data access layer. This layer is the persistence layer, it provides different parts of the application with data, and is able to insert, update or delete data from database. This is shown at the very top of Figure 4.5. This layer consists of one class only. This class is called "ESDataAccessObject.java" and contains some queries specific to the application and other methods for inserting or updating data.

This class does not depend on any of the other layers. It only depends on other helper classes, such as constant class which provides some constant values, helper classes and converter class which provides converting functionality such as converting time to other zones.

### 4.3.5 Reference Finder

In the previous sections, different layers of the backend application were presented. However, there is one part of the solution that was not mentioned, that is the reference finder.

As described in the requirement specification in section 1.3, the first part of the problem was finding and storing a reference continuously. For this, a reference finder was developed. This reference finder is part of the backend application and has one responsibility which is finding the optimal train reference for each train-line.

The reference finder algorithm was developed with a job scheduler which runs according to a defined schedule. The algorithm was set to run daily at 03:00 GMT+2, as that is about the time all train trips finish running in the Eastern line.

Initially, a list of all train lines (train numbers) was created and put in a constant java file. This list contain every unique train line that runs in the Eastern line and is used in the algorithm to find a reference for every one of the lines listed in it.

**The algorithm**

The implemented algorithm iterates over the defined list of train numbers and applies the same logic to every train number (train line) in the list.

When iterating over train numbers, it does the following:

a. Get all balises (position) reported by that train and speeds train had at each of those positions.
b. Get a list of allowed speeds to each of the positions reported by train
c. Calculate the mean absolute error (MAE) of train speed using the two speed lists, list of speeds reported by train in step a and list of allowed speeds in step b
d. Retrieve MAE of current train reference
e. If no reference of that train-line exist in the database, save this trip (today's trip) as current best reference so far. (it means we have not found a reference yet)
f. Compare MAE in step c To MAE in step d (when step e does not apply)
g. Save a new reference if today's train has better MAE score than MAE of current reference.

These same steps are repeated for each train number in the list. Each for them will either get a new reference or keep the old reference.

Each of these steps are described in more detail below:

**A: Getting balises and speeds reported by train**

All balises (positions) reported by train are fetched from database and stored in a list. Speed reported at each of those positions/balises are also fetched and stored in a separate list.

**B: Creating a list of allowed speeds**

To calculate MAE, two lists of speeds are required. List of reported speeds on each position (actual train speed) and list of allowed speeds on each position train reported. When the two lists are available, MAE can be calculated as a measure of how close the reported speeds are to the allowed speeds.

List of reported speeds by train at each position is available from step a. However, list of allowed speeds on each position is not.

To get the second list, it had to be created and have the same length as the list of reported speeds (to calculate MAE, both have to be same length).

To create this list of speeds a hash map consisting of balises as keys and allowed speeds as values is created and stored in a constant file. This is illustrated in Figure 4.7.

This hash-map tells the algorithm that a certain balise lays in a certain section which has a certain allowed/planned speed.

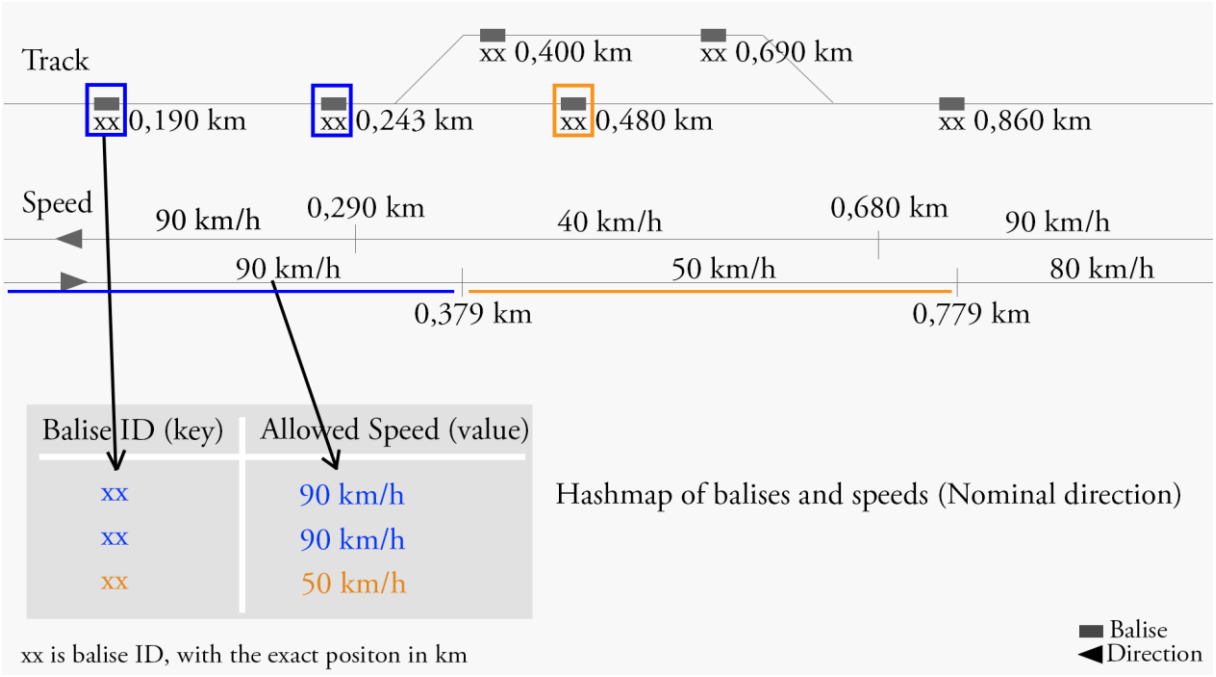



Figure 4.7: Mapping balises to allowed speeds

Since this hash-map is available and accessible by the scheduled task, the algorithm then uses list of balises (positions) reported by train to create the list of allowed speeds. Every balise reported by train is used to look up allowed speed in the hash-map. When the speed at that position is found, it is added to the list of allowed speeds. This is done to every balise reported by train, it is used to look up the speed value train should have at that position and add a new speed to the list of allowed speeds which will be used with list of actual speeds to calculate MAE.

```
reportedBalises = ["04", "04", "23", ...., "16"]; //balises
actualSpeedList = [88, 94, 65, ..., 41];//created by retrieving speed data
reported by train from database
allowedSpeedList = [90, 90, 70, ..., 40];// created looking at balise list
above and looking up mapped speeds in hashmap
```

Snippet 4.5: Lists of reported balises, actual speeds and allowed speeds at each balise

```
private int[] makeSpeedList(String[] balises){
   int[] speedList = new int[balises.length];
   int i = 0;
   for(String b: balises){
       speedList[i] = BALISE_SPEED_MAP_REVERSE.get(b);
       i++;
   }

   return speedList;
}
```

Snippet 4.6: Making a allowedSpeedList from hash-map

This entire step is illustrated in flowchart below.

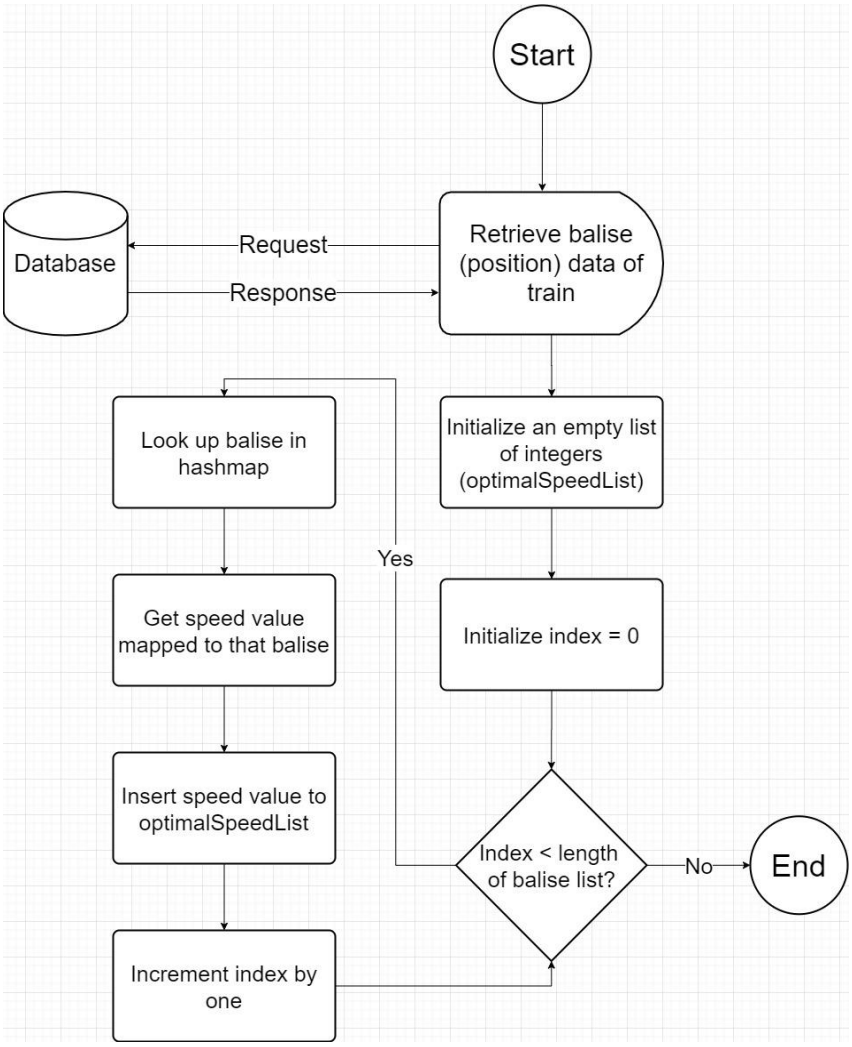



Figure 4.8: Flowchart of  step

### C: calculating MAE

Now that two lists of speeds are available, the algorithm calculates the mean absolute error. The algorithm uses formula 2.2 described in section 2.6.1. This step is illustrated in snippet below.

```
TimeRange timeRange = new TimeRange(
        trip,
        trip.getStartTime(),
        trip.getEndTime()
);

String[] positions = trainService.getReportedBalises(trip, timeRange);
int[] trainSpeedList = trainService.getReportedSpeeds(trip, timeRange);
int[] plannedSpeedList = makeSpeedList(positions);

int mae1 = calculateMAE(plannedSpeedList, trainSpeedList);
log.debug("MAE of actual train: " + mae1);
```

Snippet 4.7: Algorithm calculating MAE

```
private int calculateMAE(int[] allowed, int[] actual){
    int sum = 0;
    for(int i = 0; i < ref.length; i++){
        int allowedSpeed = allowed[i];
        int actualSpeed = actual[i];

        int diff = Math.abs(actualSpeed - allowedSpeed);
        sum += diff;
    }
    return sum/allowed.length;
}
```

Snippet 4.8: Method responsible for calculating MAE

**D: Retrieving reference MAE**

The mean absolute error (MAE) of the reference is stored in the database. This value or "score" is retrieved from database, and then is used for comparison with the new calculated MAE.

```
//Get reference information from database
JSONObject referenceInfo = this.esDataAccessObject
        .queryReference(trainNumber);
```

Snippet 4.9: Retrieving reference from database

**E: No MAE found in DB**

This is an exceptional step. This will only happen the first time the algorithm runs. When the algorithm runs for the first time, there will be no reference for any train-lines. Thus, this train line will be set as starting reference.

**F: Compare MAE**

The algorithm then compares MAE score of reference found in step D with MAE of current train. List with smallest MAE value is the one that has closest speeds to allowed speeds.

**G: update reference**

The algorithm will at the end update existing reference if the new train has better (smaller) MAE score than current reference.

The entire algorithm is illustrated in Figure 4.9: Flowchart of reference algorithm.

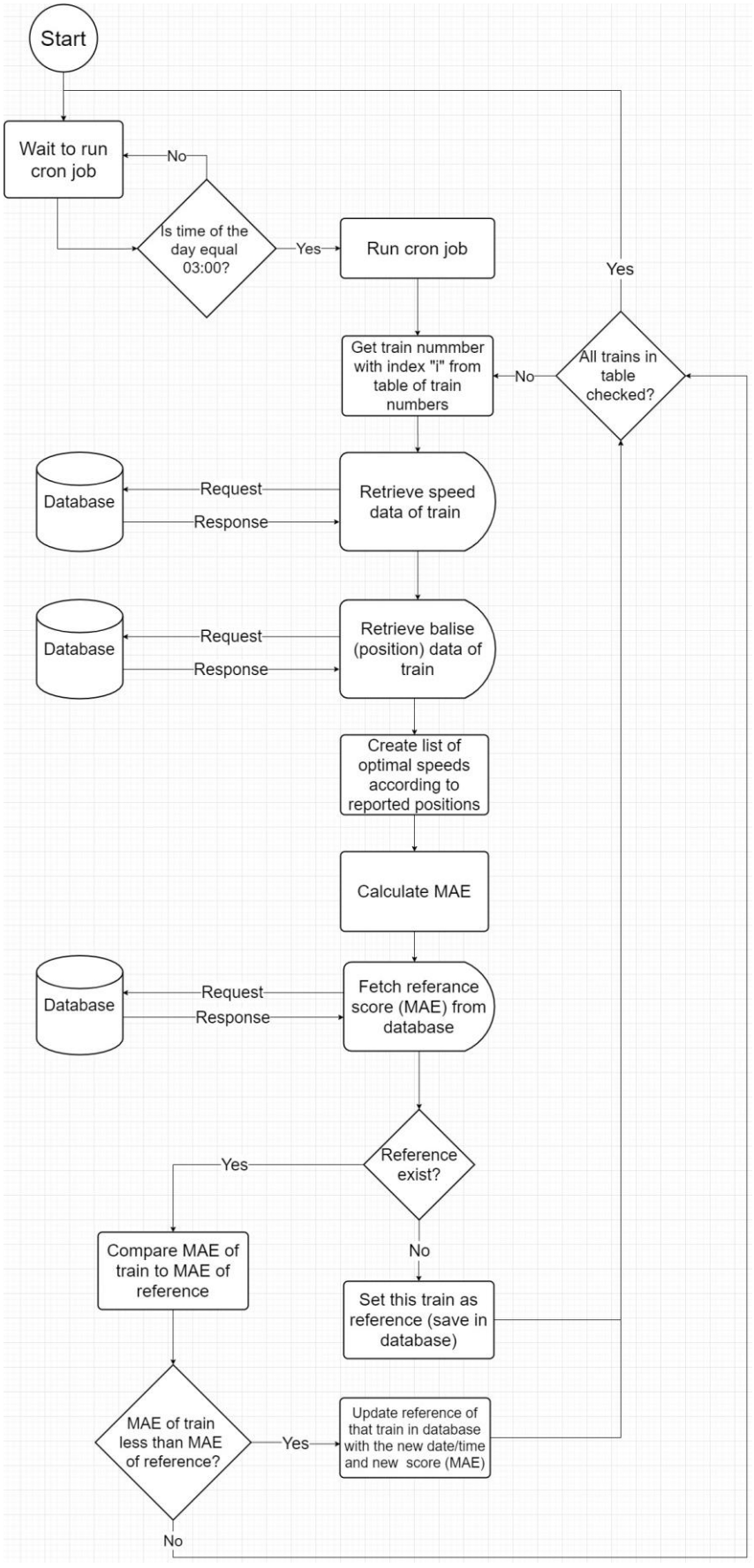



Figure 4.9: Flowchart of reference algorithm

**Storing the reference**

As mentioned, reference is stored in database. Since each train has a reference, only one reference for each train is stored in the database. This means when a new better reference if found, the old reference is updated rather than storing it as a new entry and keeping a history of references for each train-line.

When it comes to what data is stored, not the entire reference train data is stored in the database. Only a selected set of attributes (can be thought of as meta data) were selected to mimic the size of storage, and make it efficient, these attributes are the following:

- Train number
- Start and end time of trip
- Insertion date (last updated)
- Score (MAE)

There is no need to retrieve entire train data of the reference (position report messages and other messages), as MAE is already calculated and stored there. Thus, no need to repeat the process of calculating MAE of reference every time.

The database mappings (scheme) are shown below:

```
{
  "ertms-log-ref": {
    "mappings": {
      "log": {
        "properties": {
          "@timestamp": {
            "type": "date"
          },
          "from": {
            "type": "date"
          },
          "mae_score": {
            "type": "long"
          },
          "to": {
            "type": "date"
          },
          "train_number": {
            "type": "text",
            "fields": {
              "keyword": {
                "type": "keyword",
                "ignore_above": 256
              }
            }
          }
        }
      }
    }
  }
}
```

From the mappings above, the database name is "ertms-log-ref" and has five properties (columns)

- @Timestamp: insertion date/last updated
- From: start date/time of trip
- To: end date/time of trip
- Mae_score: calculate mean absolute error
- Train_number: train number

**Dependency**

The reference finder depends on other layers of the backend application, that is the data access layer for retrieving and saving data to the database, as well as the application layer for finding out information about each trip such as trip start and end time, trains used for that trip and so on. It also depends on other helper, converter and constant classes.

### 4.3.6 Exception Handling

In the backend application, exceptions are handled by a global exception handler. This handler is a class that handles any exception the application might throw. If any exception is thrown, be it a custom exception (application specific), or a general exception, it will be caught in this class.

When the exception is caught, a custom response is created and sent to requesting client. This way all errors that are specific to the application are not sent as a response, instead a human readable message is sent which the client application can then use to display to the user.

```
@ExceptionHandler(value = TrainTripNotFoundException.class)
protected ResponseEntity<Object>
handleTripNotFoundException(TrainTripNotFoundException e){
    e.printStackTrace();
    System.out.println("msg: " +e.getMessage());
    return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_FOUND);
}
```

Snippet 4.10: Handling trip not found exception

Snippet 4.10 shows how a custom exception called "TrainTripNotFoundException.java" being handled. A response entity object is created which contains the message of the thrown exception will be sent to client along with a status code. This message can then be used by the frontend application and displayed to the user as an error.

### 4.3.7 Other Parts and Classes

There are other parts and classes of the backend application not covered above. For example helper classes, converter classes, exception classes and other. These also are part of the backend application and provide functionality to other classes and are mainly meant to assist other parts of the application.

## 4.4 Frontend

### 4.4.1 User Interface

The client or the presentation tier is a single page application, which as mentioned in 2.3.5 is an application that consists of only one page and where only one HTML file exists which changes the content dynamically through JavaScript.

The first thing a user sees when accessing the application is the page in figure below.

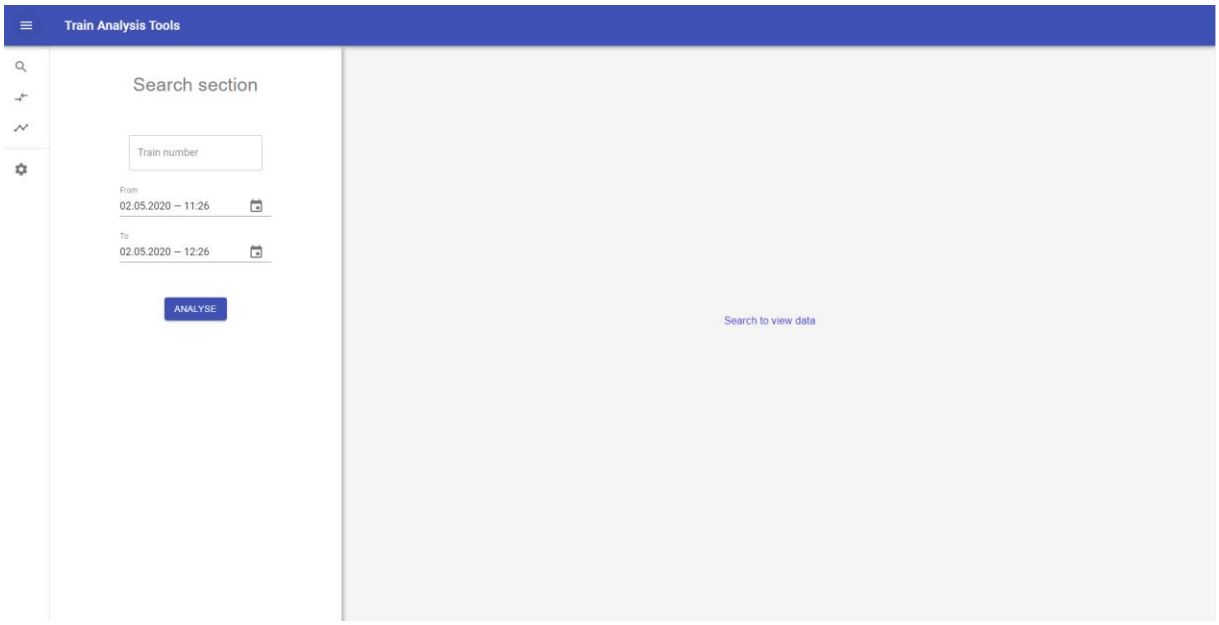



Figure 4.10: Analysis page

This is the start page of the application. It consists of a header with application title, a collapsible panel with some icon-buttons to the far left, a search section and an empty area which is the analysis area. This is where the analysis graphs will be plotted and displayed.

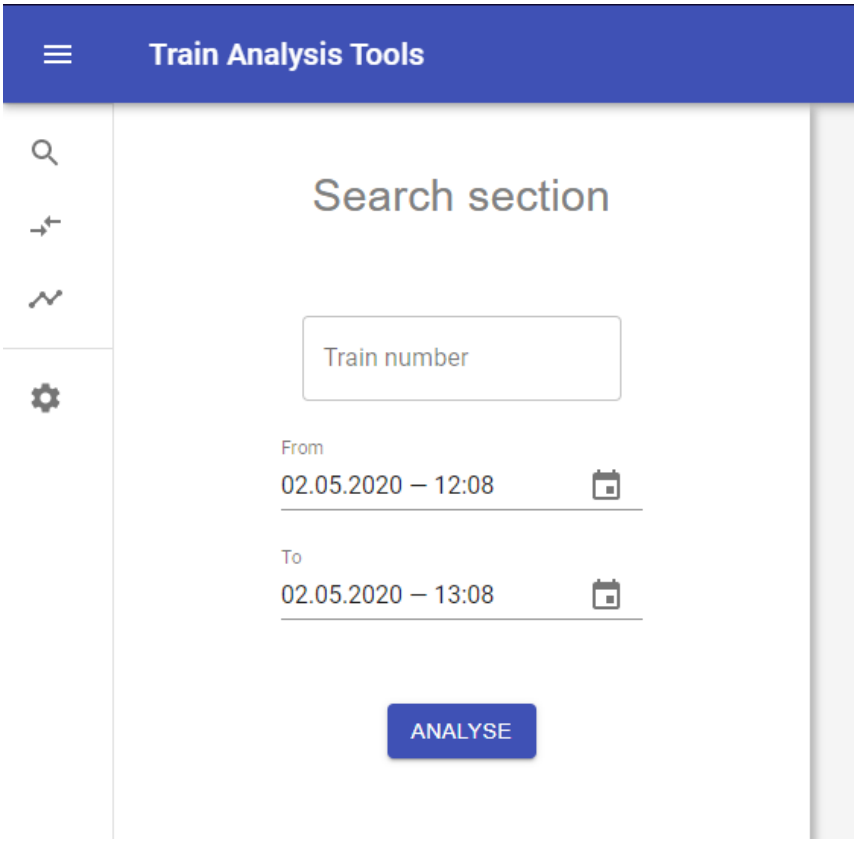



Figure 4.11: Search section

In the search section, the user is able to type a train-number, select date and time range, then search for that train. The date panel always shows today's date, from-time set to an hour ago until current time and when clicked, a popup with date/time selector will be displayed as shown in the figure below. The user is also able to enter date directly in the date-field without selecting.

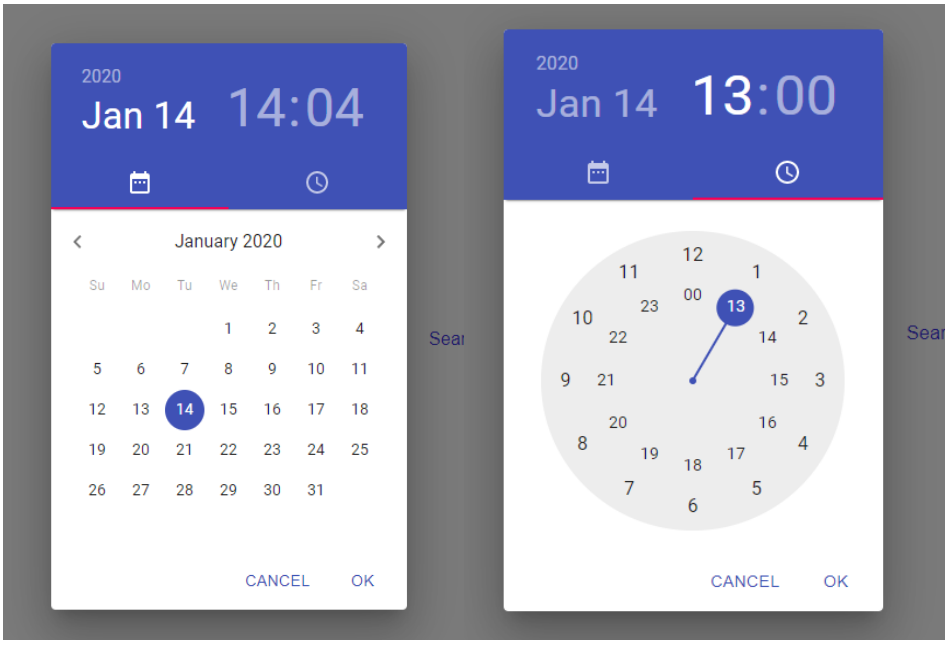



Figure 4.12: Date and time selector by Material UI

When the search button is pressed, the frontend application will display a simple loading animation while retrieving data.

When data is received, the application will display a chart where graphs and other data for analysis are shown.

### Chart

The charting library that was used as mentioned in section 3.5 is by fusioncharts. There are many other charting libraries, but this one was chosen because it provides functionality for marking specific points in time and events in the chart.


Figure 4.13: Chart displayed

Zooming in to the chart in Figure 4.14, it contains different types of information. The first thing at the very top of the chart is some information about the trip, that is the train number, and the exact start and end time of the trip.

Below that, data for analyzing and comparing the trip to the reference is shown in the chart. Different types of data are described below.

### Speed

The speed of the train is plotted in the chart as a graph. Looking at Figure 4.15, there are two speeds in the chart, one is the reference speed found by the algorithm described in section 4.3.5 and the other the actual speed of train user searched for. Both speeds are plotted in the same chart to make it easier to compare them.

When hovering over any point in the graph, both speed values at that point of time will be displayed along with the time as shown in Figure 4.15.

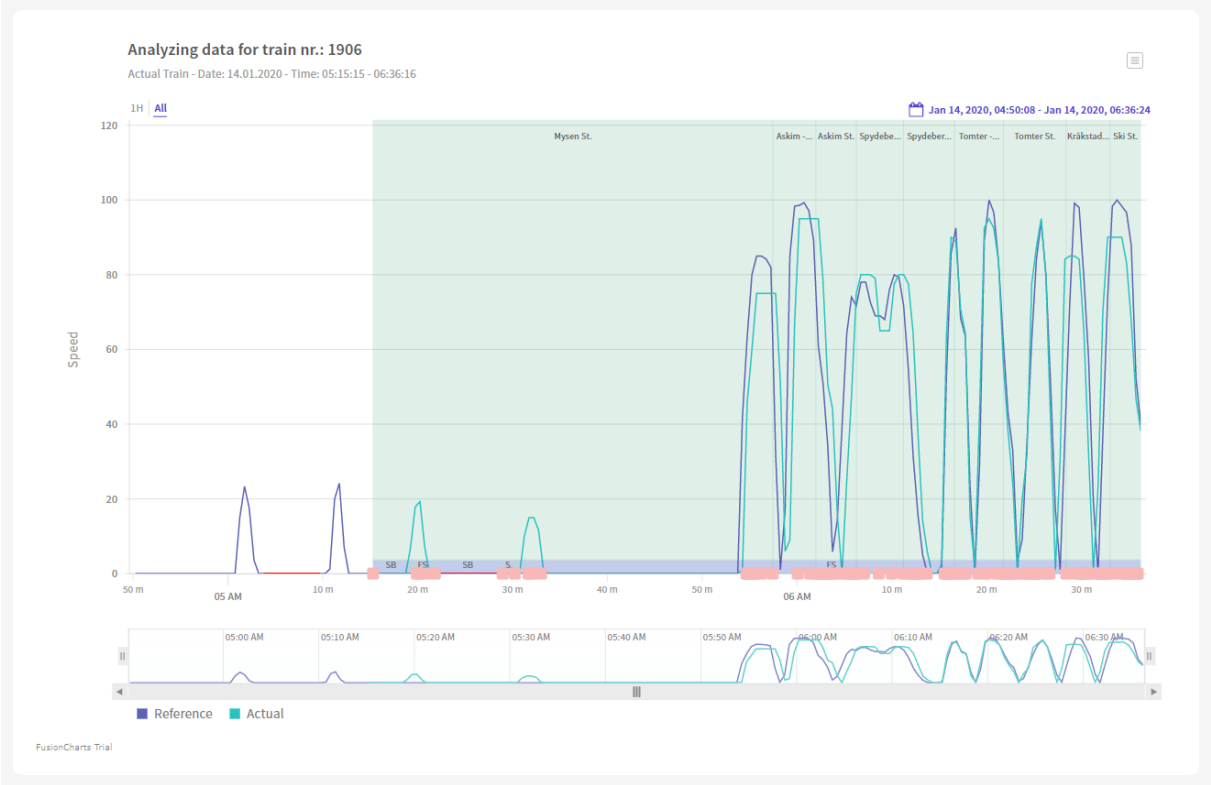



Figure 4.14: Chart with data

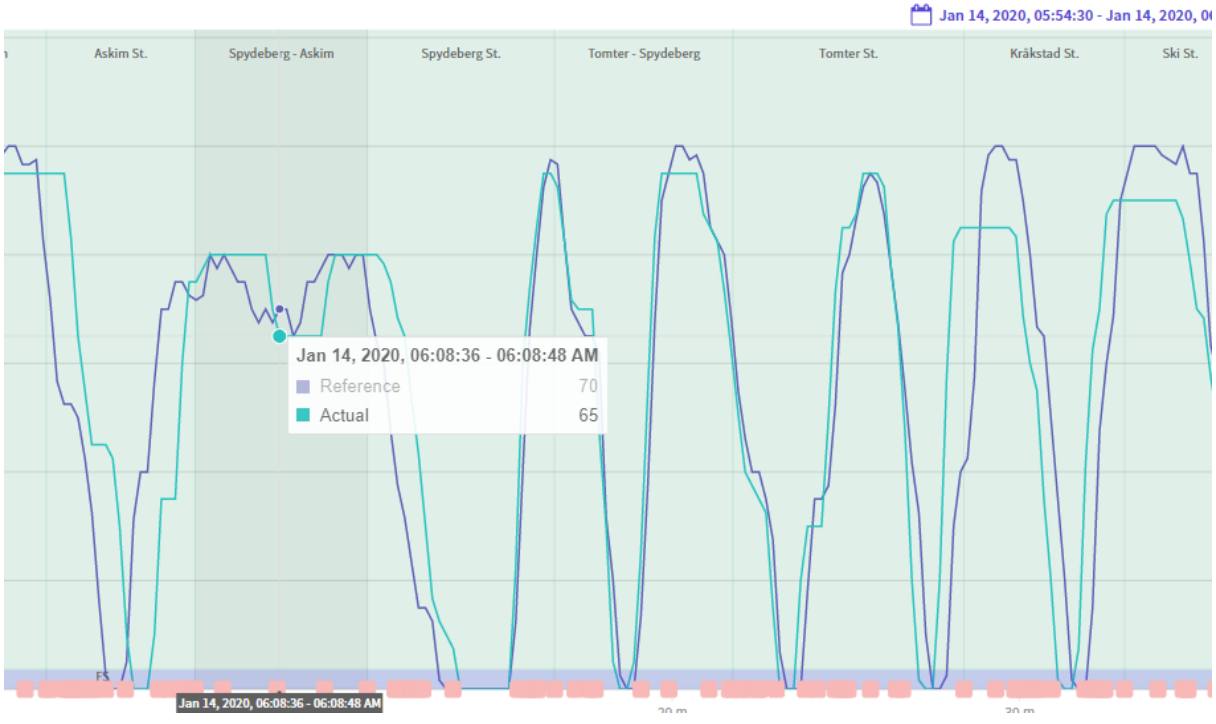



Figure 4.15: Hovering over a point of the graph

### Balises

All balises the train passed during the trip are shown at the bottom of the chart as time-markers, each placed at the exact time they were passed by train. When hovering over

any balise, the balise id, exact time passed and exact position (its placement in the real world/Eastern line) of the balise in km is displayed. This is illustrated in figure xx.

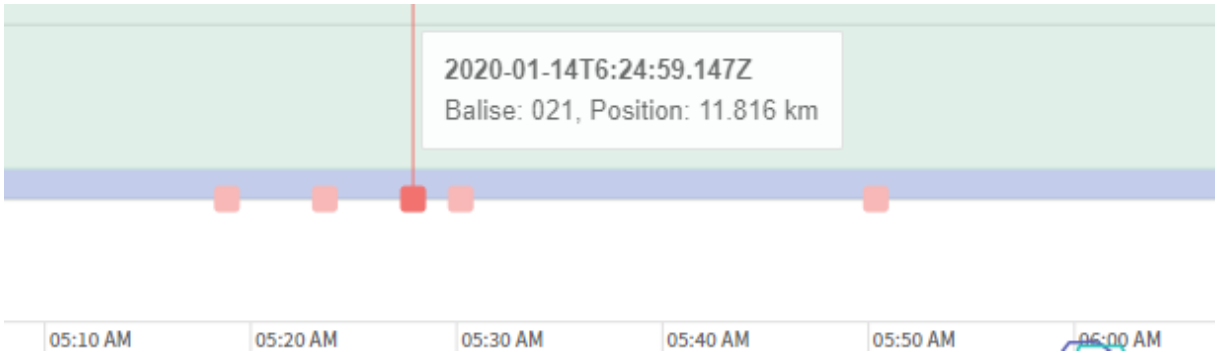



Figure 4.16: Hovering over a balise

### Train-mode

All modes train entered during the trip are also displayed at the bottom of the chart from the time train entered that mode until time it exited/switched that mode and entered another. Also when hovering over any mode-marker, marker-color changes and the exact time train entered and switched mode is shown. This is illustrated in the figure below.

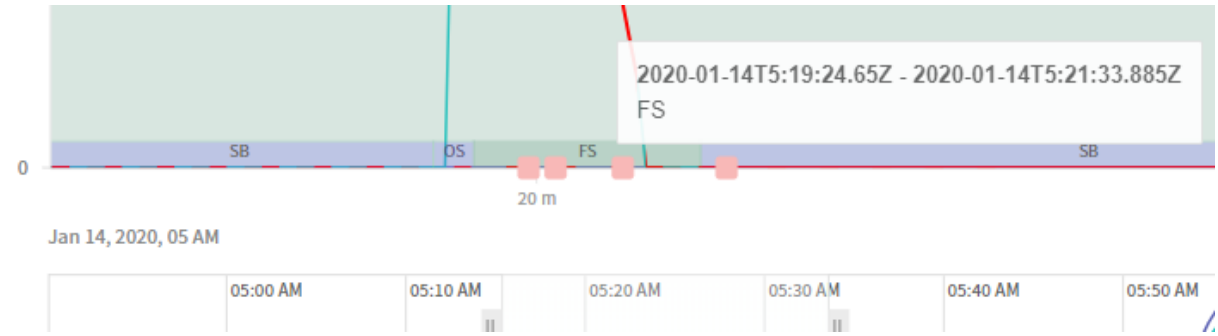



Figure 4.17: Train mode area

### Stations

Every station the train has passed or was at for a period of time is marked in the chart. Even if train is between two different stations, it will be shown in the chart. Stations are presented with time-marker areas just like train-modes, but they are full time markers, covering/occupying the entire height of the chart. This is illustrated in the figure below.

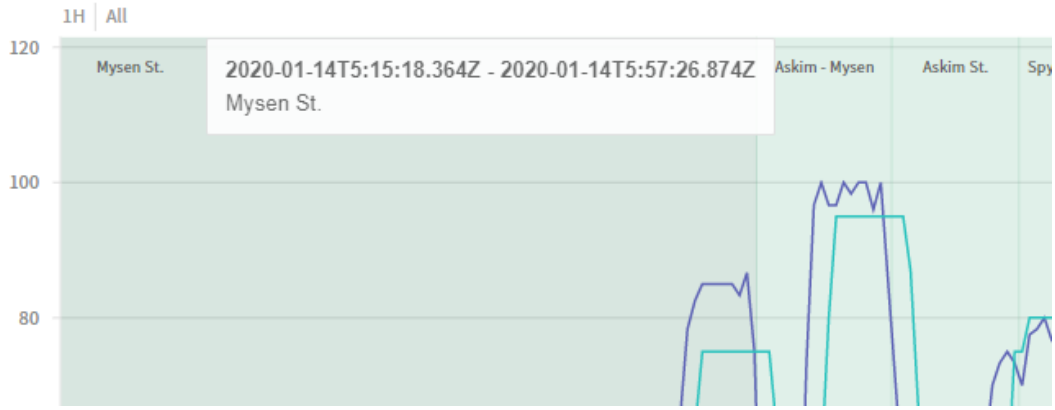



Figure 4.18: Hovering over a station area

When hovering over any area of the chart, the station name is shown along with the exact time station entered and left as illustrated in figure above.

### 4.4.2 Design

The application was designed with users in mind. When the problem was introduced in chapter 1, a lot of focus was centered around simplifying the analysis process. So designing the application to be simple-to-use and with simple design was an important part of this project.

The design choices made were based on Don Norman's design principles described in 2.2.2.

First thing that was done was choosing Material UI as a component library. The component library choice was not random, but was chosen for a reason. Not only does it provide suitable and attractive components, but it also provided the application with consistency, feedback and affordance which are three of Don Norman's design principles.

When it comes to consistency, since every component used in the application was a Material UI component (except for the chart), the application interface became consistent, meaning all elements had a similar look, and similar animations (hovering and clicking animations). Part of the choice when designing the application was to not use other component libraries as different component library's may cause inconsistency in the application design, thus the choice was to only use Material UI in the frontend application. In addition, the coloring of the application was themed which also provided consistency to the overall look and feel.

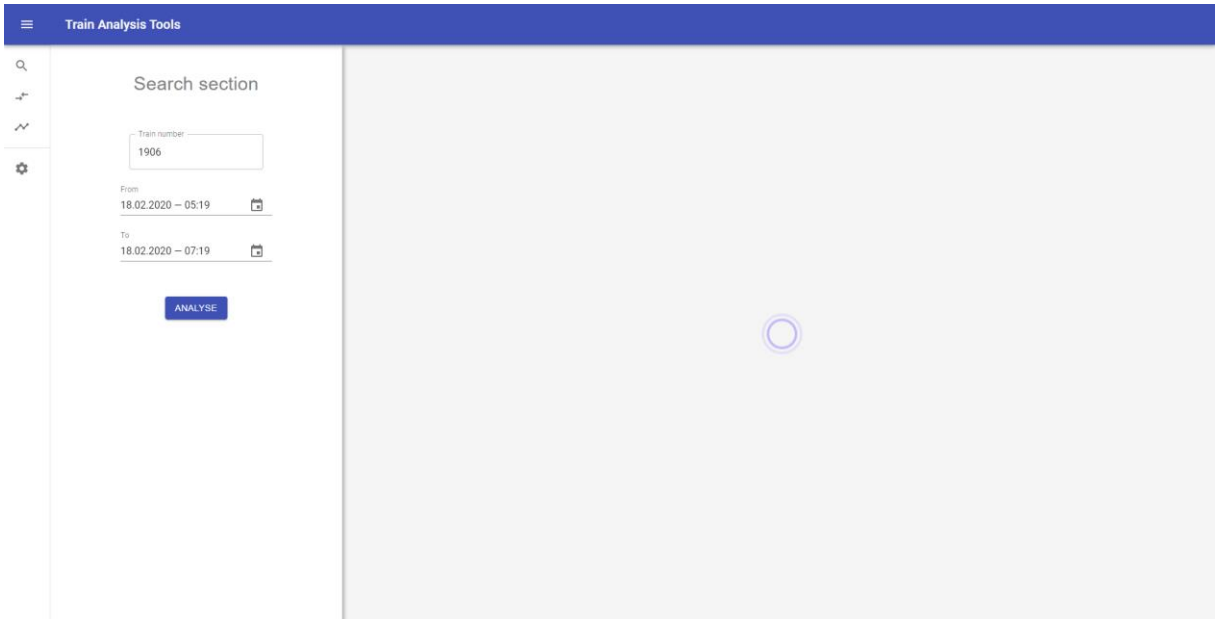



Figure 4.19: Loading data (loading animation in chart area)

When it comes to feedback, every component in the application provides feedback (By Material UI). When clicking a button, clicking in an input area or hovering over an icon, simple animations are played telling the user that an action was taken. Another part of the feedback according to Don Norman's principles mentioned in 2.2.2 was providing the user with results of the action. For example, when the user clicks the search button to search for a train, the button plays a simple animation telling the user about the action,

but then a loading animation in the graph area is displayed. This tells the user that the application is "doing something" because of the action taken, that is the application is retrieving data and loading the chart. This is shown in Figure 4.19. When this is done, the application displays the chart or a feedback message to the user if data could not be retrieved or other problems occurred. This is illustrated in Figure 4.20. Have the application not displayed any animation while retrieved data from the backend, user will be left guessing until the data is received and the chart is displayed.

