Beumer Site-report tool

Pieter Schreurs 318775

HBO student Creative Media & Game Technologies at Saxion Enschede

Company Advisor: Chiel Migchelbrink

First Assessor: Yiwei Jiang

Second Assessor: Daniel Valente de Macedo

Table of contents

1. Introduction

This is a report for the module Graduation of the HBO study at Saxion called Creative Media and Game Technologies (CMGT).

This assignment came to be at the request of BEUMER group Benelux. They are a company who develops conveyor systems for sorting packages. These conveyor systems can fail sometimes due to parts crashing or packages getting stuck. These failures have to be reported and repaired by mechanics. Reporting of the failure is essential to the company due to liability contracts with their clients, however this task is not always done correctly or even forgotten at times. Following this problem they require a solution in order to make it easier to create these reports and incentivise the mechanics to fill in the reports correctly.

1.1. Company

This project involves two companies, the first company is Bens consultancy, they are a software consultancy company who support customers who arrive with problems and provide a custom software related solution. They specialize in improving business processes by developing web or app solutions. They are a company of 7 employees and all the projects they do are for external clients.

One of these external customers is called the BEUMER Group. They are a large multinational with 5,400 employees in more than 70 countries and offer intralogistics solutions for conveying, loading, palletising, packaging, sortation and baggage handling (BEUMER 2023). The branch called BEUMER Group Benelux built the conveyor belts for the post services of the big postal companies in the Netherlands. They also built the conveyor belts for a large flight company to handle the suitcases. These systems can break or jam, in that case there needs to be a mechanic who will repair or replace the parts. In case this happens, a mechanic has to create an incident report which was a long process. Bens consultancy has helped with this problem by creating the app called Mybeumer, with this app you can scan a QR code by scanning the package the mechanic can fill in the corresponding incident report on his mobile phone.

1.2. Parties and individuals

Bens consultancy – The internship company which provide the tutorage and technical knowledge in order to assist me with this project. Their interests are selling a product to the client.

BEUMER Group Benelux – The client which has a problem and are in need of a solution. They provide me with the requirements of the project. Their interests are improving their processes as a result of this project.

Chiel Migchelbrink – My company tutor who is also the director of Bens Consultancy. He will review most of my work and give me new insights in how to problem solve. His interests are making a more complete product for the client BEUMER.

Steven Bakker – The co-director together with Chiel, he is in charge of communication between Bens and BEUMER. His interest are making a more complete product for the client as well.

Wim Spekking – He is the Director Operational Excellence of BEUMER Group Benelux. He is responsible for the vision and requirements of this project, and the contact person at BEUMER. His interests for this project is to optimise the work processes of the engineers in order to problem solve more efficiently.

Harry Emo – He is responsible for quality assuring all the site-reports at BEUMER. After his quality assurance he submits them to the clients of BEUMER for liability reasons. The final goal is to make his life a lot easier.

1.3. End users

The project to be developed is to be used by **mechanics** on the workfloor. These mechanics will use this application in combination with a smart-device to report site-reports. The site-reports will be stored in a database. These site-reports then have to be quality graded and approved or denied by a **quality assurer**.

1.4. Objective

This project has multiple goals and these goals are intertwined. The first goal is to make it easier for the engineer to fill in a site report after an incident on the job site. The second goal is to make the generated site-report's quality high enough in order to pass the quality control from the quality assurer. The third goal is to make sure the site-reports are all based on facts and measurable parameters.

2. Often used terms

In this report there will be some terms which are specific to this case which will be talked about quite frequently. These terms are explained here for ease of read.

Mybeumer – A reporting tool developed by Bens consultancy for BEUMER to make reports for the customers of BEUMER in order to inform them about incidents and maintenance, and whose liability it is.

Incident report – An incident report is a report a mechanic does when the conveyor system has a jam or gives a fault message in which the conveyor system stops and needs interfering.

Site report – A site report is a collection of incident reports with high enough severity that it severely hinders the uptime of the conveyor belt for that day.

Realwear glasses – Official name is the Realwear HMT-1, these are accessory glasses which the client supplied. These glasses only work on voice control and were chosen for their durability on sturdiness.

3. Define

In the defining stage, the goal and problems are written down in a clear and concise way in order to make the goal of the assignment clear.

3.1. Problem statement

The company BEUMER wants to improve their processes by correctly gathering information from errors or jams in their conveyor system. site-reports which are filled in by their mechanics. These site-reports are often not filled in correctly if at all. In order to solve problems of their workflow, these reports need to be of a certain quality and be filled in consistently every time an error occurs.

3.2. Research goals

The mechanic can fill in a site-report faster and more convenient by using a smart glass.

The mechanic creates a site-report of a higher quality and completeness by gamifying this process.

3.3. Main question

How can the goal of improving the quality and quantity of filing site-reports by mechanics be met by developing an application for a smart device and gamifying the rewards at sorting depots in order to get a more complete overview of incidents?

3.4. Sub questions

Can the creation of a site-report be more streamlined and quicker with a smart device on the work floor? If so, which smart device is the most suited for this?

Which medium of information is needed at minimum in order to create a site-report and can the supplied Realwear HMT-1 provide these mediums?

What are the important UI/UX design decisions in order to develop a user friendly experience for a AR/VR device?

4. Scope

The scope of this project is to supply the company BEUMER with a proof of concept project which shows the possibilities of using smart-devices to insert contents of a site-report into a database.

4.1. Deliverable

This project involves a deliverable prototype, the prototype has to contain the core functionalities in order to be rolled out as a pilot. The core functionalities are inputting text/speech, taking pictures and filling in lists.

4.2. Inclusion

The application is able to create a site-report with all the now deemed necessary items. These site-reports are stored somewhere persistent.

4.3. Exclusion

The prototype will not have every feature implemented, however the proof of concept must serve as proof that every feature can be implemented. Item recognition by QR codes are

The pilot will not be able to recognise every single component of a site, since this can there can be up to 1000 different components.

4.4. Assumptions

The smart glass is provided by the client and the mechanics can be trained to use this device.

4.5. Constraints

The application is set up to be a pilot in order to see if this working method works. It is not possible to gather a full report of insights on a particular site.

5. Ideation

The problem of the client is defined and the research goals are formulated. In order to create a prototype there are a few decisions which have to be made. Some of these decisions can be thought about beforehand.

5.1. Input method

One of these decisions is the way an user can input information in the system. Depending on the type of information is required there can be multiple ways of inputting it.

The device the client has provided is the Realwear HMT-1, however the hardware components can be swapped to another device if deemed necessary. Since the device has to be used in a working place with lots of noise from all the machines whirring, if speech to text is used it should only pick up the actual voice input.

5.2. SWOT

This SWOT analysis is for determining which input device to use.

Realwear HMT-1, all the specifications are gathered from their website specification page (Realwear HMT-1 2023) it has a price of €1669,-. This device is already provided by Beumer.

Strength	Weakness
Voice input is filtered from loud noises making sure all voice input is free from noise. Native QR scanner and barcode scanner.	No buttons for input, everything needs to be a voice command. Has a small projector as screen, slightly uncomfortable to focus sight on.
Opportunities	**Threat**
Great battery 3250 mAh, can be used a full shift of 10 hours without charging. Has a plugin for Unity via a SDK, since I'm more experienced in Unity. Has native Android support.	Relatively unknown device, less documentation online. The OS on the device itself is less customisable, you can only build a layer on top of it. Mechanics has to be trained in order to use a relatively unknown object.

Another candidate is the google enterprise glasses 2 for its reasonable price of €999- which is catered for businesses as well, all specifications are gathered from their website page (Google glass 2023).

Strength	Weakness
It has a touch input for more variable ways of handling input. Uses AR (augmented reality) for on top projection.	Natively no QR or barcode scanner. Support ends at 15 September, 2023
Opportunities	**Threat**
Great documentation for further development and lots of resources/libraries available online.	No Unity support Low battery 800 mAh so the device has to charge more often. Mechanics has to be trained in order to use a relatively unknown object.

Another option would be to issue a smartphone to the mechanic on site. For the sake of the SWOT analysis a flag ship smartphone will be used which uses android as a platform. The Samsung Galaxy S23 is such a flag ship smartphone at a cost of €799,- (Samsung 2023)

Strength	Weakness
Native keyboard for text input. Great Camera a value of 50MP.	Hands have to be occupied filling in the forms via the phone. Site report can not be filled in while working on the machine.
Opportunities	**Threat**
Great documentation for further development and lots of resources/libraries available online. Good battery at 3900 mAh with an indicated battery life of 22 hours. An assumption can be made that almost everyone knows how to operate a phone.	Pure android development.

Conclusion:

The Realwear HMT-1 will be used for development, this because it's the preferred choice of the customer and for its great noise cancelling microphone in a loud work environment. This project is set up as a pilot in order to test the capabilities of this device. The API can be set up as such that it does not matter which device is used as input.

The next SWOT analysis is about which programming language will be used in order to achieve the end project. As the device is an android device, the first option is java.

Strength	Weakness
Native android support. Rapid prototyping, by drag and dropping.	My experience in Android is pretty much non-existent.
Opportunities	**Threat**
The use of implementing WearHF metadata in order to optimize UI for voice control.	Discrepancy of front-end and back-end languages, since the back end will be using .Net

Another option will be using the Unity plugin provided by Realweal (Realwear Developer 2023). This allows the use of the Unity Engine.

Strength	Weakness
C# Experience is far greater than Java. Great control of adding voice commands.	Non-native build for the Realwear device.
Opportunities	**Threat**
The back-end can be conform to the code dependencies of the front-end.	The by Realwear managed C# plugin support can end, resulting in an outdated product.

Conclusion:

For development of the project, Unity Engine will be used for the reason that it can do anything that Android development can do for this project, if you include the plugin from Realwear. The level of knowledge of the used language far outweighs any threat of an outdated plugin.

6. Design

During the design phase the flow of the software is described by sequence diagrams, flow charts and class diagrams. This in order to create a full design of the prototype before beginning development. The general sequence work method describes what the intended use of the prototype is and how it fits the goal of solving the problem at BEUMER group. The prototype is named Beumer reporting tool in this example. It will be the implementation of filling in a site-report to a quality assurer.

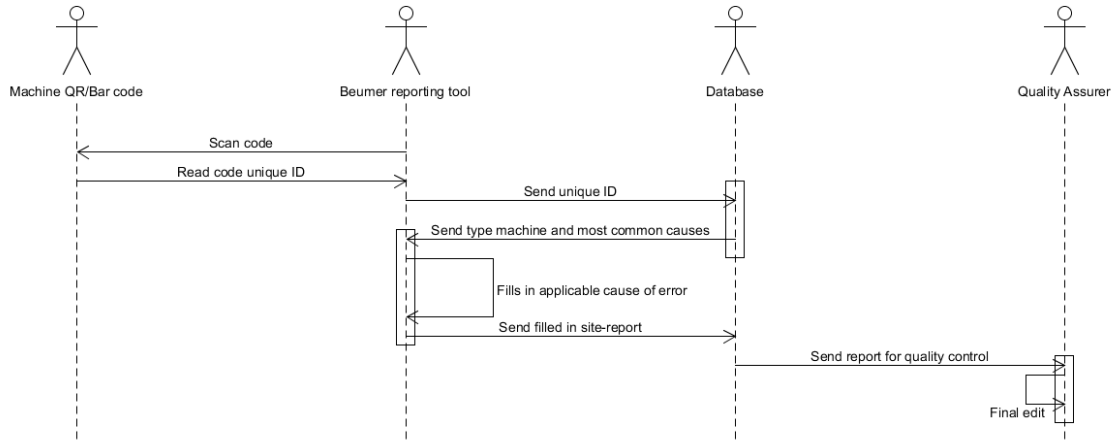


Figure 1 General sequence work method

The flowchart above shows the workflow of creation of a site-report. First the engineer takes the Realwear glasses with the Beumer reporting tool to a machine. He/She scans the machine's QR code and in a database a customised list appears for this type of machine. This customised list is a collection of common errors on the machine. The engineer then fills out the applicable error types, pictures and explanations and then the reporting tool sends the data to a database. The site-report is then sent to the quality assurer who will approve or edit the final site-report.

The following flow chart is the sequential order of steps in order to initiate a site-report. If these conditions are met then the mechanic will create a site-report. The conditions are either a broken part of a cost of higher than €1000,- or an operational standstill of more than 30 minutes.

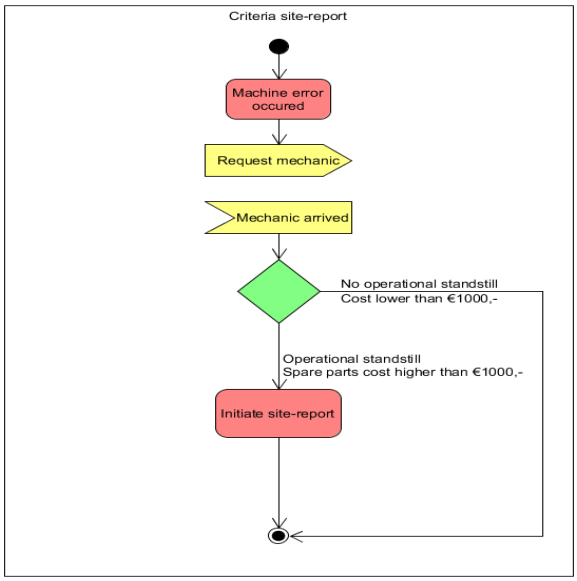


Figure 2 Flow chart criteria site-report

6.1. Site-report requirements

I received a few example site-reports of Beumer, although there were no clear coherent values used everywhere and a couple of redundancies. I made a class diagram in order to shed some insight into what a site-report has to contain.

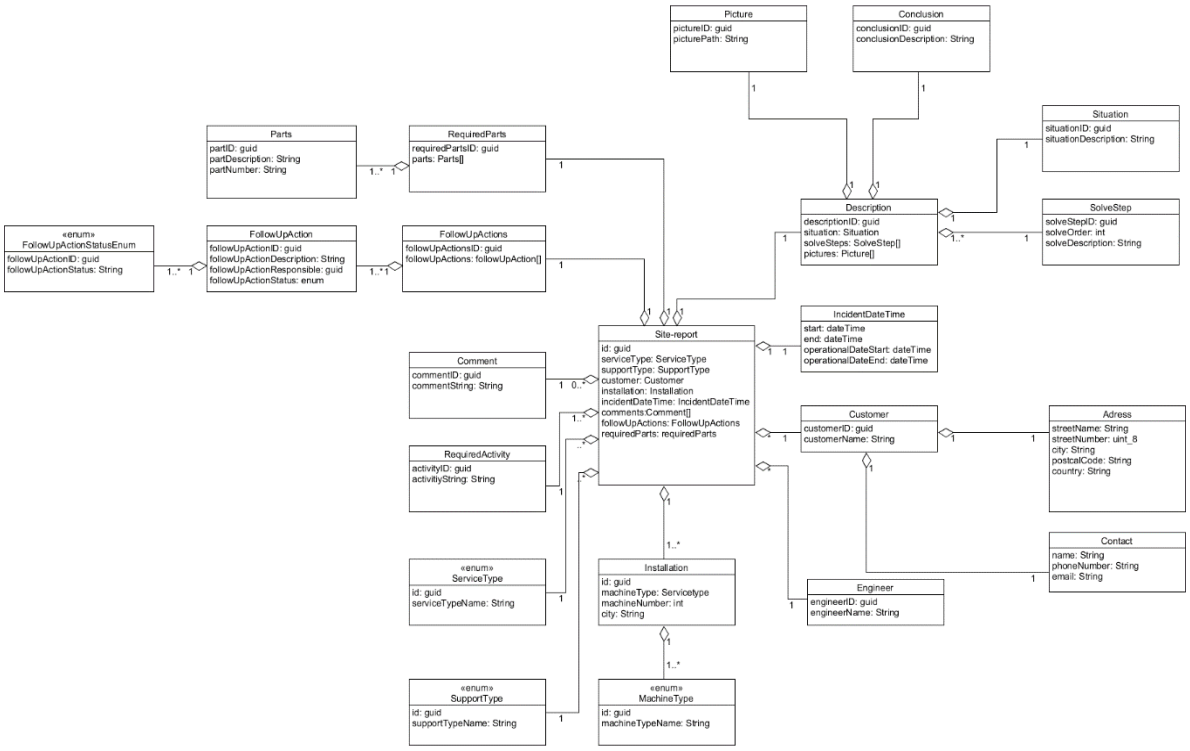


Figure 3 Class diagram first edition

Analyzing of the data types:

Site report: The site report itself is just a container that links all the connected classes, it functions as a linking table, it can hold a start and end date time for the report.

Description: The description is the core of the site-report, it contains pictures and text of what was wrong with the machine. These can be filled in with pictures and text/speech.

Engineer: This is the engineer who worked on the machine to make it operational again, this can be registered in the form of a login.

Customer: This is the location and name of company where the machine is located, this can be configured for a machine since it will only function in one location.

Servicetype & Supporttype: These are predefined types which the engineer has to check which one applies. This can however later be coupled to a customer since the customers contract will be the same for every machine in the site.

Installation: This is the machine setup which is scanned by the QR scanner, to register which machine broke. Scanned by the system.

RequiredActivity: These are the steps the user did to resolve the issue. These can be filled in by a speech file/text.

Comment: Extra comments the user can input, not used in examples I have received. Not used in the examples.

FollowupActions: Follow up actions i.e. ordering parts for repair. Not used in the examples.

Requiredparts: The parts that were used to repair or replace in the machine. Scanned by user on the QR/barcodes.

This class diagram is made according to the already in-use Site-Reports. This diagram shows all the values that are present in these reports. Due to the “limited” way of input a device with no text input has, I made a diagram which shows which values have a certain input.

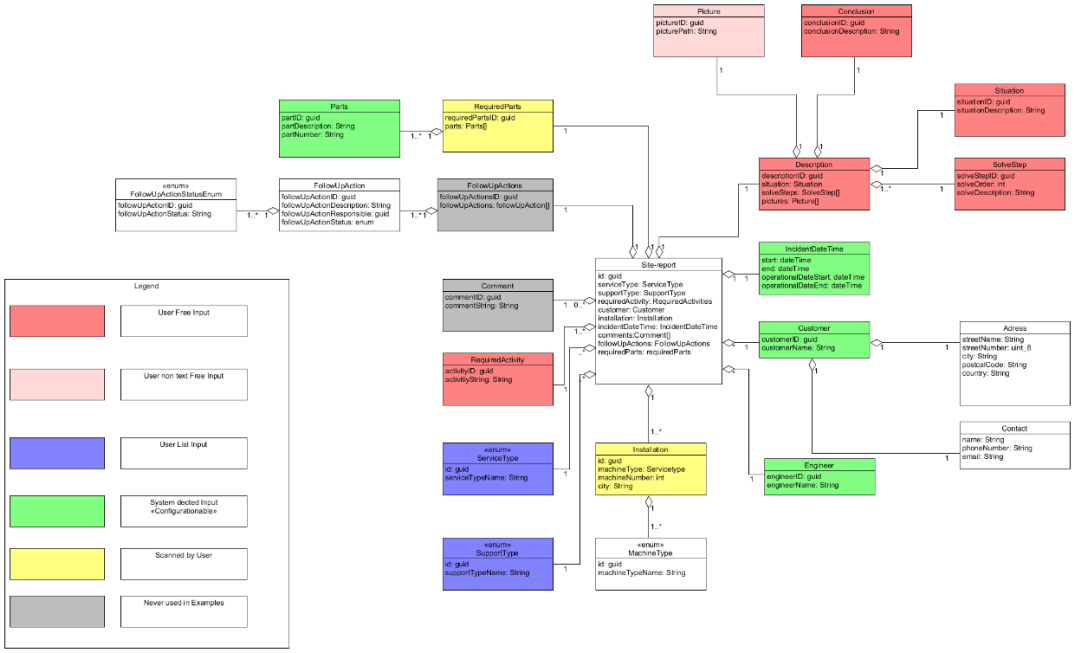


Figure 4 Class diagram with Input types

6.2. Result of analysis

By analysing this I can create a list of requirements the application has to adhere to. Namely:

- The user can Scan a machine or a part, either by barcode scanner or a QR scanner.
- The user can select an item or multiple items from a list of items.
- The user can take a picture and upload it.
- The user can input text for further explanation in any form or shape.

7. Prototyping

The minimal requirements are noted in the user stories above in order to create the minimal viable site-report. The goal of the prototype is to provide a proof of concept of the front-end functionalities and a persistent back end in which data is saved.

So in order to make the prototype I used a low-fidelity prototype which means a rough skeleton of the application of the product based on requirements (Perfomatix 2022). This ensures that the pre-defined requirements are all implemented and the presentation can be incremented on later.

Because there are so many tasks on this full-stack development project, the tasks can vary and the prototyping methods can also vary. This is because all parts are designed to be self-contained pieces of software. According to multiple sources (Perfomatix 2022, Geeksforgeeks 2023) there are four prototyping methods for software development, namely:

- Rapid Throwaway
- Evolutionary
- Incremental
- Extreme

A quick summary of the types:
Rapid Throwaway is the method of exploring ideas and getting customer feedback for each of them, functionalities created here are not necessarily be a part of the prototype.

Evolutionary is a method to develop a prototype based on well-understood requirements, these are then tried

Incremental is the type of prototyping where the final product is broken into small pieces of prototypes and developed individually which is at the end collectively merged together.

Extreme is mainly used for web development and contains creating a finished UI first and filling it with mock data at first. After that incrementally implementing actual functional methods.

For gathering feedback and testing the core functionalities. The emphasis will be on gathering input from the device and storing the data somewhere. The concept is to create small prototypes to test the functionalities of the Realwear glasses. These functionalities are made with the intent of using the core functions for the final product. So for the Realwear Unity to Android build project I will be using an **Evolutionary** development method. The first iteration of this prototype proved sufficient to embed with a reworked backend API, and is then iterated upon with new UI updates.

The server side/API will be developed along side with the current needs of the Realwear Unity to Android project. However since the API can be developed in order to hold the data persistently it requires a bigger time investment to set up properly and can be expanded when the need arises. Therefore the **Incremental** prototype method is used. It takes a small amount of the total scope required for the to be implemented user story. The first prototype proved to be sufficient to include to test the core functionalities. The second increment is with a persistent database attached to it.

8. Designing

The minimal setup in order to tests the beforementioned minimal functionalities.
Is to have a controller for the Realwear glasses, which can sent files/logs to an API, then the API will save it to a local file storage. In order to then test and read the results I can use an integrated .NET tool named Swagger or Postman.

Swagger (OpenAPI) is a language-agnostic specification for describing REST APIs. It allows both computers and humans to understand the capabilities of a REST API without direct access to the source code. Its main goals are to:

- Minimize the amount of work needed to connect decoupled services.
- Reduce the amount of time needed to accurately document a service. (ASP.NET 2022)

Postman is an API Platform for developers to design, build, test and iterate their APIs.
So either Swagger or Postman can be used to view data and send requests to the API. This is just a substitute for a well-developed website that shows the data in a clean presentable fashion.

The figure below shows the general workings of the setup.
By making use of the API, the core functionalities of the Realwear glasses are decoupled and could be potentially be replaced by another application (i.e. a smartphone)

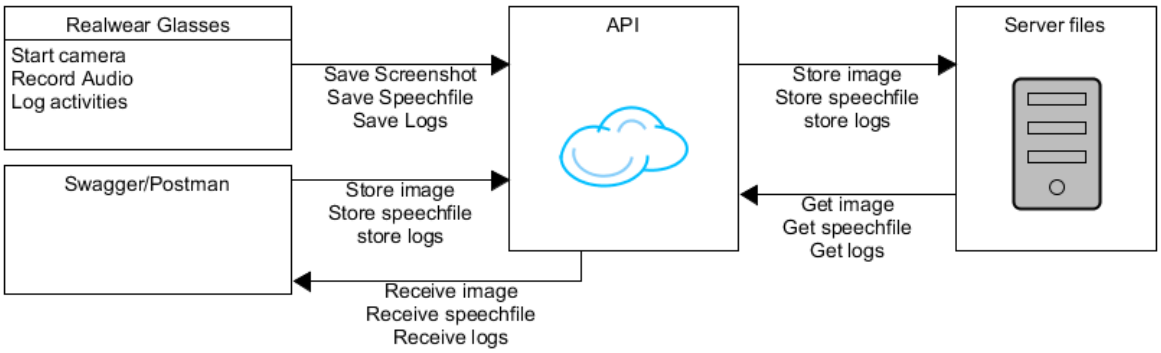


Figure 5 Flow of a minimal setup

9. Debugging

The Realwear device is by default not connected to a computer, this could prove difficult to debug once the project is taken into production. For development it is possible to connect it to a computer and read its input via USB or wireless connection. Once the device is connected the default debugging program is Android Debug Bridge (ADB). Android Debug Bridge (adb) is a versatile command-line tool that lets you communicate with a device. The adb command facilitates a variety of device actions, such as installing and debugging apps. adb provides access to a Unix shell that you can use to run a variety of commands on a device (Android Debug Bridge 2023). The ADB is a really useful tool to connect to the Android system that's running on the device. It gives the developer more control over the application and it logs all services that the device is running.
Some of the useful services that can be read are:

- The voice service that is running and what command it receives.
- The system print from the application.
- System logs
- Driver and update logs

With this there is the possibility of debugging the application, however it is far from ideal. The log fills up with all the Android services that are not necessarily required to know for the developer. The device needs to be connected to a computer in order to receive these ADB logs, this can be done during runtime or afterwards. This is far from an ideal solution because live debugging is not an option during production.

Therefore another way of debugging is desirable. Debugging by manually writing a log function which uploads its input to another device via a http request is a solution. This has the advantage of receiving all the data from different machines into one place. This makes it a prerequisite for the device to be connected to the internet, which it needs anyway in order to upload the data captured. This way of debugging creates a way to receive live feedback so it is the preferred method.

10. Software architecture

A software design before starting on the product which is meant for production is required. This ensures that the codebase follows a clear guided structure and is easy to read, expanded, can be tested on individual functions and its core functionalities are decoupled from the rest from the code.

Because there is a possibility of the Realwear glasses being replaced or substituted for another device (i.e. smartphone) there needs to be a certain level of independence of the designed part.

According to the Design there are a couple of systems that work together and have to be designed together.

The way the persistent data is handled and received is the "lead", according to the needs of the API/Database product the rest is amended accordingly.

In summary, the software architecture requires:

- Independence of code, mostly presentation layer.
- Easy testing of individual components.
- Using the API layer as a lead.

Because the API .NET is the lead, the designs will be based on web development.

10.1. Traditional 3-Layer architecture

In web development it is common to organize applications in layers, one such typical divide of layers is shown in the figure below (.NET Common web application architectures, 2023).

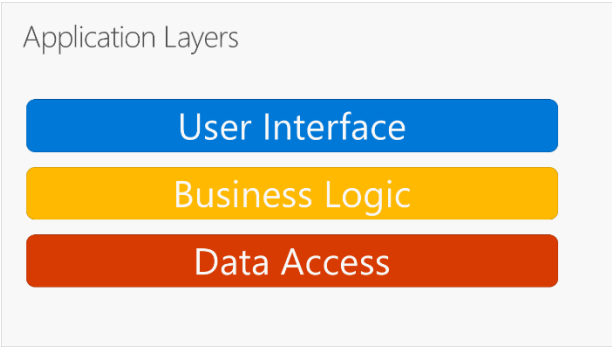


Figure 6 Application layers in web development

Using this architecture, users make requests through the user interface layer that gets picked up by the business logic layer, this in turn calls the data access layer for data. The business logic layer is the one that connects both layers and handles all the logic. According to Microsoft (.NET Common web application architecture,2023) there is a big disadvantage using this.
"That is, the user interface layer depends on the business, which depends on the data access layer. This means that the business logic, which usually holds the most important logic in the application, is dependent on data access implementation details (and often on the existence of a database). Testing business logic in such an architecture is often difficult, requiring a test database."

This design is then unfortunately does not adhere to the abovementioned requirements of the software architecture.

10.2. Clean Architecture

Another architecture suggested by Microsoft (.NET Common web application architectures, 2023) is clean architecture. The big difference between this and others is that this puts the business logic and application model at the centre of the application. The business layer is not dependent on the data layer but instead implementation is dependent on the application core.

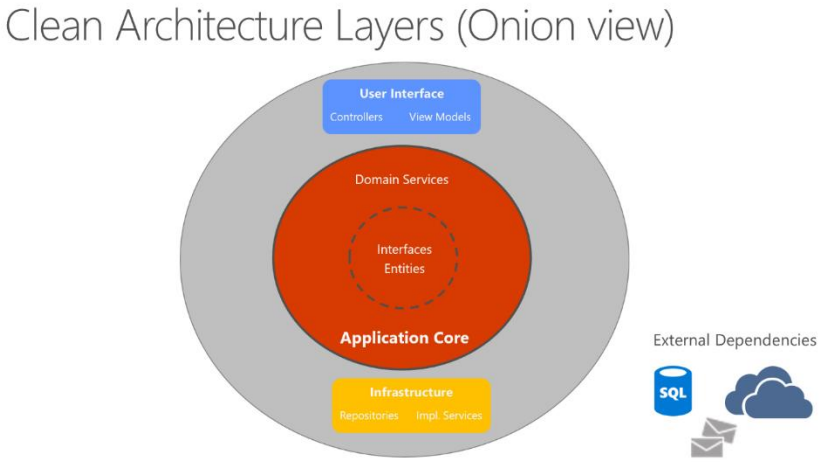


Figure 7 Clean architecture layers

In this diagram, dependencies flow toward the innermost circle. The application's entities and interfaces are at the very centre. Just outside, but still in the Application Core, are domain services, which typically implement interfaces defined in the inner circle. Outside of the Application Core, both the UI and the Infrastructure layers depend on the Application Core, but not on one another (.NET Common web application architecture, 2023).

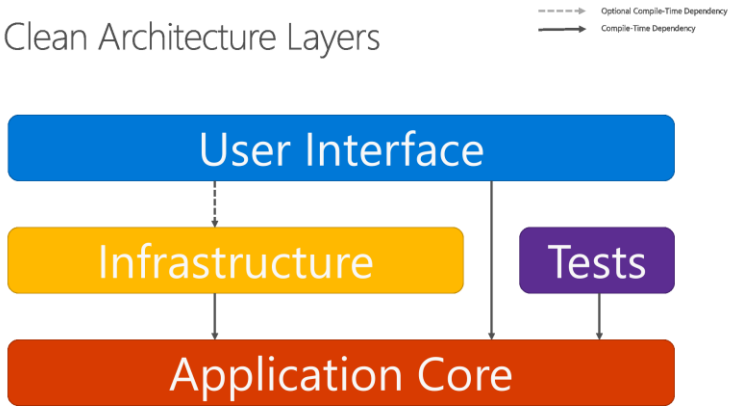


Figure 8 Application layers in clean architecture

This diagram shows the dependencies in a way as presented in figure 7. This shows that Tests can run independently on the application layer. The UI layer does not have any direct dependency on the application layer or infrastructure, therefore it can be swapped out freely.

This solves our requirements of independence, the layers are clearly defined and can be swapped out with some minor edits. The testing of individual components is also easily possible due to single dependencies. For the production I will be using this, clean architecture, software architecture.

11. UI/UX design

The design of UI/UX is not something I am specialized in, however since this project uses a quite unusual device of input it is an almost mandatory subject to briefly touch upon. In order to give better guidance to the user of what steps he has to undertake.

The two big hurdles the product has to clear are:

- How does a device with no input method except voice makes sure the user knows what the commands are.
- What are the best practices for UI design for a device that's so near your eyes.

11.1. Voice user interface (VUI)

Voice user interface (VUI) makes it possible to interact with a device using speech commands. Some of the most well-known examples are Apple's Siri, Google assistant and Amazon's Alexa. However, smartphones have buttons you can press and quite intelligent deciphering of command prompts.

The Realwear glasses allow me to create custom commands instead of responding to an open text. According to Speechly 2020 this classifies this device as a Voice command device, which makes it a device with a very specific purpose with a narrow use case. This means these commands can be made in a highly detailed structure and is less of a dialogue between user and machine.

For voice commands in general there is an anatomy of how a voice command is structured. An users' voice command consists of three key factors: **intent, utterance** and **slot** according to Toptal, 2017.

The intent or objective of the command is either a very specific task, for example: select an item, or a broader request: show me a manual.

The utterance could be "Select item A", "Select A" or "I want you to select item A". These should all mean the same thing.

The slot however is less required and is often optional it is often used for destinations, a set time or date. By analysing these key factors the list of voice commands can be made with a better intent.

Toptal, 2017 also gives some general tips for VUI which are great to keep in mind when designing all the voice commands. Keep the communication simple and conversational, fewer words are better. Confirm when a task has been completed, give the user a small voice confirming command. Create a strong error strategy, if the command is not recognised give the user a recommended voice command.

As an example of voice commands I looked for inspiration at the Microsoft windows speech recognition commands (2023) as this is a well-documented page of voice **commands**. This example was chosen because of its simplicity of commands and good documentation and because windows is universally known.

Navigation

Voice command	**Behaviour**
Back	Goes to the previous page.
Next	Goes to the next page.
Finish report	Finishes up the report and goes to the main page.
Restart report	Throws all the data away and restarts the site report.

Input functions

Voice command	Behaviour
Start QR	Starts the QR scanner.
Stop QR	Stops the QR scanner.
Start camera	Starts the camera.
Stop camera	Stops the camera.
Start microphone	Starts recording with microphone.
Stop microphone	Stops recording with microphone.
Select **#**	Selects/unselects item **#**
Screenshot	Makes a screenshot.

By presenting the user with a list of items he can easily select the desired items, upload pictures and upload spoken speech. The commands that can be spoken have a button on the UI which text corresponds with the command in order to easily identify which commands can be used.

11.2. UI design for smart glasses

The screen of the Realwear HMT-1 is in a relatively unknown spot for UI/UX design. This is because it's a combination of Virtual Reality (VR) and Augmented Reality (AR). The screen allows the user to still see the outside world just like how AR functions but does not project its screen over the outside world. The user needs to actively look at the screen in order to fully receive information from it.

However, general tips from VR and AR development can still be followed and mix and matched. Some of these general tips from Google developers (2020) for AR are as following:

- Keep it relevant, only show the relevant information, go for a minimalistic approach.
- Avoid the unexpected, don't send content too quickly and at unexpected times, make sure the user knows what the intention is.

While some of the general tips on VR interface design are to avoid using big text blocks and highly detailed UI elements (Sam appelbee and Alex dereutte (2017). While using tested wireframes is a good starting point for UI design.

While the letters might not look as crisp on the device due to the resolution, there is large text required in order to read it, this was done by testing readability by different developers. The texts are required to convey the command message.

12. Development

During the development phase a prototype was made following rapid prototyping. The first iteration of this prototype had the goal of testing the core functionalities of the Realwear glasses.

The core functionalities were as follows:

- Add voice commands to the Unity project.
- Send debugging data to a created API server for debugging purposes.
- Send data from the Realwear glasses to the created API which represents the database.
- Send a picture from the Realwear glasses to the created API and save it.
- Record an audio file and send it to the created API and save it.

12.1. Unity

For the development of the prototype the first thing to create was the interface for the glasses. This is made in Unity and had some essential parts that had to be developed in order to function.

12.1.1. Voice control

One of these key components of working with the Realwear was adding a voice command. The voice command provided by the documentation was this:

```
void Start()
{
var wearHf = GameObject.Find("WearHF Manager").GetComponent<WearHF>();
wearHf.AddVoiceCommand("Hello World", VoiceCommandCallback);
}
void VoiceCommandCallback(string voiceCommand)
{
Debug.Log("Voice command recognized: " + voiceCommand);
}
```

Figure 9 Realwear documentation

This would allow voice commands to be bound with functions. However this has two major disadvantages, one is that voice these voice commands have to be bound to a function and keyboard shortcuts for debugging purposes also have to be bound to the same function. This would result into writing a separate function to see which hotkeys are pressed and call the correct function. The second disadvantage is that there is no overarching function in which you can connect a debugging mode on. The solution I found for this is to create a class called Command. As seen in Appendix Figures of Adding commands. There is a list of commands in the WearHFConnector. A command consists of the String which gets linked to the spoken command as seen in the figure above, a keycode for desktop use and an Action link to the function which is to be executed. Then linking the spoken command to a callback allows it to have multiple checks on the command such as, add to debug, repeating the command back to the user and executing the linked command.

12.1.2. Camera, QR Scanner

For the Camera and QR scanner I used an example on github by nickdu88: https://github.com/nickdu088/Unity-QR-Scanner this used an external library to decode a QR message to a message string. The input of the camera gets streamed as a texture on a plane in the Unity scene. This behaviour is shown in the Appendix as Figure Webcamtexture & Figure screenshot & GetQR code. The screenshot method reads all the bytes from the texture and puts it in a bytearray which is then passed to the Networkcontroller. The network controller then sends to a created web API which functions as a database.

12.1.3. Audio recorder

For the audio recorder I researched how to create a .wav/.mp3 file from an audioclip in Unity and I stumbled on this page: https://gist.github.com/darktable/2317063 From this I imported the SavWav class to create a file from an audio clip. This was needed because the wav files uses a header file in it's data type. This program allows the creation from an Audioclip to a .wav file. As seen in the figure Generating Audio file the audio recorder saves a file and then reads the bytes again from the same file and sends it to the Network controller.

12.1.4. Network controller

The network controller creates forms that can be sent to the web API. The network controller uses a singleton pattern in order to not be dependant on being linked to each class which wants to send data. The example shown in figure Send byte array to web API shows how the singleton is implemented. A WWWForm is created and filled with the correct data to be sent to the web API.

12.2. Web API

For the Web API a ASP.NET 7.0 framework is used. This web API uses controllers for each connection point, such as ImageUpload, SpeechFileUpload and FileDebugController. These function as endpoints on their respective connectionpoints i.e. www.baseurl.com/ImageUpload. Each controller has the same broad functions which are: Upload, GetList, GetItem and DeleteItem. The upload function is seen in figure ImageUpload in the Appendix. A directory on the server is allocated for the respective files. The imageupload has some built in security such as not overwriting files if it already exists and a token to see if an unique token matches so that not everyone can just upload files to it. The big advantages of working in folders is that all the files are neatly separated and works really fast for a prototype. The figure ImageList GetImage DeleteImage shows how the items can be manipulated further. It works quite simple due to the folder structure. The figure below shows how these functions can be tested and executed with swagger as described in Designing

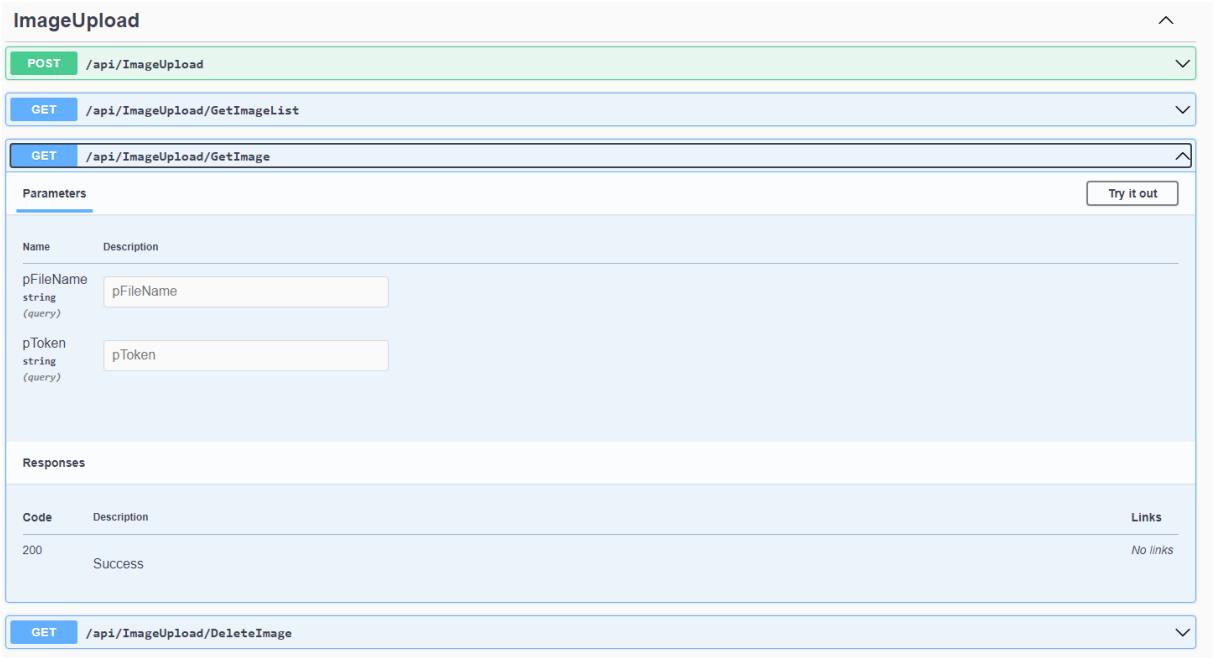


Figure 10 Swagger example

After these functionalities were tested and deemed sufficient to gather the required files, it was time to gather some feedback on these functionalities. The folder structure has outlived it's purpose for the prototype. There is no way to sort the files by which file belongs to which site-report. In order to achieve this a database with linking tables has to be created. In order to achieve this integration with a database, it is required to implement the software architecture as described previously.

12.3. Web API 2.0

As stated before, the “Clean architecture” architecture was to be implemented. The figure below shows the class diagram of saving a picture as an example. This is to give context on how the architecture is implemented.

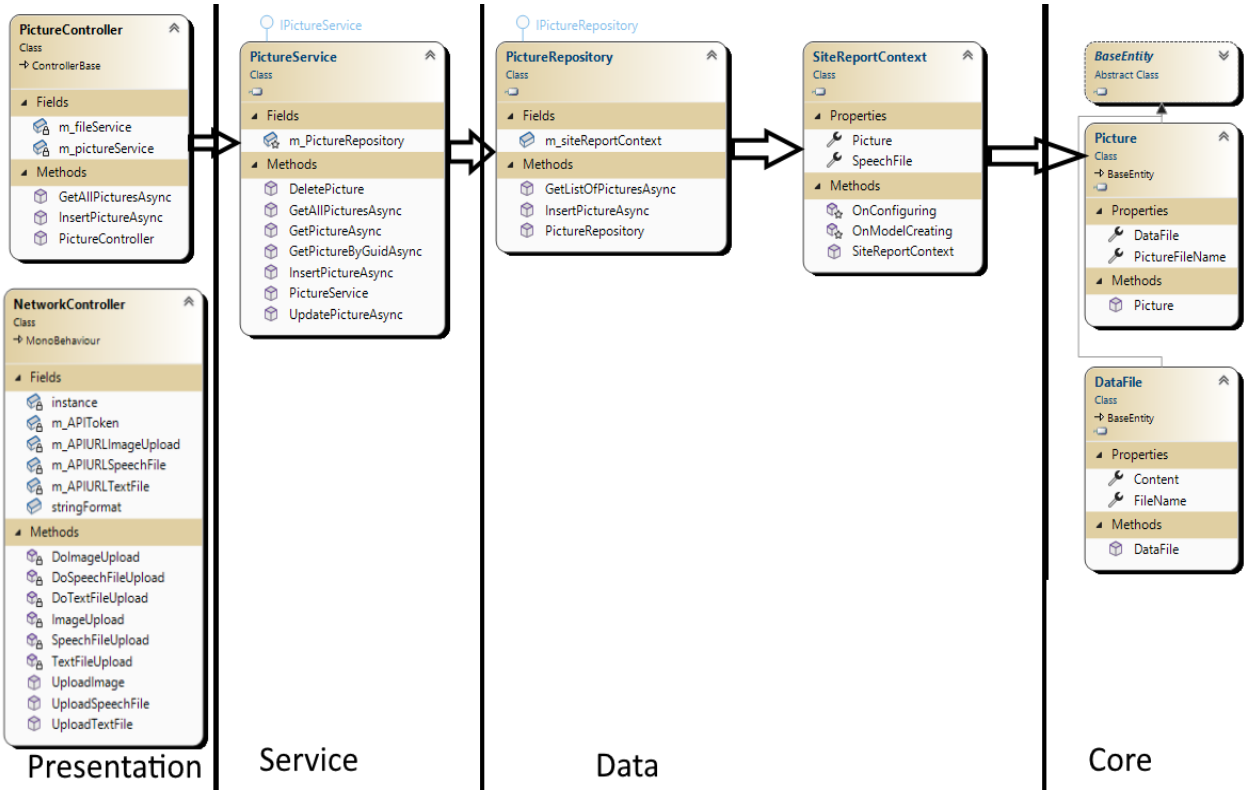


Figure 11 Class diagram example

Now to give this class diagram some context. As stated before, dependencies go from Core to Data to Service and then last to the presentation layer.

12.3.1. Core layer

In the core layer of the project are entities and aggregates located. The entities are as short as possible and hold a singular responsibility. For example the Picture only holds the filename and a reference to the datafile. The datafile exists only during runtime which is when the files on the system are read and their byte data is converted to its content. It can also hold Aggregates which are grouped classes of entities, for example an customerAddress and a customerName could be paired into a Customer aggregate.

12.3.2. Data layer

The data layer handles, as the name suggests, the data access, file system access, network calls etc. The primary function of this layer in this project is to retrieve and write data to the database. It also accesses the picture and speech files which are stored in a map on a server. For the implementation of the connection to the database I chose for Entity Framework (EF). According to Microsoft (2023) Working with Data in ASP.NET Core Apps it is “the recommended way for your application to access its data”. I also have experience albeit little with Entity Framework (EF). The big advantage is that all CRUD (Create, Read, Update, Delete) functions work pretty well out of the box. The way it works in this project is that a context is loaded from a targeted SQL database. The context is specified of one or multiple entities or aggregates defined in the Core layer and corresponding to database tables. The context is then passed on to a repository which limits the access level of the service layer so that

it can't modify the whole context. All of the application logic is in the data layer, this means checks for validation and conditional edits and saves.

12.3.3. Service layer

The service layer is the point of contact of the application so to speak. By invoking services, the presentation layers can send requests/commands and retrieve models. It also validates the input data by validating the incoming .json data with the specified model in the data layer.

12.3.4. Presentation layer

The presentation is in this case split in multiple instances. One for the API controller which acts as the first step a program encounters once it fires a request to the API. The other presentation layer is the Unity project which can fires its own request to the API. It asks for certain data and this data is then supplied by the API. This data is then built into different models consisting of aggregates and entities. Which then in turn fills an UI supplied by the model data.

12.4. Optimisation

During development of the prototype, a problem was encountered with the way Unity creates audio files. The only way to specify a new audio file is to create one with a specified length. The byte size of the audio files would become really big which would be unpractical once you're working with a device that is on the move and therefore could have slow internet. Some of the solutions to the no internet problem are:

- Make user select length before speaking
- Set amount of time 30 sec.
- Read bytes for long pauses and cut these.
- Timer between start and stop command and cut byte size to this amount.

Making the user select the length before speaking has the disadvantages of creating an extra step in the workflow of the user. It also has the disadvantages of the message being too short and having to redo this step because the user could not say everything he wanted.

Having a set amount of time limits the filesize but just as the solution before if the user has more to say he can not extend the duration and the file is cut short.

Reading bytes and checking for pauses is a bit more complex than the before mentioned solutions. However, with the machines being used in environments with loud ambient sounds (i.e. big sorting machines) the silences might be too hard to detect.

The easiest way to detect the length of an audioclip is to time the time it took between the two commands, stop and start, and cut the audio file to this size, the final result is seen in the Appendix - Generating Audio file.

Another decision was to either save the byte array as a field in a database, or saving a file path of the picture and saving the picture on a dedicated spot on the server. On a small scale it would be possible to save the files as bytes in the database. However according to Microsoft: Large object storage in a database or a filesystem (2006):"It is advised to store objects larger than 1MB in a filesystem and let the database save the physical path. This is because the read:write ratio and rate of object overwrite or replacement are important factors." Because the project is targeted to being used in all the Benelux sites it spans more than 10 sites with each up to a few incidents per day, the amount of pictures increases rapidly.

13. Improvements

For the improvement section I will list certain problems or suggestions I have encountered during development but had no time to fix or implement. In my opinion there are still a lot of things that are missing before this project can be used in production.

- Handle internet not available.
- Upload video files.
- Automatically change the spoken text to written text.
- Implement a cache function for text to speed up the program.

Solving the problem of **having no internet** while working with the device. While the product is being used in a controlled environment (i.e. inside a factory) and Wifi boosters/points can be installed this can still occur. A possible solution is to store the files locally on the device and keeping the site reports in cache memory. However this means the device can not be shut down, so another option would be to generate a file which has all the links to the speech files / pictures and other variables which then can be read on the next startup once the device has internet. After uploading the correct files the files can be cleared again.

