Academy ACT

Graduation Thesis

Student	Remy Tapper (435962)
Academy	ACT, HBO-IT
Course	Afstuderen/Graduation Project HBO-ICT
Company	Brease
Company supervisor	Colin Bosman
Saxion coordinators	Lonneke Brands; Erik Leeuw; Willem Prakken; Anja Renkema; Tina Uda;
Internship coordinator	Dick Heijink
Date	13/06/2021

Document History

Version	Modifications	Author	Date
0.1 draft	Creation of the document	Remy Tapper	15/02/2021
0.2	Created initial structure	Remy Tapper	23/02/2021
0.3	Added research questions & problem analysis	Remy Tapper	24/02/2021
0.4	Added theoretical framework chapters	Remy Tapper	25/02/2021
0.5	Added Text-to-Speech research, subchapters	Remy Tapper	09/03/2021
0.6	Added table of figures, table of tables, refactoring	Remy Tapper	10/03/2021
1.0	Processed feedback. First concept.	Remy Tapper	11/03/2021
1.1	Processed feedback of Internship coordinator	Remy Tapper	15/03/2021
1.2	Finished processing feedback of Internship coordinator	Remy Tapper	16/03/2021
1.3	The initial start of Voice-to-Text research.	Remy Tapper	17/03/2021
1.4	Finished Voice-to-Text research.	Remy Tapper	18/03/2021
1.5	Added Data collection and Analysation methods.	Remy Tapper	19/03/2021
1.6	Added Voice-to-Text tests	Remy Tapper	22/03/2021
1.7	Added Text-to-Speech results, updated accuracy test.	Remy Tapper	23/03/2021
1.8	Updated tables.	Remy Tapper	25/03/2021
1.9	Added Lo-Fi wireframe chapter, added Hi-Fi wireframe chapter.	Remy Tapper	01/04/2021
1.10	Added sequence diagrams, added Lo- and Hi-Fi designs. Finished wireframes chapter.	Remy Tapper	06/04/2021
1.11	Updated Methodology and Designs chapter.	Remy Tapper	07/04/2021
2.0	Processed feedback Company supervisor. Second concept.	Remy Tapper	09/04/2021
2.1	Processed feedback Internship coordinator	Remy Tapper	14/04/2021
2.2	Processed feedback from Internship coordinator, updated Text-to-Speech, Voice-to-Text, added Realise phase to the Methodological approach chapter.	Remy Tapper	15/04/2021
3.0	Processed feedback. Submitted concept.	Remy Tapper	06/05/2021
3.1	Processed feedback Internship coordinator, restructured Results chapter.	Remy Tapper	11/05/2021

3.2	Finished discussion, conclusion, implementation, software competencies.	Remy Tapper	25/05/2021
3.3	Finished Realisation, added Class diagram, updated existing diagrams, styling, and formatting.	Remy Tapper	26/05/2021
3.4	Styled entire document, added missing figure references	Remy Tapper	27/05/2021
3.5	Processed feedback from the Company supervisor	Remy Tapper	28/05/2021
3.6	Processed feedback from the Company supervisor	Remy Tapper	29/05/2021
4.0	Finalised report. First official concept	Remy Tapper	30/05/2021
4.1	Processed feedback of concept.	Remy Tapper	05/06/2021
4.2	Processed feedback of concept.	Remy Tapper	07/06/2021
4.3	Processed feedback of concept.	Remy Tapper	09/06/2021
4.4	Introduced each chapter, updated introduction, processed feedback of concept.	Remy Tapper	11/06/2021
4.5	Processed feedback, wrote additional information for discussion and conclusion.	Remy Tapper	12/06/2021
5.0	Styling, finalized thesis.	Remy Tapper	13/06/2021

Distribution

Name	Role	Date of issue	Version
Colin Bosman	Company supervisor	10/03/2021	0.6
Dick Heijink	Internship coordinator	11/03/2021	1.0
Colin Bosman	Company supervisor	07/04/2021	1.11
Dick Heijink	Internship coordinator	08/04/2021	2.0
Dick Heijink	Internship coordinator	06/05/2021	3.0
Colin Bosman	Company supervisor	28/05/2021	3.4
Dick Heijink	Internship coordinator	30/05/2021	4.0
Colin Bosman	Company supervisor	11/06/2021	4.4
Colin Bosman	Company supervisor	12/06/2021	4.5
Dick Heijink	Internship coordinator	13/06/2021	5.0

Abstract

In this project, an application has been created in the form of a virtual assistant to answer the question of whether or not a virtual assistant would be of added value to a reception and how particular concepts such as Voice-to-Text, Text-to-Voice in combination with a chosen chatbot could play a role in this.

To substantiate this question, a comprehensive analysis has been performed regarding the different libraries applied to this project. Next to this, a Multi-criteria analysis has been created to compare said libraries and see which library would work best concerning the context of the project.

Besides this, a survey was conducted to get a clearer view of how organisations currently fill the reception role and whether or not a virtual assistant would be a good addition or replacement for these organisations.

Findings from the analysis phase indicated that based on the criteria of the case, The Web Speech API would work best as a Voice-to-Text service, and Google-TTS would be the best choice as a Text-to-Speech service since they suffice the necessary criteria that this project requires.

The results gathered from the survey showed that a fair amount of organisations are interested in having a virtual receptionist as addition to their current reception state . