

Abdi Bako
Amr Hamcho
Mustafa Abdullah
Shamil Saidovich Khumparov

Assetfront Repair

Bachelor's project in Engineering - Computer Science
Supervisor: Seyed Ali Amirshah
May 2021

Abdi Bako
Amr Hamcho
Mustafa Abdullah
Shamil Saidovich Khumparov

Assetfront Repair

Bachelor's project in Engineering - Computer Science
Supervisor: Seyed Ali Amirshah
May 2021

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

Sammendrag av Bacheloroppgaven

Tittel:	Assetfront Repair
Dato:	20.05.2021
Deltakere:	Abdi Bako Amr Hamcho Mustafa Abdullah Shamil Saidovitch Khumparov
Veileder:	Seyed Ali Amirshahi
Oppdragsgiver:	Headit AS
Kontaktpersoner:	Ronny Kristiansen Dag Blakstad
Nøkkelord:	Angular, API, Assetfront, Headit, Scrum, TypeScript
Antall sider:	48
Antall vedlegg:	12
Tilgjengelighet:	Open

Sammendrag:

Digitalisering er en viktig prosess for behandling, lagring og utveksling av data, ettersom den gjør det mulig å få tilgang til og overføre data effektivt. Som medfører til å forbedre effektiviteten og konsistensen. Assetfront Repair vil være et delsystem for Assetfront, der mekanikere og ingeniører kan sende tekniske rapporter angående kjøretøy. Assetfront Repair vil bidra til å lette arbeidet til de som utfører vedlikehold og inspeksjoner på maskiner som er koblet til Assetfront. Applikasjonen løser to funksjoner. For det første, vil applikasjonen effektivisere prosessen med å registrere servicer og/eller inspeksjoner for et kjøretøy. For det andre digitaliserer applikasjonen prosessen med å registrere en teknisk tilstandsrapport (TTR). Dagens løsninger er både ressurs og tidkrevende. Assetfront ønsket en web-applikasjon med responsiv design for forskjellige enheter. Assetfront Repair har et forbedringsrom som vil bli diskutert senere i rapporten.

Summary of Graduate Project

Title: **Assetfront Repair**

Date: 20.05.2021

Authors: Abdi Bako
Amr Hamcho
Mustafa Abdullah
Shamil Saidovitch Khumparov

Supervisor: Seyed Ali Amirshahi

Employer: Headit AS

Contact Persons: Ronny Kristiansen
Dag Blakstad

Keywords: Angular, API, Assetfront, Headit, Scrum, TypeScript

Pages: 48

Attachments: 12

Availability: Open

Abstract:

Digitization is an important process for processing, storing and exchanging data, as it allows data to be accessed and transmitted efficiently. Which results in improving the efficiency and consistency. Assetfront Repair will be a subsystem of Assetfront, in which mechanics and engineers can submit technical reports regarding vehicles. Assetfront Repair will help to facilitate the work of those who perform maintenance and inspections on machines connected to Assetfront. The application resolves two functions. Firstly, the application streamlines the process of registering services or inspections for vehicles. Secondly, the application digitizes the process of registering a Technical Condition Report (TCR). Today's solutions are both resource and time consuming. Assetfront wanted a web-application with a responsive design to be accessed from different devices. Assetfront Repair has a room for improvement that will be discussed later on in the report.

Preface

This bachelor thesis is written in the spring of 2021 by Mustafa Abdullah, Abdi Bako, Amr Hamcho, Shamil Saidovitsj Khumparov, students at Norwegian University of Science and Technology (NTNU) in Gjøvik, department of Computer Technology and Informatics. We would like to thank Headit AS, and in particular Dag Blakstad, Ronny Kristiansen, and Geir Harald Bjerkemo for an exciting assignment, and for providing support and feedback throughout the project.

We would like to thank our supervisor Seyed Ali Amirshahi who has provided us with valuable guidance throughout the project.

Lastly, we would like to thank the staff of NTNU for assisting us with challenges throughout this project.

Contents

List of Figures	iv
List of Tables	v
1 Introduction	1
1.1 Background	1
1.2 Project Description	1
1.3 Target Audience	2
1.4 Responsibilities and Roles	2
2 Requirement Specification	4
2.1 Functional Requirements	4
2.2 Non-functional Requirements	5
2.2.1 Non-functional Product Requirements	5
2.2.2 Non-functional Organizational Requirements	7
2.3 Use cases	7
2.4 Misuse cases	12
2.5 Risk Analysis	14
3 Software Development Methods	17
3.1 Considered models	17
3.1.1 Extreme Programming (XP)	17
3.1.2 Kanban	17
3.2 Scrum	18
3.2.1 Daily Scrum meeting	18
3.2.2 Meetings with Supervisor	18
3.2.3 Milestones	18
3.3 Development Process	18
3.3.1 Scrum as development method	19

3.3.2	Pair Programming	19
3.3.3	Trello	19
4	Technologies, Design, and Implementation	21
4.1	Technology Assessment	21
4.1.1	Technologies and Tools	21
4.1.2	Choice of Technology	22
4.1.3	Git	22
4.2	Design	23
4.2.1	System Design	23
4.2.2	Object-oriented design	24
4.2.3	Prototyping	25
4.2.4	Components	26
4.2.5	services	27
4.2.6	GUI and UI Design	28
4.2.7	S3 file-system Design	29
4.3	Implementation	30
4.3.1	System Overview	30
4.3.2	Connecting to API	32
4.3.3	service overview	34
4.3.4	Component Overview	37
5	Deployment and Testing	42
5.1	Deployment	42
5.1.1	Amazon Deployment	42
5.1.2	Docker	42
5.2	Testing	43
5.2.1	Static Testing	43
5.2.2	Manual Testing	44
5.2.3	User Testing	44
6	Discussion and Conclusion	45
6.1	Development Method and Process	45
6.1.1	Team Cooperation	45
6.1.2	Meetings with The Company and Product Owner	46
6.2	Development	46

6.2.1 Security	46
6.2.2 Application Design	46
6.2.3 User Test	47
6.3 Future work	47
6.4 Conclusion	48
Bibliography	49
Appendices	A1
A Pre-project plan	A1
B Survey questionnaire	A13
C REST API Documentation	A17
D Technologies and Tools	A21
E Project assignment	A22
F Project agreement	A25
G Today's TCR	A29
H TCR JSON	A47
I TCR PDF	A49
J Assetfront on an Iphone 6	A55
K Implementation of TCR	A58
L Meetings with The Company and Product Owner	A60

List of Figures

1.1	Overall system operation.	2
1.2	Organization of the development team.	3
2.1	Functional Requirements in Assetfront Repair.	4
2.2	Use case diagram for upload functionalities.	8
2.3	Use case diagram for download functionalities.	8
2.4	Severity and likelihood for risk assessment.	15
3.1	Trello board.	20
4.1	Development with branches.	23
4.2	Overall system model.	23
4.3	Simplified system model.	24
4.4	Detailed system model.	24
4.5	Class diagram that represents the important classes and interfaces.	25
4.6	System mockups.	26
4.7	HTML structure.	27
4.8	An example of services and components interaction.	27
4.9	Phone vs PC	28
4.10	Simple and informative design.	29
4.11	S3 Hierarchical file system.	30
4.12	Sequence diagram showing how send TCR works.	31
4.13	Sequence diagram showing how send service or inspection work.	32
4.14	Sequence diagram showing how download TCR work.	32
4.15	Error message due to wrong VIN	38
4.16	Confirmation dialog window	38
4.17	TCR overview.	40
6.1	Overall system model with Keycloak implementation.	48

List of Tables

2.1	Non-functional Performance requirements.	5
2.2	Non-functional Scalability requirements.	5
2.3	Non-functional Usability requirements.	6
2.4	Non-functional Maintainability and Manageability requirements.	6
2.5	Non-functional Portability Requirements.	6
2.6	UX requirements.	7
2.7	Non-functional Deployment requirements.	7
2.8	Detailed use case – Attach file/s.	9
2.9	Detailed use case – Lookup VIN.	10
2.10	Detailed use case – Send Service/Inspection.	11
2.11	Detailed use case – Send TCR.	12
2.12	Stride threat model acronyms [38].	13
2.13	Misuse case - Data forgery.	13
2.14	Misuse case - QR-code manipulation.	13
2.15	Misuse case - Amazon S3 bucket key hijacking.	14
2.16	Misuse case - Files with malicious content.	14
2.17	Product risks.	15
2.18	Business risks.	16
4.1	List of the Services with its dependencies.	35
4.2	List of the components with its description and dependencies	37

Listings

4.1	Media queries example.	28
4.2	Interceptor Service.	33
4.3	Registration of Interceptor as a multi-provider.	33
4.4	getVehicleData method.	33
4.5	Method showing how to convert Observable to Promise.	34
4.6	Method to generate PDF for TCR.	36
4.8	Get last modified folder.	40
5.1	Dockerfile Angular	43
5.2	docker-compose.yml file	43
K.1	TCR Component	A58

Chapter 1

Introduction

1.1 Background

In this project, we aim to develop a new application for Headit AS¹, which is an IT service-provider firm for businesses of different sizes. They wish to develop a product for a company called Assetfront², which works alongside other companies to ease the maintenance and sale of machinery and vehicles. In this project, we aim on behalf of Headit AS to develop a solution that will facilitate the work of those who perform maintenance and inspections on vehicles associated to Assetfront. Also, to streamline this process with the intention that users outside of Assetfront will be able to utilize it as well. The solution used today to perform the desired functions (Service, Inspection, and Technical Condition Report (TCR)) needs to be optimized. Assetfront acquires functionalities to add a Service and an Inspection. However, the current system is outdated and only available for authenticated Assetfront users. Furthermore, Assetfront's current approach to handle TCRs is not digitized. The users have to fill out a Portable Document Format (PDF) file by hand, which will be scanned afterward to the system (see Appendix G). It is desired that the system digitizes this process, where a user can easily complete a TCR and submit it, as well as retrieving a previously submitted TCR.

Thus, the idea of a web application was born, where the users could easily search for a vehicle, and then fill out a desired report where he or she could send in a form (Service, Inspection, and TCR) or view older forms of the aforementioned vehicle.

N.B.: We will hereon refer to Headit AS as “the company”, and Assetfront as “the product owner”.

1.2 Project Description

Assetfront Repair is a web application that aims to be a platform for workshops, service centers, and certified agencies to update their designated Digital Machine Cards (DMC). The goal of the application will be to facilitate the recording of documentation issued by mechanics and engineers at Assetfront, and their partners. Our project revolves around easing the search for a vehicle using its Vehicle Identification Number (VIN), this will allow the user to choose the desired form to submit (Service, Inspection, and TCR). After the submission, the form will be uploaded to a cloud storage, alongside the attached files. Then, the user will be able to download the form as a PDF file right after submission. The form can also be retrieved later on by a user with the specified VIN, if needed. As shown in (Figure 1.1), the user starts by visiting the website using smartphones or computers (1 in Figure 1.1). Then the user can look up a vehicle by providing its VIN. This identifier is then sent to an Application Programming Interface (API), to check whether the vehicle

¹<https://headit.no/>

²<https://assetfront.com/>

exists (represented in part (1) in Figure 1.1). This is done in form of an API request. In case the given VIN does not exist, the API will send back an API response[40] where the front-end of the application will display an error message. If the vehicle is found (represented in part (3) in Figure 1.1), the user will be redirected to a page where he or she can fill out three different types of forms. After the submission of a form, it will be sent to a cloud storage (represented in part (4) in Figure 1.1) in form of a PDF file alongside any attachments. These reports can later on be obtained by users with access to the VIN-number (represented in part (5) in Figure 1.1)(represented in part (6) in Figure 1.1). To summarise, the system will:

- Search for a vehicle by VIN and view vehicles information.
- Send a Service and an Inspection along with attachments.
- Send a TCR.
- Download previous Service, Inspection, and TCR forms.
- Utilize Amazon cloud and deployment services.

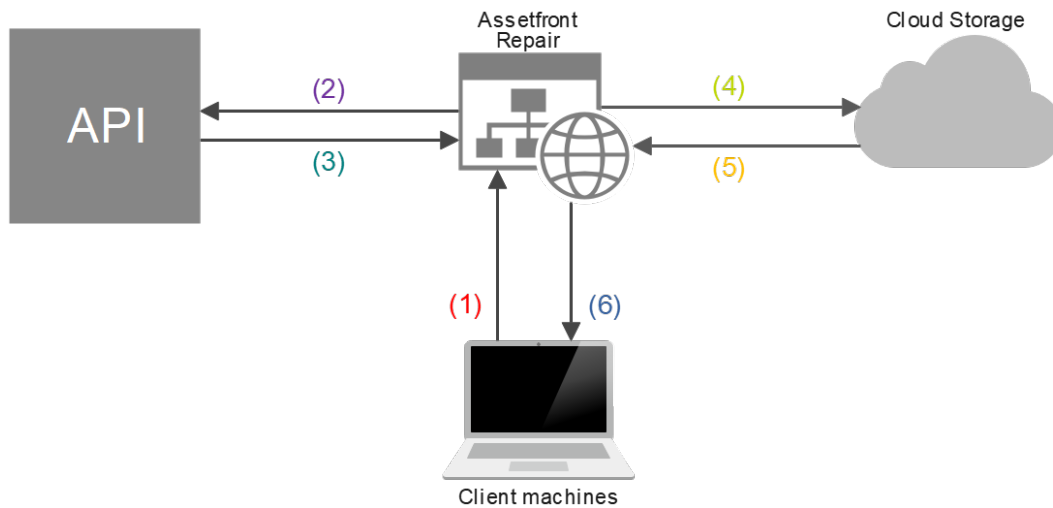


Figure 1.1: Overall system operation.

1.3 Target Audience

Assetfront Repair is a web application that can be used in a company environment to mainly digitize paper-based reports. As well as, save important files in a reliable and scalable online storage. Hence, firms that are seeking to streamline their business process are classified as secondary audience. However, the target users of the application is Assetfront team. Assetfront is an international firm that has users from different Scandinavian countries. In terms of report audience, we expect the reader of this report to have a background/knowledge in software development.

1.4 Responsibilities and Roles

All team members have the same professional background from computer science engineering program. Even though all the members have taken the same subjects, each one has different aspect in software development they like to work on, this formed a good basis for working on such a project. By voting in the group, we agreed to choose Khumparov as our project leader (see Figure 1.2). We

voted in the group on how things should be done and resolved. The choice of Khumparov as group leader was made based on his communication skills. Abdullah was responsible for documenting within the group, of which were from the meetings with the company, product owner or supervisor.

Since the team and the project was relatively small, we chose to only work as developers. Therefore, all team members have been involved in most of the project, but each team member has the main aspect they focused on.

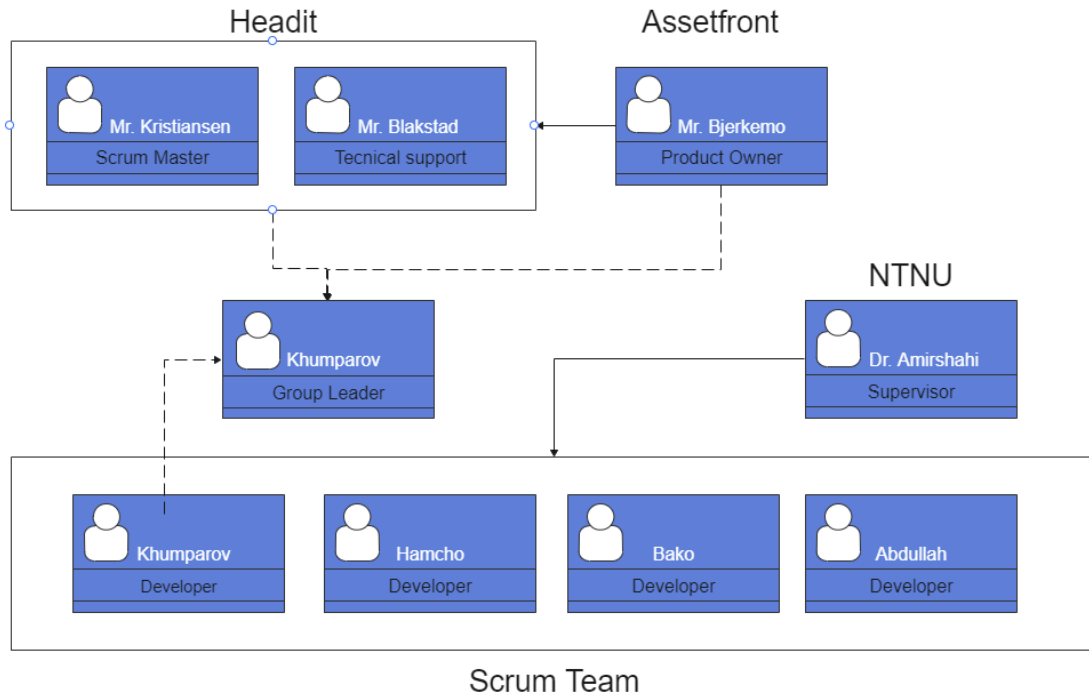


Figure 1.2: Organization of the development team.

Product Owner: Geir Bjerkemo, chairman of the board of Assetfront, used to assist with help when there was uncertainty about the functionality of the application.

Scrum Master: Ronny Kristiansen is an employee at Headit AS and is Scrum master in this project. As a scrum master, he must make sure that the sprints are completed, and that the group had fulfilled certain tasks that need to be done.

Technical support: Dag Blakstad is an employee at Headit AS. He was responsible for technical support. Dag provided technical help when the group needed it. He was also present during status meetings for assistance with technical issues.

Supervisor: Seyed Ali Amirshahi was our supervisor. He provided supervision and feedback throughout the project.

Project leader: Shamil Khumparov was responsible for communication with the company and product owner, arranging meetings.

Development team: The developer team that worked on Assetfront Repair were Abdi Bako, Amr Hamcho, Mustafa Abdullah, and Shamil Khumparov

Chapter 2

Requirement Specification

In this chapter, we will discuss both functional and non-functional requirements. As well as look into the use cases and misuse cases. The last part of this chapter is dedicated to the results of our risk analysis of the project.

2.1 Functional Requirements

The functional requirements in Assetfront Repair describe what the system is capable of doing. Mainly from the user's point of view, in addition to which functionalities would the user expect to perform.

The requirements that our team had decided to take into consideration and base the decisions upon, are provided by the assignment description. The team has managed to state the main requirements of the desired product after multiple meetings with the company and product owner. The main functions the application must be capable of doing are:

	Name	Description
FR. 1	Sending Data	The system must be able to send data in form of files (JSON and PDF) regarding a vehicle's Service, Inspection, and/or TCR.
FR.2	Download	User must be able to download previously stored files from an Amazon Simple Storage Service (Amazon S3) bucket.
FR.3	Scanning	User must be able to scan a QR-code, to ease the use of the application.
FR.4	Uploading	The system must be able to upload files from the application to an Amazon S3 bucket as objects.
FR.5	Retrieving	The system must be able to retrieve stored files from the Amazon S3 bucket as objects.
FR.6	Legal files	The system must be able to recognize legal extensions (PDF, PNG and JPEG).
FR.7	User input	The system must be able to assure that the user's inputs are validated.

Figure 2.1: Functional Requirements in Assetfront Repair.

To achieve the functionalities mentioned above, the user must be able to

- Attached files from the used device.
- Use the device’s camera to capture an image and use it as an attachment.
- Browse through data registered about a vehicle.

2.2 Non-functional Requirements

The non-functional requirements in Assetfront Repair describe how the system works and points out its different aspects. As well as how it performs certain functions. The team has decided to split the non-functional requirements into two categories.

2.2.1 Non-functional Product Requirements

The core principles of the desired product were performance, maintainability and manageability, scalability, portability, and usability. For a better overview, we have divided the Non-functional Product Requirements into tables, see Table 2.1, 2.2, 2.3, 2.4, and 2.5.

In Table 2.2 scalability requirements are listed. These requirements were constructed according to the following approach (Assetfront Repair must be capable of handling a growing number of concurrent users. As well as standing ups and downs of traffic).

Moreover, Table 2.4 contains maintainability and manageability requirements. These requirements were constructed according to the following approach (Assetfront Repair must be capable of being monitored by its administrators easily, especially the file history). This will be done by saving all the related files in an Amazon S3 bucket, which will allow the administrators to manage the file logs easily. Furthermore, the application component structure must be well-organized for further modifications.

Table 2.1: Non-functional Performance requirements.

Performance		
NFR.1.1	Load time	Assetfront Repair should be capable of delivering under two second page load time. Taking into consideration the techniques used in the development process to optimize the load time.
NFR.1.2	Responsiveness	Assetfront Repair should be capable of delivering high responsiveness. This will be done by avoiding waiting and indicating that the request is received before its done processing. Along with an informative user interface to keep the user informed.

Table 2.2: Non-functional Scalability requirements.

Scalability		
NFR.2.1	Project Structure	Assetfront Repair must be developed using the standard structure of the Angular application.
NFR.2.2	CSS Encapsulation and Styling	Assetfront Repair must use CSS encapsulation[2] by styling each component and create a reusable code.
NFR.2.3	Reactive programming RXJS	Assetfront Repair must use RxJS (Reactive Extensions for JavaScript)[46] library. In order to handle async tasks properly, watch for changes in our application and run code accordingly. This is done through the observer pattern ¹ .

Table 2.3: Non-functional Usability requirements.

Usability		
NFR.3.1	Efficiency	Assetfront Repair must be efficient for the frequent user.
NFR.3.2	Ease of remembering	Assetfront Repair must be easy to remember for the frequent user.
NFR.3.3	Ease of learning	Assetfront Repair must be for both novices and users with experience from similar systems.
NFR.3.4	User testing	Assetfront Repair must be tested by users whom have not seen the application beforehand. The test users should be employees at the company that the product is developed for.

Table 2.4: Non-functional Maintainability and Manageability requirements.

Maintainability and Manageability		
NFR.4.1	App components	Assetfront Repair must be developed through building up an organized component structure, and separating presentational from container components[8]. The component structure must be taken into consideration throughout the development phase since reusable components are essential for maintainable applications.

Table 2.5: Non-functional Portability Requirements.

Portability		
NFR.5.1	Browser requirements	Assetfront Repair must be portable. Meaning that moving from one browser to another browser does not create any problem.

To assure a successful project performance, the company and our team have selected suitable front-end development tools. As our task is to create a front-end application, we would ensure a good user experience that includes efficiency and speed. This would be done by using proper tools and methodologies for front-end development. The technologies used in the project are listed in Section 4 and Appendix D.

The front-end requirements were unclear at the beginning of the project. The team had to design prototypes and present them to the product owner and the company (see Section 4.2.3). After that, the prototypes would be adjusted according to the product owner’s wishes. This process would run through the development part as well. The product owner attended some of the scrum meetings (as described in Section 3.2) and added requirements and feedback. In the development phase, the front-end requirements were decided to be:

- The code must be reusable for future modifications.
- Optimize the application’s speed and performance.
- The development must be according to User Experience (UX) requirements (see Table 2.6).

We have created a list of requirements to fulfil and achieve the best possible results in regards to UX. The requirements in the following list are inspired by the Doctoral thesis “Laws of Interface Design”[23], and the book “Sketching user experience”[18].

Table 2.6: UX requirements.

UX requirements		
NFR.6.1	Responsiveness	The interface must be responsive on different screen sizes.
NFR.6.2	Familiarity	The design must be user-friendly in terms of that the user familiarizes with the design quickly.
NFR.6.3	Colors (UX)	The combination of colors to use in the application should be Palette[39]. For that, we have used an online tool to help us to choose the list of colors ² .
NFR.6.4	Style	The components must be styled using Angular Material ³ , which is a tool for material design in Angular.

2.2.2 Non-functional Organizational Requirements

In this Section, we will take a look at the deployment requirements. This will ensure transferring the application to the company. Further, the company will maintain the code and integrate it with their existing web application (see Table 2.7).

Table 2.7: Non-functional Deployment requirements.

Deployment		
NFR.7.1	Container deployment	Assetfront Repair must be deployed using Docker container.
NFR.7.2	Cloud storage	Assetfront Repair must use Amazon S3 bucket for saving all files and data provided by the user.

2.3 Use cases

To show the core functionalities of the system, and present its usage from a perspective of different user types, a use case diagram was created seen from the user perspective. The diagram in Figure 2.2 shows the use cases for the part of the system where the user sends out either a Service, Inspection, or a TCR. These functionalities involve uploading files to the cloud storage the company is using. Furthermore, the diagram in Figure 2.3 points out the part of the application where the user downloads previously sent files. To analyze the use cases mentioned in Figure 2.2 further, detailed use case tables were created. The main actor of the system is the user. Another user of the application is the admin who is mainly responsible for monitoring the system and managing the cloud storage in terms of configuring and access rights.

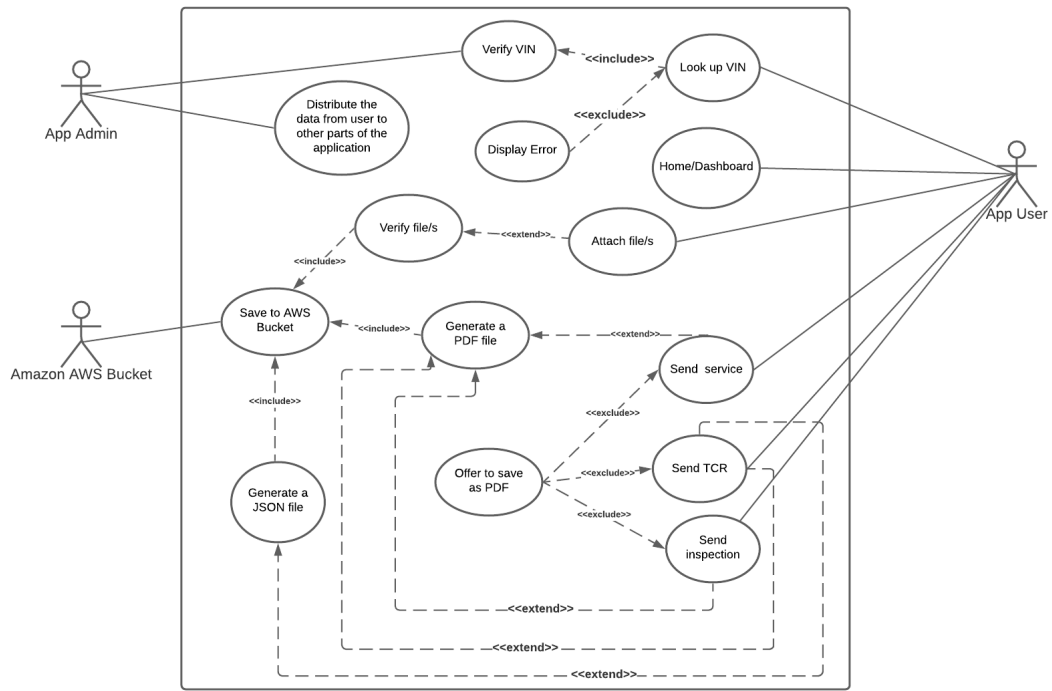


Figure 2.2: Use case diagram for upload functionalities.

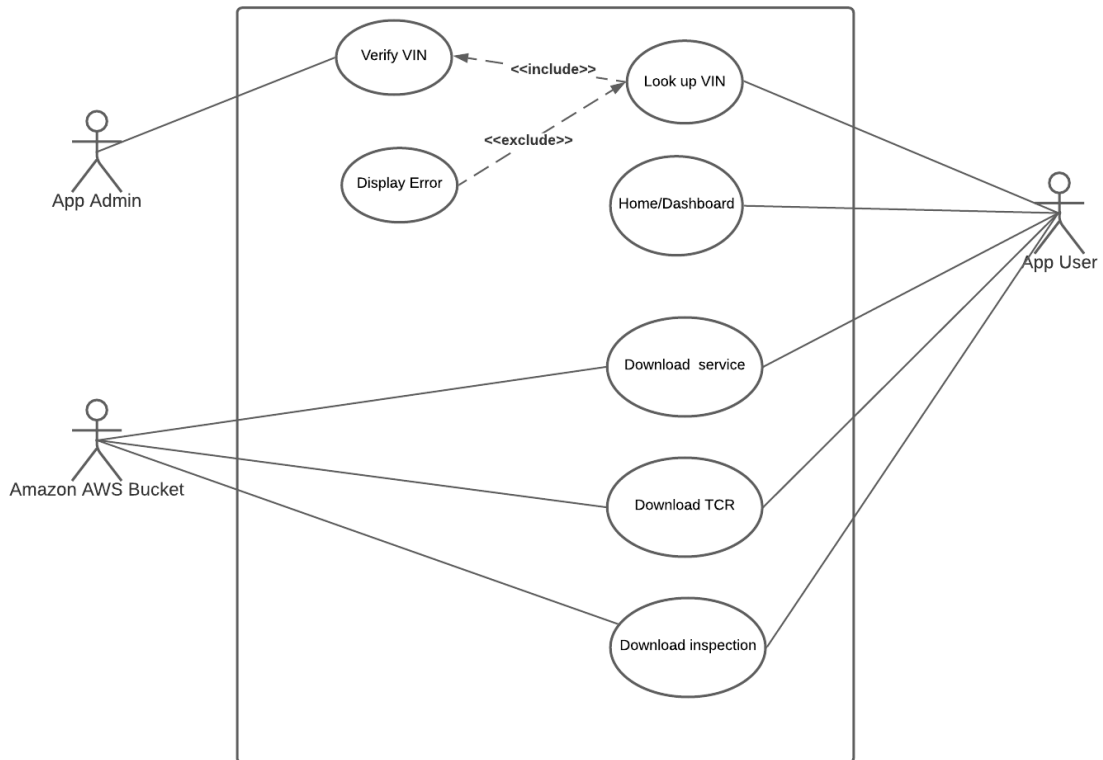


Figure 2.3: Use case diagram for download functionalities.

Table 2.8: Detailed use case – Attach file/s.

Name:	Attach file/s
Actor:	Application users
Description:	The user gets the option of attaching a file or more to Service or Inspection (see “Send Service/Inspection” use case in Figure 2.2)
Flow of Events:	
Basic Flow:	<p>This use case begins when a user clicks on attach a file:</p> <ul style="list-style-type: none"> • The system offers the user to either browse the device and choose a file, or take a picture using the device’s camera. • The system filters out the files that the user can attach. Only files with extensions PDF, JPEG, and PNG appear to the user. • The name of the file/s chosen would then appear to the user on the browser. • The system uploads the chosen file/s to the Amazon S3 bucket when the entire form is completed. • Use Case ends.
Alternative Flows:	<ol style="list-style-type: none"> 1. Attaching an illegal format. The system displays an informative error message. 2. Unless the user attaches a legal file format, the file/s will not be uploaded to Amazon S3 bucket.
Post Conditions:	<ol style="list-style-type: none"> 1. Success: The file/s are added to the system and saved in Amazon S3 bucket as a PDF and JSON file. 2. Failure: The file/s are not added to the system.

Table 2.9: Detailed use case – Lookup VIN.

Name:	Lookup VIN
Actor:	Application user
Goal:	Searching for a registered vehicle
Description:	The user gets access to the system by looking up a vehicle by using VIN to get access to the different functionalities of the application (Service, Inspection, TCR).
Flow of Events:	
Basic Flow:	<p>This use case starts when the user is not verified to the system and goes to the start page:</p> <ul style="list-style-type: none"> • The system prompts the user for inserting a valid VIN. • The user enters a VIN. • The system validates the inserted data by traversing through the API provided by the company and makes sure it is valid. • The user is validated and redirected to the home page of the vehicle. • Use Case ends.
Alternative Flows:	<p>In case the user inserted an invalid VIN, the following occurs:</p> <ol style="list-style-type: none"> 1. The system displays an informative error message. 2. The system prompts the user to re-enter valid information. 3. The Basic Flow continues where the user inserts new information.
Post Conditions:	<ol style="list-style-type: none"> 1. Success: Success the user is validated, and home page will be displayed. 2. The user is unable to access the rest of the applications functionalities.

Table 2.10: Detailed use case – Send Service/Inspection.

Name:	Send Service/Inspection
Actor:	Application user
Goal:	Sending a form regarding the vehicle’s Service/Inspection
Description:	The user gets access to the desired form by choosing either Service or Inspection from the home page.
Flow of Events:	
Basic Flow:	<p>This use case begins when a user accesses the “send a service/ Send an Inspection” feature/s of the system:</p> <ul style="list-style-type: none"> • The system asks the user to insert the necessary data (service/inspection date, hours, cost, inspection state, Service comment, and user’s details). • The system gives the user the option to attach a file(PDF, PNG and JPEG). • The system checks that the data inserted is of a correct type. For instance, the hours used in service must be of number type. • A pop-up is displayed to show a summary of data inserted, along with the option to download the data as a PDF file. • The system uploads the PDF file along with the files attached to Amazon S3 bucket. • Use Case ends.
Alternative Flows:	<ol style="list-style-type: none"> 1. Invalid information entered: The system displays an “Invalid data error” and asks the user to enter valid information 2. Cancel button: In case the user clicks on the cancel button, the system will redirect him or her to the home page.
Post Conditions:	<ol style="list-style-type: none"> 1. Success: The Service/Inspection is added to the system and saved in Amazon S3 bucket along with the files attached. 2. Failure: The Service/Inspection is not added to the system.

Table 2.11: Detailed use case – Send TCR.

Name:	Send TCR
Actor:	Application user
Description:	The user gets access to the desired form by choosing either clicking on the TCR link in the sub-header or clicking on the button on the home page.
Flow of Events:	
Basic Flow:	<p>This use case begins when a user accesses the TCR feature of the system:</p> <ul style="list-style-type: none"> • The system asks the user to fill up a multiple answer form, made of a JSON file received from the company API. • The system checks whether all questions are answered since it is mandatory. • The system generates and sends a PDF and JSON file to the Amazon S3 bucket on the user hitting the submit button. • A pop-up is displayed to show a summary of the data input of the user, alongside giving the user the option to download a PDF file. • Use Case ends.
Alternative Flows:	<ol style="list-style-type: none"> 1. Not answering any of the multiple-choice question form: The system disables the “Continue” button and does not let the user proceed. 2. Cancel button: In case the user clicks on the cancel button, the system will redirect him or her to the home page.
Post Conditions:	<ol style="list-style-type: none"> 1. Success: The TCR is added to the system and saved in Amazon S3 bucket along with the JSON file. 2. The TCR is not added to the system.

2.4 Misuse cases

This section points out some potential misuse cases of the application with the STRIDE threat model. STRIDE is a threat identification and classification model[48]. Acronyms of STRIDE is explained below in Table 2.12. Tables 2.13, 2.14, 2.15, 2.14, and 2.16 show the misuse cases that we identified in the application. These misuse cases in the Tables (2.13 - 2.16) falls at least into one of the categories of STRIDE. For example, Table 2.13 can fall under a few categories such as Spoofing, Repudiation and Tampering. Table 2.12 is an example of a misuse case, which falls under the category of Spoofing, Repudiation and Tampering where any user with access to VIN, can claim to be another user and input a report, or claim not to have submitted a report after submission. All the Tables (2.13 - 2.16) have a row where the letters of STRIDE category are used.

Table 2.12: Stride threat model acronyms [38].

Threat	Threat property
S-Spoofing	Masquerading as another user
T-Tampering	Unauthorized manipulation of data
R-Repudiation	Denial of performing an action
I-Information disclosure	Exposure of data to unauthorized parties
D-Denial of Service (DoS)	Denying legitimate users by overloading resources of the application
E-Elevation of Privilege (EoP)	Unprivileged users gains unauthorized privileges

Table 2.13: Misuse case - Data forgery.

Name:	Data forgery
STRIDE categories:	STR
Actor:	Attacker/User
Goal:	Submission of incorrect data
Description:	The user might be tempted to submit incorrect data out of mistake. This however can be abused by an attacker since the VIN number can be obtained by other means, such as Assetfront Retrade ⁴ (Subsystem of Assetfront). Which will result in someone with malicious intent to overload the storage capacity.

Table 2.14: Misuse case - QR-code manipulation.

Name:	QR-code manipulation
STRIDE categories:	STRD
Actor:	Attacker
Goal:	To manipulate the data contained in the QR-code in the PDF file that will be saved to the client's machine and Amazon S3 bucket
Description:	This can lead to QR-codes with malicious code hidden inside its contents, getting saved and executed on user's devices.

Table 2.15: Misuse case - Amazon S3 bucket key hijacking.

Name:	Amazon S3 bucket key hijacking
STRIDE categories:	STD
Actor:	Attacker
Goal:	To get access keys of the Amazon S3 bucket
Description:	Access keys to the Amazon S3 bucket can get stolen from the front end of the application since, which can be used by attacker or attackers to overload the saving capacity of the Amazon S3 bucket, which can result in higher data storage cost.

Table 2.16: Misuse case - Files with malicious content.

Name:	Files with malicious content
STRIDE categories:	TI
Actor:	Attacker
Goal:	Infecting client machines with malicious files uploaded to the Amazon S3 bucket
Description:	Files that contain malicious content can be uploaded and hidden inside legitimate files to be later on downloaded by other users.

2.5 Risk Analysis

The team had analyzed the different types of risks that might occur, this was done in the project plan (see Appendix A). However, we decided to look at a wider range of risk scenarios and study the likelihood and severity of each of them. The risk analysis was inspired by an essay called “Methods of Risk Analysis and Management” [24]. Also, an article called ‘Project Risks vs. Business Risks’ [52], which sheds the light on various risk scenarios. The categories studied are listed below:

- **Product risk analysis :** In Table 2.17, the team had identified high, medium, and low risks that might affect both the quality of the product and the ability to meet the requirements in Section 2. The analysis is inspired by a paper called “Risk management in the product development process” [1].
- **Business Risk analysis):** This category involves the organizational risks that might occur. The risks in Table 2.18 will mainly affect the business environment, which in our case is the development team.

Figure 2.4 shows the color-coding of the risk assessment, taking into consideration likelihood and severity factors.

Severity				
Likelihood		Low (L)	Medium(M)	High(H)
	Low(L)		R2.0	R1.4 & R1.8
	Medium(M)	R1.5	R1.7	R1.1 & R1.2 R1.3 & R1.6
	High(H)	R1.9		

Figure 2.4: Severity and likelihood for risk assessment.

Product risks		
ID	Description	Mitigation
R1.1	The team is unable to finish all the requirements in the product	Since most of the team members are not familiar with the technologies used in this project, the likelihood that the team spends a long time on certain tasks is relatively high. Therefore, the team members had to familiarize with most of the tools before the development phase starts.
R1.2	The requirements change drastically in the development phase	The requirements change might occur due to changes in the product owner's requirements. However, there must be an agreement between the team and the company in case this scenario occurs. As a severe change in the requirements might have high impact.
R1.3	The final product does not match the requirements	This risk might occur in case of limited interaction with the product owner, which will result in that our product and the product envisioned are significantly different. In order to avoid this risk scenario, the team will schedule frequent meetings with the product owner to show the progress.
R1.4	The final design is not responsive on different screen sizes	The application will be used from various devices(Desktop, laptop, Mobil-phones, and tablets). Therefore, it is important to maintain a responsive design to ensure a smooth user interaction with the application. This risk is likely to happen due to a lack of knowledge in web application design among the team members. Therefore, the team had decided to use the "Mobile first" approach[37].
R1.5	The libraries to be used in the project are incompatible or deprecated.	This risk can be avoided by researching libraries needed for development. As some libraries might be deprecated, the team has to look for alternative libraries.

Table 2.17: Product risks.

Business risks		
ID	Description	Mitigation
R1.6	The company and product owner add unachievable requirements	Taking into consideration that the requirements were unclear in the beginning. Therefore, the team had agreed with the company that new requirements are optional to fulfill.
R1.7	Communication risks among team members	The team must ensure effective communication. Communicating any delays or shifts in deadline to deliver sprints on time.
R1.8	Loss of interest in the project by a team member	Team members will ensure that all developers are interested in the project. In case of lack in contribution, the issue will be discussed with the supervisor.
R1.9	The absence of one or many team members due to illness or personal issues	The team members will be informed of the development progress. The group had agreed on daily meetings, which will provide the members with an overview of the workflow. In that case, the impact of an urgent absence of a team member will be limited.
R2.0	Lack of motivation among one or more team members which can have negative effect on his performance	Team members will collaborate to learn technologies and tools used in the project. Ensuring that nobody falls behind.

Table 2.18: Business risks.

Chapter 3

Software Development Methods

The purpose of this chapter is to address the various aspects of the development process in this project, which will last from January to May. The choice of the development method would majorly affect the work in terms of how problems were approached, prioritized, and eventually solved. The development method Scrum was recommended to be used in this project by the company. However, we realized that pure Scrum development method was not the optimal approach, due to the lack of clarity in the project's definition. This was caused by the fact that the requirements were slightly flexible, and the development team was engaged in defining some of the requirements. As a result of that, we chose to look into a combination of different development models to find a more suitable fit for our project, that could eventually be discussed with the company. The team conducted an assessment on various development models and their benefits.

3.1 Considered models

Based on the chapter description, the team researched a variation of development models to decide the most suitable model for the project. This final decision must be further discussed with the company.

3.1.1 Extreme Programming (XP)

XP is an agile development model with multiple values. First, XP encourages code refactoring. Also, it has a short development cycle, which results in continuous feedback and interaction with the product owner. Moreover, XP considers the customer to be an integral part of the team. Furthermore, XP relies on two main techniques, pair programming, and test driven development [13].

Since the team consists of four developers, pair programming would be an effective technique to be implemented. The code will be reviewed twice through the development, this increases code quality and allows knowledge sharing among team members.

3.1.2 Kanban

Kanban is a Lean development method[43] for improving the development process. This method has been studied due to its benefits in visualizing and managing the workflow through a tool called "virtual Kanban board". Tasks could be added to the board and classified into categories (like Work-In-Progress (WIP), to do, done, etc..). The study of Kanban method was inspired by an article called "Usage of Kanban methodology at software development teams" [30]. Applying Kanban in our project could be effective for managing the workflow due to its ability to

- Split the work into pieces, write each item on a card, and put it on the wall.
- Use named columns to illustrate where each item is in the workflow.

3.2 Scrum

Scrum is an agile framework for incremental product development, which is widely used in software development. The core of Scrum is iterations, which are repeated until all the product requirements have been met. The product owner defined the functional requirements which would be added into an artifact called “Product Backlog”. The workload would be divided into iterations called “sprints”. The duration of a sprint depended on the complexity of the iteration. In our case, the length of the iteration would be between 1 and 4 weeks according to the company’s recommendations.

Scrum is the development method we agreed upon in the end. It is a good fit for the projects size and content. Furthermore, some concepts from other development methods could be present. For instance, pair programming technique can be introduced at a certain point in development.

3.2.1 Daily Scrum meeting

The team planned to run daily scrum meetings. The meetings content were regarding the status of each member in terms of what we were working on the previous day, how far we had come with the tasks, and what we were planning to work on. These daily scrum meetings helped us to keep the team members up to date on the current state of the product. Keeping group members up to date would mitigate the probability of some risks arising. For instance, if a group member has to be absent for a certain amount of time, as mentioned in risk R1.9 in Table 2.18.

3.2.2 Meetings with Supervisor

Throughout the project, we had weekly meetings with our supervisor. The supervisor meeting were mainly about the work progress, and the state of the development. Similarly to the scrum meetings, the project progress was presented to the supervisor, and his feedback was noted and added to the upcoming sprint.

3.2.3 Milestones

During the project planning phase, we set up some milestones for the development of the project (see Appendix A). In order for the group to deliver submissions in time, these milestones must have been met.

- **15th of February:** The group must be finished with the design (mock-ups and project architecture).
- **15th of April:** The group must be finished with the first version of the code.
- **1st of May:** The group must be finished with the testing.
- **10th of May:** Code delivery.

3.3 Development Process

In this Section, we aim to give an overview of how the development process affected the project.

3.3.1 Scrum as development method

After sprints delivery, the team presented the sprints to the company and the product owner. Then, the features of the next sprint were planned based on the feedback received. If there were any new wishes or changes in existing requirements, we assessed how high on the priority list these were set. There were difficulties in getting these meetings to be exactly the same day of the week or planned period due to external factors, such as one or more members having a meeting on other subjects, or the company not having the opportunity. This was tackled in a good way, in that we used to arrange a meeting one or two days ahead so that all parties had the opportunity to participate in these meetings.

3.3.2 Pair Programming

As mentioned in Section 3.1, pair programming is a way to improve the quality of the code. It is a process that engages two developers in writing a piece of code. Pair Programming was an effective way to share knowledge among team members. The team decided to introduce this element in our project and combine it with Scrum.

3.3.3 Trello

Trello¹ was used to keep control of work tasks. “Trello is a collaboration tool that organizes a project into boards. In one glance, Trello states what is being worked on, who is working on what, and where something is in a process” [53]. In our board, we included the four steps system. In Figure 3.1 it is at the front the four different steps that were implemented in our organization board.

- **To do:** In this board column, things are set up to be done after the meetings with the company and product owner, or when the team agrees that something else could be added here.
- **Doing:** This column contains the ongoing tasks.
- **Finish:** When a task was completed, and it was approved by others in the group, it was put on this board column.
- **Put on hold:** This column includes the tasks that had to be continued later, due to lack of resources.

¹<https://trello.com/>

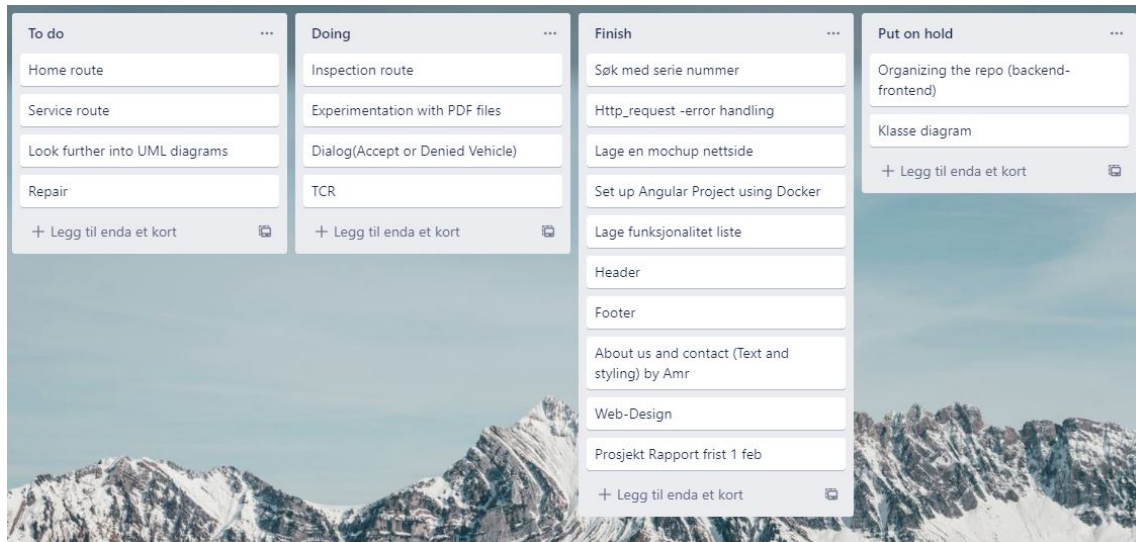


Figure 3.1: Trello board.

Chapter 4

Technologies, Design, and Implementation

This chapter will discuss multiple aspects of the development phase. Firstly, we will conduct a technology assessment and list the various technologies used in the project. Next, we will discuss the project's prototyping and the process used to achieve it. In addition, we will explicate the design of the various subsystems along with implementation.

4.1 Technology Assessment

4.1.1 Technologies and Tools

Before starting the development phase of the project, technologies and tools were examined. The examination helped to find suitable and available tools to be used throughout the development. Next, we discuss some of the specific technologies within our application. A list of all technologies and tools is shown in Appendix D.

Angular

Angular¹ is a platform and framework for building single-page client applications using HTML and TypeScript². Angular is written in TypeScript[7].

API

API is an interface that allows the interaction between two application or mixed hardware-software intermediaries[31] and[26].

REST

Representational State Transfer (REST) is a software architectural style for distributed hyper-media systems, that is used for creating web-services[54].

Swagger

Swagger is an Interface Description Language (IDL)[56] which allows the user to represent the structure of an API in order for it to be readable by a machine[51].

Amazon S3

According to Amazon documentation, Amazon S3 is a storage for the Internet, that has a simple

¹<http://angular.io/>

²<https://www.typescriptlang.org/>

web services interface that you can use to store and retrieve any amount of data[15].

Docker

According to Docker documentation, Docker is an open platform which helps developers to develop and run applications. Docker enables the user to separate the desired application the infrastructure which expedites software delivery[19].

4.1.2 Choice of Technology

The technologies used in the project were mainly compared to the project specifications. Before starting with the development, Angular was recommended to be used as a front-end development framework, because today's Assetfront solution is using the same framework and this results in a smooth transition of Assetfront Repair. Next, Spring Boot³ was recommended as a back-end framework. However, the requirements have changed as we started with the development phase. It was decided that only a front-end application would be sufficient for this project. The technology assessment was done with a tendency towards a front-end framework. In general, developing web applications is simpler with using a suitable framework as they tend to have various built-in functions to be used in the development process.

The reason behind choosing Angular is that it helps with creating interactive and dynamic applications that have single page features, Single Page Applications (SPAs). Moreover, Angular has a proper community support. They also host Angular conferences with the presence of international IT- companies for new developments in technology[4].

Amazon S3 was selected as a cloud storage resource for project files. It provides an environment to send requests to create buckets, store and retrieve data as objects. It is a suitable platform for our project as the service is highly durable and available. Amazon S3 was used for the application deployment as well. Amazon offers a free service to deploy static website files. Finally, we used HTML, CSS, and TypeScript. These technologies were required to accomplish the development of the application. As well as displaying the data and interacting with the users.

4.1.3 Git

Git is a control version system used for parallel programming between multiple developers. Git helped keeping a backup of code that was created and supported sorting through code developers have had developed at the same time. This was to minimize the amount of merge conflicts, even though we used Trello (see Section 3.3.3) to avoid any collisions with other team members, it was inevitable to happen. To further mitigate merge conflicts and/or functionality conflicts, we adapted a feature development workflow called "feature branch workflow" [12].

The aim of this development method is to create branches for each time a new functionality is to be developed. Thereafter, this branch would be deleted after merging with main development branch. The technical terminology of this merging process is called a "pull request". The pull request has to be approved by other developers before merging. Developers whom rejected a pull request had to do so with a message including the reason behind the rejection. In this project, we decided to create a "DEV" branch, where we merged to after completing the development of a function. This was to simulate different teams working on the same application 4.1. Thus, we did not merge with "Master branch" before introducing the final version of the code. As a result of that, the company could simply clone the code and continue the development process.

³<https://spring.io/projects/spring-boot>

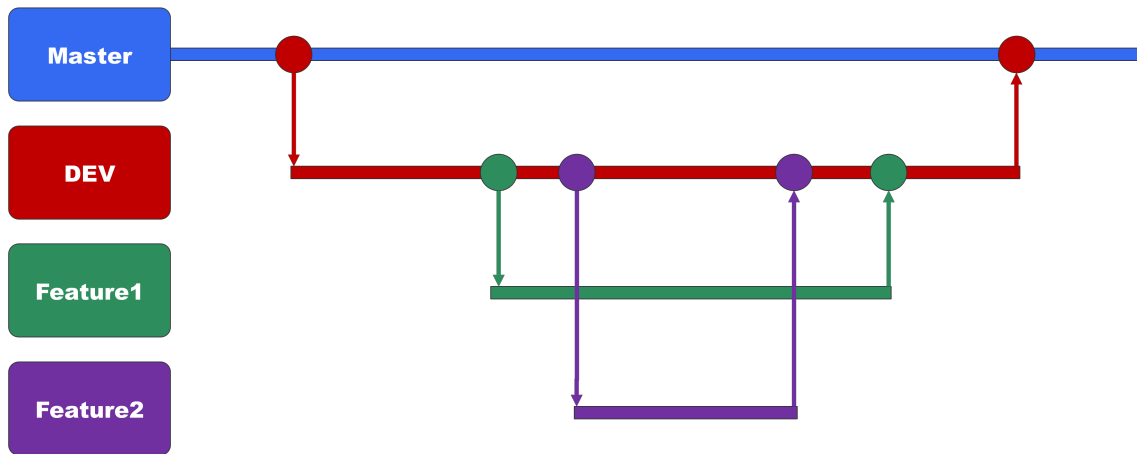


Figure 4.1: Development with branches.

4.2 Design

4.2.1 System Design

The team created a model for the system according to the project requirement. First, the generic approach to illustrate the entire system that our application would be apart of, is shown in Figure 4.2.

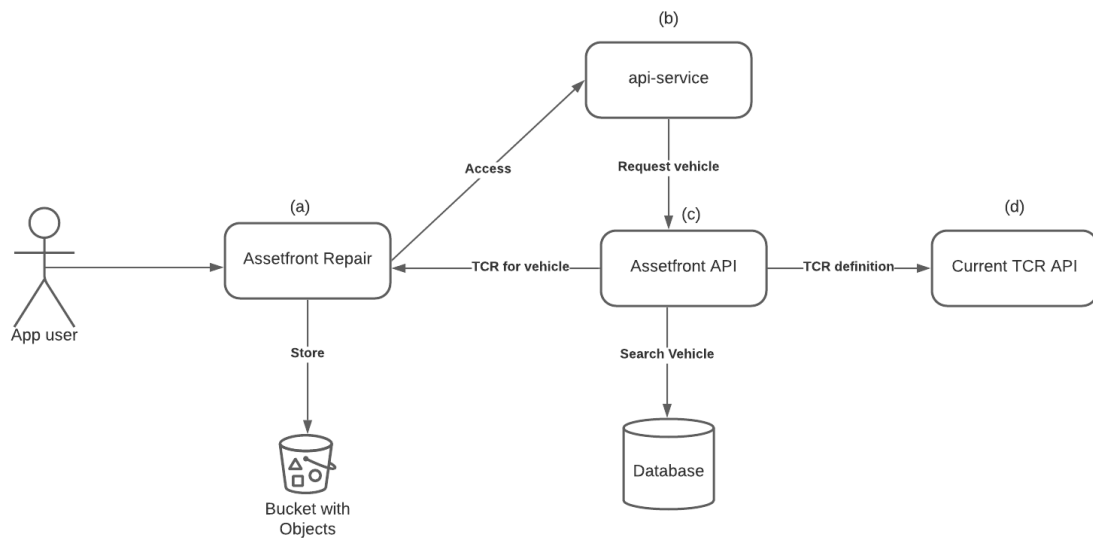


Figure 4.2: Overall system model.

Then, the main focus was set towards the interaction between Assetfront Repair, Assetfront API, api-service and Amazon S3 bucket. Since the company provided us with a ready-made API (API documentation is found in Appendix C). The system was designed to handle API- requests, and provide the user with an interactive Graphical User Interface (GUI), which would result in communicating with the storage cloud Amazon S3 bucket. This led to a simplified model of the system from an external perspective (see Figure 4.3). Furthermore, the system components in Assetfront Repair were analyzed to include the file upload restrictions, invalid API requests, and the interaction with Amazon S3 bucket (see Figure 4.4).

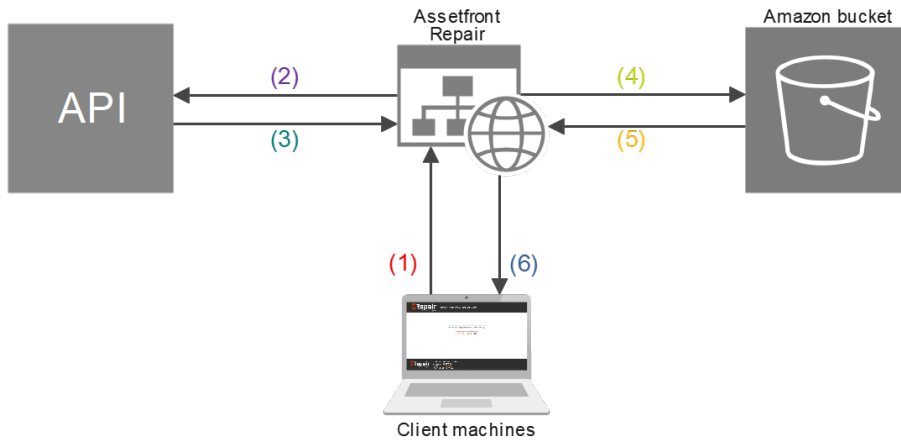


Figure 4.3: Simplified system model.

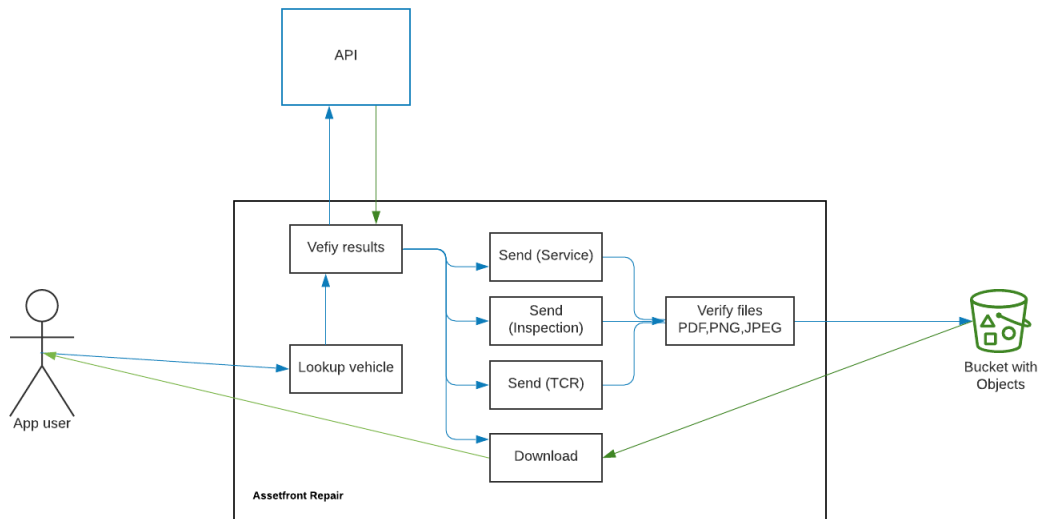


Figure 4.4: Detailed system model.

The natural approach was to display some data from the API in case of valid API-requests for the user to make sure that it is the right vehicle. Then the user gets access to the system where he or she could proceed to use the different functionalities. Furthermore, in case the user wishes to send a Service and/or an Inspection, and add attachments to it. The system will generate a PDF file with the user's input. Also, it uploads the PDF and its attachments to an Amazon S3 bucket. On the other hand, in case the user wishes to fill up a TCR. Then, the system would generate a PDF file along with a JSON file that Assetfront can request data from in form of API request. The JSON file would be used in further development to display the user's input in other Assetfront subsystems. Both files will be uploaded to the same Amazon S3 bucket. The file-system in the Amazon S3 bucket is structured according to the way the data will be retrieved later, it is explained further in Amazon S3 file-system design Section 4.2.7.

4.2.2 Object-oriented design

An important aspect of any design is the relationship between objects and interfaces in the system. Object-oriented design is mainly concerned of specifying the details of classes and objects. However,

the interface details should be avoided in such a diagram according to "Software Engineering" book by Ian Sommerville[50]. Figure 4.5 represents the important classes and interfaces in the system in term if development.

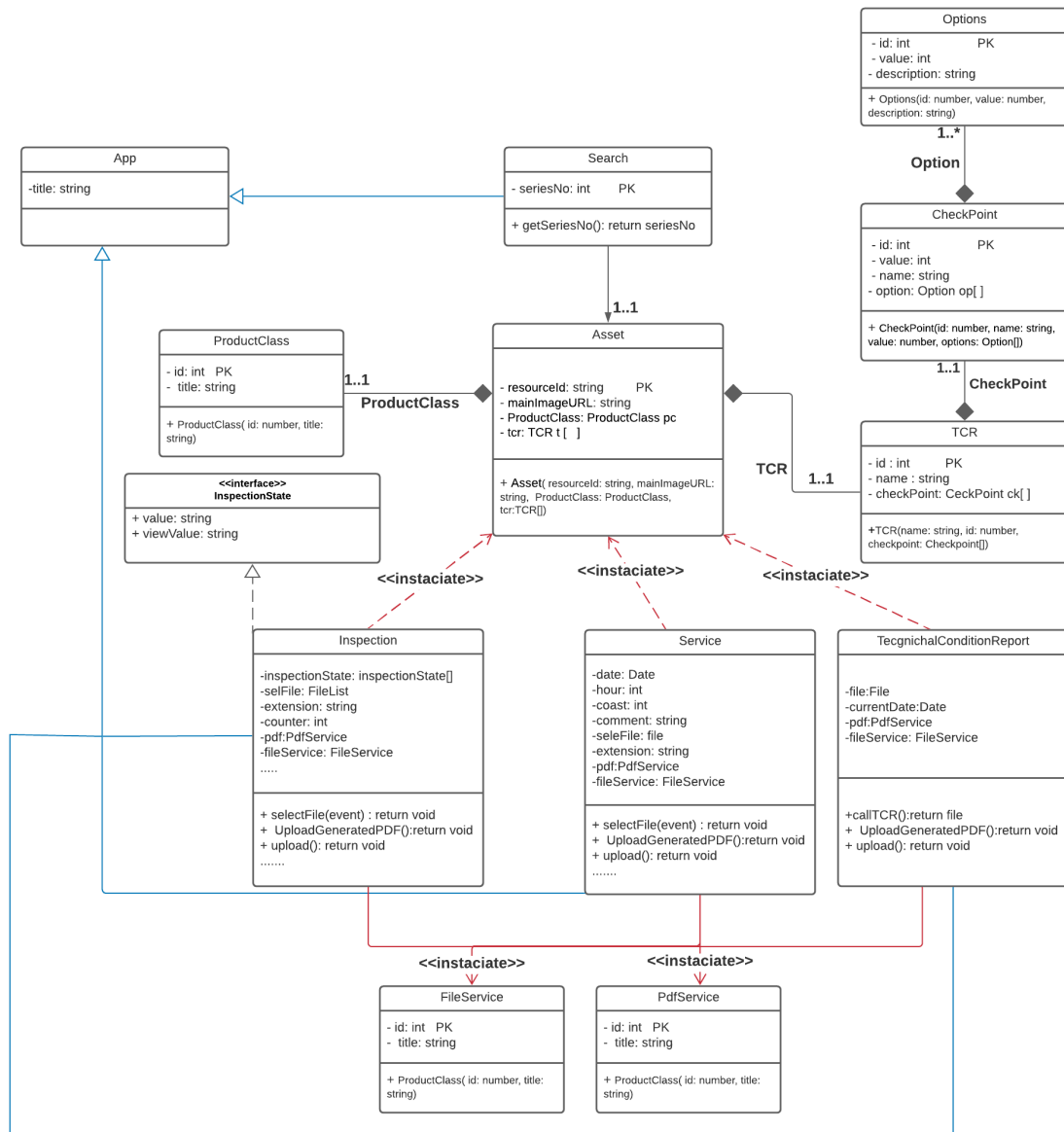


Figure 4.5: Class diagram that represents the important classes and interfaces.

4.2.3 Prototyping

Prototyping is an important process of creating in ital models of a desired product to test a certain design[16]. The team decided on creating mock-ups in the planning phase to visualize the requirements before the development phase started. According to an article written by an interaction design online school called "Interaction Design Foundation", under the title "Mock-ups"[28]. Mock-ups are used mainly to collect feedback from product owner about design in the planning phase. As the requirements in Table 2.3 are clear, the mock-ups design had to accommodate to those requirements. This has led to the mock-ups shown in Figure 4.6. The mock-ups helped us to recognize the mistaken requirements.

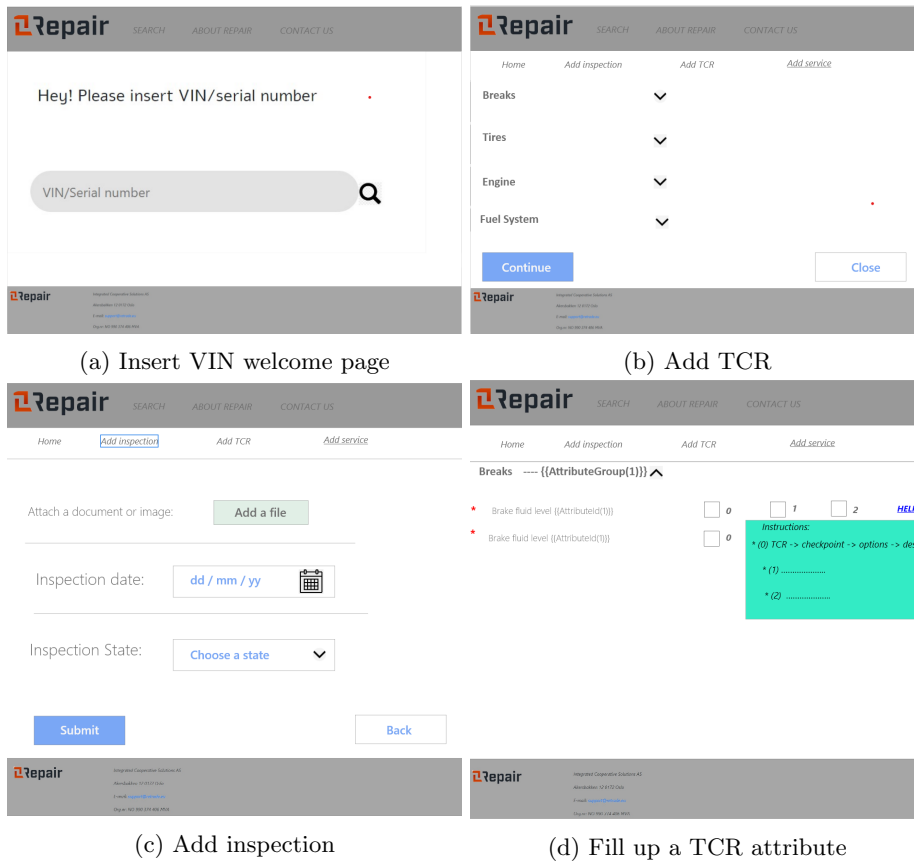


Figure 4.6: System mockups.

4.2.4 Components

According to the Angular documentation[9], components are the main building units of an Angular application. They mainly consist of three parts:

- **HTML:** An HTML template that is typically used to render the specified component. However, it is possible to create a complex structure, which is known as “nested components” [49]. In the beginning of the planning phase, the team felt they needed to design an overview of the structure of HTML template, as shown in Figure 4.7. It represents the layout of HTML templates. This was needed to acquire insight of the overall HTML structure of the application. The blue color encoded on some of the components in Figure 4.7 indicate that the HTML template of those components would be displayed in the body of App component once the button was clicked.
- **TypeScript:** A TypeScript file that contains one or multiple classes that define the behavior of the component. TypeScript component classes were decorated with @Component decorator.
- **CSS:** Every component requires a CSS selector. The selector’s role is to instantiate the component whenever the tag <“Selector”>appears in the HTML template. Also, the component’s style usually is specified in its own CSS file. The name of the file must be inserted in selectedUrl decorator of the component[10].

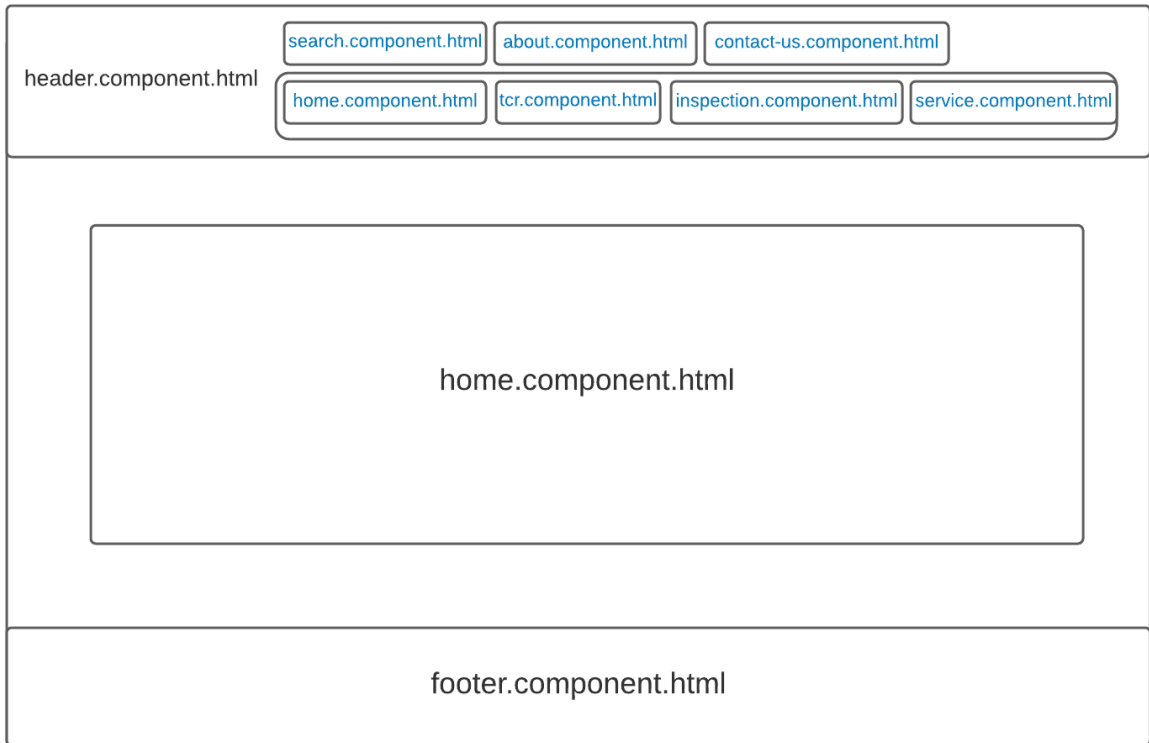


Figure 4.7: HTML structure.

4.2.5 services

An Angular service is an `Injectable`[11] class that contains functions, methods and variables to pass to other services and components. A service does not have the same structure as component and it does not require HTML and CSS templates. The main function is to pass data through the application. A service can be used in a service/component by dependency injection design pattern. Figure 4.8 is an example that illustrates the relationship between components and services in Assetfront Repair. Detailed implementation of component and services is discussed further in Section 4.3.

N.B.: An Angular service can be mistaken with the Service functionality in our application. Therefore, we decided to use Service with a capital letter to refer to the service functionality.

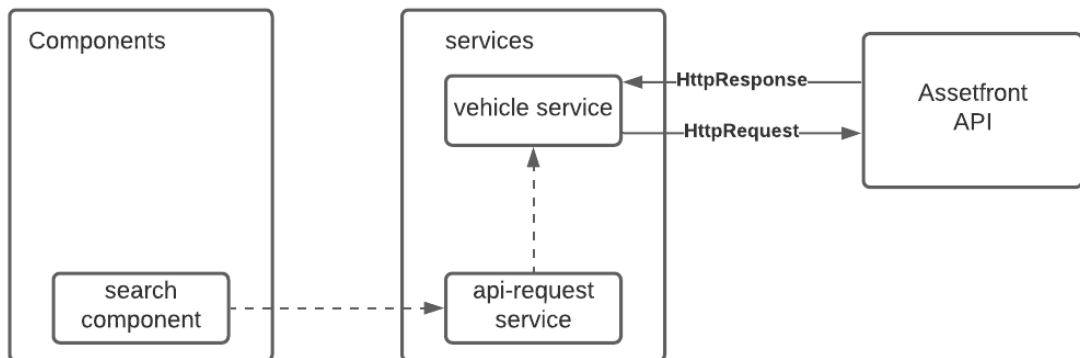


Figure 4.8: An example of services and components interaction.

4.2.6 GUI and UI Design

As mentioned in the non-functional requirements Section 2.2, the application’s overall design must fulfil the requirements. The list below illustrates how the team had approached the various requirements:

Responsive design

The home page of the application is shown in Figure 4.9. The screen sizes used in this figure are: 750 x 1334 Pixels mobile phone and 1920 x 1080 Pixels portable Personal Computer (PC). The key elements to adopt this approach in our application were media queries (illustrated in code Listing 4.1), dynamic resizing in CSS, and a flexible grid layout. The approaches were inspired by an article under the title of “Responsive Web Design” [32].

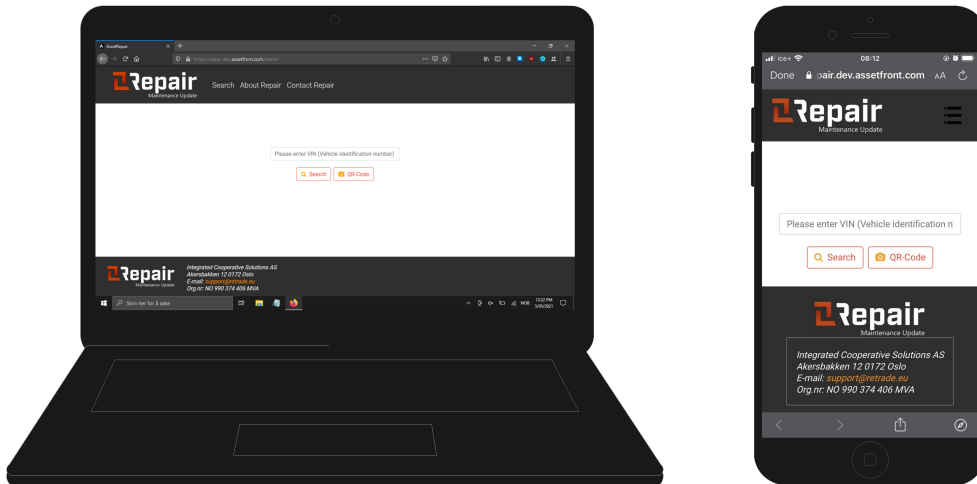


Figure 4.9: Phone vs PC

```
1
2 @media screen and (max-width: 320px) {
3   #navigation{
4     #logo{
5       height: 3rem;
6     }
7   }
8 }
9 @media screen and (min-width: 769px) {
10  #navigation{
11    #logo{
12      height: 5rem;
13      margin-left: 2rem;
14    }
15  }
16 }
```

Listing 4.1: Media queries example.

Familiarity

The application will be a part of an already existing web application called Assetfront⁴. The target audience are mainly the users of Assetfront. Thus, it is important the application’s design was inspired from the Assetfront design.

Simple and informative The main approach in this category was to maintain a clean, yet informative layout. The element displayed to the user had to have a specific useful purpose, as

⁴<https://assetfront.com/>

well as we ensured that all the fields are self explanatory (illustrated in Figure 4.10).

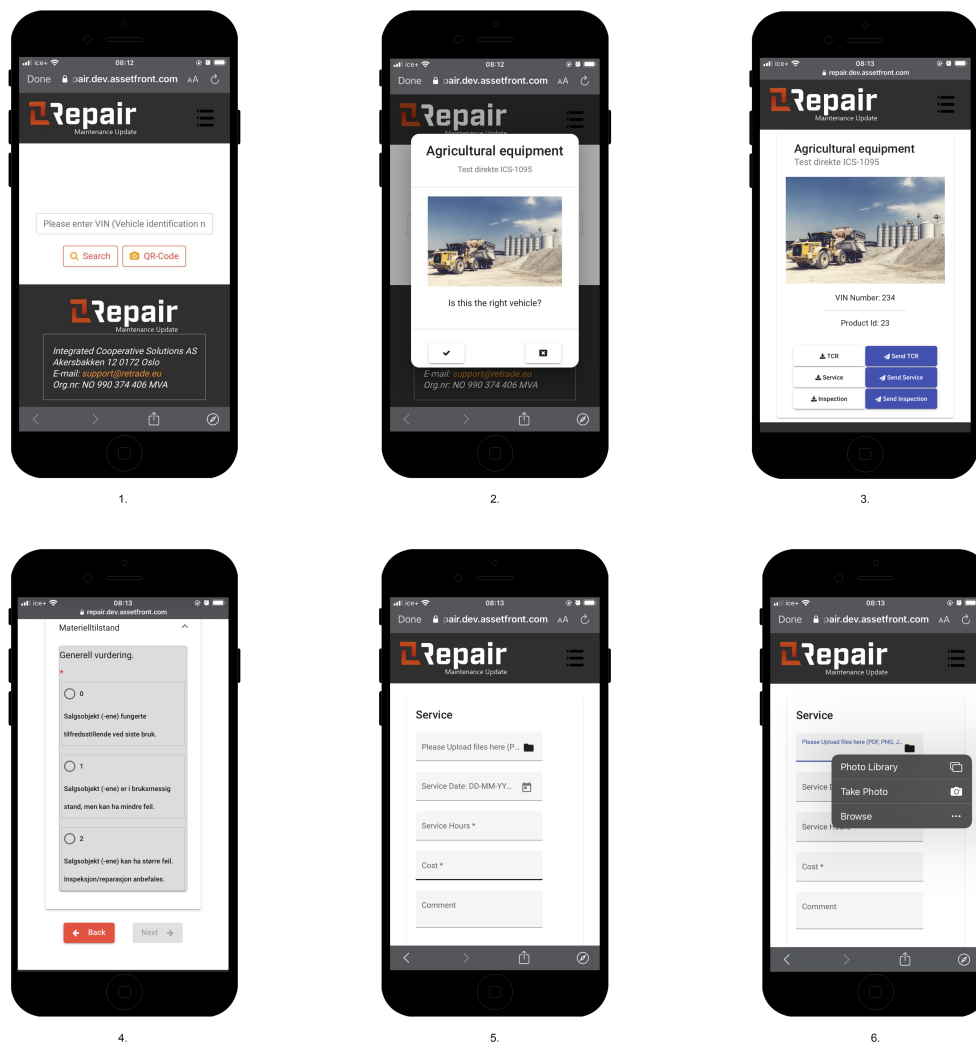


Figure 4.10: Simple and informative design.

4.2.7 S3 file-system Design

In today's solution, Assetfront runs in Amazon web Service (AWS), and uses its services. Assetfront stores application related files in an S3 bucket. Therefore, the team decided on utilizing the same tool for Assetfront-Repair as well. When a user sends a Service or an Inspection, a PDF will be generated. The PDF contains a summary of the filled form. As well as a QR-code that could be scanned in later in the application. PDF and QR-code functions are explained in Section 4.3.3. The PDF and attachments - if the user wishes to attach files to the form - will be uploaded to S3. Then, the user can download the most recent PDF form for Service, Inspection, and/or TCR, as shown in (3-in-Figure-4.10). Every vehicle in Assetfront has an unique resourceId, which it will be identified by. The approach to store files in S3 was hierarchical file system, where the data can be accessed starting from a root file and traversing downward through the levels of hierarchy, as illustrated in Figure 4.11. The root folder in the hierarchy is *asset-repair* which is the bucket name. 1 -in- Figure 4.11 show three different resourceId. Since it is not possible to attach files to a TCR, the *Attached-files* folder 4 -in- Figure 4.11 will contain a JSON file instead, see Section 4.3.3 for an explanation of the JSON file's functions.

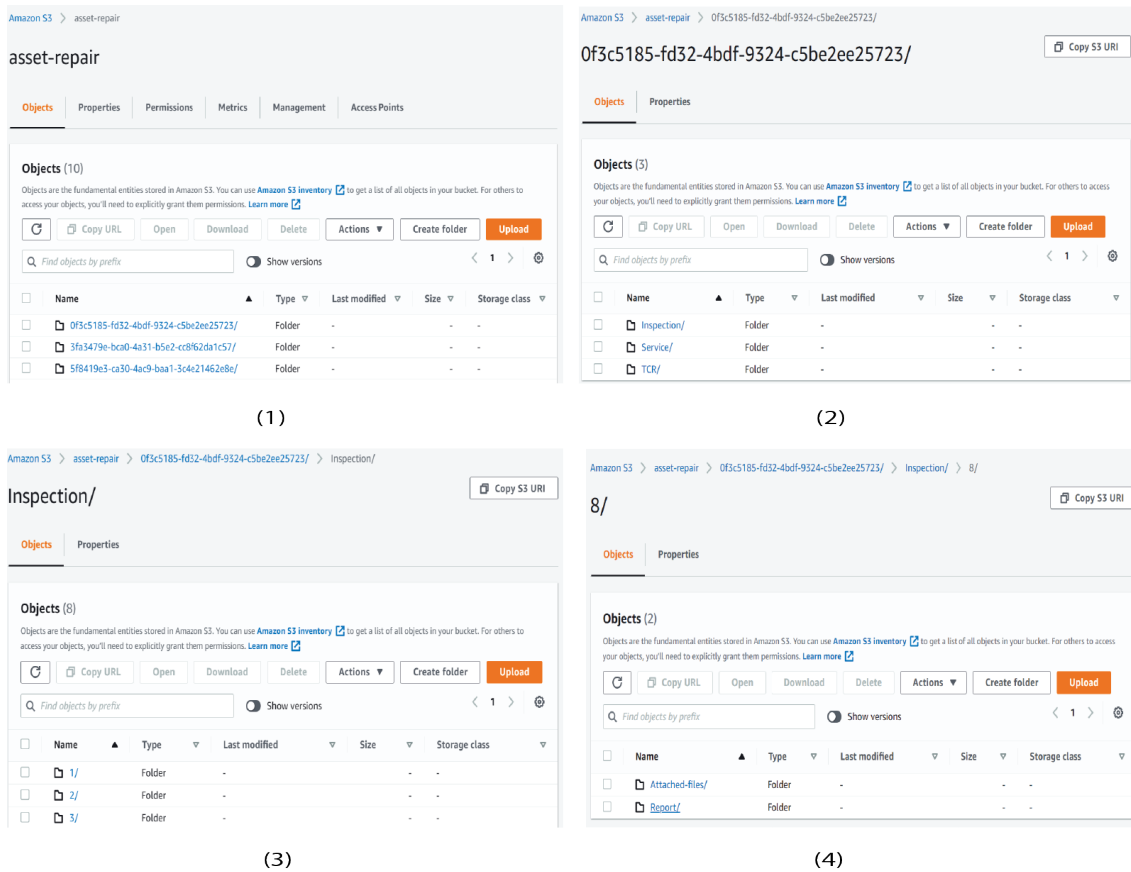


Figure 4.11: S3 Hierarchical file system.

4.3 Implementation

After a thorough discussion among the team and the company about the importance of a back-end solution in Assetfront Repair, we reached the conclusion that developing a back-end for this application is an unnecessary use of time and resources. The main reason behind this argumentation is that we already have been provided an API, where we retrieve all the data needed to be processed by the application.

Consequently, Assetfront Repair is solely a front-end application and is mainly written with Angular version 9[20]. Using a component-based system makes it relatively easy to achieve loose coupling and high cohesion[22], which helps to build scalable and manageable system.

4.3.1 System Overview

Send and Download a form

The process of submitting and downloading a form involves four different entities (user, application, API, and Amazon S3 bucket). The sequence diagrams shown in Figures 4.12, 4.13, and 4.14 illustrate the interaction between the system entities. Since the system does not have a login functionality, the main page starts with the user entering VIN. If the user entered the correct VIN, he or she will be redirected to the home page where he or she would get the opportunity to fill and send new forms, or download previously uploaded forms. Otherwise, the user will get an error message.

If the user wishes to send a TCR (Figure 4.12), he or she will be prompted to fill out the

TCR form and user details. In case something went wrong, the user will be notified. Otherwise, the application will generate PDF and JSON files from the user input and TCR definition (the functionalities are presented in Section 4.3.3). Finally, it uploads the files to the Amazon S3 bucket.

On the other hand, sending of Service and Inspection works differently. What distinguishes sending Service and Inspection from sending TCR is that the former allows the user to upload file/s. Additionally, the system gives mobile users the possibility to take a picture and upload it. Moreover, to mitigate the risk of uploading malicious files, the system restricts the file extension to only PDF, PNG, JPEG. Consequently, if the user tries to upload files with a forbidden extension, the user will get an error message. Otherwise, the file along with the generated PDF file will be uploaded to the Amazon S3 bucket. Subsequently, the user will get a successful affirmation message. Besides filling out and uploading forms, the user will get the opportunity to download previously uploaded files. The possibility to download files is restricted only to the last uploaded folder. The details of the Amazon S3 file structure is presented in Section 4.2.7. As illustrated in Figure 4.14, if the user wishes to download a TCR, the system will send a request to Amazon S3 bucket to get a list of objects in the specified folder. If the TCR folder is empty, the user will be notified accordingly. Otherwise, the list of files in the last uploaded folder will be displayed to the user. Eventually, the file/s that the user selected will be downloaded. **N.B.:** Amazon S3 uptime is 99.99% according to AWS documentation[3], which makes it unlikely for server related problems to occur.

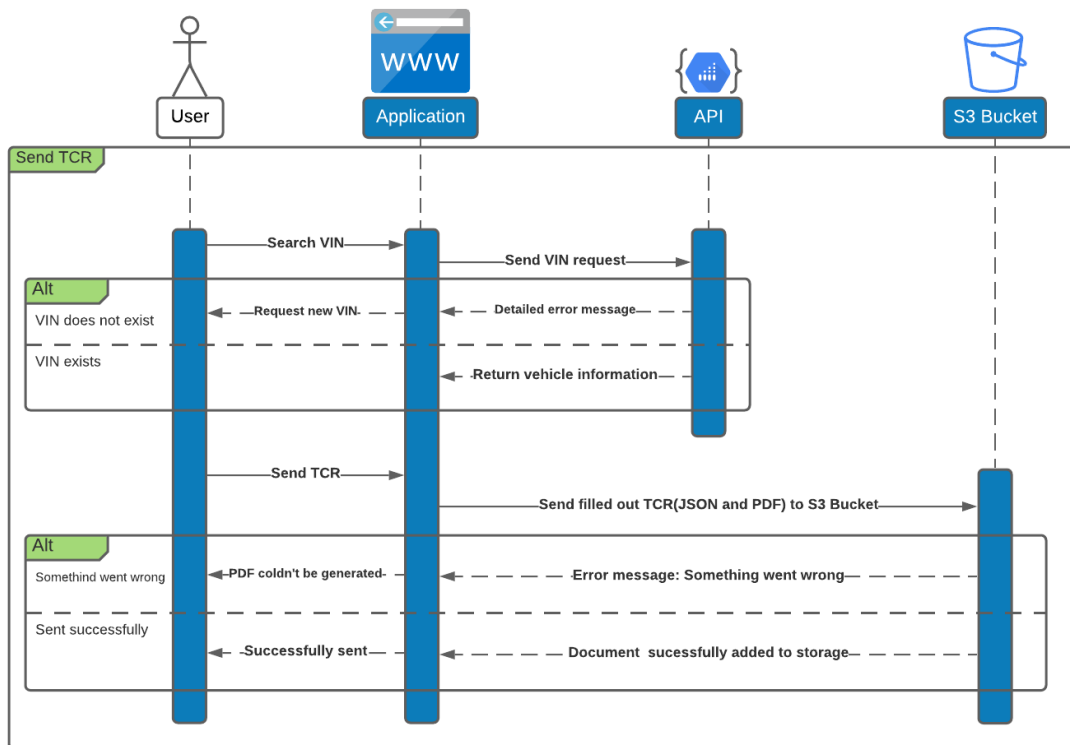


Figure 4.12: Sequence diagram showing how send TCR works.

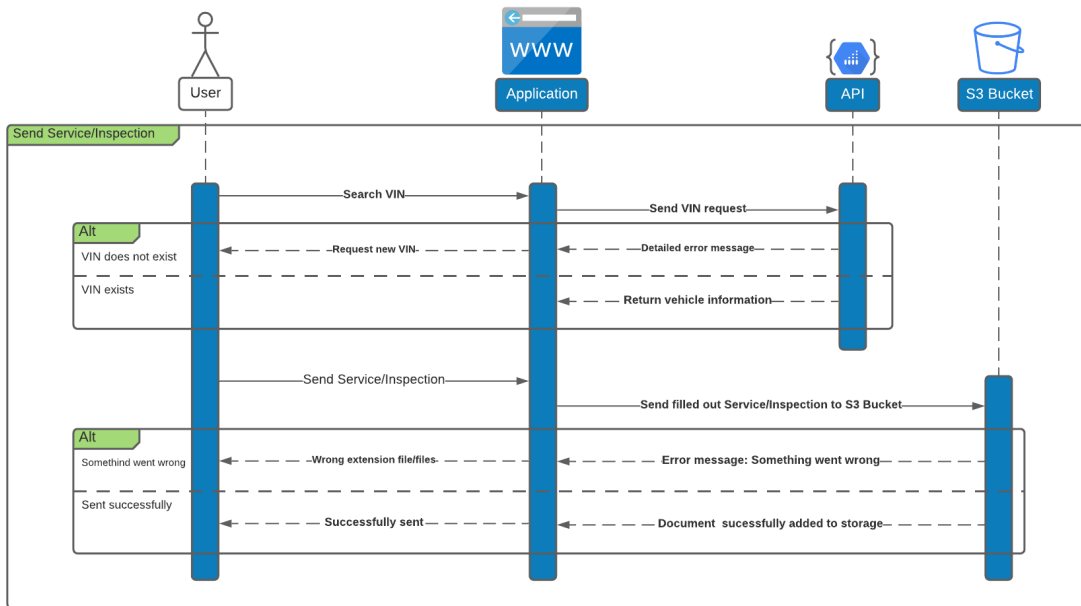


Figure 4.13: Sequence diagram showing how send service or inspection work.

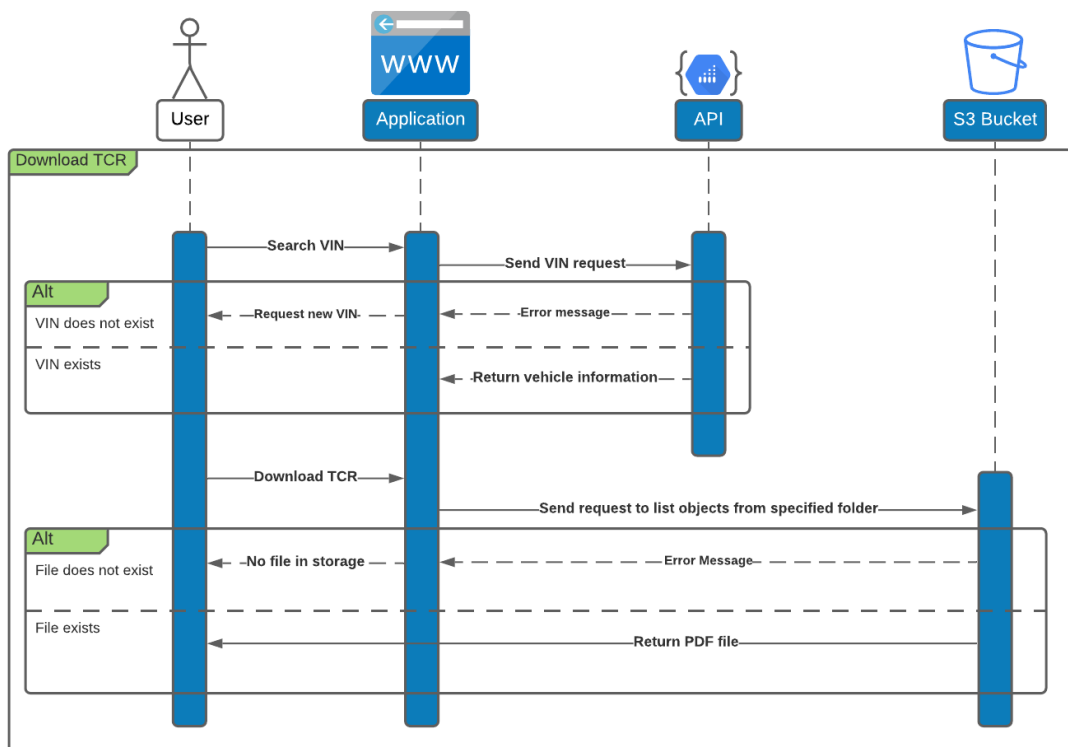


Figure 4.14: Sequence diagram showing how download TCR work.

4.3.2 Connecting to API

Assetfront Repair is entirely dependent on Assetfront API. The application is connected to the API using Interceptors. Interceptors are built-in tools for handling HTTP requests globally[6].

They allow developers to catch incoming or outgoing HTTP requests using the `HttpClient`⁵.

The “`AppHttpInterceptor`” class in Listing 4.2 implements the `HttpInterceptor` interface and overrides the `intercept` method. The `intercept` method takes `HttpRequest`⁶ and `Handler`⁷ as parameters and returns an `Observable`, which is `HttpEvent`⁸.

The aforementioned method sets the `HttpRequest` header’s content type to `application/JSON`. As shown in Listing 4.3, the interceptor is registered as a multi-provider in the “app module”. It is because there can be multiple interceptors running within an application.

```
1 intercept(req: HttpRequest<Asset>, next: Handler): Observable<HttpEvent
  <any>> {
2   req = req.clone({ headers: req.headers.set('content-type', 'application
    /JSON') });
3   return next.handle(req)
4     .pipe(
5     ... //error handling is intentionally omitted
6     )
7   );
8 }
```

Listing 4.2: Interceptor Service.

```
1 providers: [VehiclesService,
2   {
3     provide: HTTP_INTERCEPTORS,
4     useClass: AppHttpInterceptor,
5     multi: true
6   }
]
```

Listing 4.3: Registration of Interceptor as a multi-provider.

The “`VehicleService`” service invokes an interceptor by making an `HttpRequest` using the `HttpClient` service. This is done by injecting `HttpClient` into the `VehicleService` (this is illustrated Listing 4.2).

Then, the method “`getVehicleData(vin)`” takes a vehicle’s VIN as a parameter, and returns an `Observable`⁹ `HttpResponse`. The passed parameter is added to the API Uniform Resource Locator (URL) to retrieve a desired vehicle’s data (the method is presented in Listing 4.4).

```
1 getVehicleData(vin: string): Observable<HttpResponse<any>> {
2   const url = 'https://test-assetlookup.dev.assetfront.com/asset/lookup/'
3     + vin;
4   return this.httpClient.get(url, {observe: 'response'});
}
```

Listing 4.4: `getVehicleData` method.

Promise vs. Observable

Promises deal with one asynchronous event at a time, while `Observable` handle a sequence of asynchronous events over a period of time [5]. First, we attempted to use `Observable` to handle `HttpRequest`, since “`get(URL)`” returns `Observable` as well. “`get(URL)`” is a built-in method in `Http` class that returns `HttpResponse` in form of an `Observable` of `HttpClient`.

Nonetheless, we found out that `Observable` was not the right choice for our application. This is due to some components needing to access data before it is rendered. As a result of that, the `Observable` was converted into `Promise` using the “`toPromise()`” method [45], the code is presented in (Listing 4.5).

⁵<https://angular.io/api/common/http/HttpClient>

⁶<https://angular.io/api/common/http/HttpRequest>

⁷<https://angular.io/api/common/http/Handler>

⁸<https://angular.io/api/common/http/HttpEvent>

⁹<https://angular.io/guide/observables>

```

1 setSerialNo(value){
2   const promise = new Promise((resolve, reject) => {
3     this.request.getVehicleData(value)
4       .toPromise()
5       .then( resp => {
6         if(resp.status === 200){
7           this.assetDetails = resp.body;
8           resolve();
9         }
10      })
11     .catch((error) => {console.log(error); reject(error);});
12  });
13  return promise;
14 }

```

Listing 4.5: Method showing how to convert Observable to Promise.

4.3.3 service overview

As stated in Section 4.2.5, services are significant classes that help to share data between components. Accordingly, Assetfront Repair uses services to share data between its components. The application has seven services that provide data to the components and other services, these services are listed in Table 4.1).

These services needed to be injectable to pass data over to the desired components or other services (see Section 4.2.5). This is achieved by importing injectable form @angular/core and using the @injectable() decorator, as presented in Listing 4.2.

Table 4.1: List of the Services with its dependencies.

Service	Description	Dependencies	Passes data to
AppHttpInterceptor	Intercepts HttpRequest	None	AppModule
ApiRequestService	Changes an observable to promise and assignn httpResponse body to assetDetails.	VehiclesService	SearchComponent HomeComponent
FileServiceService	Responsible for setting Amazon S3 bucket.	None	TCRComponent InspectionComponent ServiceComponent, HomeService.
HomeService	Provides a functionality to upload files to Amazon S3 bucket, and download files.	ApiRequestService FileServiceService FileSaverService.	HomeComponent ServiceComponent InspectionComponent.
InputDataTransferService	Tranfers user details data between the desired components.	PDFService VehiclesService	PersonalDataComponent ServiceComponent InspectionComponent DialogServiceComponent.
PDFService	Provide a functionoality to generaten and dwnload PDF functionality.	None	ServiceComponent InspectionComponent DialogServiceComponent.
TCRService	Sets and gets the newly filled TCR.	None	PersonalDataComponent
VehiclesService	Implements API request for a particular vehicle.	HttpClient	ApiRequestservice PDFservice HomeComponent

PDF file generation

The application generates a PDF file from the user input in each Service, Inspection, and TCR. In the case of a TCR, the application also generates a PDF file from the newly filled TCR's JSON file. Moreover, the PDFService class (see Table 4.1) handles the functionality of a PDF file generation. The class makes use of jsPDF[21] open-library to generate the needed file. As the structure of the PDF file content is highly dependent on the form to be filled out, it was necessary to have a specific method that deals with all three forms separately. However, it is truly only the TCR generation that stands out from Service and Inspection.

PDF generation for newly filled TCR

The generated PDF file is a combination of the TCR definition for a particular vehicle (represented in (d) component in Figure 4.2), personal details of the user, and a QR-code (see Section 4.3.3). Appendix I shows a sample of the PDF file in a random vehicle.

The code in Listing 4.6 presents the method "PlaceForm()" that is responsible for generating the PDF file. This method takes a TCR as a JSON file (see Appendix H) and personal details as parameters. Furthermore, it loops through the JSON file and adds data to the PDF file accordingly, as described in Section 4.3.4. TCR JSON file has a nested data structure that cannot be extracted

with a single loop (see Appendix H). Consequently, the method uses triple nested for-loops to generate the desired PDF file.

```
1  PlaceForm(json: any, Company, Name, Date, Email, PhoneNR){
2      this.Person(Company, Name, Date, Email, PhoneNR):
3      //Loops through each TCR[] and adds its name to the PDF
4      for (let tcri = 0; tcri < json.length; tcri++){
5          tcrName = json[tcri].name;
6          //Write to the file
7          this.doc.text( tcrName , this.FormStartX - 2 ,this.PdfStartY += 4.5);
8          // Loops through each checkpoint [] and add its name to the PDF
9          for(let cpi = 0; cpi < json[tcri].checkpoint.length; cpi++){
10             this.PageLimit(240);
11             CheckpointName = json[tcri].checkpoint[cpi].name ;
12             this.doc.text(CheckpointName, this.FormStartX, this.PdfStartY +=
13                 4.5);
14             //Loops through options[]
15             for(let opi = 0; opi < json[tcri].checkpoint[cpi].options.length;
16                 opi++){
17                 this.optionHolder[opi] = json[tcri].checkpoint[cpi].options[opi].
18                     description;
19                 if ( opi === json[tcri].checkpoint[cpi].value) {
20                     this.SmallRectangle(true, xNow - 20, this.PdfStartY += 3); //
21                         Filled
22                     this.doc.text(this.optionHolder[opi], xNow - 7, this.PdfStartY
23                         += 3.5);
24                 } else {
25                     this.SmallRectangle(false, xNow - 20, this.PdfStartY += 3); //
26                         Hollow
27                     this.doc.text(this.optionHolder[opi], xNow - 7, this.PdfStartY
28                         += 3.5);
29                 }
30             }
31         }
32     }
33     return this.doc.output('arraybuffer'); // Returns file object
34 }
```

Listing 4.6: Method to generate PDF for TCR.

QR-Code

QR-code and its scanner play a significant role in speeding up the process of looking up vehicles in Assetfront Repair. The QR-Code is a part of each generated PDF for every vehicle. The scanner reads and puts the value of the VIN directly to the search input field and starts the searching process by bypassing the button click. It eases the process of searching for a vehicle by its VIN, which is up to seventeen long characters (digits and capital letters). Entering those characters into the input field is a bit time-consuming, especially while sending forms for multiple vehicles. However, these advantages introduce security issues into the application (see Section 2.14).

Neither QR-Code nor scanner was in the requirement in this project at the beginning. Nevertheless, after discussing the pros and cons of QR-code with the company, the latter concluded that the advantages weigh. QR-Code generation takes place in the HTML templates of ServiceComponent, InspectionComponent, or PersonalDataComponent (see Table 4.2). Each of these components will generate a QR-Code from VIN that the user used to search for a particular vehicle. Then the PDFService class retrieves it from canvas and converts it to a Base64[36]. Finally, the Base64 string is added to the PDF of the wanted form. For generating QR-Code, the “ngx-qr-code”[41] open-source library is used. Similarly, the QR-Code scanner is implemented in SearchComponent using an open-source library called “Zxing”[42].

4.3.4 Component Overview

Although Assetfront Repair consists of 15 components that work together to achieve the required functionalities. Only the essential components will be discussed in this Section. Table 4.2 presents an overview of component names, description, and dependencies.

Table 4.2: List of the components with its description and dependencies

Component Name	Description	Dependencies
About	Page to display information about assetfront Repair.	None
App	Root component.	None
Contact-Us	Page to displays contact information of assetfront.	None
Dialog-Window	Dialogue window to prompt user to verify vehicle.	ApiRequestService MatDialogRef Router, VehiclesService
Fallback	Page to displays 404 Error message.	None
Footer	Application footer.	None
Header	Application's Navar consisting links that redirects to search, About and contact-us components.	None
Home	Main page to download and send TCR, Inspection and Service.	ApiRequestService VehiclesService HomeService
Inspection	Page where a user sends Inspection form or attachment.	ApiRequestService FileServiceService FormBuilder PDFService, HomeService InputDataTransferService MatDialog RouterVehiclesService
User-Detail	Page to collect a TCR user information.	ApiRequestService FileServiceService FormBuilder HomeService InputDataTransferService MatDialog, PDFService Router, TcrService VehiclesService
Search	The application's search page.	ApiRequestService MatDialog VehiclesService
Service	Page where a user sends Service form or attachment.	ApiRequestService FileServiceService FormBuilder HomeService InputDataTransferService MatDialog, PDFService Router
TCR	Page to Fill out TCR and upload it to Amazon S3 bucket in both JSON and PDF format.	ApiRequestService ChangeDetectorRef FormBuilder Router, TcrService
TCR-Dialog	Dialogue window that gives user an opportunity to download the newly filled TCR in PDF format.	MatDialog

Search Component

This component is responsible for accepting user's input. The user is given the possibility to either scan a QR-code or enter a value into the input field. Once the user enters a VIN and clicks the search button, the "setSerialNo()" method triggers, which calls the method "apiRequest.setSerialNo(value)". The latter method sets the VIN to the parameter "value", the code is presented in Listing 4.7. If the entered value is a valid VIN, a confirmation dialog window will appear and prompt the user to verify the vehicle (see Figure 4.16). Otherwise, an error message would be displayed on the browser (see Figure 4.15). Based on the response, the user would be either redirected to the home page (shown in 3 Figure 4.10), or back to the search page (shown in 1 Figure 4.10).

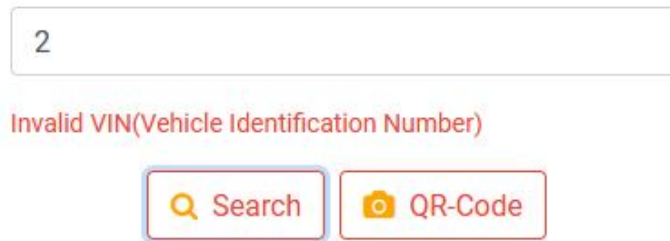


Figure 4.15: Error message due to wrong VIN

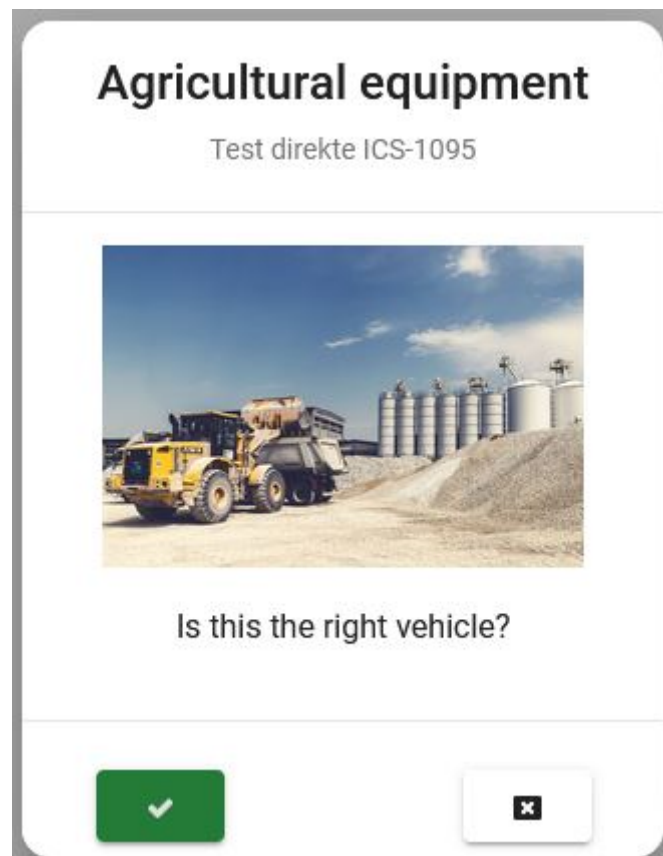


Figure 4.16: Confirmation dialog window

```
28 setSerialNo(value: string) {  
29   const regExpr = new RegExp(/^[A-Za-z0-9]/);
```

```

30  if (value.match(regExpr)) {
31      try {
32          this.request.setSerialNo(value).then(() => {
33              if (this.request.assetDetails.length > 0) {
34                  this.openDialog();
35              }
36              else {
37                  this.errorMessage = 'Invalid VIN(Vehicle Identification Number)';
38              }
39          });
40      } catch (error){
41      }
42  }
43  }

```

Listing 4.7: Method to set Serial Number

TCR Component

As briefly stated in Table 4.2, the TCR component is where a mechanic fills and sends a new TCR form, after examining a vehicle. Each vehicle has its own TCR definition, see Appendix C. During an HTTP request, the apiRequest service retrieves the TCR definition from Assetfront database through Assetfront API (see Figure 4.2), and passes it to the TCR component by injection. The data retrieved is of a type JSON file. Every vehicle has at least one TCR, and a TCR itself has at least one checkpoint. Again, checkpoint itself has at least one option see Appendix C.

TCR:

- ID
- Name
- Checkpoint
 - Checkpoint:**
 - Id
 - Name
 - Value
 - Option
 - Option:**
 - Id
 - Value
 - Description

To avoid unnecessary use of time and resources, barely relevant data to a mechanic should be rendered on the browser. Thus, the data to be displayed on the browser is: TCR name - checkpoint name - checkpoint value - options description. What distinguishes an unfilled TCR from the newly filled one is that the unfilled TCR's checkpoint value is -1, which means that the TCR is untouched. In Appendix H, a filled TCR JSON file is shown. The value of checkpoint is 0, which means that the user chose the first option.

Whenever a mechanic browses through the TCR page, he or she will get an untouched TCR regardless of the previous status. That is due to the application getting data from an API, and the application was not given the privileges to communicate with the database. However, it is uploaded to an Amazon S3 bucket in both JSON and PDF format, which allows other Assetfront subsystems like the Assetfront Retrade to retrieve the data from the JSON file when needed (see Appendix H and I).

To make the TCR user-friendly, a combination of “mat-accordion”, “mat-expansion panel” [34], and “mat-card” [33] was used. Displaying this element in HTML template is done through for-loops. It is mainly dependent on the length of a particular TCR and its checkpoint. Likewise,

Angular nested ngFor directive¹⁰ were used in the component’s template to get and render the desired data on the browser (as presented in Figure 4.17). The implementation of this method is included in Appendix K.

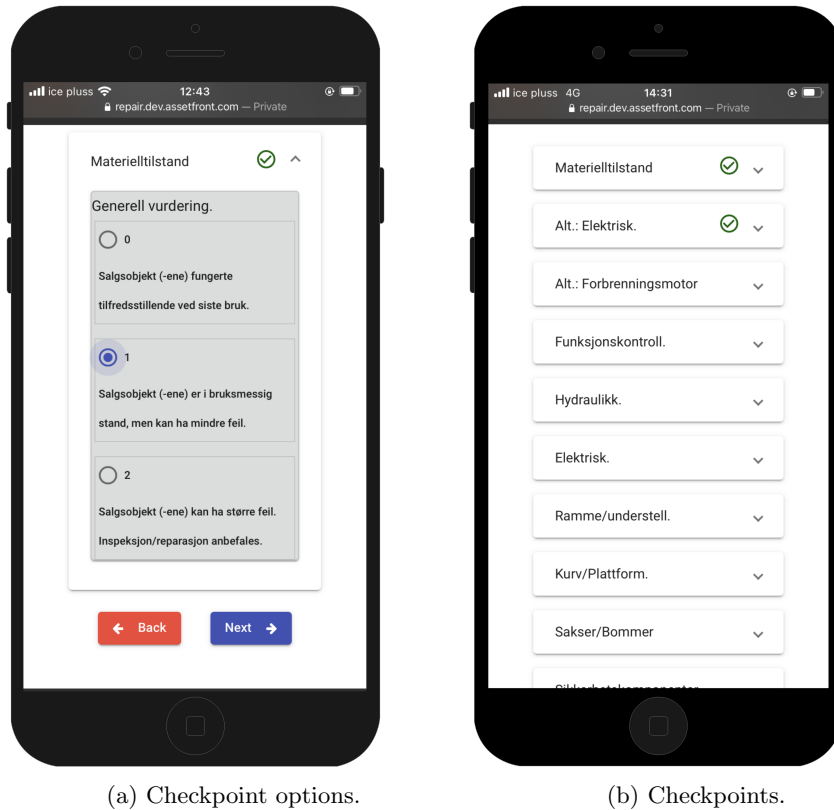


Figure 4.17: TCR overview.

Home Component

The Home Component is where the user gets the opportunity to download previously uploaded Service, Inspection or TCR. Additionally, the user can be redirected to the desired components from home page. The logic of downloading a file from Amazon S3 bucket is handled in “HomeService” class, where it is injected in to Home component. This will allow the component to get access to the necessary data to download the desired form. The download buttons (in 3 Figure 4.10) triggers the download method, which calls “home.getListObjects” method (presented in Listing 4.8). The “home.getListObjects method” takes a folder as a parameter which is either Service, Inspection, or TCR, and pushes the last modified files of the desired folder to a global variable “fileArray”. However, retrieval of objects, and sorting the retrieved objects in a certain folder can be resource-intensive over time. Therefore, the file structure of the Amazon S3 bucket is manipulated. This is done by introducing a sub-folder that must be incremented during uploading the file (see Section 4.2.7).

```

1
2 getListObject(folder){
3   this.resourceId = this.apiRequest.getAssetDetails()[0].resourceId;
4
5   this.fileService.getS3Bucket().listObjectsV2(params, (err, data) => {
6     if (err){
7       console.log(err);
8       this.lastModified = '';
9     }
10    else {
11      this.commonPrefix = data.CommonPrefixes.length;

```

¹⁰<https://angular.io/api/common/NgForOf>

```

12     this.data = data;
13     if (this.commonPrefix > 0 ) {
14         this.lastModified = data.CommonPrefixes[this.commonPrefix - 1].
            Prefix + 'Report' + '/';
15     }
16     this.fileService.getS3Bucket().listObjectsV2(paramsForList, ((
            error, data) => {
17         if (error) {
18             console.log(error);
19         } else {
20             for (let i = 0; i < data.Contents.length; i++){
21                 this.arrayOfFiles[i] = data.Contents[i].Key.split(this.
                    lastModified)[1];}
22             }
23         }));
24     }
25     });
26 }

```

Listing 4.8: Get last modified folder.

Chapter 5

Deployment and Testing

In this chapter, we will discuss both deployment and testing phases of Assetfront Repair. The team used “Automate static website deployment to Amazon S3” [14], which is a tool provided by Amazon Web Services (AWS¹) to deploy static web-applications (see Section 5.1.1). In addition, the company required us to dockerize the application for convenient future deployment.

The second part of the chapter describes the testing procedure. First, static testing was conducted early in the development phase. Then, the team ran manual testing throughout the development phase. Finally, the team conducted user tests with the participation of multiple employees who will use the application once it is deployed to an actual production environment. User testing is an important step, where we ensured that the users did not experience any major bugs in the system, as well as the responsive design and simplicity elements were present.

5.1 Deployment

5.1.1 Amazon Deployment

To store the applications content (HTML, CSS, TypeScript, images, and text files), Amazon S3 service was used. Amazon S3 allows storing data in form of objects. It is also used to host static websites and client-side scripts².

5.1.2 Docker

Docker is a containerisation platform that gives the ability to deploy and run applications by using containers. Containers are isolated work-spaces that provide all tools that a software needs to run. Docker containers help developers to run a piece of code written in a local system in any computing environment. We have used Docker Command Line Interface (CLI³) to run Docker commands. For instance, creating and running docker images⁴ as containers[44].

According to the official web page of Docker[19], Docker is written in the Go programming language, and utilizes multiple features of Linux kernel to provide its functionalities. Moreover, it uses a technology called “namespaces” to provide Docker containers. As Docker creates a set of “namespaces” for a certain container once it starts running.

In order to run an application using Docker, a Docker file must be created. This file is a text file that contains multiple commands to build a Docker image. Initially, we have created Docker files

¹<https://aws.amazon.com/>

²<https://docs.aws.amazon.com/AmazonS3/latest/userguide/WebsiteHosting.html>

³<https://docs.docker.com/engine/reference/commandline/cli/>

⁴<https://docs.docker.com/engine/reference/commandline/images/>

for both front-end and back-end. In that case, a Docker Compose⁵ was needed to define and run multiple containers. However, the requirements changed early in the development phase, where the company decided that a back-end solution was not needed. Therefore, back-end Docker file and Docker Compose were not used in the project. Nonetheless, since the files were created, we decided to leave them in the project structure for future development.

Docker Front-end

Listing 5.1 shows the implementation of front-end Docker file. The Docker file uses Node image from Docker Hub⁶ to download needed dependencies. Additionally, the Docker file uses the command *RUN* to download Node Package Manager (npm⁷) and Angular CLI. Finally, *EXPOSE* command informs Docker that the container listens at the specified network port, which in our application is 4200.

```
1 FROM node:12
2 RUN mkdir -p /app/frontend
3 WORKDIR /app/frontend
4 COPY package.json /app/frontend
5 RUN npm install -g @angular/cli
6 RUN npm install
7 COPY . /app/frontend
8 CMD ["npm", "start"]
9 EXPOSE 4200
```

Listing 5.1: Dockerfile Angular

Docker Compose

As mentioned above, Docker Compose defines and runs multiple containers in the application. Compose uses a YAML^[57] file to manage and configure the application's services, Listing 5.2 illustrates the configuration of Angular service in our application.

```
1 version: "3"
2
3 services:
4   angular-app:
5     build:
6       context: .
7       dockerfile: ./frontend/Dockerfile
8     stdin_open: true
9     ports:
10      - 4200:4200
11     volumes:
12      - ./frontend/src:/app/frontend/src
13     command: npm run start
14     restart: always
```

Listing 5.2: docker-compose.yml file

5.2 Testing

5.2.1 Static Testing

Early in the development phase, static testing was conducted to examine various aspects in the project. Static testing was applied for an early detection of defects before other types of testing

⁵<https://docs.docker.com/compose/>

⁶https://hub.docker.com/_/node/

⁷<https://www.npmjs.com/>

were applied. The team started with testing the requirement specifications, this led to an analysis of functional and non-functional requirements in the sprint review meetings, with the presence of the Scrum master. Then, the API provided by the company was analysed along with API documentation (see Appendix C), this was done by using tools to visualize the data in the API, and ensuring that it was sufficient for the development needs [51]. Moreover, the team created an overview of the libraries needs to develop the project. In case of deprecated or incomplete libraries, the team had to find alternatives. Finally, as mentioned in Section 4.1.1, WebStorm was selected to be used for development, this tool analyses the code and detects problematic code in a given file. This tool helped us during the development phase to keep the code clean and efficient [25].

5.2.2 Manual Testing

It is a type of software testing that runs without using automated tools. Manual testing was ran constantly during the development. Developers were assigned specific tasks, and they had to merge the code of the feature they implemented into the branch all the developers interact with DEV through pull requests (see Section 4.1). Before a pull request is sent, the code needed to be manually tested and code bugs needed to be fixed. Similarly, when the code is merged with DEV branch and further into the developers machines. The process might cause merge conflicts, that needed to be spotted and fixed as well through manual testing.

5.2.3 User Testing

Since the project evolved around making an application to be used in production in Assetfront, it was a high priority to meet the product owner's requirements. Therefore, the team planned to conduct user test, and invite several employees of Assetfront to test the application. To fulfill this step, the team developed a set of survey questionnaire, where the main features of the application were listed and formulated into survey questions. The survey consists of two parts. First part consisted of "to what extent do you agree with the following statement", the answers to these questions ranged from "Strongly disagree" to "Strongly agree". This allowed us to capture the users opinion on certain features, and motivated us to improve it. Speaking of which, the features that were included in user testing were both GUI and functionalities related. The second part of the survey consisted of open-ended questions, where a participant could add a feedback, provide an opinion, and write features he or she liked and did not like. The survey was written throughout the development phase, and was modified accordingly. The survey questionnaire along with the results were presented in Appendix B.

Chapter 6

Discussion and Conclusion

As we have accomplished our goal and looked into the various aspects of the development phase, this chapter will focus on our reflection of the entire process. We will start with discussing the development methods and its aspects. Then, we will shed the light on the development phase and what went into it that could have been done differently, or improved. Likewise, the user test results were studied to determine to what extent the goals have been met. Furthermore, we will conclude with future work section, which will discuss possibilities for future development of the application.

6.1 Development Method and Process

Scrum development model was suitable for our project. The team members managed to hold daily scrum meetings and meet scrum sprints deadlines. In the beginning, the team had planned to have a meeting with the company and product owner weekly. However, as it turned out, the team did not need weekly meeting, instead we had meetings by the delivery of each sprint which was between 1 to 4 weeks, unless it was necessary. Furthermore, two team members had to be absent for a short period of time due to illness and/or personal problem. Hence, this was not a major problem since this case was studied in the risk analysis process. The mitigation of this case was that the team had to ensure that all team members are on the same page in terms of knowing the various aspects in the code and design. As mentioned in the business risk analysis Table 2.18. We have managed to overcome this obstacle successfully.

Later on and towards the end, the daily scrum meeting were reduced, since the main focus was to work on the thesis report. However, we managed to meet the deadlines that were set in the planning phase.

Ultimately, as some of the requirements were unclear in the beginning and changed throughout the development, it was challenging to define the sprints scope. Yet, the team managed to clarify the requirements and get on the track shortly.

6.1.1 Team Cooperation

The team members were used to working with physical attendance from previous course-projects. Therefore, being unable to have physical meetings was challenging. The team has members and relatives whom are under Covid-19 risk group, for that reason it was chosen not to take risks with physical meetings, and this may have affected the workflow. Although the group had daily scrum meetings digitally, it became more challenging to collaborate than physical meetings. Therefore, difficult challenges became more time-consuming. However, this challenge benefited us to develop experience in that regard.

6.1.2 Meetings with The Company and Product Owner

The team had 14 meetings with the company and the product owner throughout the project. There have been various topics at these meetings ranging from functionality, design and technical assistance. The first meeting between the company and the product owner took place on the 11th of November, where we got to introduce ourselves and get to know the client better. At this stage, the company presented the assignment abstractly. Most meetings were attended by everyone involved in the project, the company, the product owner and the development team, see Appendix L. Once all parties were present, the team held presentations where the company and product owner were updated with the current project status. The final meeting with the company was held on the 10th of May, where the final code was explained and handed over for future development.

6.2 Development

6.2.1 Security

The application lacks quite a bit in terms of security as shown in Section 2.4. The group members wanted from the beginning to implement login functionality. However, this was not implemented due to the company arguing that the information handled by the application is public and can be obtained by other sales and auction pages such as Assetfront Retrade. The other reason for the rejection of the proposal was that they thought it was outside the scope of the project. This opinion changed in the final stages of the development phase, when the company encouraged us to implement a login system, mainly to automate user data submission and to mitigate some of the misuse cases of the application to a certain degree (as described in Section 2.4). However, this functionality was not implemented due to the short notice. Although a login system would have lessened the likelihood of anyone abusing the weaknesses in the application. Although it would not have solved all the issues. Misuse cases -like the manipulation of the QR-code on the PDF file- would have been mitigated by having a back-end solution to the application for the construction of the PDF files. While this would have been smart to implement from the beginning, reasons for not developing a back-end solutions were mentioned in Section 4.3. Additionally, the PDF file and QR-code generation were not in the scope of the project.

6.2.2 Application Design

Most of the group members had not directly worked with web application development. Thus, we were motivated to take an online course in Angular [55]. This was done in the planning phase, and it is documented in Gantt diagram (see Appendix A). The time spent was just enough to put us into the basics of the various tools.

The design of the final product was not determined at the start. Therefore, the company decided to engage the team in this process. In the planning phase, the team had to create prototypes for the product, and it was presented to the company and product owner 4.2.3. The company agreed on the most part of the prototypes. Nevertheless, they provided constructive feedback design-wise. As the API was not provided in the beginning of the planning phase, the structure of the TCR was misinterpreted. The team overcame this challenge as soon as the API was provided and visualized by Swagger [51]. Later in the development phase, the design aspects evolved gradually into the current product.

When it comes to tools used in the design of the application, the team assessed various tools and decided to use the most suitable ones. Tools and technologies are presented in Section 4.1.1 and Appendix D. Early in the design process, the team used Bootstrap as a CSS framework to design various components. Since there was a room for improvement in the design, we have combined Bootstrap with Angular Material. This was inspired by the course taken earlier in the planning phase. A combination of these technologies helped to achieve the desired design and functionalities. Similarly, we used Cascading Style Sheets (CSS) as a tool for styling HTML templates early in

the design process, this also changed to Syntactically Awesome Style Sheets (SASS) later on, as we found out that SASS is more efficient and easier to maintain since it uses nested syntax[35].

6.2.3 User Test

The User test was conducted to checkout if the goal of making the application user-friendly while being able to run on different machines and browsers. While creating the questions for the user test, our goal was to make questions that would not manipulate the answers of the testers. This was done by asking direct questions while giving the user the option to choose from a range of answers anonymously. Later on, the users could express their opinions about the application freely. The users were from different countries (Denmark, Finland, Norway and Sweden) and it was interesting to watch these diverse users try to use the application without any guidance, although we were watching them live.

The opinion of the test users were to be taken highly, since these users were experienced with filling out Service, Inspection, and TCR forms. After getting some feedback from the test users, we tried to address the issues that they brought up. For the most part, the test users were happy with the application. Main issues that was raised was that the TCR filling form had a lock on it that made the user fill out the whole thing before letting the user submit the report. The other issue was that it was not clear when the QR-code scanner was finished reading. Both of these issues were addressed swiftly after the user tests. Forcing the user to fill out the report issue was fixed by removing the lock on the submit button, while the scanner issue was taken care of by bypassing the search button right after the scanner finds a value.

6.3 Future work

Even though the team had managed to implement the requirements mentioned in Chapter 2, there is still a room for improvement. The currently developed application lacks the factor of security as it is not need for nowadays use. However, this factor can make the application more complete. Moreover, the application can have an extra feature where the user can register an account that consists of his or her personal data. In the current solution, the user must enter personal data every time a form is filled. With the intention of creating an ideal application, the team thought of implementing these features in the planning phase. However, this was outside of the scope of the assignment. One way to implement the aforementioned functionalities is to use Keycloak which is an open source software to permit access management and add authentication to secure applications [29]. Future implementation of Keycloak requires that the overall system model to change accordingly. Thus, only authenticated users will be able to access the system and utilize its operations. As represented in Figure 6.1, the new component added to the system (e in Figure 6.1) will be responsible for the user-authentication factor.

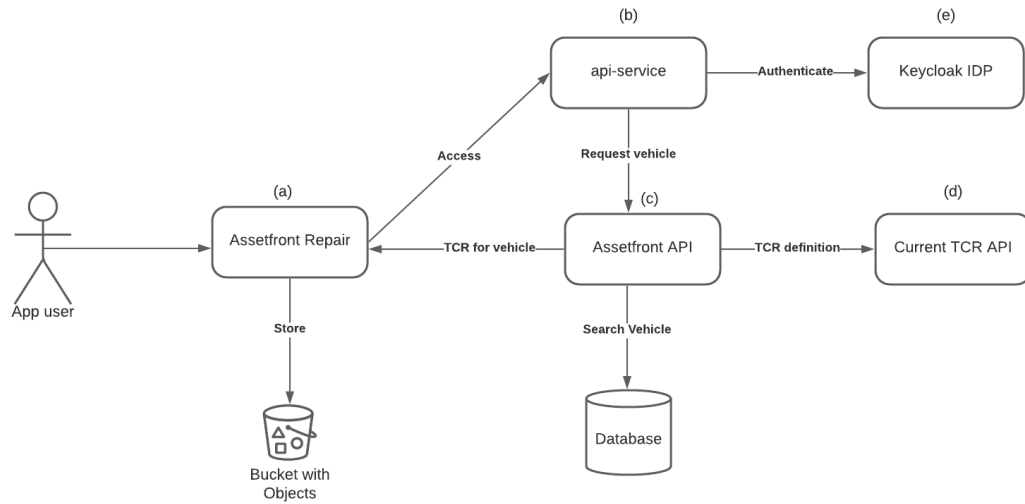


Figure 6.1: Overall system model with Keycloak implementation.

6.4 Conclusion

The objective of the project was to deliver a web-based solution to modernize the creation and storage of documentation created for different type of vehicles. The desired product was a front-end application that has a responsive design and reliable performance. In the end, the development team managed to meet all of the requirements set by the company and product owner, this was clearly stated at the last meeting with the company and the product owner, where the project's source-code was delivered and presented. As it was mentioned in the introduction, the primary audience of Assetfront Repair is Assetfront employees, who do not necessarily have an IT background. Therefore, the application was built to be practical for users with no technical background. Consequently, the team managed to develop an user-friendly application, according to user test results (see Appendix D).

Conclusively, we believe that this project has been a great learning opportunity. As it helped us to acquire experience in multiple fields like: web application development, programming, design, communication, and software development. Although we faced obstacles throughout the project, we managed to overcome them. We believe this is an important skill to bring into our professional lives.

Bibliography

- [1] Dr. A.V. *RISK MANAGEMENT IN PRODUCT DEVELOPMENT PROCESS*. URL: https://www.daaam.info/Downloads/Pdfs/proceedings/proceedings_2012/0225_Susterovaatal.pdf (visited on 05/04/2021).
- [2] ABRAHAM. *View encapsulation*. URL: <https://angular.io/guide/view-encapsulation> (visited on 05/03/2021).
- [3] Amazon. *General S3 FAQs*. URL: <https://aws.amazon.com/s3/faqs/vg> (visited on 05/11/2021).
- [4] Angular. *Angular Events*. URL: <https://angular.io/events> (visited on 04/28/2021).
- [5] Angular. *comparing-observables*. URL: <https://angular.io/guide/comparing-observables> (visited on 05/14/2021).
- [6] Angular. *intercepting-requests-and-responses*. URL: <https://angular.io/guide/http#intercepting-requests-and-responses> (visited on 05/14/2021).
- [7] angular. *What is angular?* URL: <https://angular.io/> (visited on 03/21/2021).
- [8] angular-university.io. *Angular Architecture - Container vs Presentational Components Common Design Pitfalls*. URL: <https://blog.angular-university.io/angular-component-design-how-to-avoid-custom-event-bubbling-and-extraneous-properties-in-the-local-component-tree/> (visited on 05/05/2021).
- [9] angular.io. *Angular Components Overview*. URL: <https://angular.io/guide/component-overview> (visited on 05/05/2021).
- [10] angular.io. *Component*. URL: <https://angular.io/api/core/Component> (visited on 05/09/2021).
- [11] angular.io. *Dependency injection in Angular*. URL: <https://www.pluralsight.com/guides/understanding-the-purpose-nested-components> (visited on 05/09/2021).
- [12] atlassian.com. *Git Feature Branch Workflow*. URL: <https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow> (visited on 05/18/2021).
- [13] AUSFCA-edu. *Extreme Programming*. URL: <https://www.cs.usfca.edu/~parrr/course/601/lectures/xp.html> (visited on 05/18/2021).
- [14] AWS. *Automate static website deployment to Amazon S3*. URL: <https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/automate-static-website-deployment-to-amazon-s3.html> (visited on 05/11/2021).
- [15] AWS. *What is Amazon S3?* URL: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html> (visited on 03/21/2021).
- [16] Amy Hackney Blackwell and Elizabeth Manar. *Prototype*. URL: https://go.gale.com/ps/i.do?p=SCIC&u=dclib_main&v=2.1&it=r&id=GALE%7CENKDZQ347975681&asid=1620273600000~6a20b37f (visited on 05/05/2021).
- [17] Bootstrap. *getting-started*. URL: <https://getbootstrap.com/docs/5.0/getting-started/introduction/> (visited on 05/12/2021).
- [18] William Buxton. *Sketching user experience : getting the design right and the right design*. San Francisco, Calif.: San Francisco, CA : Morgan Kaufmann, 2007.
- [19] Docker documentation. *Docker overview*. URL: <https://docs.docker.com/get-started/overview/> (visited on 03/21/2021).

- [20] Angular guide. *what-is-angular*. URL: <https://angular.io/guide/what-is-angular>.
- [21] James Hall. *A library to generate PDFs in JavaScript*. URL: <https://www.npmjs.com/package/jspdf> (visited on 05/17/2021).
- [22] Wilhelm Hasselbring. *Component-based software engineering*. World Scientific Publishing Company, 2002.
- [23] <https://angular.io/>. *Arthur Abraham's Humane User Interface talk, 2 November 2004*. URL: <https://web.archive.org/web/20120326125547/http://chandlerproject.org/Journal/HumaneUserInterface20041102> (visited on 05/03/2021).
- [24] <https://rxjs.dev/>. *Methods of Risk Analysis*. URL: <https://www.ukessays.com/essays/statistics/risk-analysis-methods.php> (visited on 05/03/2021).
- [25] <https://www.jetbrains.com/>. *Run inspections*. URL: <https://www.jetbrains.com/help/webstorm/running-inspections.html> (visited on 05/12/2021).
- [26] hubspire.com. *What is an API?* URL: <https://www.hubspire.com/resources/general/application-programming-interface/> (visited on 05/03/2021).
- [27] IETF. *The JavaScript Object Notation (JSON) Data Interchange Format*. URL: <https://datatracker.ietf.org/doc/html/rfc8259> (visited on 05/12/2021).
- [28] Interaction-design.org. *Mock-ups*. URL: <https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/mock-ups> (visited on 05/05/2021).
- [29] keycloak.org. *getting-started*. URL: <https://www.keycloak.org/> (visited on 05/15/2021).
- [30] Nevenka Kirovska and Saso Koceski. “Usage of Kanban methodology at software development teams”. In: *Journal of applied economics and business* 3.3 (2015), pp. 25–34.
- [31] Kin Lane. *Intro to APIs: History of APIs*. URL: <https://blog.postman.com/intro-to-apis-history-of-apis/> (visited on 03/21/2021).
- [32] Ethan Marcotte. *HTML5 - Responsive Web Design*. URL: <https://docs.microsoft.com/en-us/archive/msdn-magazine/2011/november/html5-responsive-web-design> (visited on 05/06/2021).
- [33] Angular material. *card overview*. URL: <https://material.angular.io/components/card/overview> (visited on 05/14/2021).
- [34] Angular material. *expansion overview*. URL: <https://material.angular.io/components/expansion/overview> (visited on 05/14/2021).
- [35] Carlos Mauri. *7 benefits of using SASS over conventional CSS*. URL: <https://www.mugo.ca/Blog/7-benefits-of-using-SASS-over-conventional-CSS> (visited on 05/15/2021).
- [36] MDN. *Base64*. URL: <https://developer.mozilla.org/en-US/docs/Glossary/Base6> (visited on 05/18/2021).
- [37] All The Way Up Media. *MOBILE WEBSITES*. URL: <https://allthewayupmedia.com/mobile-website/> (visited on 05/10/2021).
- [38] Microsoft. *The STRIDE Threat Model*. URL: [https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878\(v=cs.20\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)?redirectedfrom=MSDN) (visited on 05/07/2021).
- [39] microsoft.com. *Color Palettes (Windows GDI)*. URL: <https://docs.microsoft.com/en-us/windows/win32/gdi/color-palettes?redirectedfrom=MSDN> (visited on 05/03/2021).
- [40] microsoft.com. *REST API Reference*. URL: <https://docs.microsoft.com/en-us/rest/api/azure/devops/?view=azure-devops-rest-6.1> (visited on 05/09/2021).
- [41] npmjs.com. *How to use ngx-qrcode?* URL: <https://www.npmjs.com/package/@techiediaries/ngx-qrcode> (visited on 05/18/2021).
- [42] npmjs.com. *What is ZXing?* URL: <https://www.npmjs.com/package/@zxing/browser>.
- [43] Mary Poppendieck. “Lean software development”. In: *29th International Conference on Software Engineering (ICSE'07 Companion)*. IEEE, 2007, pp. 165–166.
- [44] Vivek Ratan. *Automate static website deployment to Amazon S3*. URL: <https://www.docker.com/blog/docker-is-the-devops-world-favorite/> (visited on 05/11/2021).
- [45] rxjs.dev. *Convert observable to promise*. URL: <https://www.learnrxjs.io/learn-rxjs/operators/utility/topromise> (visited on 05/17/2021).

- [46] rxjs.dev. *RXJS Overview*. URL: <https://rxjs.dev/guide/overview> (visited on 05/03/2021).
- [47] Sass. *Sass documentation*. URL: <https://sass-lang.com/documentation> (visited on 05/12/2021).
- [48] Adam Shostack. *Threat Modeling Designing for Security*. John Wiley Sons, 2014.
- [49] Gaurav Singhal. *Understanding the Purpose of Nested Components*. URL: <https://angular.io/guide/component-overview> (visited on 05/05/2021).
- [50] Ian Sommerville. *Software Engineering*. Edinburgh Gate Harlow Essex CM20 2JE England: Addison-Wesley, 2007.
- [51] swagger.io. *What Is Swagger?* URL: <https://swagger.io/docs/specification/2-0/what-is-swagger/> (visited on 05/09/2021).
- [52] thinkingportfolio.com. *Project Risks vs. Business Risks*. URL: <https://thinkingportfolio.com/project-risks-vs-business-risks/> (visited on 05/03/2021).
- [53] Powered by Help Scout Trello 2021. *What is Trello?* URL: <https://help.trello.com/article/708-what-is-trello> (visited on 2021).
- [54] RESTful API Tutorial. *What is REST*. URL: <https://restfulapi.net/> (visited on 03/21/2021).
- [55] udemy.com. *Angular - The Complete Guide (2021 Edition)*. URL: <https://www.udemy.com/course/the-complete-guide-to-angular-2/> (visited on 05/15/2021).
- [56] Wikipedia. *Interface description language*. URL: https://en.wikipedia.org/wiki/Interface_description_language (visited on 05/09/2021).
- [57] YAML.org. *YAML Ain't Markup Language*. URL: <https://yaml.org/> (visited on 05/11/2021).

Appendices

Appendix A

Pre-project plan

PRE PROJECT REPORT

Abdi Bako - StudNr: 488104

Amr Hamcho - StudNr: 488087

Mustafa Abdullah - StudNr: 472447

Shamil Khumparov - StudNr: 997722

BIDAT39 BACHELOROPPGAVE - DATAINGENIØR
NTNU I GJØVIK

Dato: 31.Januar 2021

Contents

1	Goals and Boundaries	1
1.1	Background	1
1.2	Goals	1
1.3	Frames	1
1.3.1	Technical	1
1.3.2	Practical	1
2	Scope	2
2.1	Subject Area	2
2.2	Delimitation	2
2.3	Task description	2
3	Project organization	2
3.1	Responsibilities and roles	2
3.2	Rules in group	2
3.3	Measures in case of violation of rules	3
4	Planning, follow-up and reporting	3
4.1	Main division of the project	3
4.2	Plan for status meetings and decision points in the period	3
5	Organization of quality assurance	4
5.1	Documentation, standard use and source code	4
5.2	Risk analysis	4
5.2.1	Risk Identification	4
5.2.2	Risk scenarios and risk Analysis	4
6	Plan for accomplishment	7
6.1	Gantt diagram	7
6.2	Milestones and decision points	8

1 Goals and Boundaries

1.1 Background

Those whom are responsible for those vehicles would like to know which vehicles are available and which ones are unavailable due to problems. Those whom are responsible for fixing those vehicles would like an easier and dependable way of updating the documentation of the vehicles in the system.

Headit is a software development company, which is behind the development of many IT solutions for a different range of costumers.

1.2 Goals

The primary goal of this project is to deliver a working application, where the user can easily search for a vehicle and create a technical report of the vehicle, or download the technical report of the report.

Beside completing and running the application, the subsidiary goals will be to learn how it is to be working on a real life project, and learn from both our fellow students and the more experienced employees at Headit. Additionally Familiarize with software tools used to develop an application from start to finish.

1.3 Frames

1.3.1 Technical

- Making a responsive web design (RWD), which means that the website can be viewed on different screen sizes.
- The application should be delivered and ran though docker containers.
- The web application should be accessible through most browsers that support HTML5.

1.3.2 Practical

Meeting with the scrum master and the product owner will be essential moving forward in the project, thus making it a must meeting with them once every week. The corona-virus situation made it challenging to meet in person, thus we held the meetings over internet over Microsoft Teams.

2 Scope

2.1 Subject Area

Assetfront is a platform that keeps track of an organization's assets, enabling Asset Management to more efficiently keep track of their asset credential. Assetfront enables value-based decision-making throughout the organization. The transparent real-time information and the Assetfront network provide maximum machine coverage and cash flow by letting you strategically introduce and request assets directly in profitable machine communities [1]. Assetfront Repair is part of Assetfront, which will keep track of the condition of vehicles at a company. Workshops, service centers and certification agencies have easy access to update designated Digital Machine Cards directly. Real-time and efficient for both the service company and the machine owner.

2.2 Delimitation

2.3 Task description

Assetfront Repair has the following functionalities:

- Search engine takes serial number to search for specific vehicle.
- Register a service/inspection for the companies assets.
- Digital vehicle documentation view/update.
- Send in a file (images/pdf).
- Dynamically generate PDF files from the registered service.

3 Project organization

3.1 Responsibilities and roles

- Scrum Master - Ronny Kristiansen
- Product owner - Geir Bjerkemo
- Group leader - Shamil Khumparov
- Supervisor - Seyed Ali Amirshahi
- Scrum team - Amr Hamcho, Abdi Bako, Mustafa Abdullah, Shamil Khumparov

3.2 Rules in group

1. In case of sickness, the other group members should be informed as soon as possible.
2. Being on time for group meetings.
3. Maximum three undocumented absences from supervisors meeting.
4. Communication with other group members when an issue arises.

5. A weekly meeting with our supervisor.
6. Sprint planning meeting.
7. Weekly scrum meetings.

3.3 Measures in case of violation of rules

1. Oral warning followed by a written one in case of more than 3 three undocumented absences.
2. The person receives a written warning. It must be pointed out what violation has occurred and be informed of the consequences.
3. If the group is unable to resolve the issue internally, discuss with the supervisor.
4. If the group member does not follow the written warning after meeting with the supervisor, he/she will be excluded from the group.