Adding the function to **upload videos** should be a relatively easy task. It involves adding a new property to the Core-Data-Service architecture and calling the video function on the device and recording this in Unity. This is not all implemented due to time constraints and was not high on the priority list for the proof of concept.

The end result in a complete site-report has to be supplied in a written format. In order to prevent someone to have to listen and dictate every speech file it would be wise to put a system in place to rewrite **speech to text**. There are two products I want to advice to use. The first one is Whisper (2022), this is a project by OpenAI and uses AI training models for speech recognition. They are the leading source of AI at this moment with their Chat-GPT.

Whisper is an automatic speech recognition (ASR) system trained on 680,000 hours of multilingual and multitask supervised data collected from the web. We show that the use of such a large and diverse dataset leads to improved robustness to accents, background noise and technical language. When we measure Whisper's zero-shot performance across many diverse datasets we find it is much more robust and makes 50% fewer errors than those models. About a third of Whisper's audio dataset is non-English, and it is alternately given the task of transcribing in the original language or translating to English (Whisper 2022).

According to these statements made by the whisper team it would seem like a great candidate for using the API to convert the speech to Text.

The other candidate is google with their cloud speech-to-text service. This service also supports uploading speech files to an API and receiving a written text. Furthermore, they have a few core functionalities.

Speech-to-Text can handle noisy audio from many environments without requiring additional noise cancellation. Support your global user base with Speech-to-Text's extensive language support in over 125 languages and variants. Customize speech recognition to transcribe domain-specific terms and rare words by providing hints and boost your transcription accuracy of specific words or phrases. Automatically convert spoken numbers into addresses, years, currencies, and more using classes (Google Text-to-speech 2023).

14. Play test and review

For testing I used usability tests, ideally it would be with done with the engineers who are supposed to work with the final product, however due to the completeness statuness prototype this was. It

proved a difficult to go into an external work site to test the prototype. So as a solution I've done this with people around my age 25-30 and who have a decent knowledge of technology i.e. smartphone usage and are aware of VR. This was to test how the UI/UX aspects work of the prototype.

The first test was to see if the program could be understood without any interference from me or guidance. This was also to see whether the Realwear glasses could be controlled without any guidance. The objective was to create a site-report, so select a contract status, add a picture of something and create a description of the situation.

First participant

Able to select the correct program	Was able to select the program after trying various command "select #", "*name".*
Able to navigate through the program	Was able to figure out the text on buttons to navigate.
Able to create pictures	Was able to start the camera, didn't take screenshots.
Able to upload speech file	Started the speech controller but didn't stop it.
Able to select items from a list	Was not able to select items.
Able to finalize the site-report	Did finalize it, but didn't correctly save speech files.

Second participant

Able to select the correct program	Was not able to select the program on its own.
Able to navigate through the program	Was able to figure out the text on buttons to navigate.
Able to create pictures	Able to start the camera, forced a next.
Able to upload speech file	Able to start microphone, forced a next.
Able to select items from a list	Not able to select.
Able to finalize the site-report	Did finalize it, didn't upload anything relevant.

Because the result of the first test was not a great success, I've done a second test with the same people. This time with more guidance. The same test plan but the questions that arose I've wrote down to see where the UI has to improve to give feedback.

First participant

Able to select the correct program	Was able to select the item again.
Able to navigate through the program	Was clear, maybe introduce a step counter #/#
Able to create pictures	Was able to start the camera, asked on how to take a picture. Asked how to see if it was successful.
Able to upload speech file	Started and stopped the speech. Asked if it was successful.
Able to select items from a list	Was not able to figure out the items. *Items description are in dutch while the commands were in English.*
Able to finalize the site-report	Finished it successfully.

Second participant

Able to select the correct program	Was able to select it the second time.
Able to navigate through the program	Was able to go back and further with commands.
Able to create pictures	Asked for the take a picture command. Was able to create multiple screenshots.
Able to upload speech file	Successfully started, stopped and uploaded the file. Was asking if it was recording.

Able to select items from a list	Was not able to figure out the items. *Items description are in dutch while the commands were in English.*
Able to finalize the site-report	Finished it successfully.

The results were more satisfactory in the end. The final goal of creating a site-report has been achieved.

14.1. Results

The common observation between all were:

- Able to navigate forward through the menus.
- Able to start the camera.
- Able to start the microphone.
- Lack of visual feedback on success.
- Was not able to detect non-implicit mentioned functions.

For reference, here are the menus that the user see:

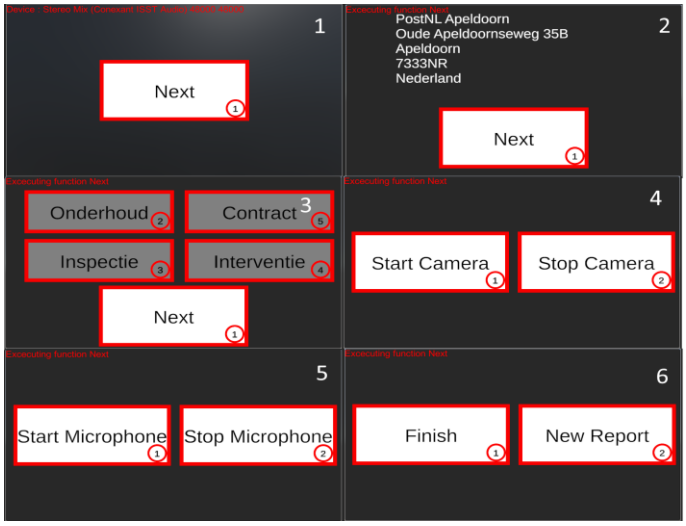


Figure 12 Menus the user were shown

The applicants were able to be guided through the functions by the visual feedback alone since it is literally the command on a singular “button” like menu. The “next” button has at one point disappeared, however the command is still able to be called, the users picked up on this no problem. The start camera and start microphone also did work, however it was unclear if the microphone was working because there was no audio/visual feedback. With the start camera function, the screen literally changes in just the camera output, however there is no clear indication of how to take the screenshot/picture. Same with the microphone function, there is no clear indication on whether the functioning is running, apart from the small debug text above, and is completed and saved. However these results have also shown that doing this loop a second time had a better result than the first. Since the end-users are engineers on site and can be trained, I suspect a training of 15 to 30 minutes should be sufficient to show all the ins and outs of the program. Once the engineer is trained the process should be a piece of cake and he should be able to walk through the program in mere minutes.

The conclusion is that this prototype still needs a lot of visual/audio feedback in order to feel more intuitive. However, as I am not a UI/UX designer I can not provide this for a proficient level. Trained engineers can be made to work with this prototype quite easily.

15. Conclusion

After collecting all the data, developing the prototype and having usability tests it is time to look back at the initial questions that needed answering. Let's start with the sub question: "Can the creation of a site-report be more streamlined and quicker with a smart device on the work floor?"

The creation of a site-report is now way quicker than someone walking back to a computer after fixing the error of the machine. This is done by the smart device supplied, albeit there are still a lot of things that can be streamlined in order to make this even more robust and quicker.

"Which medium of information is needed at minimum in order to create a site-report and can the supplied Realwear HMT-1 provide these mediums?"

The site-reports as of now are designed with a lot of useless and duplicate information. By normalizing the data as seen in figure 3 & 4, this can be shrinked down a lot. After doing this the mediums can be broken down intotext, spoken or written, and pictures. The prototype proved that the Realwear HMT-1 can provide these in combination with the API. This can also be extended with video recording which I have no doubt will not be an issue.

The main question: "How can the goal of improving the quality and quantity of filing site-reports by mechanics be met by developing an application for a smart device and gamifying the rewards at sorting depots in order to get a more complete overview of incidents?" with this prototype is answered. The quality and quantity of site-reports can be increased because of the new ease of use of making a site-report. The gamification of the reward is unfortunately not tested because lack of time and planning. However, engineers can be trained quite easily to use these glasses in order to work with them.

16. Recommendation

As my project comes to an end and the goal to create a proof of concept project for an app for the Realwear HMT-1 in order to create site-reports has been achieved. It is time to draw conclusions and advice BEUMERGROUP on how to continue, or not, with this product.

The proof of concept has been deemed feasible, it provides the user with all the tools to create a full site-report. This project allows the engineer to report the cause of an incident more truthfully, which was the first wish of BEUMERGROUP. Automatically filling in a complete site-report in document form however has not been totally achieved, however it removes guessing work from the person which has to create these reports and has now concrete proof of the incidents.

In order to pilot this product in a real environment it is advised to complete these steps :

- Creation of a website which shows all the site-report data with an attached speech file and pictures.
- Expand some missing important features, i.e. selecting users and video support.
- Add more extensive log files which automatically uploads itself once it has internet access.
- Creation of a PDF document out of the collected data of a site-report.
- A more intuitive UI update to the glasses interface.

Once these steps are implemented, the product is ready for the pilot in order to further develop the product.

17. Personal reflection

For the personal reflection I like to give my personal experience with this project, the company I worked for and the company which it was developed for.

The experience I had with this project was a fun one, being able to set up everything myself like the Android Unity platform, the web API and the database. It gave me a lot of freedom to experiment and learn new things. It was a tough challenge because there was so much emphasis on things I haven't learned within the academy of Saxion. The product I produced in the end was a decent enough proof of concept, however I would have liked for the product to be have been more developed. The end-result however is a great framework in order to continue development if the project is to be continued. I had overestimated the amount of work I could have done on the product and thought I would be way further done. Working in an organized way is not my forte, I worked in sprints of 2 weeks and more often than not I overestimated the things I would be able to do in those weeks. Either that or underestimated the difficulty of the task. Daily logging of activity went pretty well for a month and a half until deadlines showed up. I think I have to be more of a routine worker and implement planning tasks in this routine.

The company I worked for had a great atmosphere, there were great colleagues and everyone has a lot of technical knowledge of web-development and software development. Although I would have liked a better form of daily/weekly progress updates. There were bi-daily standups, they however did not involve the individual tasks on each project since everyone is working on a different project. However having meetings over architecture and design with Chiel was useful and insightful, this made me do more research on the advantages and disadvantages of certain architectures. In my opinion the one thing this company could profit from is more updates within the team, so in case someone needs help on a different project it is somewhat clear what is going on in there and what's been worked on.

The company I developed the solution for I had contact with bi-weekly. The contact persons from this company are easy to talk to. Unfortunately the prototype I made was, in my opinion, not in a good enough state in order to launch the pilot. With this pilot I could receive feedback from the final end-user instead of the people at this company. If I were to do such a project again, I think a good idea is to be more involved with the company I make this for, in terms of contact hours and sprint planning. Try to set a hard deadline when to showcase the product (which they did) and then according to that showcase steer more of the product. Once it's in an accepted state, present it to the engineers and ask for their feedback. But I'm happy to report that the client is still interested in the product after a showcase and continues to want development on this product.

All in all, I am happy with the result of this project. It is not as completed as I would have liked it to be and the clients problem is not solved yet. I think I learned a lot about web development and it's intricacies, even though it was not the hurdle I thought I had to clear initially. I have to work harder to discipline myself to take every day some time free for planning and logging.

18. References

.NET Common web application architecture (2023), Common web application architectures. Retrieved June 09, 2023 from https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures

Android Debug Bridge (2023), Android Debug Bridge (adb) Retrieved June 09, 2023 from https://developer.android.com/tools/adb

ASP.NET (2022), ASP.NET Core web API documentation with Swagger/OpenAPI. Retrieved June 06, 2023 from https://learn.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-7.0

BEUMER (2023) Welcome to Beumer group. Retrieved February 22, 2023, from https://www.beumergroup.com/.

Google Cloud Speech to Text (2023) Speech-to-Text. Retrieved June 15, 2023 from https://cloud.google.com/speech-to-text/docs/speech-to-text-requests

Google developers (2020) Glass design principles. Retrieved June 13, 2023 from https://developers.google.com/glass/design/principles

Google glass (2023) Glass Enerprise edition 2 tech specs. Retrieved March 02, 2023 from https://www.google.com/glass/tech-specs/.

Microsoft (2023) Working with Data in ASP.NET Core Apps. Retrieved June 15,2023 from https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/work-with-data-in-asp-net-core-apps

Microsoft: Large object storage in a database or a filesystem (2006). Retrieved June 15, 2023 from https://www.microsoft.com/en-us/research/publication/to-blob-or-not-to-blob-large-object-storage-in-a-database-or-a-filesystem/?from=https://research.microsoft.com/apps/pubs/default.aspx?id=64525&type=exact

Microsoft Windows speech recognition commands (2023) Windows speech recognition commands. Retrieved June 09, 2023 from https://support.microsoft.com/en-us/windows/windows-speech-recognition-commands-9d25ef36-994d-f367-a81a-a326160128c7

Performatrix (2022) Software Prototyping methods you need to know. Retrieved June 06, 2023 from https://www.perfomatix.com/4-software-prototyping-techniques-you-need-to-know/

Realwear Developer (2023) Unity. Retrieved 02 May 2023 from https://support.realwear.com/knowledge/hmt-development-unity.

Realwear HMT-1 (2023) REALWEAR HMT-1®. Retrieved February 28, 2023 from https://www.realwear.com/hmt-1/.

Sam appelbee and Alex dereutte (2017). Getting Started With VR UI Interface Design. Retrieved June 15, 2023 from https://marvelapp.com/blog/getting-started-vr-interface-design/

Samsung (2023) Galaxy S23 | S23+. Retrieved 02 May 2023 from https://www.samsung.com/us/smartphones/galaxy-s23/.

Speechly (2020) What are voice user interfaces? Retrieved June 09, 2023 from https://www.speechly.com/blog/what-is-voice-user-interface

Toptal (2017). Designing a VUI – Voice User Interface retrieved June 09,2023 from https://www.toptal.com/designers/ui/designing-a-vui

Whisper (2022) Introducing whisper. Retrieved June 15, 2023 from https://openai.com/research/whisper

19. Appendix

230126 BEUMER Site Report Apeldoorn Carrier 131, 132 en cart V1.0.pdf

230123 BEUMER Site Report Born Cart 34, Carrier 67 en carrier 68 V1.0.pdf

221213 BEUMER Site Report Born Crash 1 cart 2 carriers V1.0.pdf

220928 BEUMER Site Report Amersfoort Carrier 267 en 268 V1.0.pdf

220922 BEUMER Site Report Amersfoort Carrier 50 V1.0.pdf

```
54      private void LogCommandCallback(string pVoiceCommand)
55      {
56          foreach (Command iCommand in m_commandList)
57          {
58              if (pVoiceCommand == iCommand.VoiceCommand)
59              {
60                  iCommand.ExecuteFunction();
61              }
62          }
63      }
            1 reference
64      private void InitializeKeyCommands(WearHF pWearHF)
65      {
66          AddCommandAndBindMethod(new Command("Next", KeyCode.N, m_menuController.ActivateNextPanel));
67      }
            1 reference
68      public void AddCommandAndBindMethod(Command pCommand)
69      {
70          m_wearHF.AddVoiceCommand(pCommand.VoiceCommand, LogCommandCallback);
71          m_commandList.Add(pCommand);
72      }
73
74  }
```

Figure 13 Adding commands

```
9   public class Command
10  {
11      private Action m_referenceToFunction;
        1 reference
12      public Command(string pVoiceCommand, KeyCode pKeyCode, Action pDelegate)
13      {
14          VoiceCommand = pVoiceCommand;
15          KeyCode = pKeyCode;
16          m_referenceToFunction = pDelegate;
17      }
        3 references
18      public string VoiceCommand { get; set; }
19
20
        2 references
21      public KeyCode KeyCode { get; set; }
22
        2 references
23      public void ExecuteFunction()
24      {
25          if (m_referenceToFunction != null)
26          {
27              m_referenceToFunction();
28          }
29      }
30  }
31
        Unity Script (1 asset reference) | 0 references
32  public class WearHFConnector : MonoBehaviour
33  {
34      private List<Command> m_commandList = new List<Command>();
            Unity Message | 0 references
35      void Start()
36      {
37          m_wearHF = m_wearHFManagerGameObject.GetComponent<WearHF>();
38          InitializeKeyCommands(m_wearHF);
39          SceneManager.sceneUnloaded += SceneUnloaded;
40      }
            Unity Message | 0 references
41      private void OnGUI()
42      {
43          if (Event.current.isKey && Event.current.type == EventType.KeyDown)
44          {
45              foreach (Command iCommand in m_commandList)
46              {
47                  if (Event.current.keyCode == iCommand.KeyCode)
48                  {
49                      iCommand.ExecuteFunction();
50                  }
51              }
52          }
53      }
        1 reference
54      private void LogCommandCallback(string pVoiceCommand)
55      {
56          foreach (Command iCommand in m_commandList)
57          {
58              if (pVoiceCommand == iCommand.VoiceCommand)
59              {
60                  iCommand.ExecuteFunction();
61              }
62          }
63      }
```

Figure 14 Adding Commands

```
Unity Message | 0 references
void Start()
{
    var renderer = m_cameraTexturePanel.GetComponent<RawImage>();
    m_webcamTexture = new WebCamTexture(512, 512);
    renderer.texture = m_webcamTexture;
}
1 reference
public void StartQRScan()
{
    HideMenuAndShowCamera();
    m_QRCoroutine = StartCoroutine(GetQRCode());
}
2 references
private void HideMenuAndShowCamera()
{
    if (!m_cameraTexturePanel.activeSelf)
    {
        m_cameraTexturePanel.SetActive(true);
    }
    if (m_exitQRScan)
    {
        m_exitQRScan = false;
    }
    m_webcamTexture.Play();
    m_menuItemsHolder.SetActive(false);
}
```

Figure 15 Webcamtexture

```
private IEnumerator ScreenShot()
{
    yield return new WaitForEndOfFrame();
    Texture2D renderResult = new Texture2D(Screen.width, Screen.height, TextureFormat.ARGB32, false);
    Rect rect = new Rect(0, 0, Screen.width, Screen.height);
    renderResult.ReadPixels(rect, 0, 0);

    byte[] byteArray = renderResult.EncodeToPNG();
    siteReportBuilder.AddImageFileName(String.Format("{0}", DateTime.Now.ToString(m_stringFormat)) + ".png");
    NetworkController.UploadImage(byteArray);
}

1 reference
IEnumerator GetQRCode()
{
    IBarcodeReader barCodeReader = new BarcodeReader();
    if (!m_webcamTexture.isPlaying) { m_webcamTexture.Play(); }
    var snap = new Texture2D(m_webcamTexture.width, m_webcamTexture.height, TextureFormat.ARGB32, false);
    while (string.IsNullOrEmpty(m_qrCode))
    {
        try
        {
            if (m_exitQRScan)
            {
                break;
            }
            snap.SetPixels32(m_webcamTexture.GetPixels32());
            var Result = barCodeReader.Decode(snap.GetRawTextureData(), m_webcamTexture.width, m_webcamTexture.height, RGBLuminanceSource.BitmapFormat.ARGB32);
            if (Result != null)
            {
                m_qrCode = Result.Text;
                if (!string.IsNullOrEmpty(m_qrCode))
                {
                    Debugger.Log("DECODED TEXT FROM QR: " + m_qrCode);
                    break;
                }
            }
        }
        catch (Exception ex) { Debugger.Log(ex.Message); }
        yield return null;
    }
    ResetCameraAndReturnToHomeMenu();
}
```