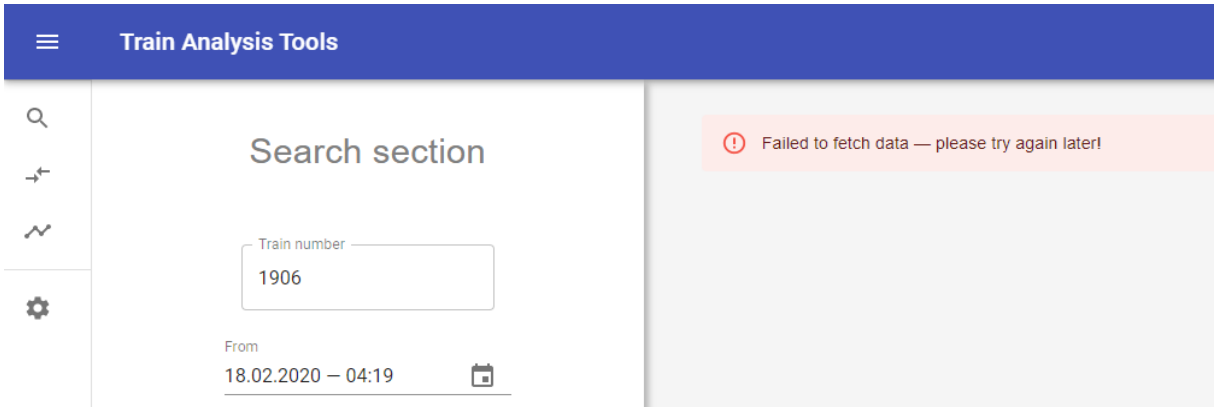



Figure 4.20: Error message about fetching data failure

This way, principle of feedback by Don Norman has been applied in this application.

Another principle that was applied in the design of the application was visibility. Everything that is important is visible in this application.

With all of that done, the application applies good practices and many of Don Normans design principles, and should be easy-to-use. Thanks to functionality provided by material UI.

### 4.4.3 Pages and Extra Features

As mentioned in 4.4.1 and the application contains a collapsible panel to the left far left. This panel contains different buttons which were suppose to represent different pages (dynamically rendered by React as a "page" in a SPA). These were not a requirement but extra features. They are not implemented yet but are there for further development.

Another feature that was included in the application is data export. The user can export the chart as PDF or as an image file. This was not a requirement either but was added since fusion chart provides this functionality out of the box and it is a useful functionality to have.

# 5 DISCUSSION

## 5.1 Technical Results

The overall results of the project does satisfy the initial requirements given by the client, with movement authority being an exception. However, as with almost every other technological product, there is room for improvements.

### 5.1.1 Uncertainty in Reference

An interesting topic of discussion when it comes to the overall results is the solution to obtaining an optimal train reference.

Although the application satisfies all initial requirements given by client including the reference, with movement authority being an exception, the solution quality of each requirement is in many cases as important as fulfilling the requirement itself.

The current reference solution provides a good start or entry level train reference, reason being reference currently is only dependent on speed. As mentioned in section 1.4, the optimal train reference may depend on many other factors than train speed only, each with own weight. However, this was out of the scope of this project as mentioned, due to the level of complexity these other factors may add into the project.

Looking into our solution, the algorithm was straight forward. Every balise on the Entire eastern line was mapped to a speed value according to its position, and stored in a hash map as a key-value pair. This is illustrated in figure below.

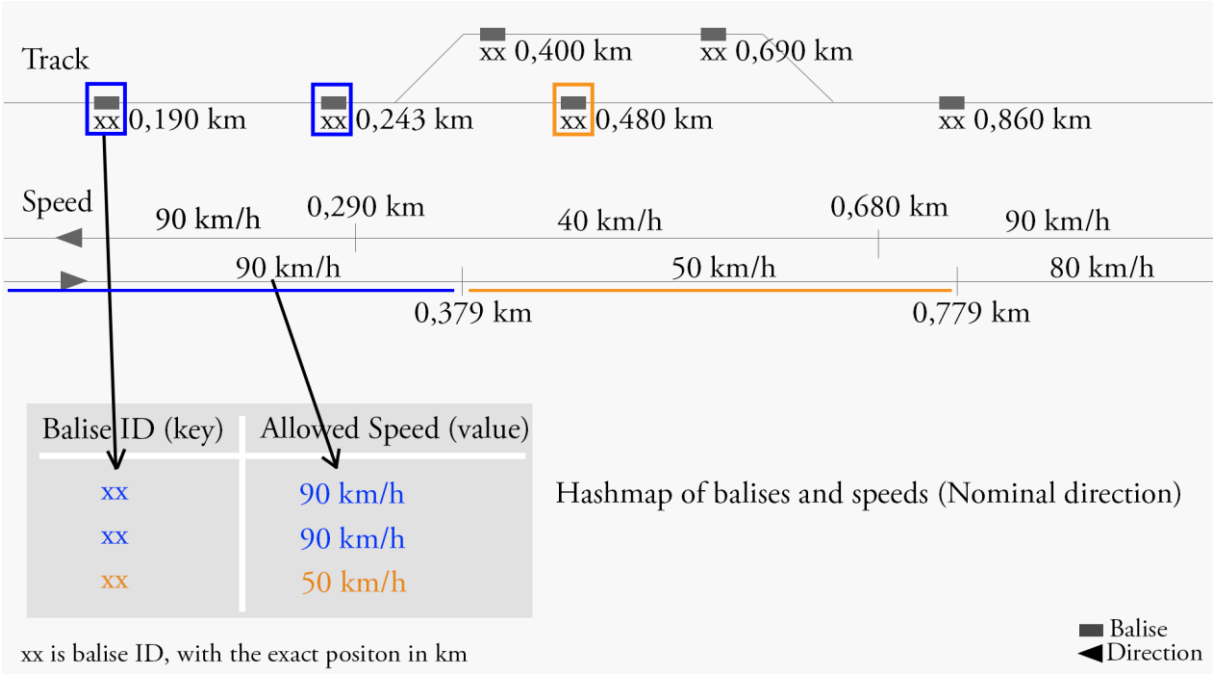



Figure 5.1: Mapping of balises to allowed speeds in Eastern line

When the train sends a position report message (which as mentioned in section 2.5.4 contains speed, last read balise, train-mode and so on) the balise reported is used to look up the allowed speed at that position (using the hash map). This speed is then stored in a list of "optimal" speeds. Reported speeds by train are also stored in a separate list of "actual" speeds.

The difference between the two data lists, the "optimal" speed list (allowed speeds at reported positions) and the "actual" speed list is then calculated using the MAE method described in section 2.6.1. This MAE score is then compared with the MAE score of the current reference stored in database, the smallest MAE is the one closest to the "optimal" speed. Based on that comparison, reference is either updated or kept.

Although the solution mentioned above does the job, however, it has some down sides to it and some uncertainties. These uncertainties are mainly related to mapping.

Looking at the figure below, if the train enters a new section which has different allowed speed and did not yet read any balise in the new section entered, the train will still report the last read balise group which is laying in the previews section and is mapped to the speed of that previews section. This will cause the algorithm to think train should be driving at 90 km/h (because balise being reported is mapped to 90 km/h). Therefore, if train is driving at 50 km/h (which is the allowed speed) the algorithm will consider this as a bad thing, because it thinks train should drive at 90 km/h as mapped. Algorithm will calculate a difference or an absolute error (AE described in 2.6.1) of 90 km/h – 50 km/h = 40 km/h, which will contribute to inaccurate results in total MAE score.

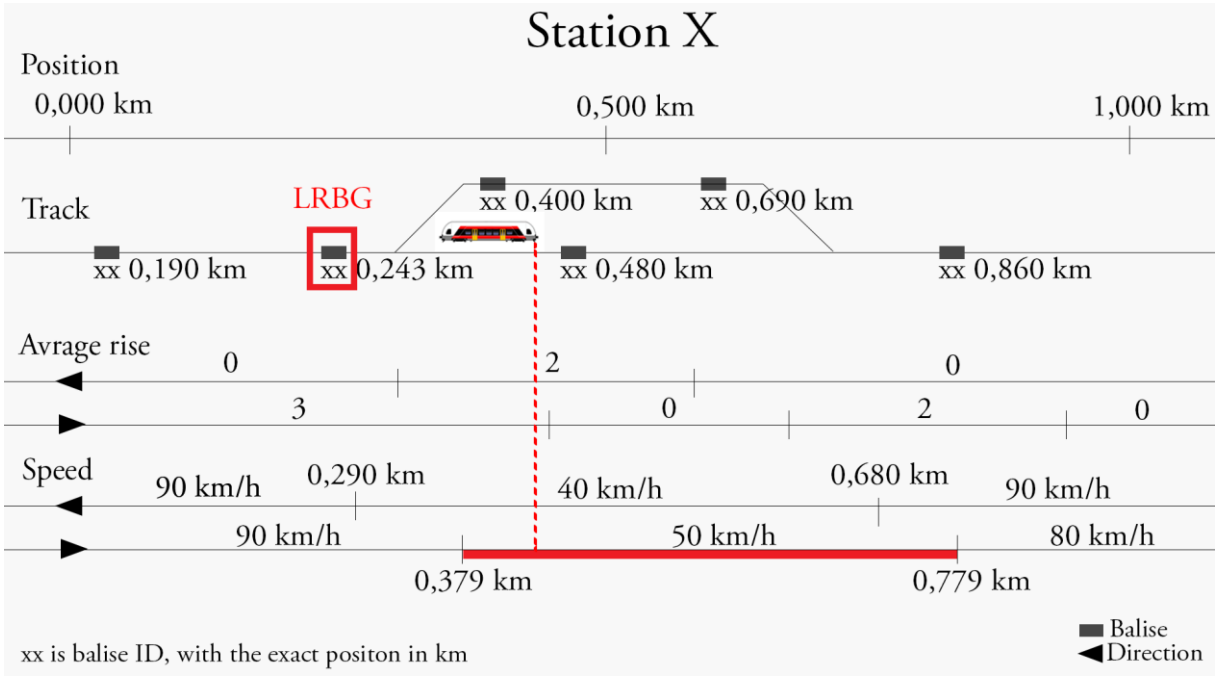



Figure 5.2: Reference uncertainty when entering new section

During the period train enters a new section and reads the next balise in the new section, train would have travelled 0.101 km in example above (Figure 5.2). Considering the train is driving at about 50 km/h, it will take about 7.3 seconds to reach next balise after doing the math. Since train sends one position report every six seconds, there will be at least one position report sent during that period. This one report will contribute to the error. However, since it is only one position report sent during that period, it will not contribute to error by much, thus not affect the score by much. However, on other positions and locations of the eastern line, this scenario may be repeated but the distance may be longer, thus more position reports would be sent which will increase the uncertainty in the MAE score.

This make the solution not produce the perfect reference but a very close one, depending on length of these "uncertain sections".

### Better solutions

One of the solutions that was considered when this challenge appeared was mapping position-ranges to a speed value instead of mapping balises to speeds. This means for every section, the start positon in km and the end position of section are used as a range key, and speed of that section is mapped to the range as a value.

As an example looking at Figure 5.3, from position 0.379 km to 0.779 km allowed speed is 50 km/h.

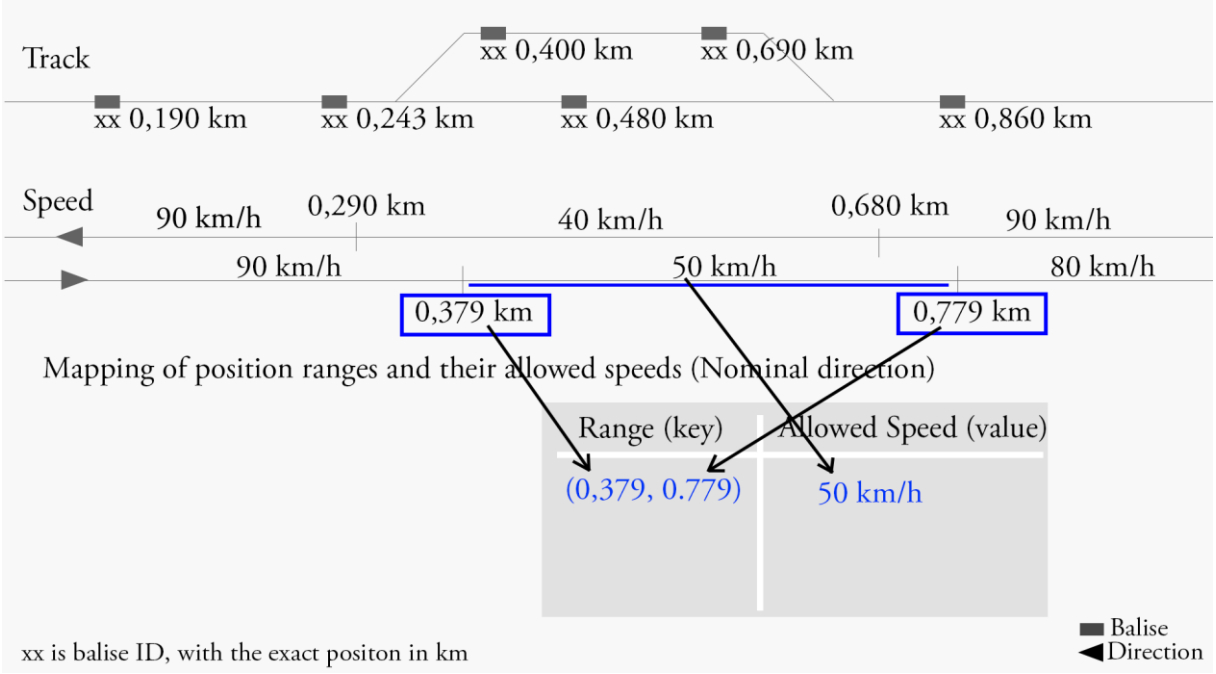



Figure 5.3: Mapping section range to allowed speed

This can be mapped as a range for example by using google guava library, which provides such mapping functionality.

```
((0.379, 0.779), 50)  //passing 0.400 as key returns 50 as value
```

Snippet 5.1: Mapping section range to allowed speed

So if train is reporting xx as last read balise (balise at position 0,243 km in Figure 5.3), and an estimated train frontend position by 0.157 km is reported by train, it means train's exact location in km is 0.400 km (0.243 + 0.157) which is in the range of speed 50 km/h according to the mapping shown above. Passing 0.400 as the key in the map above, will return 50 as the value.

This seemed to be the perfect solution, especially that train reports estimated frontend position from last read balise (how many km it drove since it last read balise reported in the message), which means almost exact position of train is available to use.

However, the road may consist of multiple tracks on some positions as mention in section 2.5.4, especially in a station stop. Which causes an issue illustrated in Figure 5.4. This is an issue where the position in known, but not the track if using a mapping of position ranges and speeds only. The problem is that each track may have different allowed speed as mentioned in 2.5.4.

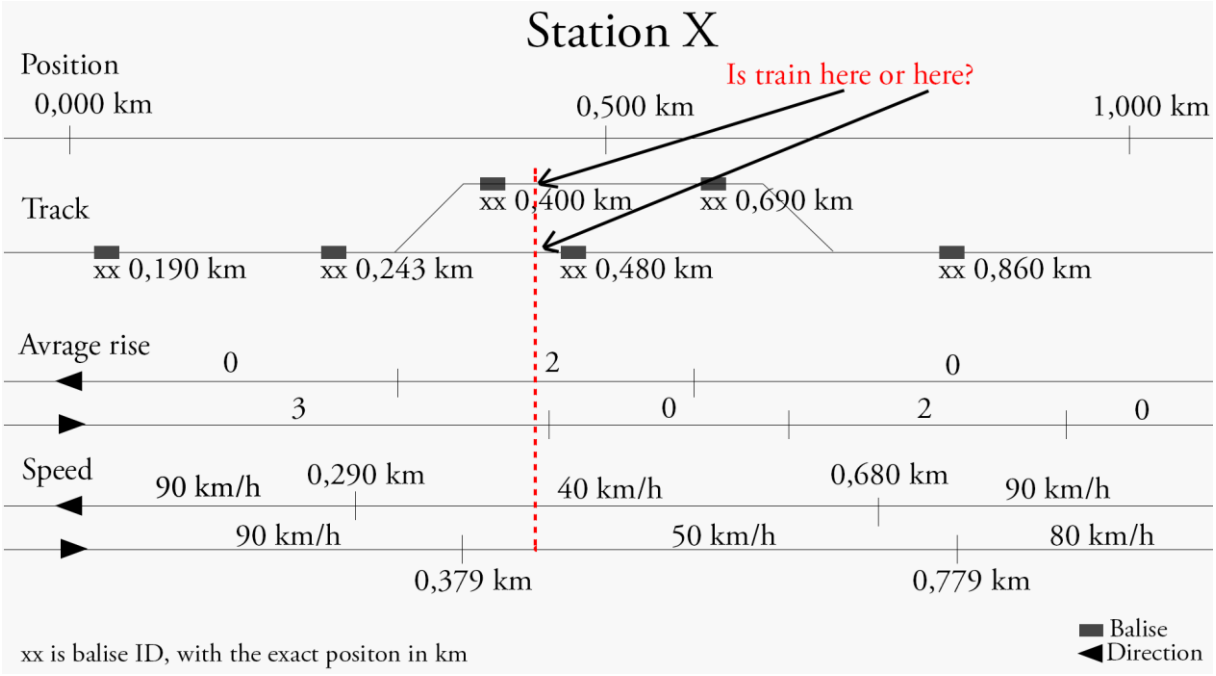



Figure 5.4: Track not known by algorithm

This means in addition to mapping position ranges to speeds, we need to extend the key in the hash-map with another parameter to tell us which track train is on. That third parameter is the last read balise. If last read balise I known, track train is on is also known.

Here is where things started to become more complicated. Even if we were able to create a mapping where a key consists of a balise combined with a position range and a value of speed, other challenges will be met. This is illustrated in Figure 5.5.

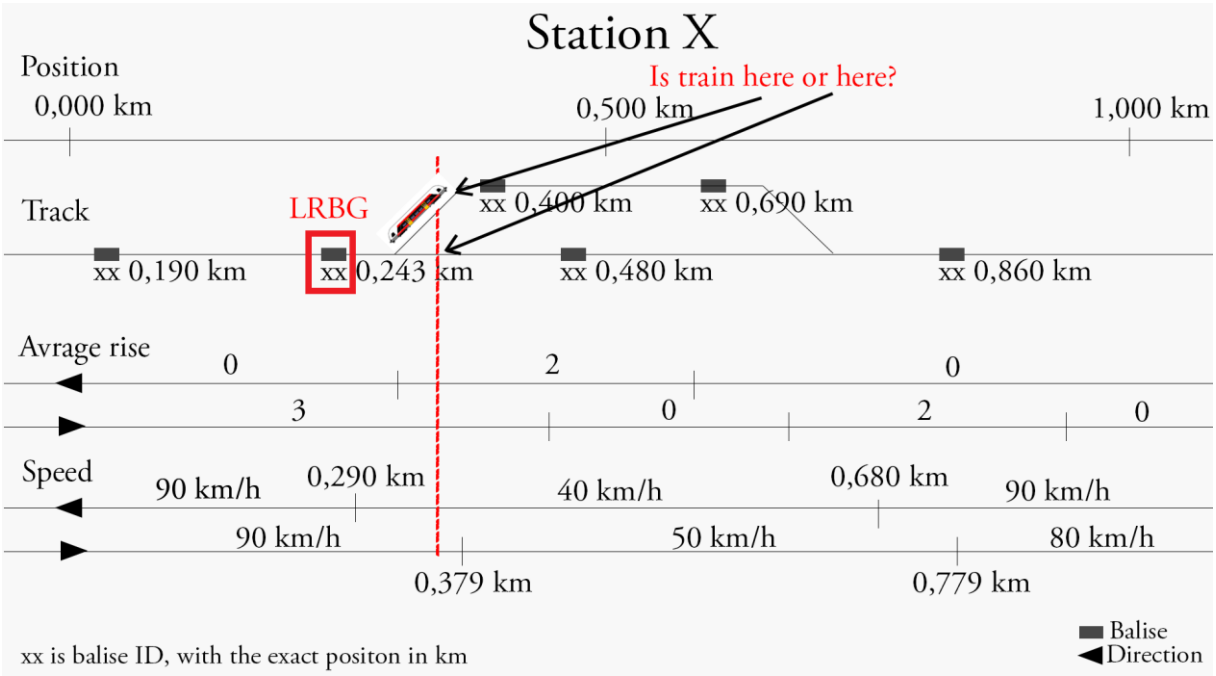



Figure 5.5: Track not known even with extended mapping key

In the scenario shown in Figure 5.5 above, the last read balise is the one on track I. Track II has a balise only further at position 0.400 km. The train is driving towards that balise, which means the algorithm will not know the train is on that track II until it registers the balise on that track. With a mapping where the key consisting of last read balise and position range, algorithm only knows the track based on last read balise and estimated train frontend position. Thus, it will not detect that train is going towards track II.

Reading though the official documentation by the *European railway agency*, I could not find an indicator to switching tracks sent by train or something similar to solve this in an easier or less complicated way. There could be more information to this, but that's how far I was able to get due to limited time and limited resources in this project.

One solution that was considered for last mentioned track problem was to read next balise to determine where train is heading towards. However, this becomes very complicated, especially the mapping part. Mapping many track, different directions and speeds on different stations and implementing a complicated algorithm was time consuming, thus was left out for future development, as time in this project was limited.

## 5.2 Development

### 5.2.1 Methodology

Earlier, in chapter 3, it was mentioned that scrum was used as a methodology under the development of the project. However, because of the team size, consisting of a single student, not all aspects of scrum were applied. Applying the exact values and principles of scrum as a solo developer was not possible. During the development, it was not really experienced as "scrum" like I had earlier in a team, rather it felt more like a modified version of scrum or a "scrum-like" methodology.

Even though it was not experienced as "scrum", it still offered many advantages. One of the main advantages experienced during this project when using scrum as a solo developer was that it helped keep the focus during each spring on the development of planned requirements that are in the sprint backlog and kept the focus on the sprint goal.

Sometimes when developing a software project one might attempt to add other functionality one thinks are "cool", which shifts the focus during the development on undesired functionality or less important work. This did however not happen under this project. The focus was always on reducing the sprint backlog as much as possible before sprint end and sprint meeting.

With that said, other aspects of scrum were followed in a great manner. The development consisted of short iterations (sprints), sprints were planned, sprint meetings were held with the product owner and stockholders and product backlog was updating constantly. The progress was also tracked using the JIRA software. All of that created a good development experience and made the development process much cleaner and easier.

### 5.2.2 Estimation

One of the challenges experienced during this project was related to estimation. As a single student, I tried to estimate weather it will be possible to complete all requirements in time.

At the beginning of the project, I was very positive and had high expectations. I expected to finish all requirements and do more of the "nice to have" requirements. However, my estimation and expectations were a little too positive and optimistic. Many things took longer than expected.

One of the things that consumed a lot of time of this project was the final report. Being a single student, it took a lot of time to complete this report.

Another thing is research. Researching good practices, theories to follow and principles to apply when designing the product and developing the system was time consuming. Not only research itself, but also finding trustworthy resources.

The main focus during the project was centered around good system design and architecture as well as clean code and good documentation. Implementing a lot of undocumented functionality with bad code, unclear design and unstructured architecture was not given a thought.

Because of that, there was not enough time to implement requirement related to movement authority. This is the only requirement not done because of positive estimation, optimistic expectations and larger focus on system design and architecture.

However, at the end this ended up working very well. The project was conducted professionally, the report written successfully and the application was developed, well documented and deployed.

## 5.3 Further Work

The system is well developed with clear architecture, clean code style and is well documented, making it easy for next developer/developers to further expand it.

### 5.3.1 Mapping in Application

Earlier it was mentioned that balises were mapped to speeds to solve reference finding problem. However, mapping is hardcoded in the application in a file containing constants. The problem with the mapping being hardcoded in the applications is if any updates to the line happen in future. If a balise is removed, replaced or a new balise is added, or if speed restrictions are changed on any section, this will produces errors and uncertainties in the reference algorithm.

One thing that could be done to solve this issue is to move all mappings to database. That way it is possible to update mappings in case of any updates to the Eastern-line regarding speed or balises happening in the future. Due to limited time, this was not done.

### 5.3.2 Improving Reference Finder

Uncertainties in reference finding algorithm were discussed earlier in section 5.1.1

The current implemented solution produces good results with small margin of error depending on the length of the "uncertain section" as mentioned. If more arcuate results are desired by client, the suggested and more complicated solutions described in 5.1.1 could be implemented.

### 5.3.3 Functionality and Features

As the system currently is, it is not rich in functionality, but provides most important functionality and data according to requirements given by client.

For further development, client Abraham Mosye is responsible for providing extra requirements if desired. However, there are some requirements given by client later as the project progressed which are "nice to have" or "should have" requirements. All of these are noted in the product backlog.

One of the requirements given by client later in time and noted in the backlog was to be able to search by station and present only data from that station. Another requirement that came later in the project was balise error detection where if any deprecated balise in the railway or the eastern line is detected it will be presented. Another requirement was to present in a visual way all places train was later than it should be. All other requirements are listed in product backlog and are there for further development.

## 5.4 Usage on Other Lines

Even though most of the focus was centered around the Eastern line during this project, the system developed and the algorithms implemented can be applied and used in any line in Norway driven by ERTMS (operating on ERTMS).

Only thing required to do this is to expand any mappings or list with data from other lines. For example, list containing train numbers used in reference finder algorithm should be expanded with train numbers from other lines, and mapping of balises to speeds should be expanded with data from other lines. No modification in algorithms or logic is required at all.

The solution is general, thus the same application and logic should work on any line in Norway. Not only that, but possibly any country using ERTMS should be able to use this application with no problems if correct mapping exists.

## 5.5 Evaluation

Overall, I think I did a great job in this project, the required functionality is implemented and well documented. The client is also satisfied with the final product. Supervisors expressed their satisfaction with the work done. Thus, the project is considered successful by different parties.

The requirements in this project were adapted to the size of the group, a single student, which made it possible to complete most of requirements in time.

Being a solo developer in this project was a bit of a disadvantage. However, I gained a lot of experience on the total process of developing and documenting scientific work, as I did everything on my own and with the help of supervisors and client throughout the project.

# 6 CONCLUSION

The project intention was to simplify the analysis process and make it easier to detect deviation, errors and adverse events when analyzing data. This was done by developing comparison software and algorithms to compare speed along with other train data to a reference.

The reference part of the problem was solved and produces great results. The algorithm behind reference finding runs daily and stores a new reference if a better one was found. For more accurate results, better solutions does exist, but are more complex and time consuming. Suggestion for more accurate results were presented in this report but weren't implemented due to time limitations along with being a single student conducting this project.

When it comes to the second part, that is developing analysis software, all requirements requested by client were implemented, except for the movement authority part. Estimation during the project was a big challenge. Many tasks consumed longer time than expected and estimated. Other than that, the system has been developed with good architecture and known software engineering and design principles. The system is also well documented making it easy to further develop and expand on existing solutions.

By carrying out this project, I was able to gain experience on planning and conducting a project, and I was able to integrate previously acquired knowledge throughout my study at NTNU to solve new challenges and problems. Not only that, I was also able to acquire new knowledge and experience to solving new challenges to be faced in the future.

The project is considered successful by me and the client, the results have met expectation and provide bases for further work and development.

It was a great experience to work with ERTMS and train data and experiment with different algorithms and solutions. I am looking forward to learn more and do better job further in my future carrier.

# 7 REFERENCES

[1] BaneNOR, "ERTMS - Fakta," Bane NOR, [Online]. Available: https://www.banenor.no/Prosjekter/prosjekter/ertms/fakta/. [Accessed May 2020].

[2] Jernbaneverket, "ERTMS for Dummies 1 - Grunnleggende funksjonalitet," 12 June 2014. [Online]. Available: https://www.banenor.no/globalassets/documents/ertms/1-ertms-for-dummies-grunnleggende-001.pdf. [Accessed February 2020].

[3] IBM, "Software development," [Online]. Available: https://www.ibm.com/topics/software-development. [Accessed February 2020].

[4] S. B. J. L. Michael Bloch, "Delivering large-scale IT projects on time, on budget, and on value," October 2012. [Online]. Available: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value. [Accessed February 2020].

[5] "Why do projects fail?," [Online]. Available: http://calleam.com/WTPF/?page_id=1445. [Accessed February 2020].

[6] ScienceDirect, "Waterfall Model," [Online]. Available: https://www.sciencedirect.com/topics/computer-science/waterfall-model. [Accessed February 2020].

[7] A. Styve, "Prenstaion by Arne Styve at NTNU - Prosjektledelse i IT-prosjekter," [Online]. Available: https://learn-eu-central-1-prod-fleet01-xythos.s3-eu-central-1.amazonaws.com/5def77a38a2f7/3806388?response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27Arne%2520Styve_Prosjektledelse%2520i%2520IT%2520Prosjekter%25202020.pdf&response-content-t. [Accessed February 2020].

[8] A. Alliance, "Agile 101," agilealliance.org, [Online]. Available: https://www.agilealliance.org/agile101/.

[9] Developers, "Agile Manifesto," agilemanifesto.org, [Online]. Available: https://agilemanifesto.org/.

[10] Wikipedia, "Scrum (software development)," [Online]. Available: https://en.wikipedia.org/wiki/Scrum_(software_development). [Accessed February 2020].

[11] J. S. Ken Schwaber, "The Scrum Guide," *The Definitive Guide to Scrum: The Rules of the Game,* pp. 1-19, November 2017.

[12] H. M. Hafedh Mili, "Concerned about separation," 16 October 2005. [Online]. Available: https://www.researchgate.net/publication/244446574_Understanding_separation_of_concerns. [Accessed February 2020].

[13] Wikipedia, "Separation of concerns," [Online]. Available: https://en.wikipedia.org/wiki/Separation_of_concerns. [Accessed February 2020].

[14] S. Rekhi, "Don Norman's Principles of Interaction Design," 23 January 2017. [Online]. Available: https://medium.com/@sachinrekhi/don-normans-principles-of-interaction-design-51025a2c0f33. [Accessed 29 February 2020].

[15] Wikipedia, "Software architecture," [Online]. Available: https://en.wikipedia.org/wiki/Software_architecture. [Accessed May 2020].

[16] "Client-server model," wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/Client%E2%80%93server_model.

[17] "Front end and back end," wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/Front_end_and_back_end.

[18] Wikipedia, "Multitier architecture," [Online]. Available: https://en.wikipedia.org/wiki/Multitier_architecture. [Accessed February 2020].

[19] M. Richards, "Chapter 1. Layered Architecture," February 2015. [Online]. Available: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html. [Accessed March 2020].

[20] Medium, "Layered Architecture," 16 October 2019. [Online]. Available: https://medium.com/java-vault/layered-architecture-b2f4ebe8d587. [Accessed March 2020].

[21] Oracle, "Chapter 1 Designing Applications," [Online]. Available: https://docs.oracle.com/cd/E19644-01/817-5448/dgdesign.html.

[22] lhotka.net, "Should all apps be n-tier?," 21 July 2005. [Online]. Available: http://www.lhotka.net/weblog/ShouldAllAppsBeNtier.aspx. [Accessed March 2020].

[23] Wikipedia, "Single-page application," [Online]. Available: https://en.wikipedia.org/wiki/Single-page_application. [Accessed March 2020].

[24] "Spring Framework," wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/Spring_Framework. [Accessed April 2020].

[25] "React - A JavaScript library for building user interfaces," reactjs.org, [Online]. Available: https://reactjs.org/. [Accessed 13 February 2020].

[26] "What is Babel?," BabelJS.io, [Online]. Available: https://babeljs.io/docs/en/index.html.

[27] "Concepts - Webpack," webpack.js, [Online]. Available: https://webpack.js.org/concepts/.

[28] "Apache Lucene," apache.org, [Online]. Available: https://lucene.apache.org/.

[29] "What is an Elasticsearch Index?," Elastic.co, [Online]. Available: https://www.elastic.co/blog/what-is-an-elasticsearch-index.

[30] Elastic, "What is Elasticsearch?," Elastic.co, [Online]. Available: https://www.elastic.co/what-is/elasticsearch.

[31] "Removal of mapping types," Elastic.co, [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/7.x/removal-of-types.html.

[32] "Logstash," Elastic.co, [Online]. Available: https://www.elastic.co/logstash.

[33] "How Logstash Works," Elastic.co, [Online]. Available: https://www.elastic.co/guide/en/logstash/current/pipeline.html.

[34] B. NOR, "GSM-R," Bane NOR, [Online]. Available: https://www.banenor.no/kundeportal/jernbanen-i-norge/tjenester-og-priser/tilhorende-tjenester/gsm-r/gsm-r-mobilnett/. [Accessed February 2020].

[35] BaneNOR, "Signalsystem," Bane NOR, [Online]. Available: https://www.banenor.no/Jernbanen/Jernbanedrift---eit-komplisert-samspel/Jernbaneverket-satser-pa-ny-teknologi-for-signalanlegg/. [Accessed February 2020].

[36] ERA, "Chapter 7 - ERTMS/ETCS Language," in *Subset 26 - System Requirements Specification*, European Union Agency for Railways, 2016.

[37] ERA, "Chapter 8 - Messages," in *Subset 26 - System Requirements Specification*, European Union Agency for Railways, 2016.

[38] ERA, "Subset 23 - Glossary of Terms and Abbreviations," 12 5 2014. [Online]. Available: https://www.era.europa.eu/sites/default/files/filesystem/ertms/ccs_tsi_annex_a_-_mandatory_specifications/set_of_specifications_2_etcs_b3_mr1_gsm-r_b1/index003_-_subset-023_v310.pdf. [Accessed May 2020].

[39] Wikipedia, "Balise," [Online]. Available: https://en.wikipedia.org/wiki/Balise. [Accessed February 2020].

[40] "Absolute Error & Mean Absolute Error (MAE)," statisticshowto.com, [Online]. Available: https://www.statisticshowto.com/absolute-error/. [Accessed 12 April 2020].

[41] "Build beautiful web & mobile dashboards," Fusioncharts, [Online]. Available: https://www.fusioncharts.com/. [Accessed January 2020].

[42] Wikipedia, "Waterfall model," [Online]. Available: https://en.wikipedia.org/wiki/Waterfall_model. [Accessed 2020].

# APPENDICES

| | |
|---|---|
| Appendix 1 | Installation and Testing Guide |
| Appendix 2 | Minutes 1 to 9 |
| Appendix 3 | Status reports 1 to 7 |
| Appendix 4 | Preliminary report |
| Appendix 5 | Initial wireframe sketches |
| Appendix 6 | Initial Flowcharts |

# Appendix 1
# Installation and Testing Guide

# Installation and Testing Guide

In order to test the system, there are few software installations needed and some steps to configure database and add data which needs to be followed. All of that is described in this document.

## Frontend

To run the frontend application, you need to do the following:

1. Download node.js
2. Using CMD navigate to the frontend project and go to the folder where `package.json` file is located.
3. While in that folder install the dependencies by running `'npm install'`
4. Start the frontend app by running `'npm start'`

A new browser window or tab should open and the application should run.

## Backend

To run the backend application, you need to do the following:

1. Use Java 1.8 as an SDK (it may not work with newer versions)
2. Download Maven
3. Access the backend project where pom.xml file is located
4. Install maven dependencies by running `'mvn install'`
5. To run the backend app you can run the commend: `'mvn spring-boot:run'`

For other running methods, you can refer to this link.

## Database

For the backend application to work as intended, it needs data. This data is stored in an elsticsearch database. Follow the steps below to download and configure the database.

1. Download Elasticsearch database version 5.5.3 and unzip it.
2. Sett the cluster name to the same name as in in configuration of the backend application. This can be done by following the steps below:
    a. After unzipping the downloaded elasticsearch zip file go to elasticsearch-5-5-3>config
    b. Open a file called config.yml
    c. Scroll all the way to the end of the file

    d. And insert this line at the end of file: `'cluster.name: ertmslog'` (do not add #, this is used for commenting)
    e. save and exit
3. Run elasticsearch by accessing elasticsearch-5-5-3>bin then double clicking elasticsearch.bat (this will open a cmd, wait until it says "started", make sure do DON'T close the command prompt, closing it means stopping elasticsearch from running!)
4. Let it run for about one minute, after that close the cmd (this will create a data folder).
5. Download the attached data.
6. While in the data folder, copy both folders inside of the data folder
7. Go to elasticsearch-5-5-3>data>nodes>0>indices then paste both folders (if you don't find a folder named "indices" make sure to create it, then paste copied folders in it
8. Run elasticsearch again by accessing elasticsearch-5-5-3>bin then double clicking elasticsearch.bat

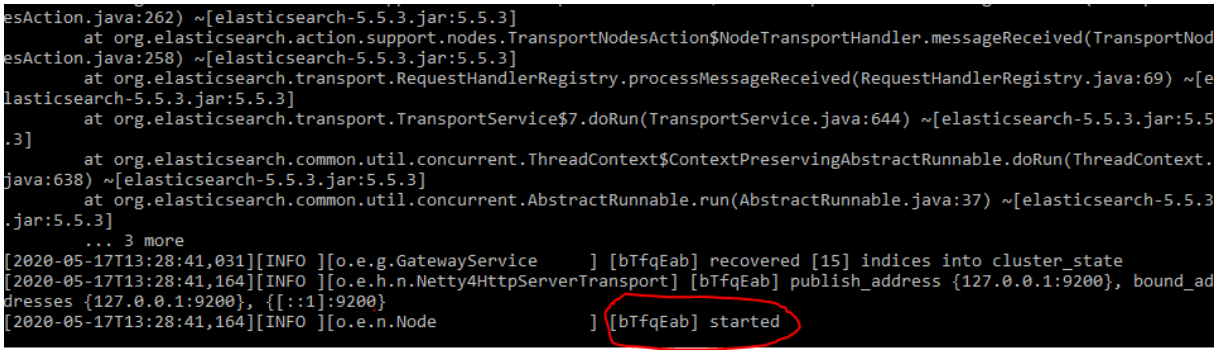



Figure above shows Elasticsearch started.

Now that elasticsearch have started, you can test the applications.

## Important information

It is very important to download the same versions listed above. Otherwise things may not work, especially pay attention to elasticsearch version. This needs to exactly be 5.5.3 for compatibility reasons.

The attached data folder only contains data of one train-line for three different days. Please read "READ ME" file inside of the data folder which tells you the date and time each of these lines run.

# Appendix 2
# Minutes

NTNU
Department of ICT and
Natural Sciences

# Minutes

Date/time: 23 January 2020 / 02:00 PM
Place: NMK B block
Arranger: Sarmad Saeed
Regards: Bachelor thesis “Train route analysis”

Participants:

| Present | Name | Title | Phone | Email |
|---|---|---|---|---|
| ☑ | Sarmad Saeed | Student | 45559633 | sarmadsa@stud.ntnu.no |
| ☑ | Ibrahim A. Hameed | Supervisor | | ibib@ntnu.no |
| ☑ | Anniken Susanne T. Karlsen | Supervisor | | anniken.t.karlsen@ntnu.no |
| ☑ | Abraham Mosye | Contracting | | mosye.abraham@banenor.no |

☒ Not present

Initially met supervisors Anniken and Ibrahim and discussed around the project. After that we agreed on meeting dates and times. There are going to be one meeting every second week with the main supervisor Ibrahim and optionally Anniken. The place and form of the meeting may vary, some meetings may be done through skype, others will be done by meeting in person. As for the company, the student will be responsible for contacting them and arranging meetings depending on the need. Main supervisor Ibrahim will also be able to contact the company if needed.

After meeting supervisors, student and main supervisor Ibrahim met contractor Abraham from the company Bane NOR through skype. Abraham described the product to be developed in more details, the objective/use of the product, and the problem this product will solve. After the meeting with Abraham, few possible solutions to the given problem were discussed with Ibrahim.

Until next meeting, the main focus will be working on finishing the preliminary thesis report and presentation of the thesis.

NTNU
Department of ICT and
Natural Sciences

# Minutes

Date/time: 5 February 2020 / 12:30 PM
Place: NMK B block
Arranger: Sarmad Saeed
Regards: Bachelor thesis “Train route analysis”

Participants:

| Present | Name | Title | Phone | Email |
|---|---|---|---|---|
| ☑ | Sarmad Saeed | Student | 45559633 | sarmadsa@stud.ntnu.no |
| ☑ | Ibrahim A. Hameed | Supervisor | | ibib@ntnu.no |
| ☒ | Anniken Susanne T. Karlsen | Supervisor | | anniken.t.karlsen@ntnu.no |
| ☒ | Abraham Mosye | Contracting | | mosye.abraham@banenor.no |
| ☑ | Saleh Alayat | Guest | | |

☒ Not present

In this meeting we discussed different important things around the thesis, and the student had some questions which were clarified during the meeting.

Initially we talked about the preliminary report, mainly the schedule under chapter 5, whether it was a good schedule and if there should be any adjustments or changes to it. Saleh was a guest in this meeting and he suggested to make the plan more clear by using gantt chart which will show tasks that run in parallel more clearly. After that we discussed the relevant frameworks for visualization of the frontend application, Ibrahim suggested a framework called “D3” that is short for “Data-Driven-Documents”, he also suggested to do some more research to see what is out there.

After that we talked about the next phase in the schedule, the system design phase, about UML diagrams, wireframes and so on. At the end Ibrahim had some important information about the final thesis report, so he presented the information and we discussed that.

Until next meeting the main focus is going to be on designing the system, that is creating UML diagrams for the system, activity diagrams, wireframes and other important diagrams and mockups before starting the implementation phase.

NTNU
Department of ICT and
Natural Sciences

# Minutes

Date/time: 19 February 2020 / 12:30 PM
Place: Skype meeting
Arranger: Sarmad Saeed
Regards: Bachelor thesis “Train route analysis”

Participants:

| Present | Name | Title | Phone | Email |
|---|---|---|---|---|
| ☑ | Sarmad Saeed | Student | 45559633 | sarmadsa@stud.ntnu.no |
| ☑ | Ibrahim A. Hameed | Supervisor | | ibib@ntnu.no |
| ☑ | Anniken Susanne T. Karlsen | Supervisor | | anniken.t.karlsen@ntnu.no |
| ☒ | Abraham Mosye | Contracting | | mosye.abraham@banenor.no |

☒ Not present

In this meeting both supervisors where present. As for the client from Bane NOR, Abraham, a meeting with him was arranged earlier, so there was no need for him in this meeting.

In this meeting the progress was shown to the supervisors. The main focus for this phase was system design, which involved planning the system, the frontend, the backend and the reference algorithm. Multiple activity diagrams showing the logic of the system where shown to the supervisors. Complete wireframes/mockups of the system were presented to the supervisors as well. The student also presented the graph library that was found which could satisfy the requirements of the client.

After showing progress, we talked about the bachelor report. The report so far is written in English, but we had a small discussion about what would be the best language to use, should we continue in English or rewrite in Norwegian instead. Anniken suggested the student to talk to the study program manager, Girts Strazdins in case he still cannot decide whether to change the report content to Norwegian or to keep it in English.

After that, the student and Ibrahim discussed a solution to the reference problem (the optimal train route reference speed algorithm). Ibrahim presented a quick solution, which mainly relays on integration and graph areas.

At the end Ibrahim suggested to create a folder in the cloud and put everything related to this project and which the supervisors should have access to there in the cloud folder. This includes the bachelor report, wireframes, UML diagrams, documentation and other documents the supervisors may need.

NTNU
Department of ICT and
Natural Sciences

# Minutes

Date/time: 11 March 2020 / 12:30 PM
Place: Skype meeting
Arranger: Sarmad Saeed
Regards: Bachelor thesis “Train route analysis”

Participants:

| Present | Name | Title | Phone | Email |
|---|---|---|---|---|
| ☑ | Sarmad Saeed | Student | 45559633 | sarmadsa@stud.ntnu.no |
| ☑ | Ibrahim A. Hameed | Supervisor | | ibib@ntnu.no |
| ☑ | Anniken Susanne T. Karlsen | Supervisor | | anniken.t.karlsen@ntnu.no |
| ☒ | Abraham Mosye | Contracting | | mosye.abraham@banenor.no |

☒ Not present/seperate meeting

In this meeting Anikken an Ibrahim were present. As for the client Abraham, a separate meeting was arranged as usual.

In this meeting we talked about different things and the student had different questions about the writing, more specifically the theoretical part. Questions specific to writing were cleared during the meeting by Anikken.

The student also had some Technical questions related to programming which Ibrahim helped to answer.

At the end of the meeting, Anniken suggested that she would take care of the writing part, while Ibrahim should take care of the technical part and things related to programming.

NTNU
Department of ICT and
Natural Sciences

# Minutes

Date/time: 25 March 2020 / 12:30 PM
Place: Skype meeting
Arranger: Sarmad Saeed
Regards: Bachelor thesis “Train route analysis”

Participants:

| Present | Name | Title | Phone | Email |
|---|---|---|---|---|
| ☑ | Sarmad Saeed | Student | 45559633 | sarmadsa@stud.ntnu.no |
| ☑ | Ibrahim A. Hameed | Supervisor | | ibib@ntnu.no |
| ☒ | Anniken Susanne T. Karlsen | Supervisor | | anniken.t.karlsen@ntnu.no |
| ☒ | Abraham Mosye | Contracting | | mosye.abraham@banenor.no |

☒ Not present/own meeting

This meeting was held with Ibrahim only. As for the client Abraham, a separate meeting was held as usual.

In this meeting student showed progress and asked some questions related to the thesis. One challenge was finding a good mathematical comparison method for comparing speeds of different trains.

Ibrahim suggested to do some research on the mean absolute error method (MAE) and the mean squared error (MSE). He underlined that both can be used for comparison purposes, but that each should have own effect and produce different results.

Other questions were also cleared by Ibrahim during this meeting.

NTNU
Department of ICT and
Natural Sciences

# Minutes

Date/time: 8 April 2020 / 12:30 PM
Place: Skype meeting
Arranger: Sarmad Saeed
Regards: Bachelor thesis “Train route analysis”

Participants:

| Present | Name | Title | Phone | Email |
|---|---|---|---|---|
| ☒ | Sarmad Saeed | Student | 45559633 | sarmadsa@stud.ntnu.no |
| ☒ | Ibrahim A. Hameed | Supervisor | | ibib@ntnu.no |
| ☒ | Anniken Susanne T. Karlsen | Supervisor | | anniken.t.karlsen@ntnu.no |
| ☒ | Abraham Mosye | Contracting | | mosye.abraham@banenor.no |

☒ Not present/own meeting

This meeting was planned to be held with Ibrahim only. However, due to overlapping in Ibrahim’s schedule in which he had to participate in sim & vis webinar, this meeting was cancelled.

NTNU
Department of ICT and
Natural Sciences

# Minutes

Date/time: 22 April 2020 / 12:30 PM
Place: Skype meeting
Arranger: Sarmad Saeed
Regards: Bachelor thesis “Train route analysis”

Participants:

| Present | Name | Title | Phone | Email |
|---|---|---|---|---|
| ☑ | Sarmad Saeed | Student | 45559633 | sarmadsa@stud.ntnu.no |
| ☑ | Ibrahim A. Hameed | Supervisor | | ibib@ntnu.no |
| ☑ | Anniken Susanne T. Karlsen | Supervisor | | anniken.t.karlsen@ntnu.no |
| ☒ | Abraham Mosye | Contracting | | mosye.abraham@banenor.no |

☒ Not present/Separate meeting

In this meeting Anniken and Ibrahim were present. As for Abraham, a separate meeting was held.

In this meeting student showed the progress. Most of the functionality was completed including reference finder algorithm in the backend, except for displaying reference speed in the frontend part of the application.

After showing the progress, we discussed the bachelor report, which was sent to both supervisors for feedback earlier.

Anniken gave feedback on the report and highlighted different parts that needs to be improved. She suggested to write in a way that clearly show a red thread in the text and to tie different parts together.

At the end of the meeting, Ibrahim suggested to focus more on the report. Anniken suggested to send it again in about ten days after improvements.

NTNU
Department of ICT and
Natural Sciences

# Minutes

Date/time: 12 May 2020 / 12:00 PM
Place: Skype meeting
Arranger: Sarmad Saeed
Regards: Bachelor thesis “Train route analysis”

Participants:

| Present | Name | Title | Phone | Email |
|---|---|---|---|---|
| ☑ | Sarmad Saeed | Student | 45559633 | sarmadsa@stud.ntnu.no |
| ☑ | Ibrahim A. Hameed | Supervisor | | ibib@ntnu.no |
| ☑ | Anniken Susanne T. Karlsen | Supervisor | | anniken.t.karlsen@ntnu.no |
| ☒ | Abraham Mosye | Contracting | | mosye.abraham@banenor.no |

☒ Not present/Separate meeting

This is not a planned meeting, rather a meeting arranged after a request by Anniken. She wanted to give feedback on the bachelor report. Thus, this meeting was arranged.

In this meeting Ibrahim and Anniken were present. Anniken gave feedback on the state of the bachelor report after reviewing carefully in the last few days. We also discussed other things regarding the report, including attachments, source code and publishing.

Anniken asked the student to discuss publishing with client to clarify everything before submitting the final report.

NTNU
Department of ICT and
Natural Sciences

# Minutes

Date/time: 14 May 2020 / 14:00 PM
Place: Skype meeting
Arranger: Sarmad Saeed
Regards: Bachelor thesis “Train route analysis”

Participants:

| Present | Name | Title | Phone | Email |
|---|---|---|---|---|
| ☑ | Sarmad Saeed | Student | 45559633 | sarmadsa@stud.ntnu.no |
| ☑ | Ibrahim A. Hameed | Supervisor | | ibib@ntnu.no |
| ☒ | Anniken Susanne T. Karlsen | Supervisor | | anniken.t.karlsen@ntnu.no |
| ☒ | Abraham Mosye | Contracting | | mosye.abraham@banenor.no |

☒ Not present/Separate meeting

This was the final formal meeting held during this project. This meeting was held with Ibrahim. A day before, a meeting with client Abraham was held.

In this meeting student showed the report and had final discussion around the thesis. Ibrahim expressed that he is satisfied with the work student have done though the semester and with the written bachelor report.

# Appendix 3
# Status Reports

| IE303612 Bacheloroppgave | Prosjekt: Togrute analyse | Antall møter denne periode: 1 møte(r) | Firma – Oppdragsgiver: Bane NOR | Side 1 av 1 |
|---|---|---|---|---|
| **Rapport fra prosess Framdriftsrapport** | Periode/uke(r): Uke: 3,4 og 5 | Antall timer denne per. (fra logg): 44t | Prosjektgruppe (navn): Sarmad Saeed Abbas | Dato: 04.02.2020 |

Hovedhensikt / fokus for arbeidet i denne perioden

Fokuset denne perioden har vært å jobbe med forprosjektet, å fullføre både rapporten og presentasjonen.

Planlagte aktiviteter i denne perioden

Å jobbe med forprosjektet.

Virkelig gjennomførte aktiviteter i denne perioden

Forprosjektet. Forprosjektrapporten er ferdigskrevet og presentasjonen er gjennomført.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Ingen avvik.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

Ingen.

Erfaring fra denne perioden

Meste parten av erfaringen jeg fikk fra denne perioden gikk ut på planlegging av prosjekt og rapportskriving.

Hovedhensikt/fokus neste periode

Design av systemet er hovedfokuset neste periode. Å planlegge systemets design før implementasjon.

Planlagte aktiviteter neste periode

Å jobbe med systemets design, planlegge designet, lage nødvendige diagrammer som UML, aktivitetsdiagrammer og lignende diagrammer som viser hvordan de forskjellige delene er koblet sammen. I tillegg, lage wireframes/skisser som visualiserer utseende på programvaren etter ferdigutvikling.

Annet

Ingen.

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

Ønsker hjelp med å finne et graframmeverk (flerdimensjonale eller tilsvarende) for visualisering av flere typer data samtidig.

| IE303612 Bacheloroppgave | Prosjekt: Togrute analyse | Antall møter denne periode: 1 møte(r) | Firma – Oppdragsgiver: Bane NOR | Side 1 av 1 |
|---|---|---|---|---|
| **Rapport fra prosess Framdriftsrapport** | Periode/uke(r): Uke: 6 og 7 | Antall timer denne per. (fra logg): 49t | Prosjektgruppe (navn): Sarmad Saeed Abbas | Dato: 18.02.2020 |

Hovedhensikt / fokus for arbeidet i denne perioden

Fokuset denne perioden har vært å jobbe med systemets design.

Planlagte aktiviteter i denne perioden

Å jobbe med systemets design, lage wireframes/skisser og UML diagrammer som beskriver planlagt implementasjon, finne graf bibliotek/rammeverk som tilfredsstiller kravene, i tillegg til å begynne med bachelor rapport skriving. Systemets design handler om å planlegge design på systemet før implementasjon og å lage aktivitets diagrammer, wireframes/skisser og andre nødvendige diagrammer.

Virkelig gjennomførte aktiviteter i denne perioden

Jobbet med design av systemet, laget flere aktivitets diagrammer, laget wireframes/skisser som beskriver systemets utseende, funnet og prøvde flere graf bibliotek/rammeverk, begynte å skrive bachelor rapport i tillegg leste en god del av dokumentasjon av tog data om forskjellige meldinger og pakker som blir sendt til og fra toget.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Ingen avvik.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

Ingen.

Erfaring fra denne perioden

Meste parten av erfaringen jeg fikk fra denne perioden gikk ut på å designe og planlegge et system, lage wireframes/skisser og å jobbe med UML for å planlegge løsninger. I tillegg fikk jeg litt erfaring om skriving av bachelor rapport, hovedsakelig å skrive en god innledning.

Hovedhensikt/fokus neste periode

Implementasjon av systemet er hovedfokuset neste periode.

Planlagte aktiviteter neste periode

Å begynne med implementasjonen av systemet og lage et enkelt prototype. Kommer til å jobbe litt med frontend, litt med backend og litt med implementasjon av referanse algoritme. Skal bruke aktivites diagrammer, wireframes/skisser og andre UML diagrammer som var laget forrige periode for å implementere de forskjellige delene av systemet.

Annet

Ingen.

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

Det er noen problemer med JIRA som er hostet i http://jira.uials.no:8080/secure og confluence, Arne Styve har laget et prosjekt til denne bachelor oppgaven i JIRA, men når jeg trykker på et av bordene i Jira, så vises kun en «loading» side. Confluence får jeg ikke logget inn. (trenger ikke confluence lenger, kun JIRA)

| IE303612 Bacheloroppgave | Prosjekt: Togrute analyse | Antall møter denne periode: 1 møte(r) | Firma – Oppdragsgiver: Bane NOR | Side 1 av 1 |
|---|---|---|---|---|
| **Rapport fra prosess Framdriftsrapport** | Periode/uke(r): Uke: 8, 9 og 10 | Antall timer denne per. (fra logg): 70t | Prosjektgruppe (navn): Sarmad Saeed Abbas | Dato: 10.03.2020 |

Hovedhensikt / fokus for arbeidet i denne perioden

Fokuset denne perioden har vært å begynne med implementasjon av systemet.

Planlagte aktiviteter i denne perioden

Å jobbe med implementasjon av både frontend og backend. Mer spesifikt, utvikle et søke form der brukeren skal kunne søke etter et togtur, legge til grafer for å visualisere data, implementere algoritmer for å kunne søke etter et togtur ved bruk av tog nummer i et bestemt tidsrom, lage noen «queries» for data henting fra databasen. Også bachelorrapport skriving.

Virkelig gjennomførte aktiviteter i denne perioden

Jobbet med implementasjon av både frontend og backend. Laget et søke form. Lagt til graf område i frontend der data og grafer skal vises, implementerte algoritme for å kunne søke etter et togtur ved bruk av tog nummer og tid, skrevet noen database «queries» som skal gjenbrukes under utviklingen av applikasjonen, og skrevet mer i bachelorrapporten.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Valgte å utsette implementasjon av referanse algoritmen. Grunnen til dette er at det var viktigere å implementere funksjonaliteten i applikasjonene nå i starten.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

Ingen.

Erfaring fra denne perioden

Meste parten av erfaringen jeg fikk fra denne perioden gikk ut på å implementere algoritmer for søk, og å optimalisere disse. I tillegg så fikk jeg mer erfaring i skriving av rapport.

Hovedhensikt/fokus neste periode

Å fortsette med implementasjon av systemet er hovedfokuset neste periode.

Planlagte aktiviteter neste periode

Det er planlagt å fortsette med implementasjon av systemet, både frontend og backend. Skal legge til mer funksjonalitet i søkeformet, koble det mot backend applikasjonen, lage graf skjema for data i frontend applikasjonen, behandle hastighetsdata, lage en web API for at frontend skal kunne hente data fra backend, og skrive mer i bachelorraporten.

Annet

Ingen.

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

Ingen.

| IE303612 Bacheloroppgave | Prosjekt: Togrute analyse | Antall møter denne periode: 1 møte(r) | Firma – Oppdragsgiver: Bane NOR | Side 1 av 1 |
|---|---|---|---|---|
| **Rapport fra prosess Framdriftsrapport** | Periode/uke(r): Uke: 11 og 12 | Antall timer denne per. (fra logg): 38t | Prosjektgruppe (navn): Sarmad Saeed Abbas | Dato: 24.03.2020 |

Hovedhensikt / fokus for arbeidet i denne perioden

Fokuset denne perioden har vært å fortsette med implementasjon av backend og frontend applikasjonene.

Planlagte aktiviteter i denne perioden

Det var planlagt å fortsette med implementasjon av systemet, både frontend og backend. legge til mer funksjonalitet i søkeformet, koble det mot backend applikasjonen, lage graf skjema for data i frontend applikasjonen, behandle hastighetsdata, lage en web API for at frontend skal kunne hente data fra backend, og skrive mer i bachelorraporten.

Virkelig gjennomførte aktiviteter i denne perioden

Jobbet med søkeformet, lagt til flere funksjonalitet, brukeren kan nå velge dato og tid ved søk, koblet det mot backend, laget et grafskjema i frontend, jobbet med logikken i backend for behandling av hastighetsdata, laget en API som frontend nå bruker for å hente data. Nå plottes data i grafer ved søk. I tillegg så jobbet jeg mer med bachelorrapporten.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Det har vært litt mindre jobb i uke 11 på grunn av eksamen i emnet: IF300114, Ingeniørfaglig systemteknikk og systemutvikling. Måtte bruke mye tid på å øve for eksamen.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

Ingen.

Erfaring fra denne perioden

Fikk mer erfaring med implementering og god API design.

Hovedhensikt/fokus neste periode

Hovedfokuset neste periode er å jobbe med balise informasjon og tog posisjon, i tillegg fikse noen feil og bugs.

Planlagte aktiviteter neste periode

Å behandle balise og posisjons informasjon, presentere informasjonen i grafene, fikse tidssone problemer ved søk og inkludere alle vogn ved søk med et tognummer. I tillegg til å jobbe med bachelorrapporten.

Annet

Ingen.

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

Ingen.

| IE303612 Bacheloroppgave | Prosjekt: Togrute analyse | Antall møter denne periode: 1 møte(r) | Firma – Oppdragsgiver: Bane NOR | Side 1 av 1 |
|---|---|---|---|---|
| **Rapport fra prosess Framdriftsrapport** | Periode/uke(r): Uke: 13 og 14 | Antall timer denne per. (fra logg): 53t | Prosjektgruppe (navn): Sarmad Saeed Abbas | Dato: 07.04.2020 |

Hovedhensikt / fokus for arbeidet i denne perioden

Fokuset denne perioden har vært å fortsette med implementasjon av frontend og backend applikasjonene, i tillegg til bachelorrapport skriving.

Planlagte aktiviteter i denne perioden

Det var planlagt å implementere logikken for baliser og tog posisjon i backend og presentere informasjonen i frontend, å forbedre søk med tog nummer algoritmen og å fikse problemer med søk, i tillegg til skriving av bachelorrapporten.

Virkelig gjennomførte aktiviteter i denne perioden

Jobbet hovedsakelig med implementasjon av logikken til tog posisjon og baliser, forbedring av søke algoritmen, i tillegg til skriving av bachelorrapport.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Ingen avvik.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

Ingen.

Erfaring fra denne perioden

Meste parten av erfaringen jeg fikk fra denne perioden gikk ut på å debugge og å finne feil i algoritmene som var lagd tidligere, for eksempel søke algoritmen.

Hovedhensikt/fokus neste periode

Å fortsette med implementasjon av backend og frontend, å jobbe med referanse algoritmen, i tillegg til å jobbe mer med bachelorrapporten.

Planlagte aktiviteter neste periode

Implementasjon av tog modus og stasjoner toget har kjørt gjennom både i backend og i frontend (presentasjon). I tillegg så kommer jeg til å jobbe med referanse algoritmen. Bachelor rapporten kommer jeg også til å jobbe med mye mer enn tidligere da det nærmer seg slutten.

Annet

Ingen.

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

Ingen.

| IE303612 Bacheloroppgave | Prosjekt: Togrute analyse | Antall møter denne periode: 1 møte(r) | Firma – Oppdragsgiver: Bane NOR | Side 1 av 1 |
|---|---|---|---|---|
| **Rapport fra prosess Framdriftsrapport** | Periode/uke(r): Uke: 15 og 16 | Antall timer denne per. (fra logg): 71t | Prosjektgruppe (navn): Sarmad Saeed Abbas | Dato: 21.04.2020 |

Hovedhensikt / fokus for arbeidet i denne perioden

Fokuset denne perioden har vært på å fortsette med implementasjon av frontend og backend applikasjonene i tillegg til referanse algoritmen og bachelorrapport skriving.

Planlagte aktiviteter i denne perioden

Det var planlagt å implementere logikken for tog moduser og stasjoner i backend og presentere informasjonen i frontend. Legge til mulighet for data eksport, fikse styling, implementere referanse algoritmen, i tillegg til skriving av bachelorrapporten.

Virkelig gjennomførte aktiviteter i denne perioden

Jobbet hovedsakelig med implementasjon av logikken til tog moduser og stasjoner, data eksport, forbedring av styling, implementasjon av referanse algoritmen, i tillegg til skriving av bachelorrapport.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Ingen avvik.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

Ingen.

Erfaring fra denne perioden

Meste parten av erfaringen jeg fikk fra denne perioden gikk ut på utvikling og rapport skriving.

Hovedhensikt/fokus neste periode

Å fokusere mest på bachelorrapport skriving. I tillegg til å ferdigstille applikasjonene og lage en installasjons guid for testing av systemet.

Planlagte aktiviteter neste periode

Bachelorrapport skriving, gjøre siste endringer i applikasjonene, lage en installasjons guid for testing av systemet.

Annet

Jeg føler at jeg ligger back når det gjelder bachelorrapporten, og det begynner å bli litt for lite tid. Men hovedfokuset kommer til å være rapporten fra neste periode.

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

Ingen.

NTNU
Kunnskap for en bedre verden

| IE303612 Bacheloroppgave | Prosjekt: Togrute analyse | Antall møter denne periode: 1 møte(r) | Firma – Oppdragsgiver: Bane NOR | Side 1 av 1 |
|---|---|---|---|---|
| **Rapport fra prosess Framdriftsrapport** | Periode/uke(r): Uke: 17, 18 og 19 | Antall timer denne per. (fra logg): 115t | Prosjektgruppe (navn): Sarmad Saeed Abbas | Dato: 11.05.2020 |

Hovedhensikt / fokus for arbeidet i denne perioden

Fokuset denne perioden har vært på å fortsette med bachelorrapport skriving, i tillegg til å ferdigstille applikasjonene og lage en installasjons guid for testing av systemet.

Planlagte aktiviteter i denne perioden

Det var planlagt å skrive bachelorrapporten, gjøre siste endringer i applikasjonene og lage en installasjons guid for testing av systemet.

Virkelig gjennomførte aktiviteter i denne perioden

Jobbet hovedsakelig bachelorrapporten. Ferdigstilt applikasjonene og laget en installasjons guid for testing av systemet.

Beskrivelse av/begrunnelse for eventuelle avvik mellom planlagte og virkelige aktiviteter

Ingen avvik.

Beskrivelse av /begrunnelse for endringer som nå ønskes i selve prosjektets innhold eller i den videre framgangsmåten - eller framdriftsplanen

Ingen.

Erfaring fra denne perioden

Meste parten av erfaringen jeg fikk fra denne perioden gikk ut på ferdigstilling av prosjektet og rapport skriving.

Hovedhensikt/fokus neste periode

Å Ferdig stille bachelorrapporten, forberede en presentasjon og en video.

Planlagte aktiviteter neste periode

Å Ferdig stille bachelorrapporten, kvalitet sikre rapporten, forberede en presentasjon og en video.

Annet

Ingen.

Ønske om /behov for veiledning, tema i undervisningen – drøfting ellers

Ingen.

# Appendix 4
# Preliminary Report

TITTEL:

**Sammenligning av togrute med referanse**

KANDIDATNUMMER(E):

**Sarmad Saeed Abbas**

| DATO: | EMNEKODE: | EMNE: | DOKUMENT TILGANG: |
|---|---|---|---|
| **31.01.2020** | **IE303612** | **Bacheloroppgave** | - Åpen |

| STUDIUM: | ANT SIDER/VEDLEGG: | BIBL. NR: |
|---|---|---|
| **BACHELOR I INGENIØRFAG, DATA** | 12/1 | - Ikke i bruk - |

OPPDRAGSGIVER(E)/VEILEDER(E):

Oppdragsgiver: Bane NOR, Abraham Mosye.
Veiledere: Ibrahim A. Hameed og Anniken Susanne T. Karlsen

OPPGAVE/SAMMENDRAG:

Denne oppgaven handler om å utvikle et produkt (programvare) for analyse av tog data.

Oppgaven kan deles i to deler. I den første delen er det ønskelig at hastighetsdata for alle togrutene skal plottes på grafer, deretter skal det lages algoritmer for sammenligning av hastighetsgrafene for å finne den beste hastigheten og bremsekurven for toget blant grafene. Den beste hastighets og bremsekurvegrafen skal lagres som referanse og bruks for sammenligning med en hastighetsgraf for et valgt tog. Algoritmen skal kjøre kontinuerlig og alltid lagre den beste referanse som finnes så langt. Den «beste» hastighetsgraf og bremsekurve er de som er nærmest til den planlagte hastigheten i hver strekning.

I den andre delen skal det utvikles programvare hvor brukeren skal kunne søke etter et tog så vil togets hastighets og bremsekurvegraf plottes og sammenlignes med referansen som var funnet i første delen. I tillegg så skal andre informasjon også vises, som tog modus, posisjon i kilometer, tid, baliser langs veien som toget har passert og annen data som er nødvendig for analyse. Nødvendig funksjonalitet for programvaren skal også utvikles.

*Denne oppgaven er en eksamensbesvarelse utført av student(er) ved NTNU i Ålesund.*

# INNHOLD

# 1 INNLEDNING

## 1.1 Bakgrunn

Jernbanen i Norge blir trådløs i løpet av år 2034, og Norge blir det første landet i verden hvor togtrafikken styres gjennom et datasenter. Et felles trafikkstyringssystem for hele Europa kalt ERTMS (European Rail Traffic Management System) innføres. Dette handler om at alt skilt og optiske signaler/lyssignaler forsvinner og erstatters med databaserte ERTMS hvor et datasenter styrer trafikken ved at det sendes kjøretillatelser, hastighetsbegrensninger og annet data fra et senter kalt RBC via GSM-R til et tog og fra toget til RBC senteret. Dette produserer store mengder data (Big data) på grunn av den kontinuerlige kommunikasjonen mellom toget og RBC. Data som produseres og lagres kan brukes til analyse ved feil eller ved forekomsten av andre uønskede hendelser. I forbindelse med dette, så trenges det verktøy/applikasjoner som gjør det enklere å analysere data.

Allerede nå så har vi i Norge en linje som benytter seg av ERTMS. Østrelinjen som strekker seg fra Ski til Sarpsborg. Det er fortiden den eneste linjen i Norge som styres av ERTMS i dag. Det er denne linjen som er hoved fokuset i denne oppgaven. Men det er alltid mulig å anvende/bruke den samme logikken/applikasjonen/verktøyet for alle andre linjer i hele Norge og mulig linjer i hele Europa.

## 1.2 Om oppdragsgiver

Oppdragsgiveren for denne oppgaven er Bane NOR også kalt/tidligere kalt Jernbaneverket (JBV). Bane NOR er et statlig selskap som er ansvarlig for jernbaneinfrastrukturen i Norge. Bane NOR har som ansvar å planlegge, utvikle og vedlikeholde jernbanenettet, trafikkstyring og administrasjon og utvikling av jernbaneeiendom.

Bane NOR består av mange avdelinger og har kontorer i flere steder i Norge. Avdelingen som er ansvarlig for denne bacheloroppgaven er drift og vedlikehold avdeling.

## 1.3 Den grunnleggende problemstillingen

På grunn av store datamengder, er det ønskelig å analysere data på en enkel måte. Dermed trenges det verktøy/applikasjoner og andre smarte algoritmer som kan forenkle dataanalysen.

Denne oppgaven handler om å utvikle et produkt (programvare) for analyse av tog data. Mer spesifikt er det ønskelig først at hastighetsdata for all togrutene skal plottes på grafer, deretter skal det lages algoritmer for sammenligning av hastighetsgrafene for å finne den beste hastigheten og bremsekurven for toget blant grafene. Den beste hastighets og bremsekurvegrafen skal lagres som referanse og bruks for sammenligning med et valgt hastighetsgraf for et valgt tog. Det vil si at brukeren skal kunne søke etter et tog så vil togets hastighet og bremsekurve plottes i en graf og sammenlignes med referansen som var funnet i starten. I tillegg så skal andre informasjon også vises, som tog modus, posisjon i kilometer, tid, baliser langs veien som toget har passert og annen data som er nødvendig for analysen.

# 2 BEGREPER

- **ERTMS**: European Railway Traffic Management System, felles Europeisk standard for togtrafikk fremføring.
- **RBC**: Radio Block Center er hjernen i signalsystemet ERTMS. Den er ansvarlig for blant annet å kommunisere med tog og å gi kjøretillatelser.
- **GSM-R**: mobilnett som er lukket og er utviklet kun for jernbaner i Europa. Brukes for kommunikasjon mellom RBC og toget.
- **Balise**: en elektronisks enhet som ligger midt i jernbanespor. Når toget kjører over den, sender den informasjon til toget, og forteller toget om posisjonen sin, i tillegg kan en balise ha andre oppgaver.
- **Tog modus**: toget kan være i forskjellige moduser, hver av disse har eget formål.

# 3 PROSJEKTORGANISASJON

## *3.1* Prosjektgruppe

| **Studentnummer(e)** |
|---|
| 492852 |

Tabell: Studentnummer(e) for alle i gruppen som leverer oppgaven for bedømmelse i faget ID 302906

### 3.1.1 Oppgaver for prosjektgruppen

Siden gruppen for denne bachelor oppgaven består av kun en person så må alle rollene gjennomføres av studenten selv. Dette vil si at studenten må ta for seg ansvaret for prosjektledersoppgaver, sekretærs oppgaver og andre oppgaver. Det betyr at studenten er ansvarlig for å planlegge prosjektet, gjennomføre prosjektet, reportere til styringskomiteen, planlegge møter, invitere til møter, sende nødvendig dokumentasjon før møter, skrive møtereferat, skriving av framdriftsrapport og skriving av bachelorrapport i tillegg til implementasjon og utvikling av nødvendige algoritmer og programvare.

## *3.2* Styringsgruppe (veileder og kontaktperson oppdragsgiver)

Styringsgruppen for denne bacheloroppgaven består av Ibrahim A. Hameed, Anniken Susanne T. Karlsen og Abraham Mosye.

Ibrahim A. Hameed er hovedveilederen for denne oppgaven, Anniken Susanne T. Karlsen er biveilederen, mens Abraham Mosye er oppdragsgiveren fra Bane NOR og er senior ingeniør – ERTMS operasjoner.

# 4 AVTALER

## *4.1* Avtale med oppdragsgiver

Det er avtalt med oppdragsgiver at de skal være tilgjengelig for møter og spørsmål angående prosjektet. Alt nødvendig utstyr skal være tilgjengelig, dette gjelder også andre ting som er nødvendig for å utføre prosjektet som data, programvare og lignende. Ingen formelle kontrakter er signert, kun muntlige avtaler.

## *4.2* Arbeidssted og ressurser

Oppdragsgiver tilbyr tilgang kontor i arbeidsstedet med alt utstyr som er nødvendig for å utføre prosjektet. I tillegg til dette er det også mulig å jobbe fra skolen eller andre steder ved bruk av en tildelt bærbar PC. Nødvendig data er tilgjengelig i arbeidsstedet, men det er også mulig å få tak i nødvending data ved å koble seg gjennom VPN.

Kontor er tilgjengelig når som helst til og med lørdager og søndager ved bruk av tildelt adgangskort.

Ved spørsmål eller behov for personer/hjelp så tilbyr også oppdragsgiver muligheten til å stille spørsmål eller få hjelp av personer. Veiledere er også tilgjengelig ved behov, og det er avtalt møter med veileder annen hver uke.

Dersom det oppstår problemer med å få tilgang til data, med tildelt bærbar PC eller andre tekniske problemer så er det mulig å kontakte IT avdelingen, brukerstøtte eller andre avhengig av type hjelp som trenges. Men må være obs på at de ikke er tilgjengelig hele tiden og det kan være at andre også trenger hjelp, dermed kan dette ta litt tid avhengig av situasjonen.

# 5 PROSJEKTBESKRIVELSE

## *5.1* Problemstilling - målsetting - hensikt

**Problemstilling:**
Problemstillingen kan deles i to deler, den ene er å finne et referanse hastighet og bremsekurve som kan brukes for sammenligning med et valgt tog. Dette innebærer å lage en algoritme som skal kjøre daglig og som skal sammenligne hastigheten og bremsekurven til alle togrutene så langt (fra dag x til i dag), og finne den beste hastighets og bremsekurvegraf blant alle togene så langt, deretter lagre det som referanse for sammenligning. Dersom en dag algoritmen finner et bedre bremsekurve og hastighet, så skal da de nye lagres som referanse.

Den andre delen er å utvikle et programvare/analyseverktøy der brukeren skal kunne søke etter et tog, og få plottet hastighets og bremsekurvegrafen til toget, og får samtidig plottet grafene fra referansen for å sammenligne de. Andre informasjon som bør vises er tid, togets modus, posisjon i kilometer (hvor langt toget har kommet, eller hvor toget er) og baliseinformasjon.

**Effektmål:**

- Automatisere sammenligning av data og lagring av referanse.
- Simplifisere analysen av tog data.

**Prosessmål:**

- Samarbeid med oppdragsgiver.
- God dokumentasjon utforming under utviklingsprosessen.
- Bruke det vi har lært hittil for å utføre prosjektet.

**Resultatmål:**

- Utvikle et metode/algoritme for å finne den optimale/beste hastighetsgrafen og bremsekurven for tog i et gitt rute.
- Utvikle verktøy/programvare som forenkler analysen av tog data.

## *5.2* Krav til løsning eller prosjektresultat – spesifikasjon

Hoved fokuset er funksjonelle krav, andre krav som utseende, design og lignende er ikke definert.
Kravene for denne oppgaven er definert som følgende:

- Det skal utvikles en algoritme som sammenligner hastigheten og bremsekurven for alle togene.
- Et referanse skal lagres for sammenligning.
- Algoritmen skal kjøre kontinuerlig (daglig eller etter hver togtur)
- Det skal utvikles et verktøy/programvare for analyse av data, hovedsakelig sammenligning av tog data, programvaren skal ha følgende funksjonalitet:
    - Brukeren skal kunne søke etter et tog
    - Det skal plottes hastighetsgrafen og bremsekurven til søkt tog i tillegg til referansen for sammenligning.
    - Andre informasjon som skal vises i tillegg er
        - Tid
        - Posisjonen i km.
        - Togets modus.
        - Baliseinformasjon
        - Andre ting kan komme i tillegg senere.

Noen justeringer og enderinger kan forekomme etterhvert avhengig av oppdragsgiverens behov.

## *5.3* Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)

For dette prosjektet er det planlagt å bruke et smidig utviklingsmetode, mer spesifikt Scrum. Dette er et veldig populært utviklingsmetode som er testet og fungerer kjempe godt for programvareutvikling. Prosessen er slik:

1. Møt oppdragsgiveren
2. Skriv ned alle kravspesifikasjon i form av fortellinger «stories»
3. Estimer hvor mye arbeid/tid hvert av disse fortellingene utgjør.
4. Tilordne poeng «story points» til hvert av fortellingene.
5. Planlegg lengden på et sprint
6. Velg noen «stories» som er nok å utføre i løpet av et sprint fra produktets «backlog»
7. Start et sprint og begynn med utviklingen og prøv å fullføre mest mulig av planlagte «stories»

8. I slutten av hvert sprint, møt oppdragsgiveren, vis produktet hittil, og skriv ned nye krav, eller oppdater eksisterende krav, deretter start et nytt sprint.

I dette prosjektet så er planen å bruke 14 dagers sprintlenge.

## *5.4* Informasjonsinnsamling – utført og planlagt

Gruppen har skaffet seg dokumentasjon for alt tog data som sendes til toget og fra toget. Denne beskriver alle type meldinger som sendes fra og til toget, alle pakker, parametere, attributter, og annet som er nødvendig for denne oppgaven. I tillegg har gruppen anskaffet seg relevant litteratur om ERTMS og tog kommunikasjon med RBC som er fint å kunne litt om. Informasjon om databaser, eksisterende systemer og programvare er også på plass.

Meste parten av informasjonen, dokumentasjon, litteratur, kan gruppen anskaffe seg fra bedriften dersom det er mer informasjon som trenges.

Det er planlagt å samle annen informasjon som er relatert til utvikling, som rammeverk og lignende. Spesielt med tanke på et graframmeverk. Dette er ikke noe som kan skaffes fra bedriften. Planen er da å undersøke mer via nett søk og lignende. NPM er et populært register for frontend rammeverk og bibliotek, og kan brukes til søk etter rammeverk som mulig kan tilfredsstiller kravene.

## *5.5* Vurdering – analyse av risiko

Faktorer for suksess:

- Å sette realistiske mål som er mulig å oppnå.
- Ha en god plan for gjennomføring av prosjektet, og følge planen godt.
- Å jobbe systematisk.

Faktorer mot suksess:

- Dårlig planlegging (urealistiske forventninger og tilsvarende).
- Mangel på støtte og/eller veiledning fra styringsgruppen.

## *5.6* Hovedaktiviteter i videre arbeid

| Nr | Hovedaktivitet | Kostnad | Tid/omfang | Start dato |
|---|---|---|---|---|
| A1 | Planlegging (forprosjekt) | -- | 3 uker | 13.01.2020 |
| A2 | Systemets design | -- | 2 uker | 03.02.2020 |
| A2.1 | Wireframes | -- | -> | |
| A2.2 | UML, aktivitet diagrammer osv… | -- | -> | |
| A3 | Utvikling/Implementasjon | -- | 11 uker | 17.02.2020 |
| A3.1 | Implementasjon av referanse algoritme | -- | -> | |
| A3.2 | Utvikling av frontend applikasjonen | -- | -> | |
| A3.3 | Utvikling av backend applikasjonen | -- | -> | |
| A4 | Testing | -- | 2 uker | 04.05.2020 |
| A5 | Rapport skriving | -- | 16 uker | 03.02.2020 |

| | | | | |
|---|---|---|---|---|
| A5.1 | Kvalitet sikring av rapport | -- | -> | |
| A5.2 | Dokumentasjon | -- | -> | |

"->" Betyr en del av hovedaktivitetens omfang/tid.
Aktivitet A5 skal kjøres parallelt med alle andre aktiviteter, som systemets design, utvikling og testing.

## *5.7* Framdriftsplan – styring av prosjektet

### 5.7.1 Hovedplan

Vi har fem hoved aktiviteter, hvert av disse aktivitetene deles inn i delaktiviteter. Hoved aktivitetene er planlegging (A1), design av systemet (A2), utvikling (A3), Testing (A4), Rapport skriving (A5).

Startdatoen på en fase, markerer sluttdatoen på forrige fasen. Det vil si at 03.02.2020 avsluttes planleggings fasen (A1) og starter design fasen (A2). Rapport skriving fasen (A5) starter sammen med design fasen (A2) og kjører parallelt med alle andre aktiviteter fra og med A2.

Design fasen er en fase hvor vi tenker gjennom hvordan vi har lyst til å designe systemet, det vil si hvordan ting skal settes sammen, for eksempel at backend skal være koblet til et database, behandler data, og leverer data til frontend applikasjon. Alt dette skal planlegges, og det skal lages wireframes, UML diagrammer, aktivitet diagrammer og lignende som beskriver systemets design. Målet her er å ha en konkret ide og plan om hvordan systemet skal implementeres før utviklingsfasen. Denne skal da presenteres og godkjennes av oppdragsgiver og veileder før utviklingsfasen.

I utviklingsperioden kommer vi til å implementere det som var planlagt under designperioden. For å starte så kommer vi først til å løse den første delen av oppgaven som er å utvikle en algoritme som kjører daglig og som finner en referanse for sammenligning. Når dette er ferdiggjort så skal vi gå over til utviklingen av frontend applikasjonen, deretter går vi over til å utvikle/implementere backend applikasjonen.

Når alt dette er ferdig så har vi et ferdig utviklet programvare som er klar til testing og publisering. Vi kommer til å bruke litt tid på å teste programvaren og bug fikse før levering.

Tesing er også en delaktivitet under hver delaktivitet under utvikling, så under A3.1 trenges det testing, samme gjelder A3.2 og A3.3. Systemet skal testes kontinuerlig. Så Testing kjører parallelt med utvikling, men samtidig må vi ha en testing fase helt tilslutt for å kvalitet sikre produktet og eventuelt rette feil og fikse bugs før levering.

### 5.7.2 Styringshjelpemidler

For å styre dette prosjektet er det flere hjelpemidler som skal brukes. Det viktigste hjelpemiddelet er Jira som er spesielt bygd for Scrum prosjekter. Ved å bruke Jira, vil en kunne definere oppgaver «tasks» og «subtasks», oppgi type oppgave, viktighet og det er også mulig å legge til vedlegg og kommentarer for hvert oppgave. Dette gjør det enkelt å holde oversikt over alle oppgaver som skal utføres.

Et prosjekt i Jira har en «backlog» for produktet med alle oppgaver som skal utføres, og hvor mye tid hver av disse oppgavene trenger (et estimat som man definerer). I tillegg vil man kunne jobbe i sprints ved brukt av Jira. Da har man mulighet til å starte et sprint, definere lengden på hvert sprint, og velge oppgaver som skal utføres i løpet av sprintet. Man har et bord med tre kolonner «å gjøre», «under arbeid» og «fullført» hvor det er mulig å flytte oppgaver fra et kolonne til et annet for å oppdatere status på oppgaven.

Andre ting Jira tilbyr er å generere diagramrapporter som for eksempel «burndown chart» som viser hvor mange av eksisterende oppgaver er fullført, hvor mange nye oppgaver har blitt lagt til og hvor mye som gjenstår ved bruk av «story points». Det er mange andre ting Jira tilbyr som kan være nyttig for å styre et utviklingsprosjekt som dette.

### 5.7.3 Utviklingshjelpemidler

Dette er et utviklingsprosjekt, dermed er det flere hjelpemidler som tas i bruk. Det viktigste er en IDE, i dette tilfellet IntelliJ IDE fra JetBrains. Det er også ønskelig å holde oversikt over de forskjellige versjonene av produktet, dermed tas Git i bruk. Itillegg så tas Gitlab, Github eller begge i bruk for å kunne ha programvaren og versjonene lett tilgjengelig i skyen.

### 5.7.4 Intern kontroll – evaluering

Intern kontroll i prosjektet vil bli gjennomført ved å bruke Jira som hjelper med å holde oversikt over fremgangen, ved enten å se på rapportene som for eksempel «burndown chart» og tilsvarende eller se at flere oppgaver av de som er planlagt å bli utført i sprintet blir faktisk ferdiggjort innen sprintet.

Ved å bruke et smidig utviklingsmetode som Scrum, vil det bli arrangert møter med oppdragsgiver og veiledere ofte, i dette tilfellet kommer vi til å ha et formelt møte i slutten av hvert sprint i tillegg til andre uformelle. Dersom oppdragsgiveren er fornøyd med resultatene som blir representert i slutten av hvert sprint, er det et kjennetegn på at mål er nådd.

## *5.8* Beslutninger – beslutningsprosess

Under arbeidet med forprosjektet har beslutninger om avgrensinger av oppgaven blitt tatt sammen med oppdragsgiver, siden de har mest kunnskap om oppgaven. Informasjon om oppgaven og avgrensinger ble presentert til oppdragsgivere i statusmøtet.

Viktige beslutninger og avgjørelser under arbeid med hovedprosjektet blir tatt sammen med styringsgruppen.

# 6 DOKUMENTASJON

## *6.1* Rapporter og tekniske dokumenter

Dokumentasjon som skal utarbeides er følgende:

- Bachelor rapport
- Møtereferater
- Framdriftsrapport
- Møteinnkallingsdokumenter (avhengig av behovet)
- System dokumentasjon (dokumenter om systemets design, wireframes, UML diagrammer, aktivitetsdiagrammer, kode dokumentasjon og lignende)

Andre dokumenter som skal utarbeides er avviksdokumenter, sammendrag, poster og lignende. Det er også Mulig å skrive brukerdokumentasjon (hvordan bruke applikasjonen og systemet). Dette skrives dersom systemet blir for avansert eller at det er vanskelig å bruke systemet og om det er ønskelig av oppdragsgiveren.

Alle dokumentene sikkerhetskopieres og lagres i skyen, både i Google drive og Microsoft OneDrive. Alle dokumenter er også lagret lokalt

# 7 PLANLAGTE MØTER OG RAPPORTER

## *7.1* Møter

### 7.1.1 Møter med styringsgruppen

Møter med styringsgruppen holdes hver 14. dag. Planen er å kombinere dette møtet sammen med Scrum møtet (altså i et møte) hvor resultatene vises til oppdragsgiveren og veiledere. I dette møtet vil en kunne også få veiledning fra veiledere etter ønske. Vi vil også kunne diskutere hva som har blitt gjort, hva som skal gjøres og om det er noe i veien så vil det bli diskutert sammen med veiledere eller oppdragsgiver.

Invitasjoner via mail har blitt sendt til veileder for alle møte datoene og veiledere har akseptert møtene via mail. Under er det et oversikt over alle datoene for møter:

| Møte nr. | Dato | Kl. |
|---|---|---|
| 1 | 22.01.2020 | 14.00 |
| 2 | 05.02.2020 | 12.30 |
| 3 | 19.02.2020 | 12.30 |
| 4 | 11.03.2020 | 12.30 |
| 5 | 25.03.2020 | 12.30 |
| 6 | 08.04.2020 | 12.30 |
| 7 | 22.04.2020 | 12.30 |
| 8 | 14.05.2020 | 12.00 |

I tabellen over er det oppgitt kun de formelle møtene som er avtalt, andre uformelle og korte møter med oppdragsgiver eller veiledere er ikke inkludert i tabellen.

Etter hvert møte skal det skrives et kort møtereferat som inneholder det viktigste som ble diskutert i møtet. Dette skal sendes til alle veiledere etter møtet på epost.

## *7.2* Periodiske rapporter

### 7.2.1 Framdriftsrapporter (inkl. milepæl)

Planlagte datoer for framdriftsrapport er hver 14. dag. Planen er å levere en rapport før møtedatoen slik at veiledere får gått gjennom den i forhånd. Rapport som skal leveres kommer til å være i form av tekst, der vi tar for oss arbeid utført i forrige periode, planlagt arbeid som skal gjennomføres neste periode, beskrivelse/begrunnelse for avvik mellom planlagte og virkelige aktiviteter, beskrivelse/begrunnelse for endringer, erfaringer og annet som kan være viktig.

# 8 PLANLAGT AVVIKSBEHANDLING

Avhengig av type avvik, grunnen til avvik og alvorlighetsgraden vil passende handling blir utført. Dersom avviket kan løses internt så skal det løses internt, å løse det internt skal prøves først. Om det ikke er mulig å løse det internt vil styringsgruppen bli varsle. Det kan arrangeres et møte dersom avviket er alvorlig. Veiledere vil kunne gi veiledningen og hjelpe med å løse avviket. Studenten er ansvarlig for å varsle om avvik som kan være alvorlig så fort som mulig. Veiledere har også et ansvar, dersom de ser eller finner utfordringer i oppgaven bør de informere om dette.

Dersom prosjektet ikke går som planlagt skal Studenten komme med forslag til løsninger og endringer og føre de i sakslisten. Dette skal da tas opp med styringsgruppen neste status møte. Oppgaven kan omdefineres dersom det er den eneste måten å løse avviket på.

# 9 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING

Det eneste programvaren som vi ikke har tilgang til i dette prosjektet er Jira for å styre Scrum prosjektet. Dette har blitt tatt opp med universitetslektoren Arne Styve som hadde kontroll over Jira som tilhører skolen og som studenter kunne bruke. Andre programvare som en IDE, et versjonskontroll system og lignende er tilgjengelige, og det er ingen spesielle utstyr eller programvare som bør kjøpes så langt.

# 10 REFERANSER

[1] Om ERTMS, https://www.banenor.no/Prosjekter/prosjekter/ertms/

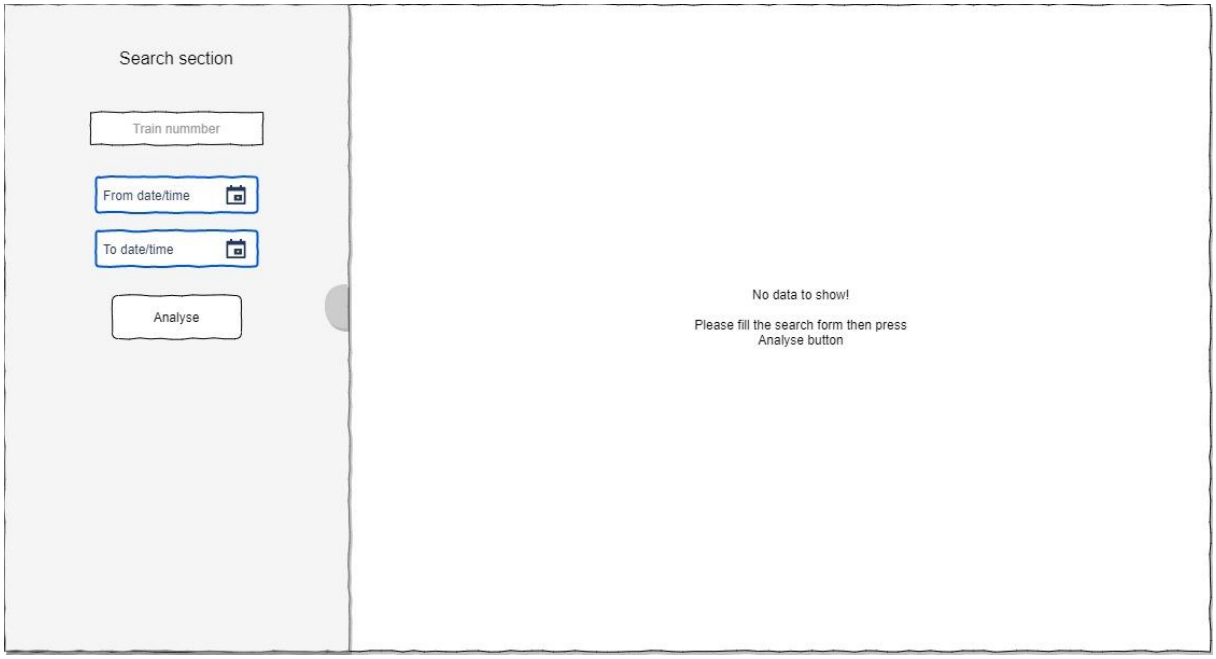
[2] Balise, https://no.wikipedia.org/wiki/Balise

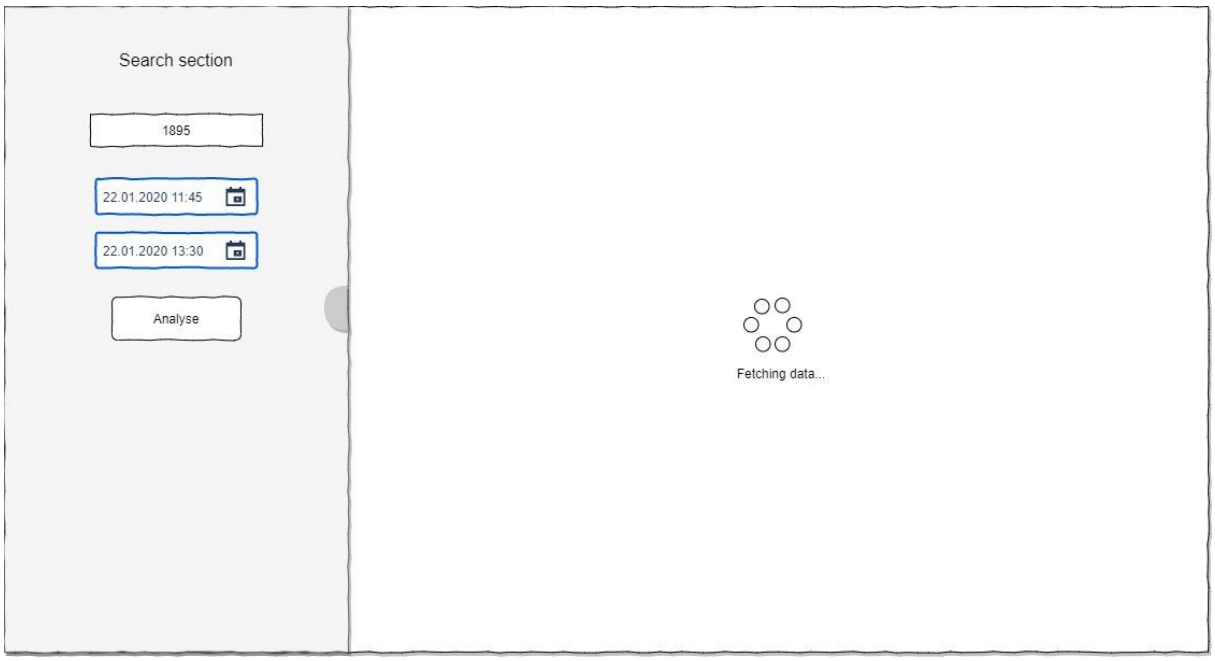
[3] Johansen, Langlo, Rolstadås og Olsson. Praktisk Prosjekt-ledelse. Fagbokforlaget, 2014. isbn: 978- 82-450-1690-1

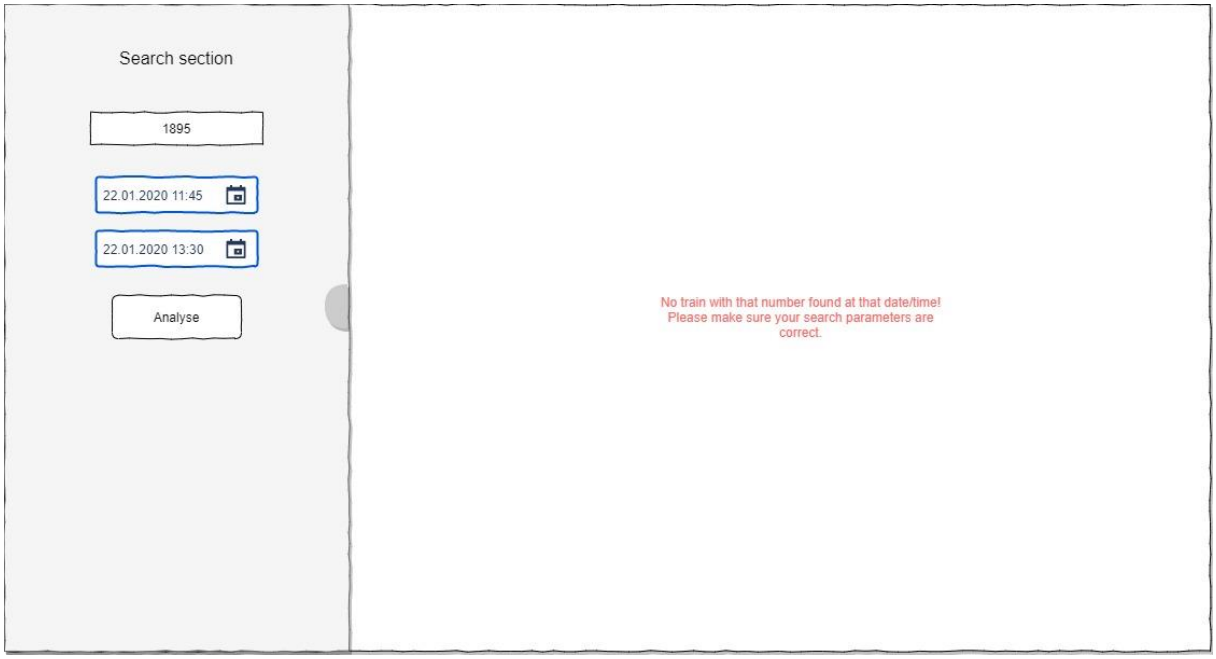
# VEDLEGG

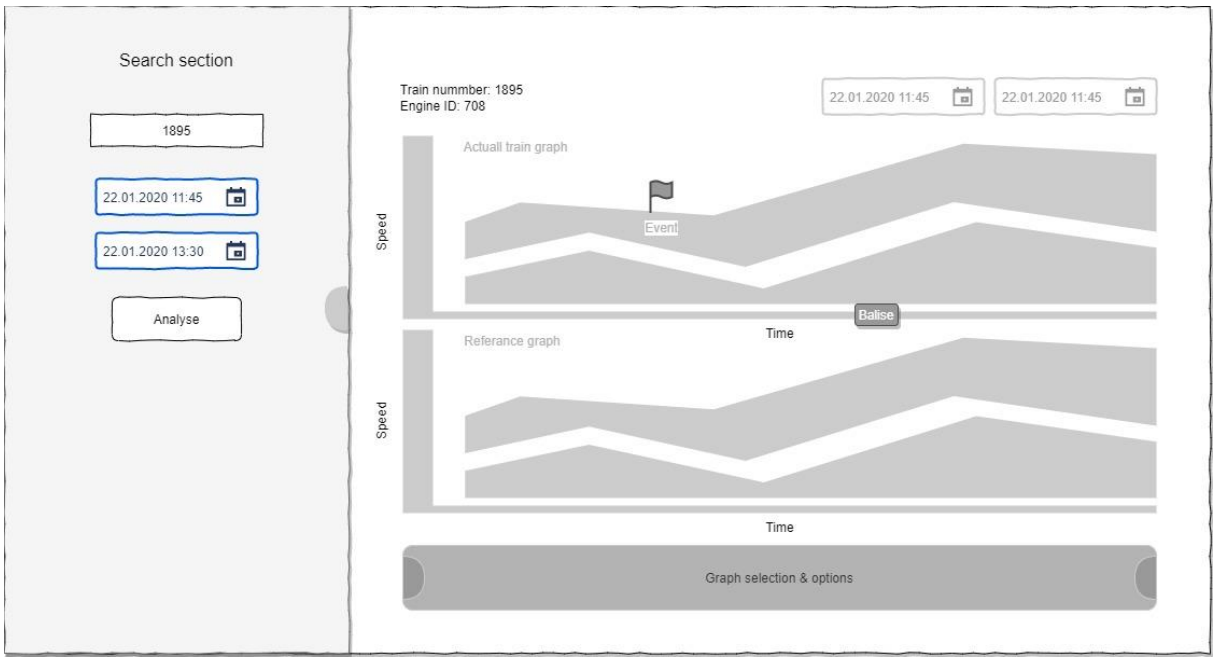
Vedlegg 1 Oppgavetekst fra oppdragsgiver.

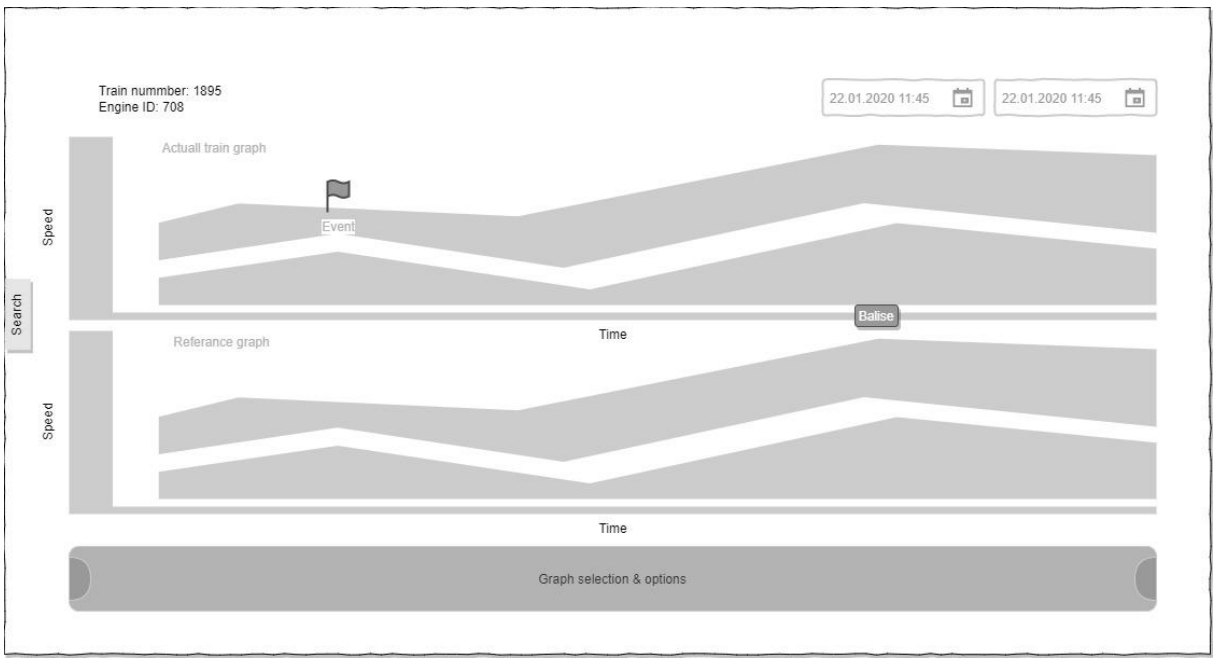
# Appendix 5
# Initial Wireframes

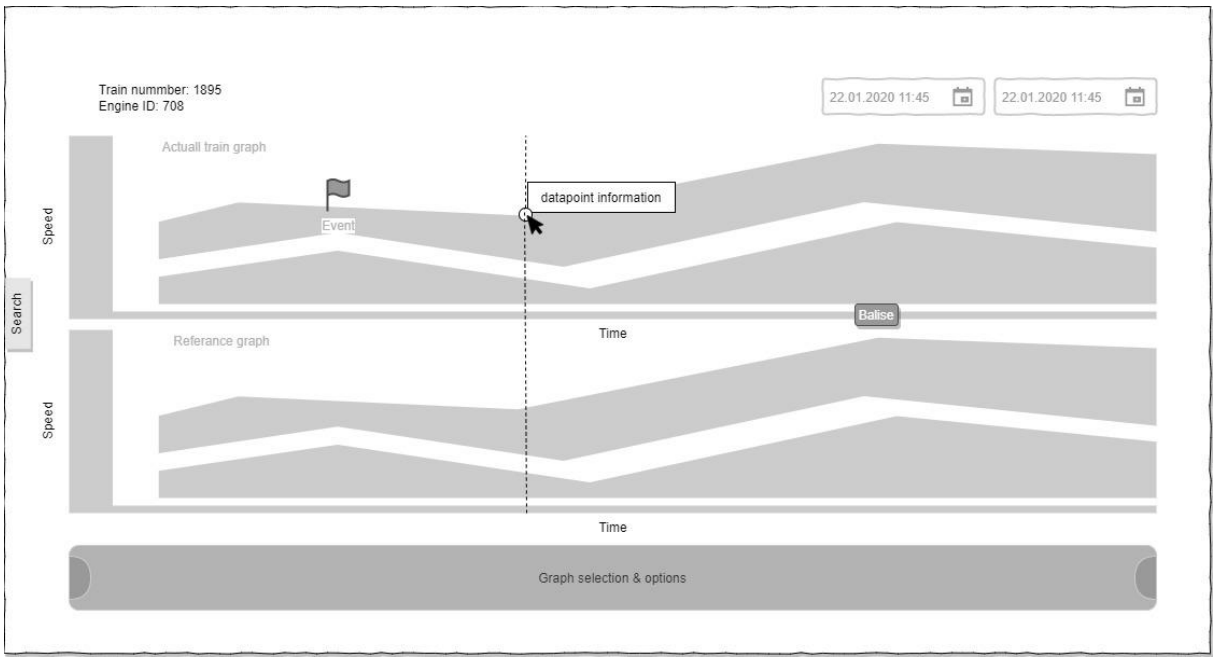

Search section
Train nummber
From date/time
To date/time
Analyse
No data to show!
Please fill the search form then press
Analyse button

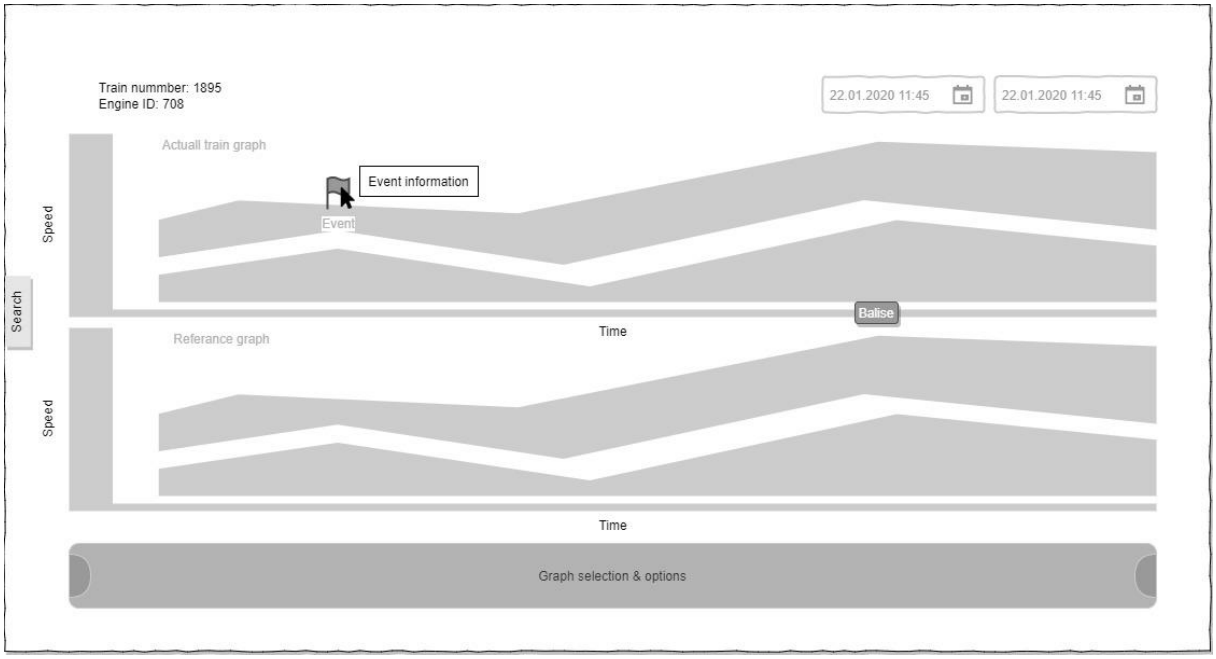

Search section
1895
22.01.2020 11:45
22.01.2020 13:30
Analyse
Fetching data...


Search section
1895
22.01.2020 11:45
22.01.2020 13:30
Analyse
No train with that number found at that date/time!
Please make sure your search parameters are
correct.

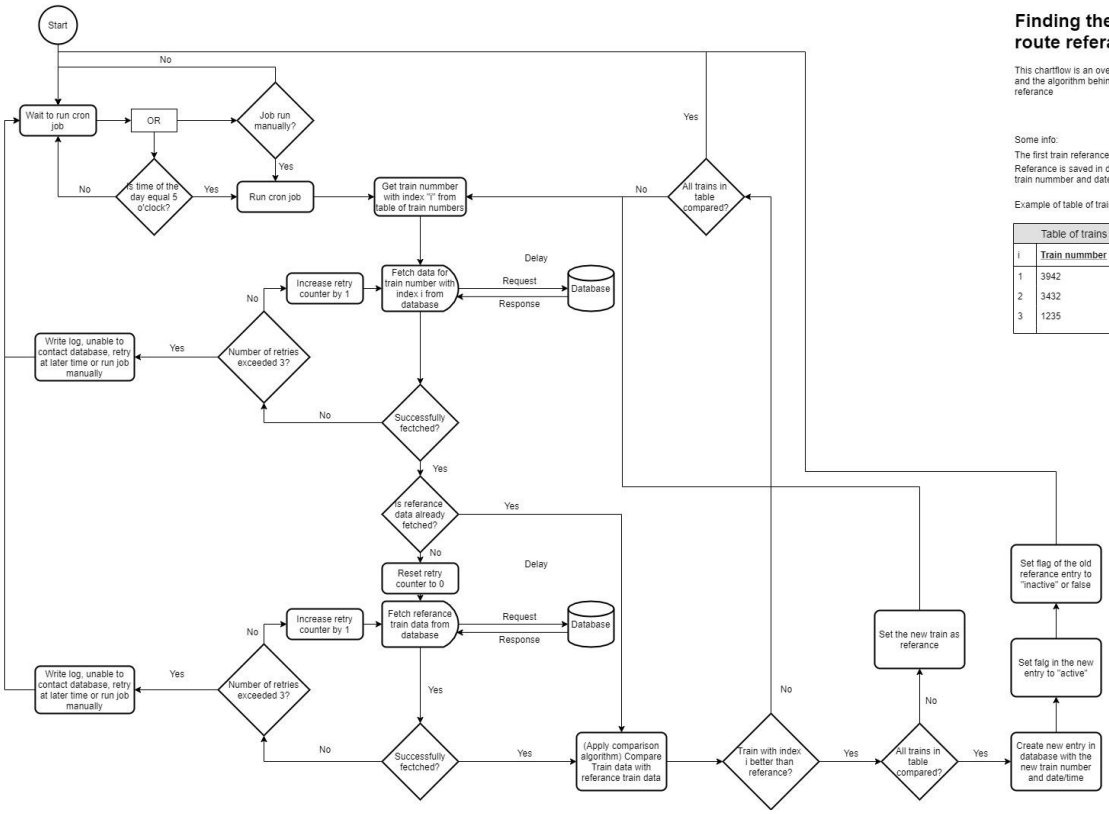

Search section
1895
22.01.2020 11:45
22.01.2020 13:30
Analyse
Train nummber: 1895
Engine ID: 708
22.01.2020 11:45
22.01.2020 11:45
Actuall train graph
Event
Speed
Balise
Time
Referance graph
Speed
Time
Graph selection & options


Train nummber: 1895
Engine ID: 708
22.01.2020 11:45
22.01.2020 11:45
Actuall train graph
Speed
Event
Search
Balise
Time
Referance graph
Speed
Time
Graph selection & options


Train nummber: 1895
Engine ID: 708
22.01.2020 11:45
22.01.2020 11:45
Actuall train graph
Speed
Event
datapoint information
Search
Balise
Time
Referance graph
Speed
Time
Graph selection & options


Train nummber: 1895
Engine ID: 708
22.01.2020 11:45
22.01.2020 11:45
Actuall train graph
Event information
Event
Speed
Search
Balise
Time
Referance graph
Speed
Time
Graph selection & options

# Appendix 6
# Initial Flowcharts




## Finding the best train route referance

This chartflow is an overview over the cron job and the algorithm behind finding the best train referance

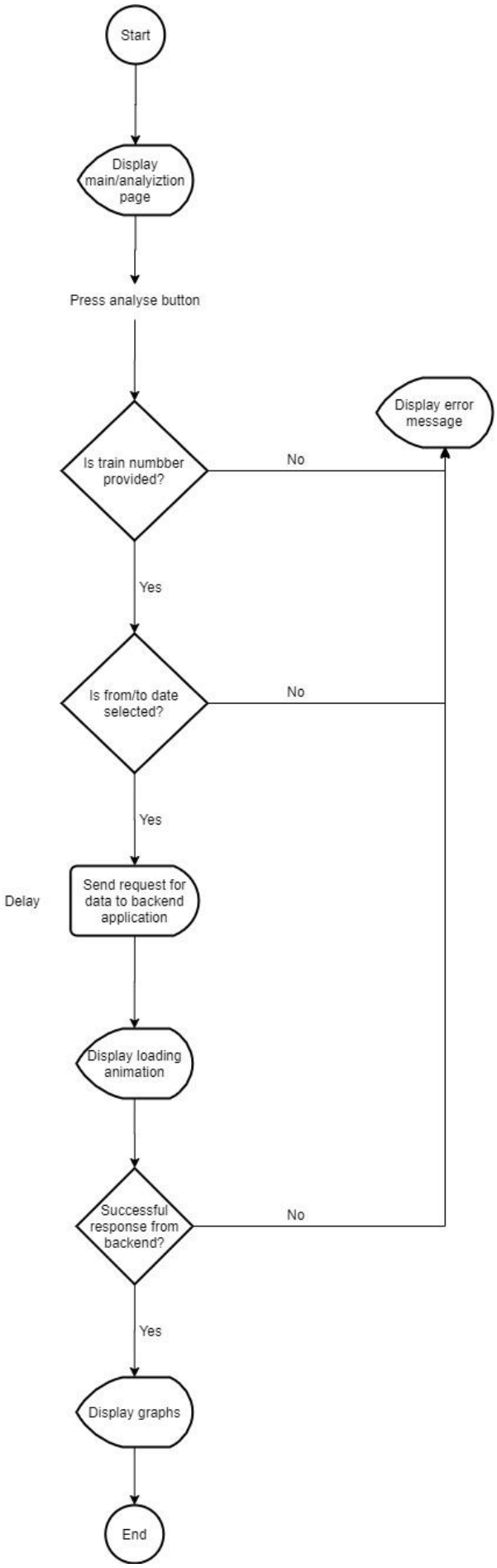
Some info:

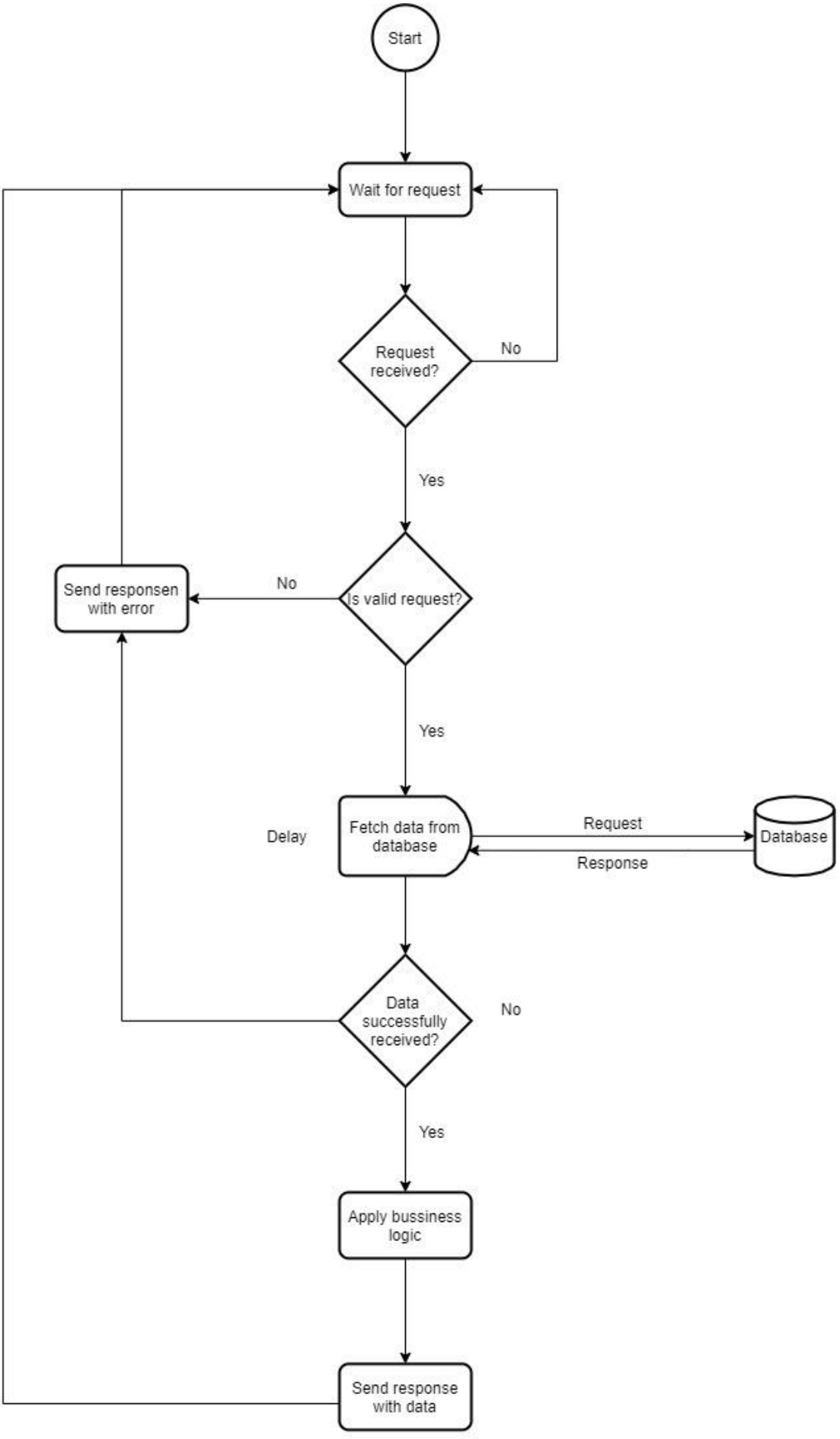
The first train referance is any train route

Referance is saved in database as train nummber and date/time

Example of table of train numbers:

| Table of trains | |
|---|---|
| i | **Train nummber** |
| 1 | 3942 |
| 2 | 3432 |
| 3 | 1235 |


Start
Display main/analyiztion page
Press analyse button
Is train numbber provided?
No
Display error message
Yes
Is from/to date selected?
No
Yes
Delay
Send request for data to backend application
Display loading animation
Successful response from backend?
No
Yes
Display graphs
End


Start
Wait for request
Request received?
No
Yes
Is valid request?
No
Send responsen with error
Yes
Delay
Fetch data from database
Request
Response
Database
Data successfully received?
No
Yes
Apply bussiness logic
Send response with data