4 Planning, follow-up and reporting

4.1 Main division of the project

Scrum is the development model which will be used throughout the project development. Scrum is an agile framework for incremental product development, which is widely used in software development. The core of Scrum is iterations, which is repeated until all the product requirements have been implemented. Scrum would be a suitable development model in our project. The product owner has defined the functional requirements which will be added into an artifact called "Product Backlog". The workload will be divided into iterations called "sprints". The duration of a sprint depends on the complexity of the iteration. In our case, the length of the iteration would be between 1 to 4 weeks.

4.2 Plan for status meetings and decision points in the period

The scrum team has planned a weekly scrum meetings on Fridays from 8:30 to 9:30. The meeting will be mainly about sprint planning. The group members will meet 3-4 times a week, and one other obligatory meeting with the supervisor on Wednesday from 14:00 til 15:00.

5 Organization of quality assurance

5.1 Documentation, standard use and source code

The group members have agreed on creating a google drive folder where all the written documents would be available and possible to be modified. As well as making a team on Microsoft Teams we could communicate and held meeting. All the meeting with the supervisor and scrum meetings must be documented. Moreover, the group has created a project planning document using Trello[2] Where we can easily assign a task to a group member and check the development towards the end.

5.2 Risk analysis

We will only be analyzing the risks that affect the project negatively.

5.2.1 Risk Identification

- **Time:** Lack of time. All the group members must learn new technologies.
- **Motivation:** Positive work environment to keep the team members motivated.
- **Cost:** Too ambitious projects for a limited budget.
- **Change in requirements** possible scenarios: change in programming. language or change in project architecture.
- **Change in external systems:** possible scenarios would be: a new product owner.

5.2.2 Risk scenarios and risk Analysis

In this section, we will assess the risk and its affect on different aspects of the project

- **Probability of Occurrence(OV):** likelihood that an identified risk could occur (**0 - 10**) 0 being not present, and 10 being certain.
- **Severity of Consequences Value(CV):** rating based on the impact of an identified risk to safety, resources, work performance, property, and/or reputation. (**0 - 10**) 0 being not present, and 10 being high.
- **Risk Rating (RR) = Probability of Occurrence (OV) x Severity of Consequences Value (CV).**

Possible risk scenarios might be, these are sorted from highest risks to the lower ones:

No.	Name	Description	Probab	Consec	Risk value	Measures
1	Delays	Postponements in sprints	4	9	36	Flexible work assignments through the sprint. The group should have good communication with the supervisor, product owner and between group members. When a group member feels the work assignment is too much, s/he should ask for assistance from other team members.
2	Data Leak	Leakage of sensitive data to unwanted parties	3	10	30	Severity of data leakage may vary on the data leaked. That's why it is important to maintain a level of security. Such as keeping the repositories used private and only invite group members. Data from the company and made for the company should be handled with care and precaution.
3	Lack of experience and knowledge	Lack of knowledge may causes postponements and work on defective technology since the group members are working on new technology to finish the project	3	8	24	Setting aside plenty of time for research. The group members should utilize their supervisor if such problems arises.
4	Loss of data/document		2	10	20	Backup frequently both cloud based and hard storage
5	Illness	Illnesses causing absences	2	7	15	Taking this into consideration in the planning phase
6	Lack of motivation	Overwork may lead group members to lose their motivation down the line	2	5	10	Group members should not overwork themselves and communicate transparently to each other

	Low consequences	Medium consequences	High consequences
Low Probability	6	5	4
Likely		3	
High Probability		2	1

6.2 Milestones and decision points

- The group must be finished with the project plan by 31st of January.
- The group must be finished with the design (mock-ups and project architecture) by 15th of February.
- The group must be finished with the first version of the code by 15th of April.
- The group must be finished with the testing by 1st of May.
- Project submission is on the 10nd of May.
- The group must be finished with the report writing by 20th of May.
- Project presentation is on the between 5th - 10th June.

References

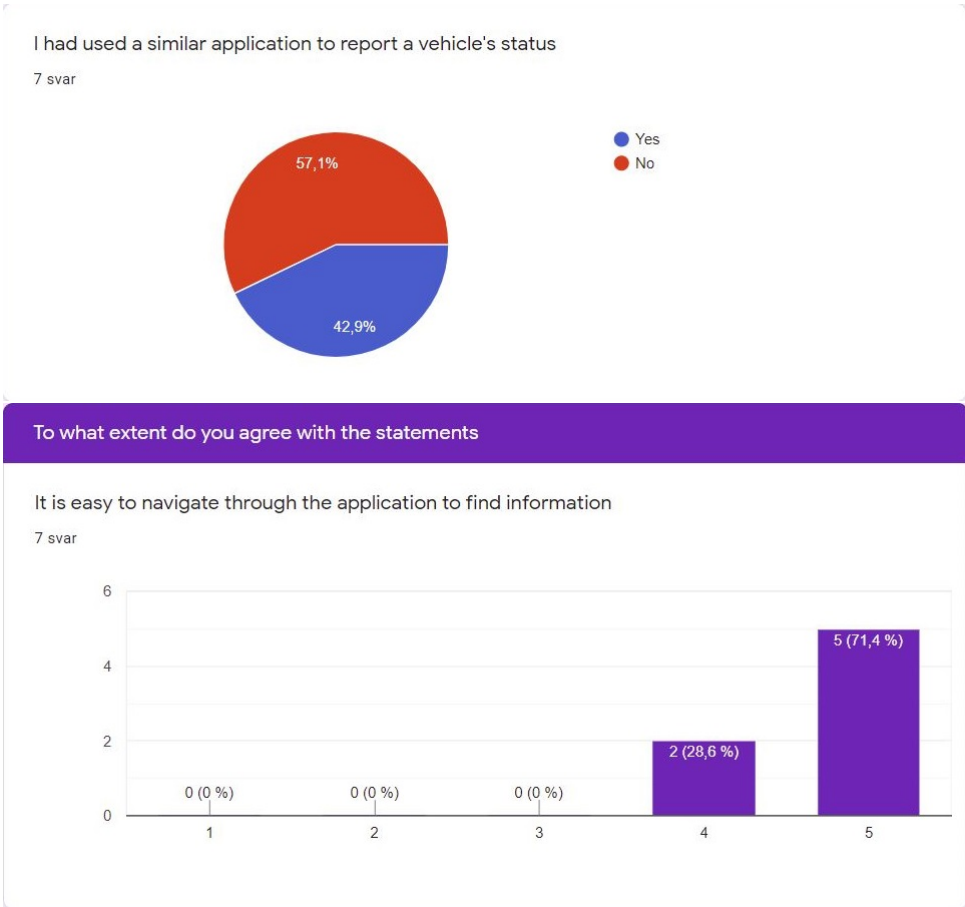
- [1] Integradet Cooperative Solutions. Assetfront. URL <https://assetfront.com/>.
- [2] Atlassian. Trello. URL <https://trello.com/>.
- [3] GanttProject. 2.8.11, 2020. URL <https://www.ganttproject.biz/>.

List of Figures

1	Gantt Diagram	7
---	-------------------------------	---

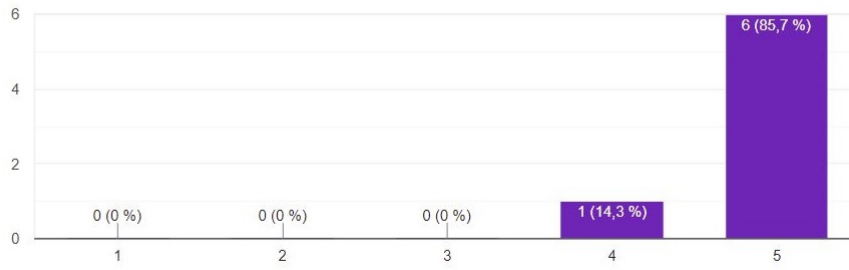
Appendix B

Survey questionnaire



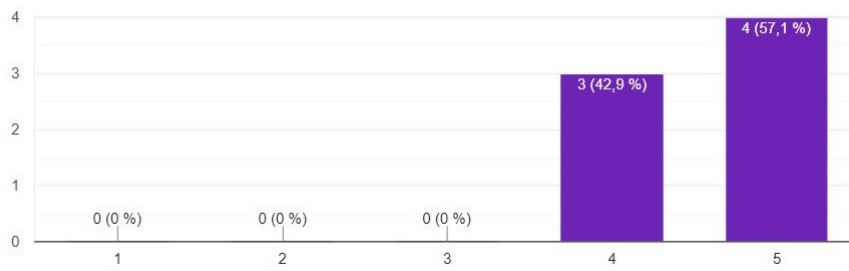
The application is easy to use in general

7 svar



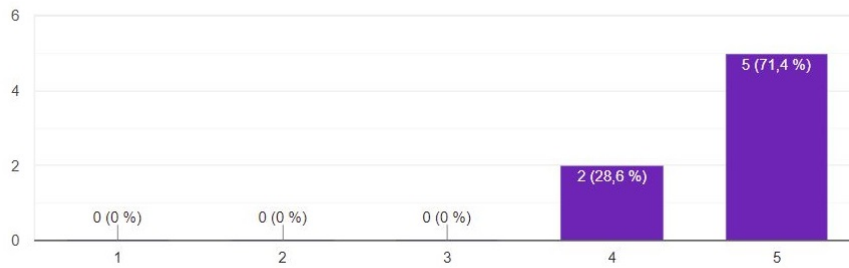
The application is a helpful tool and timesaving

7 svar



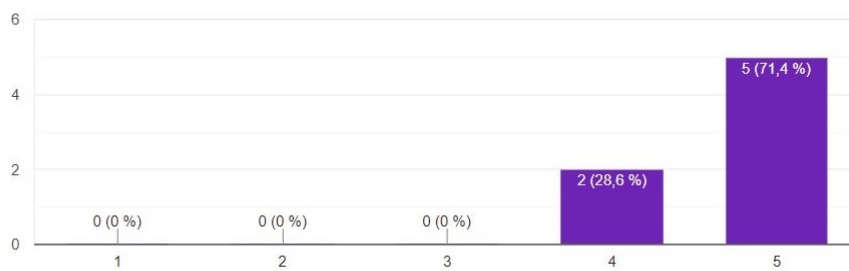
It is clear where you can download a previous Service/ TCR (Technical Condition Report) / Inspection 

7 svar



It is clear where you can send a Service/ TCR / Inspection

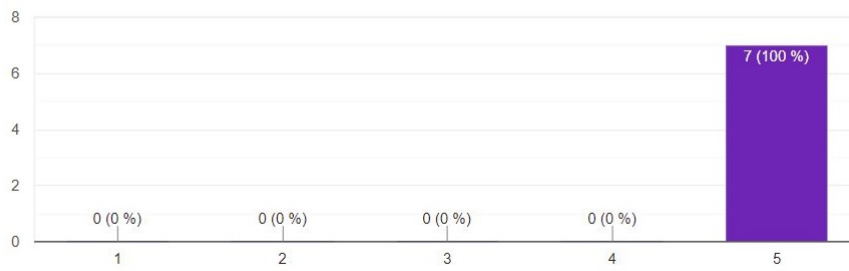
7 svar



The application is responsive on the screen size you are using



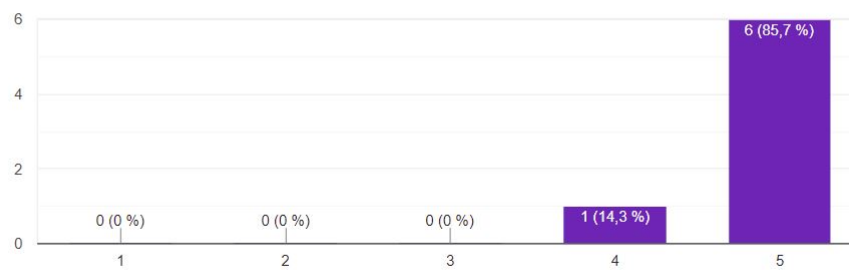
7 svar



The colors used in the application combine well and eye friendly



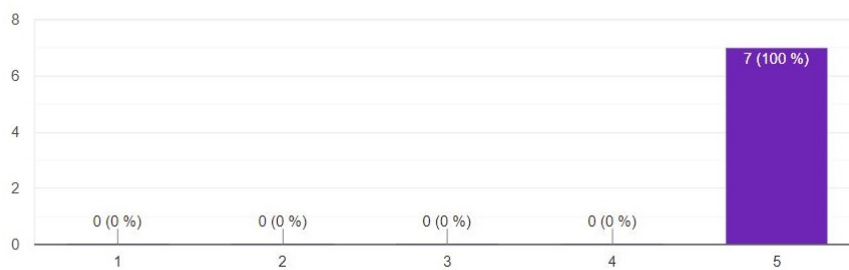
7 svar



The application is a good tool to improve the process of handling and documenting vehicle's status



7 svar



Feedback

Please list the feature/functionality you liked in the application.

4 svar

Mobil venlig, og nemt at bruge

Enkel, rask

QR code, TCR

TCR send, QR code, camera/picture possibility

Please list the feature/functionalities you did NOT like in the application and why you did not like them.

5 svar

The choice to click "QR CODE" do not open the product, it only puts in the VIN number, can irritate the first time you use the application. When you realize what the QR code is doing then its no problem anymore

At alle skal udfyldes

QR-kode bør åpne app og korrekt objekt direkte

Should maybe be clearer what the QR gives, you expect to get directly to the item. Help text would be enough.

None

Do you have any suggestions for us to improve the application?

7 svar

..

QR CODE needs to open the product

No

Qr kod kan öppna product direkt

QR og ikke-obligatoriske elementer i TCR

Remove the * in the check boxes in the TCR- only the general report should be required

None

Please add a feedback / comment here.

4 svar

Good job, its going to be very helpful

Umiddelbart rigtig godt product

Great work!

Great design, main features comes forward correctly

Appendix C

REST API Documentation

The document attached below (next page) is a REST-API documentation generated by Swagger and later exported as a PDF file.

AssetFront

This is the API for <https://assetfront.com>.

To use this API, you must have an active user with the required roles in AssetFront, or a registered OAuth2 client.

(You can not register this client yourself. Please contact ICS Partner AS to request access).

Endpoints marked with a padlock require authorization. Click the "Authorize"-button to log in.

More information: <https://headit.no>

Contact Info: post@headit.no

Version: V2.0.0

BasePath: /

All rights reserved

<http://apache.org/licenses/LICENSE-2.0.html>

Access

1. APIKey KeyParamName:api_key KeyInQuery:false KeyInHeader:true

Methods

[[Jump to Models](#)]

Table of Contents

[Lookup](#)

- [GET /asset/lookup/{serialNo}](#).

Lookup

[Up](#)

GET /asset/lookup/{serialNo}

Returns basic information for assets with corresponding Serial No or Object Ref (lookupAssetsUsingGET)

Path parameters

serialNo (required)

Path Parameter – serialNo

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Return type

array[array[Object]]

Example data

Content-Type: application/json

```
[ [ "{}", "{}" ], [ "{}", "{}" ] ]
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

OK

401

Unauthorized

403

Forbidden

404

Not Found

Models

[[Jump to Methods](#)]

Table of Contents

1. [Asset - Asset](#)
2. [Checkpoint - Checkpoint](#)
3. [Option - Option](#)
4. [ProductClass - ProductClass](#)
5. [TechnicalCondition - TechnicalCondition](#)

Asset - Asset

[Up](#)

mainImageUrl (optional)

[String](#)

productClass (optional)

[ProductClass](#)

resourceId (optional)

[String](#)

tcr (optional)

[array\[TechnicalCondition\]](#)

title (optional)

[String](#)

Checkpoint - Checkpoint

[Up](#)

id (optional)

[Long](#) format: int64

name (optional)

[String](#)

options (optional)

[array\[Option\]](#)

value (optional)

[Long](#) format: int64

Option - Option

[Up](#)

description (optional)

[String](#)

id (optional)

[Long](#) format: int64

value (optional)

[Long](#) format: int64

ProductClass - ProductClass

[Up](#)

id (optional)

[Long](#) format: int64

title (optional)

[String](#)

TechnicalCondition - TechnicalCondition

[Up](#)

checkpoint (optional)

[array\[Checkpoint\]](#)

id (optional)

[Long](#) format: int64

name (optional)

[String](#)

Appendix D

Technologies and Tools

Overleaf: Cloud-based L^AT_EX editor. The team used this tool to write this project.

Adobe XD¹: vector-based user experience design tool. The team used this tool to design mock-ups.

Git: Version control system designed to handle project of different sizes.

Bitbucket: Git-based repository. It is web-based.

WebStorm: Integrated Development Environment (IDE) for web, JavaScript and TypeScript development.

Node.js: Server-side platform, and a JavaScript runtime for building scalable and event-driven applications.

npm²: A package manager for JavaScript and Node.js.

Trello: A project management platform for task organizing.

JSON: is a lightweight, text-based, language-independent data interchange format [27].

Bootstrap: is a CSS framework to build responsive and mobile-first applications [17].

Sass³: Sass is a stylesheet language that is compiled to CSS [47].

¹<https://www.adobe.com/no>

²Node Package Manager

³syntactically awesome style sheets

Appendix E

Project assignment

Oppdragsgiver

Oppdragsgiver: Headit AS
Kontaktperson: Rune Kollstrøm
Adresse: Løvstadvegen 7, 2312 Ottestad
Telefon: 625 10 052
E-post: rune.kollstrom@headit.no

Bakgrunn

Headit AS utvikler unike fag- og innsiktsløsninger som hjelper deg til å jobbe enklere, og ta riktige beslutninger. Vi er et stort og innflytelsesrikt miljø med tverrfaglig kompetanse innen UX, data science, forretnings- og systemutvikling. Headit er et selskap som jobber sammen med våre kunder om å realisere strategi og mål. Hver dag skriver vi videre på vår unike historie. Vi har hovedkontor på Hamar, og er i dag 33 ansatte.

Lytte, forstå og løse!

Assetfront- Networked Lifecycle Machine Management (Bransjen sin facebook for maskiner) er en *Command & Control platform* for ulike typer transportkjøretøy, maskiner og verktøy (Hub, Dashboard, Exchange). Assetfront sitt dashboard gir brukerne full oversikt over sine maskiner på tvers av organisasjonen. Mer info om assetfront kan finnes på assetfront.com

Oppgaven - Assetfront Repair

«Assetfront Repair» skal bidra med å lette arbeidet for de som utfører vedlikehold og inspeksjoner på maskiner tilknyttet Assetfront. Systemet skal sørge for en enkel måte å oppdatere og tilknytte opplastet dokumentasjon til maskinene som er registrert i Assetfront.

Det ønskes at det lages et system der man enkelt kan registrere service eller inspeksjoner for maskiner, og laste opp vedlegg. Slik funksjonalitet finnes tilgjengelig i dag for brukere av Assetfront, men det er et ønske om kunne effektivisere denne prosessen, slik at brukere utenfor Assetfront kan gjøre denne type dokumentasjon tilgjengelig for Assetfront.

Oppgaven som er tenkt løst ved å utvikle en app/webseite der man kan laste opp service- og inspeksjonsmetadata sammen med vedlagt dokument. Dette skal kunne finnes igjen og importeres i Assetfront via et REST API, og knyttes til korrekt maskin.

Det må utvikles både en frontend- og backend-del til løsningen.

Dagens Assetfront kjøres i AWS, og benytter flere av AWS sine tjenester. Løsningen består av en Spring Boot-applikasjon, med tilhørende Angular-app og bruk av Material design.

Studentene vil få teknisk støtte fra Headit sine utviklere. En representant fra ICS Partner vil være produkteier, og vil sikre at oppgaven understøtter Assetfront sine behov og krav.



Oppgaven er egnet for to til fire utviklere, noe som også vil bidra til en innføring i Scrum-metodikk og team-samarbeid, og vil gi innsikt innen følgende områder:

- Backend
- Frontend
- UX
- Skyagringstjenester (vedlegg)
- Scrum
- REST API
- Kommunikasjon
- Programmering

Appendix F

Project agreement



Norges teknisk-naturvitenskapelige universitet

Vår dato 26/01-2020 Vår referanse

1 av 3

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

Headit as v/ Rune Kollstrøm (oppdragsgiver), og **Shamil Saidovitch Khumparov, Mustafa Abdullah, Amr Hamcho, Abdi Mohammad Bako** (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 11.01.2020 til 20.05.2021.

Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
 - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon, reiser og nødvendig overnatting på steder langt fra NTNU i Gjøvik. Studentene dekker utgifter for ferdigstilling av prosjektmateriell.
 - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.
3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.
4. Alle beståtte bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv NTNU Open.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.
6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.
7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.
8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.
9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.
10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.
11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.
12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Seyyed Ali Amir-shahi

Oppdragsgivers kontaktperson (navn): **Rune Kollstrøm**

Student(er) (signatur): Abdi Bako dato 29.01.2021
Mustafa Abdullah dato 29.01.2021
Sh. Khumbarov dato 29.01.21
Amr Hamcho dato 30.01.21

Oppdragsgiver (signatur): Rune Kollberg dato 27.01.2021

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato _____

Appendix G

Today's TCR

The document attached below (next page) is today's method to handle TCRs. After a user fills the form, it has to be scanned in the system.

Excavator Belt

Data

Internal Ref:

Title:

Location:

Description:

English/German:

Features

— Producer



Brand:

Unknown

Not applicable



Model:

Unknown

Not applicable



Year model:

Unknown



1st time registered:



Chassisnumber:

Unknown

Not applicable



Hours:

Not applicable



Milage (km):

Not applicable

Material

Transport information



**Transport dimension
(L/W/H) :**

Unknown

Not applicable



Transport weight (kg) :

Unknown


Not applicable

Inspection/Sertification




CE-marked

- **1: Yes**
- **2: No**
- Unknown**

 **Latest inspection:**

Unknown


Not applicable

 **Latest certification:**

Unknown


Not applicable

Hydraulics

 **Hydraulics:**

Unknown


Not applicable

 **Additional hydraulics:**

Unknown

Not applicable


Cabin

 **Cabin type:**

Unknown

Not applicable

Excavator details

 **Belts:**

Unknown

Not applicable



Tyres:

Unknown

Not applicable



Bucket:

Unknown

Not applicable



Bucket capacity:

Unknown

Not applicable



Undercarriage:

Unknown

Not applicable



Accessories:

Unknown

Not applicable

Availability



Status

-

1: Available

-

2: Ordered

-

3: Not Ordered

-

4: Delivered





Type of availability


-


1: Sale


- **2: Rental**
- **3: Transfer**


 **Deadline for interest:**


 **Available from:**

 **Available to:**

 **Price:**

 **Currency:**

 **Owner contact details:**

 **Comments / Terms for transfer:**



Buyer contact details:

Status

—
Operational status



Select

—

1: Operative

—

2: Spare

—

3: Maintenance

Service

—
Service updates



Latest service:



Hours / km latest service:



Cost:



Next service:

Inspection

—
Inspection



Inspection status

- **1: Approved**
- **2: Not approved**



Latest inspection:



Next inspection:



Comments:

Purchase

Purchase



Type of purchase

- **1: Stock**
- **2: Reseller**
- **3: End user**
- **4: Demo**




Order date:




Order status

- **1: Ordered**
- **2: Confirmed**
- **3: Received**

- **4: Delivered**
- **5: Cancelled**

 **Confirmed date:**


 **Estimated arrival week:**

 **Arrived date / week:**


 **End user contact details:**

Conditions

— Technical conditions report (TCR)

 **Date:**

General

 **Overall condition**

- **0: Object(s) working satisfactory when last used.**
- **1: Object(s) in working condition, but may have minor flaws**
- **2: Object (s) may have more extensive flaws or omissions. Inspection/repair is recommended.**

Not applicable

Hours



Hours:

Unknown

Function check



Start the engine

-

0: Correct display on instruments and warning lights

-

1: Instruments/light signalling fault

-

2: Engine does not start, neither with starting aid

Not applicable



Test all maneuvering gear

-

0: All maneuvering gear and levers function without slack. Brake, steering and gear change all function. No leakage

-

1: Maneuvering gear and levers do not function. Slack. Lacking brakes, steering or gear changing. Air or exhaust leakage

-

2: Critical maneuvering gear and levers do not function. Defect brakes, steering or gear/clutch. Larger leakages

Not applicable



Heating/Defrosting system

-

0: Heating / defrosting system working satisfactory

-

1: Heating / defrosting system has low effect

-

2: Heating / defrosting system is defect

Not applicable



Air Condition (A/C)

-

0: A/C for the driver and passengers appear satisfactory

-

2: A/C for the driver and passengers do not work

Not applicable



Parking brake

-

0: Parking brake working satisfactory

-

1: Parking brake not working

Not applicable

Engine

**Oil level**

- **0: Level between min and max mark**
- **1: Below min but shows on dipstick**
- **2: Oil level not showing on dipstick**
- Not applicable**

**Coolant**

- **0: Minus 35C or lower. Correct fluid levels. No visible leakages**
- **1: Unknown, or higher than minus 35C. Tubes are rotten. Drive belts worn**
- **2: Visible leakages**
- Not applicable**

**Exhaust leakages**

- **0: No visible leakage**
- **2: Visible leakage**
- Not applicable**

Fuel system**Mud container for water and impurities**

- **0: No impurities**
- **1: Water and dirt in mud container/fuel filtre**
- **2: Water in glass/filtre during winter conditions**
- Not applicable**

**Water separator**

- **0: Water separator working satisfactorily. Drainage and heating works**
- **1: Water separator does not work. Can not be drained. Faulty heating**
- Not applicable**

**Leakages**

- **0: No visible leakage**
- **1: Visible leakage**
- Not applicable**

Electric system



Battery control

- **0: Battery in place and OK**
- **1: Flat battery, rusty terminals**
- **2: Defect/ missing battery/batteries**
- Not applicable**



Dynamo with bolts

- **0: Dynamo with bolts in order**
- **1: Dynamo does not give satisfactory charge power**
- **2: Dynamo is defect**
- Not applicable**



Starter

- **0: Starter mounted. Connection cables are without irr and mounted**
- **1: Starter loose. Loose connection wire**
- Not applicable**

Sight/Lights



Wind screen

- **0: Wind screen has little wear, and has no cracks or chips needing repair**
- **1: Wind screen needs repair**
- **2: Wind scren has cracks and/or major wear. Needs replacing**
- Not applicable**



Main-/park-/stop lighs/blinkers/horn/mirror and window wipers

- **0: All lights working. Horn working. Mirrors complete and clean. Window wipers cleaning satisfactory**
- **1: One or more lights not working. Mirror broken, cracked or missing. Wiper blades defect**
- Not applicable**

Body/Interior



Body damage and corrocive

- **0: No visible damage or corrocive**

- **1: Minor damage, cracks or deformations. Corroctive attack on painted surface**
- **2: Major damage and deformations. Corroctive break through on bodywork**
- Not applicable**



Paint

- **0: No visible damage. Satisfactory surfaces**
- **1: Minor damages, stains and oxidations/dullness**
- Not applicable**



Condition - Interior/Furnishing

- **0: Interior/furnishing/seats are in satisfactory condition**
- **1: Defects/ worn but still usable**
- Not applicable**

Truck body/Frame



Corrosion and damages

- **0: Frame without corrosion and damage**
- **1: Minor corrosion and/or damage on frame**
- **2: Frame has major corrosion or damages**
- Not applicable**

Working hydraulics



Oil level

- **0: Level between min and max mark**
- **1: Oil level below MIN but registered**
- **2: Oil level cannot be registered. Leakages**
- Not applicable**



Hydraulic pump

- **0: Hydraulic pump working and properly fastened to frame**
- **2: Defect hydraulic pump**
- Not applicable**



Funktion

- **0: Works evenly, no sagging**

- **1: Works unevenly and/ or sagging**
- Not applicable**



Hydraulic tank and leakage hydraulic cylinders

- **0: No leakages**
- **2: Major leakage, breach or damages**
- Not applicable**



Additional hydraulic

- **0: Additional hydraulic function properly**
- **2: Additional hydraulics do not work**
- Not applicable**

Enhancement



Chains

- **0: Chains suitable for all wheels are present and residual value more than 50%**
- **1: Chains suitable for all wheels are present, and residual value between 20 and 50%**
- **2: One or more chains are missing or worn**
- Not applicable**



Fire Extinguisher

- **0: Present, correct size, and function tested >3 year**
- **1: Function tested < 3 years**
- **2: No Fire Extinguisher**
- Not applicable**

Hydraulic transmission



Oil level

- **0: Level between min and max mark**
- **1: Oil level below MIN but registered**
- **2: No oil is registered. Leakages**
- Not applicable**



Leakages

- **0: No leakage**

- **2: Leakage**
- Not applicable**

Disperse gear box



Leakages

- **0: No leakage**
- **2: Leakage**
- Not applicable**

Load-/Digging equipment



Cracks, attrition or deformations

- **0: No cracks or deformations. Normal attrition**
- **1: Minor cracks, damages and/or deformations. Extensive attrition, but still repairable**
- **2: Equipment worn out**
- Not applicable**

Shear/Tear equipment



Cracks, attrition or dformations

- **0: No cracks or deformations. Normal attrition**
- **1: Minor cracks, damages and/or deformations. Extensive attrition, but still repairable**
- **2: Equipment worn out**
- Not applicable**

Sustainers



Sustainers

- **0: Sustainers working without damages**
- **1: Minor damages or deformations. Still working satisfactory**
- **2: One or more sustainer not working**
- Not applicable**

Steering brake



Functional Control of steering brake

- **0: Steering brake is functioning satisfactorily**
- **2: Steering brake does not work**
- Not applicable**

Belts



Supporting wheels and supporting rolls, sprocket- and stretch wheel for slack

- **0: No slack**
- **1: Slight movement**
- **2: Major movement. One or more wheels are missing**
- Not applicable**



Sprocket wheel and attaching bolts

- **0: Little or no wear. No loose or missing bolts.**
- **1: Wear but teeth are not sharp.**
- **2: Belt skipping teeth. Crack or deformation in crown. Bolts missing.**
- Not applicable**



Belt tension

- **0: More than 50% left of belt tension**
- **1: Belt tension has 20-50% left**
- **2: Belt tension finished**
- Not applicable**



Belt plate and bolts - steel belts

- **0: No loose bolts or missing. All belt plates in place**
- **2: Loose or missing belt plates and/or bolts**
- Not applicable**



Belt chains

- **0: No defect cases / chains**
- **2: Defect belt chains**
- Not applicable**



Belt wearing/damage on rubber belts

- **0: No cracks go through**
- **1: Smaller damages/cracks**
- **2: One or more belts are defect**
- Not applicable**

Auction Details

Sales Details

General sales details



From date:



Note! Closing date must be Monday - Thursday.

To date:



Reserve price:



Updated by the auction system. Price ex. tax and markup



Updated by the auction system.

Protected Information

Protected information

General



Comments:



Improvements costs:

Economy



Rest value:

Unknown



Acquisition value:



Acquisition year:

Valuation



Latest valuation value:



Latest valuation dato:



Latest valuation performed by:



Comments valuation:

Appendix H

TCR JSON