Figure 16 Screenshot & GetQR code

```
1 reference
public void SaveTextFile()
{
    Microphone.End(null);//End audio recording
    float actualClipLength = Time.time - m_startTime; //Determine length of actual clip
    int clipChannels = m_audioSource.clip.channels; //Get the channel amount
    int clipFrequency = m_audioSource.clip.frequency; //Get the frequency
    float divideOrignalClipLengthByActualClipLength = m_audioSource.clip.length / actualClipLength; //Ratio of actual clip length versus initial clip size
    int sampleSizeUnClipped = m_audioSource.clip.samples * m_audioSource.clip.channels; //Total amount of samples from the initial clip
    float[] samples = new float[(int)(sampleSizeUnClipped / divideOrignalClipLengthByActualClipLength)]; //Total samples divided by the ratio to get the actual clip
    m_audioSource.clip.GetData(samples, 0); //Fills the data with the data in the new samples.

    //Return Audio clip of proper length
    AudioClip audioClipToSave = CreateNewTrimmedClip(samples, clipChannels, clipFrequency);
    //Saves audio input, to local file.
    string fileName = String.Format("{0}", DateTime.Now.ToString(m_stringFormat));
    SavWav.Save(fileName, audioClipToSave);

    var filepath = Path.Combine(Application.persistentDataPath, fileName);
    byte[] byteFiles = File.ReadAllBytes(filepath + ".wav");
    siteReportBuilder.AddSpeechFileName(String.Format("{0}", DateTime.Now.ToString(m_stringFormat)) + ".wav");
    NetworkController.UploadSpeechFile(byteFiles);
}

1 reference
private AudioClip CreateNewTrimmedClip(float[] pSamples, int pChannels, int pFrequency)
{
    AudioClip generatedClip = AudioClip.Create("GeneratedClip", pSamples.Length, pChannels, pFrequency, false);
    generatedClip.SetData(pSamples, 0);
    m_audioSource.clip = generatedClip;
    return generatedClip;
}
}
```

Figure 17 Generating Audio file

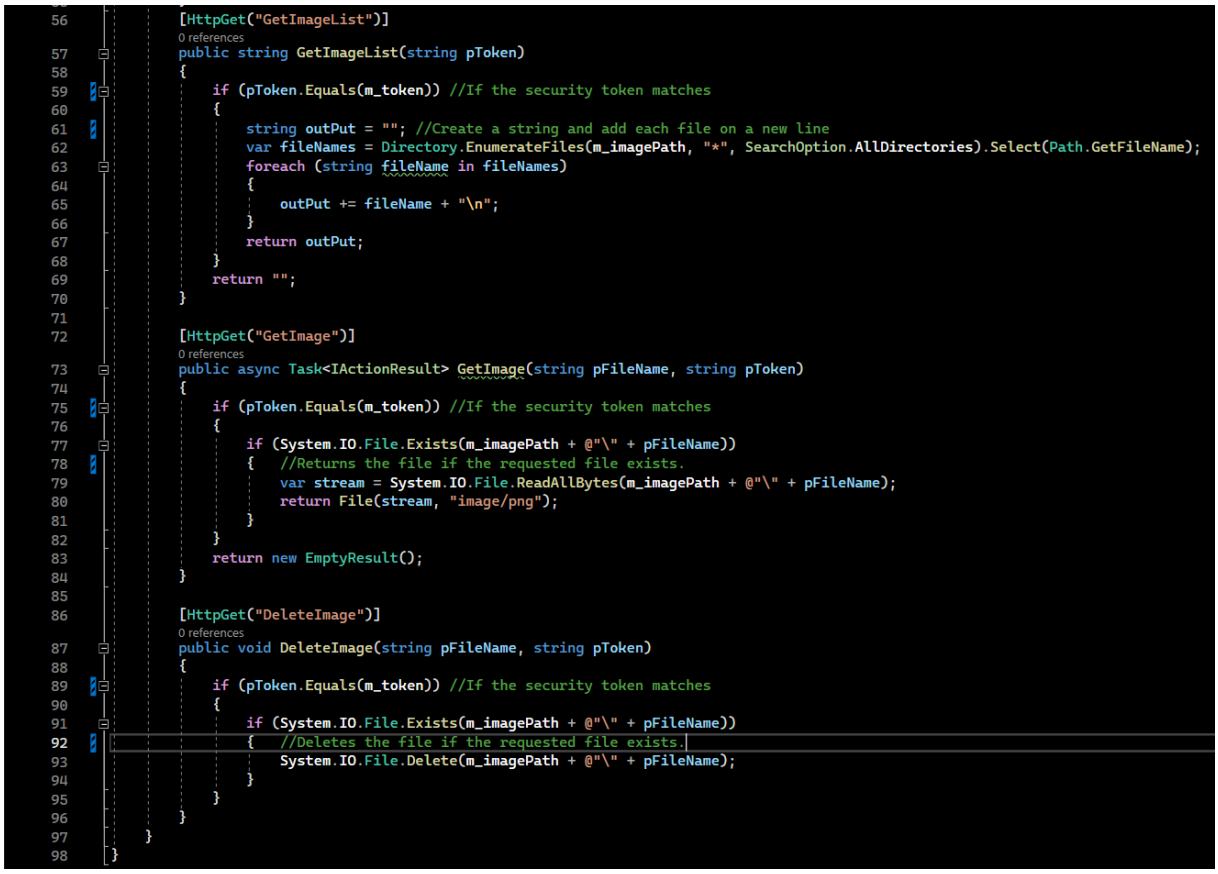
```
1 reference
public static void UploadImage(byte[] pImageBytes) //Create an instance if it doesn't exist yet
{
    if (!instance)
    {
        instance = FindObjectOfType<NetworkController>();
        if (!instance)
        {
            instance = new GameObject("NetworkController").AddComponent<NetworkController>();
        }
    }
    instance.DoImageUpload(pImageBytes);
}

1 reference
private void DoImageUpload(byte[] pImageBytes)
{
    StartCoroutine(ImageUpload(pImageBytes));
}
1 reference
private static IEnumerator ImageUpload(byte[] pImageBytes)
{
    WWWForm form = new WWWForm(); //Create a new uploadform
    form.AddField("ImageID", 1);  //Fill the form ID
    form.AddField("Token", m_APIToken); //Fill the form Token for security
    string fileName = String.Format("{0}", DateTime.Now.ToString(stringFormat)); //Generates file name as current time
    form.AddBinaryData("File", pImageBytes, fileName + ".png", "image/png"); //Set the byte array form
    using (UnityWebRequest uwr = UnityWebRequest.Post(m_APIURLImageUpload, form)) //Connect to API
    {
        yield return uwr.SendWebRequest(); //Sends the form
        if (uwr.result == UnityWebRequest.Result.ConnectionError || uwr.result == UnityWebRequest.Result.ProtocolError)
        {
            Debugger.Log("On sending image " + uwr.error);
        }
        else
        {
            Debugger.Log("Succesfully uploaded image.");
        }
    }
}
```

Figure 18 Send byte array to web API

```
[HttpPost]
0 references
public Task<ImageFile> Post([FromForm] ImageFile objFile)
{
    ImageFile imgFile = new ImageFile();
    try
    {
        if (objFile.Token.Equals(m_token)) //If the security token matches
        {
            imgFile.ImageID = objFile.ImageID;
            imgFile.File = objFile.File;
            imgFile.Token = objFile.Token;
            string fileName = imgFile.File.FileName;
            if (!Directory.Exists(m_imagePath)) //Check if directory exists else create it
            {
                Directory.CreateDirectory(m_imagePath);
            }
            int i = 0; //Check if filename already exists, else increment versions with (#)
            string nameWithoutExtension = Path.GetFileNameWithoutExtension(fileName); //Get file name
            string extension = Path.GetExtension(fileName); //Get the file type
            while (System.IO.File.Exists(m_imagePath + @"\" + fileName)) //If the name already exists, put a (#) behind it
            {
                fileName = nameWithoutExtension.Trim() + " (" + i + ") " + extension;
                i++;
            }
            //Create the file
            using (FileStream filestream = System.IO.File.Create(m_imagePath + @"\" + fileName))
            {
                imgFile.File.CopyTo(filestream);
                filestream.Flush();
            }
        }
    }
    catch (Exception ex)
    {   //If an error occurs, write it to the debug log.
        FileDebugController fileDebugController = new FileDebugController();
        fileDebugController.Write(ex.ToString(), objFile.Token);
        throw;
    }
    return Task.FromResult(imgFile);
}
```

Figure 19 ImageUpload



```
56          [HttpGet("GetImageList")]
            0 references
57          public string GetImageList(string pToken)
58          {
59              if (pToken.Equals(m_token)) //If the security token matches
60              {
61                  string outPut = ""; //Create a string and add each file on a new line
62                  var fileNames = Directory.EnumerateFiles(m_imagePath, "*", SearchOption.AllDirectories).Select(Path.GetFileName);
63                  foreach (string fileName in fileNames)
64                  {
65                      outPut += fileName + "\n";
66                  }
67                  return outPut;
68              }
69              return "";
70          }
71
72          [HttpGet("GetImage")]
            0 references
73          public async Task<IActionResult> GetImage(string pFileName, string pToken)
74          {
75              if (pToken.Equals(m_token)) //If the security token matches
76              {
77                  if (System.IO.File.Exists(m_imagePath + @"\" + pFileName))
78                  {   //Returns the file if the requested file exists.
79                      var stream = System.IO.File.ReadAllBytes(m_imagePath + @"\" + pFileName);
80                      return File(stream, "image/png");
81                  }
82              }
83              return new EmptyResult();
84          }
85
86          [HttpGet("DeleteImage")]
            0 references
87          public void DeleteImage(string pFileName, string pToken)
88          {
89              if (pToken.Equals(m_token)) //If the security token matches
90              {
91                  if (System.IO.File.Exists(m_imagePath + @"\" + pFileName))
92                  {   //Deletes the file if the requested file exists.
93                      System.IO.File.Delete(m_imagePath + @"\" + pFileName);
94                  }
95              }
96          }
97      }
98  }
```

Figure 20 ImageList GetImage DeleteImage