```

"resourceId": "0f3c5185-fd32-4bdf-9324-c5be2ee25723",
"productClass": {
  "id": 29,
  "title": "Garden- and Landscaping equipment"
},
"tcr": [
  {
    "id": 2080,
    "name": "Materielltilstand ",
    "checkpoint": [
      {
        "id": 2242,
        "name": "Generell vurdering.",
        "value": 0,
        "options": [
          {
            "id": 2315,
            "value": 0,
            "description": "Salgsobjekt (-ene) fungerte tilfredsstillende ved siste bruk."
          },
          {
            "id": 2316,
            "value": 1,
            "description": "Salgsobjekt (-ene) er i bruksmessig stand, men kan ha mindre feil."
          },
          {
            "id": 2317,
            "value": 2,
            "description": "Salgsobjekt (-ene) kan ha større feil. Inspeksjon/reparasjon anbefales."
          }
        ]
      }
    ]
  }
]
}]

```

Appendix I

TCR PDF

Company: Hundalen As

Name: John Thomas

Date: 8.5.2021

Email: john.tom@gmail.com

Phone Number: 94000011



Materielltilstand

Generell vurdering.

- Salgsobjekt (-ene) fungerte tilfredsstillende ved siste bruk.
- Salgsobjekt (-ene) er i bruksmessig stand, men kan ha mindre feil.
- Salgsobjekt (-ene) kan ha større feil. Inspeksjon/reparasjon anbefales.

Alt.: Elektrisk.

Batterikontroll.

- Batteri tilstede og i orden.
- Utladet batteri. Irrede poler.
- Defekt(e)/manglende batteri(er).

Lader.

- Lader tilstede og gir tilfredsstillende ladestrøm.
- Lader ikke.

Motorfunksjon.

- Starter og går tilfredsstillende.
- Starter ikke.

Alt.: Forbrenningsmotor

Kjølevæsknivå.

- Mellom min og max merke.
- Under min, men synlig.
- Væskebeholder er tom. Lekkasje.

Kjølevæskens frysepunkt.

- Minus 35 grader C eller lavere.
- Ukjent, eller høyere enn minus 35 gr C.

Lekkasjer.

- Ingen synlige lekkasjer.
- Synlige lekkasjer. Drivstoff siver ut.

Oljenivå.

- Mellom min og max merke.
- Under min, men vises på peilestaven.
- Oljestand vises ikke på peilestav.

Start motoren.

- Riktig utslag på instrumenter og varsellamper.
- Instrumenter/lamper varsler feil.
- Starter ikke, heller ikke med hjelpestart.

Funksjonskontroll.

Funksjonstest for styrefunksjoner, lift opp/ned, ut/inn, sving. Kjøring fremover/bakover, samt belastning/momentprøver.

- Alle funksjoner fungerer uten feil. Ingen lekkasjer.
- Mangler ved en eller flere av funksjonene. Lekkasjer.
- En eller flere av funksjonene virker ikke.

Prøv alle manøveringsorganer.

- Riktig utslag på instrumenter og varsellamper. Alle manøveringsorganer fungerer uten feil.
- Instrumenter/lamper varsler feil. Et/flere manøveringsorganer fungerer dårlig.
- Manøveringsorganer av betydning fungerer ikke.

Hydraulikk.

Hovedsentral oppe/nede.

- Begge hovedsentralene er uten lekkasjer og virker tilfredsstillende.
- Hovedsentral oppe eller nede har mangler/lekkasjer av betydning for funksjonaliteten.

Oljenivå.

- Nivå mellom min og max.
- Oljenivå under min, men synlig.
- Oljestand kan ikke registreres. Lekkasjer.

Elektrisk.

Brytere/kabler.

- Fungerer og er uten synlige skader.
- Skadede kabler. Ødelagte bryterhendler.
- Kabelbrudd og/eller defekte brytere.

Manøverpanel oppe/nede.

- Begge manøverpanelene fungerer tilfredsstillende.
- Ett eller begge manøverpanelene fungerer ikke.

Ramme/understell.

Ramme/fundament for skader.

- Uten synlige skader.
- Mindre skader, sprekker eller deformasjoner.
- Større skader og/eller deformasjoner.

Støttebein.

- Støttebein fungerer uten skader.
- Mindre skader eller deformasjoner. Fungerer fortsatt tilfredsstillende.
- Ett eller flere støttebein fungerer ikke.

Kurv/Plattform.

Innfesting av kurv/plattform.

- Tiltrukne bolter.
- Løse og/eller manglende festebolter.

Rekkverk og port med lukkemekanisme.

- Uten synlige skader.
- Deformert rekkverk.
- Manglende port eller defekt lukkemekanisme.

Sving med låsing.

- Fungerer tilfredsstillende.
- Defekte betjeningsorganer/ brytere for sving og/eller stopp.

Sakser/Bommer

Innfestinger.

- Tiltrukne bolter.
- Manglende festebolter.

Sveisekontroll

- Uten synlige skader.
- Mindre skader, sprekker eller deformasjoner. Større slitasje, men fortsatt reparerbart.
- Større skader og/eller deformasjoner.

Sikkerhetskomponenter.

Klembeskyttelse.

- Fungerer tilfredsstillende.
- Defekt eller manglende klembeskyttelse.

Nødsenk/sving/stopp.

- Fungerer tilfredsstillende.
- Defekte betjeningsorganer/ brytere for nødsenk/ sving/ stopp.

Vater/nivellering.

- Fungerer tilfredsstillende.
- Defekte indikatorer/libeller.

Bremser.

Bremsesystem.

- Alle hjul har tilfredsstillende bremsevirkning.
- Ett eller flere hjul har ikke tilfredsstillende bremsevirkning.
- Tilhengerbremsene virker ikke.

Lys.

Lys.

- Alle lys virker tilfredsstillende.
- Ett eller flere lys virker ikke.

Lasteplan/karmer.

Lasteplan med karmer, luker og låser.

- Uten synlige skader.
- Mindre skader, sprekker eller defekter.
- Defekte låser.

Trekkrok/sikkerhetslenker.

Sikkerhetslenker.

- Begge sikkerhetslenkene er tilstede.
- En eller begge sikkerhetslenkene mangler.

Trekkrok.

- Ingen synlige skader på trekkrok.
- Skader på trekkrok eller krokens innfesting i rammen.

Hjul/Dekk.

Luftrykk.

- Korrekt luftrykk.
- Tydelig feil luftrykk.
- Punkterte hjul.

Ramme.

Rust og skader.

- Rammen/bærende konstruksjon er uten synlig rust eller skader.
- Rammen/bærende konstruksjon har mindre rustangrep og/eller skader.
- Rammen/bærende konstruksjon har betydelige rustangrep og skader.

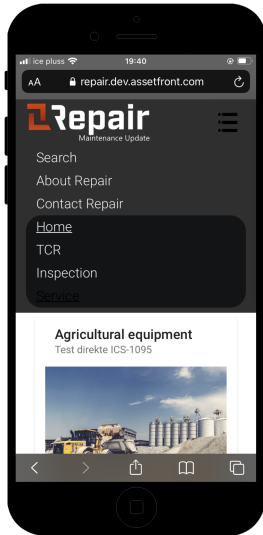
Selvgående funksjon

Selvgående funksjon.

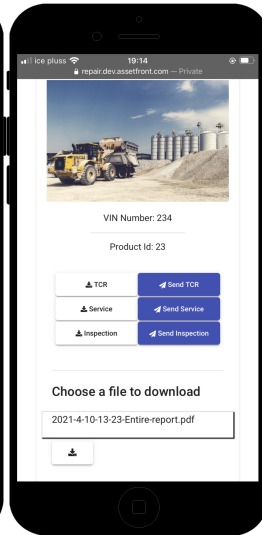
- Fungerer tilfredsstillende.
- Funksjonen virker ikke.

Appendix J

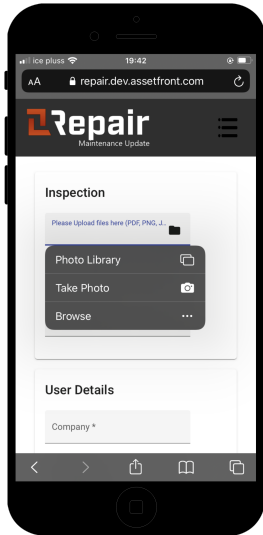
Assetfront on an Iphone 6



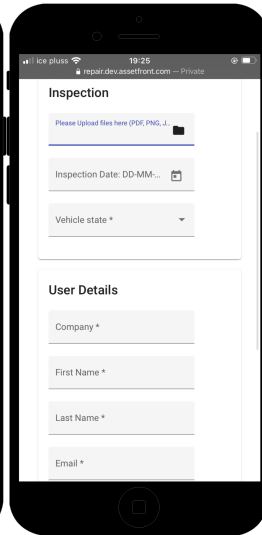
(a) Sub header



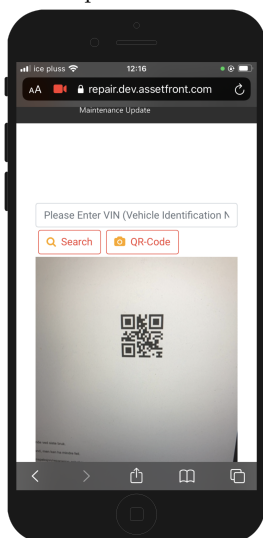
(b) Home and Download



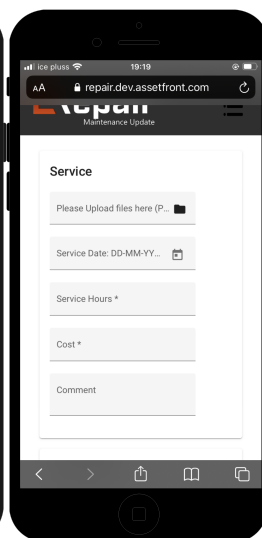
(c) File upload in Service and Inspection



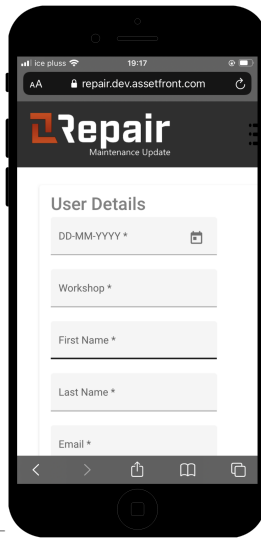
(d) Inspection and User details



(e) Scanning QR-code



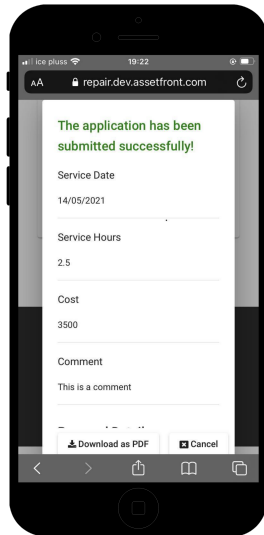
(f) Service Data



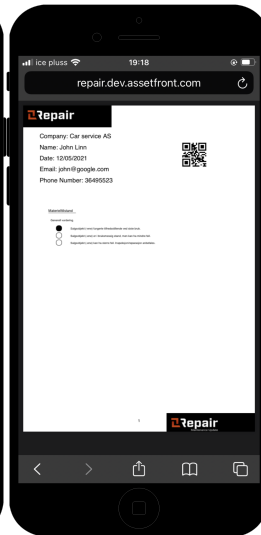
(a)
User
de-
tail



(b) Service PDF



(c) After submission



(d) TCR PDF File

Appendix K

Implementation of TCR

```
1 <div *ngIf="onSuccessfulSearch()">
2   <mat-accordion>
3     <div *ngFor="let item of tcrCopy.tcr; let indexTcr = index">
4       <mat-expansion-panel>
5         <mat-expansion-panel-header matRipple [matRippleColor]="
6           color" >
7           <mat-panel-title fxLayoutAlign="space-between center end"
8             >
9               <div>{{tcrCopy.tcr[indexTcr].name}}</div>
10              <div *ngFor="let n of tcrCopy.tcr[indexTcr].checkpoint
11                let in = index">
12                {{allCheckPointSelected(indexTcr)}}
13                <div *ngIf="allSelected && allCheckPoint.length >=
14                  tcrCopy.tcr[indexTcr].checkpoint.length">
15                  <mat-icon *ngIf="tcrCopy.tcr[indexTcr].
16                    checkpoint.length-1 === in">
17                    check_circle_outline
18                  </mat-icon>
19                </div>
20              </div>
21            </mat-panel-title>
22            <ng-template matExpansionPanelContent>
23              <h1 *ngFor="let checkp of tcrCopy.tcr[indexTcr].
24                checkpoint;"
25                let indexCp = index">
26                <mat-card class="card" id="card">
27                  <mat-card-content id="body">
28                    <h1 id="subTitle" >{{tcrCopy.tcr[indexTcr].
29                      checkpoint[indexCp].name}}
30                    </h1>
31                    <h2 *ngFor="let op of tcrCopy.tcr[indexTcr].
32                      checkpoint[indexCp].options;"
33                      let value = index" class="value">
34                    <div class="row" id="description">
35                      <div class="col-lg-1">
36                        <mat-radio-button
37                          value="value"
38                          labelPosition="after"
39                          (change)="updateValue(value, indexTcr,
40                            indexCp)"
41                          color="primary"
42                          name="{{tcrCopy.tcr[indexTcr].checkpoint[
43                            indexCp].name}}">
44                          {{tcrCopy.tcr[indexTcr].checkpoint[
45                            indexCp].options[value].value}}
```

```

37         </mat-radio-button>
38     </div>
39     {{getNewlyFilledTcr()}}
40 <div>{{allCheckPointSelected(indexTcr)}}</div>
41     <div class="col-lg-7">
42         {{tcrCopy.tcr[indexTcr].checkpoint[indexCp
43             ].options[value].description}}
44     </div>
45 </h2>
46 </mat-card-content>
47 </mat-card>
48 </h1>
49 </ng-template>
50 </mat-expansion-panel-header>
51 </mat-expansion-panel>
52 </div>
53 </mat-accordion>
54
55 <mat-card-actions>
56     <div fxLayoutAlign="center">
57         <button mat-button
58             id="back"
59             (click)="toHome()"
60             mat-raised-button
61             color="warn">
62             <i class="fa fa-arrow-left"></i>Back</button>
63
64         <button [disabled]="!atLeastOneSelected" mat-button
65             id="next"
66             (click)="personData()"
67             mat-raised-button
68             color="primary">
69             Next<i class="fa fa-arrow-right"></i></button>
70     </div>
71 </mat-card-actions>
72 </div>

```

Listing K.1: TCR Component

Appendix L

Meetings with The Company and Product Owner

Date	Participated	Description
11. Nov	Headit(Kristiansen and Blakstad), Assetfront(Bjerkemo), Development Team	Presentation of assignment
13. Jan	Headit(Kristiansen and Blakstad), Assetfront(Bjerkemo), Development Team	Detailed project introduction
22. Jan	Headit(Kristiansen and Blakstad), Assetfront(Bjerkemo), Development Team	Detailed project introduction
25. Jan	Assetfront(Bjerkemo), Development Team	Detailed description of functionalities and design
01. Feb	Headit(Kristiansen and Blakstad), Assetfront(Bjerkemo) Development Team	Document signing and technical API access
09. Feb	Assetfront(Bjerkemo and Kristin), Development Team	Access to existing platform(vendo)
16. Feb	Headit(Kristiansen and Blakstad), Assetfront(Bjerkemo), Development Team	Presentation of Mock-ups and Status report
05. Mar	Headit(Kristiansen and Blakstad), Assetfront(Bjerkemo), Development Team	Status Report
24. Mar	Headit(Kristiansen and Blakstad), Assetfront(Bjerkemo), Development Team	Status Report and Technical help
14.Apr	Headit(Blakstad), Development Team	Technical Help
21.Apr	Headit(Blakstad), Development Team	Tecnical Help
27. Apr	Headit(Kristiansen and Blakstad), Assetfront(Bjerkemo), Development Team	Last presentation before user test
3. Mai	Assetfront employees	User Test
10. Mai	Headit and Dvelopment team	Project delivery

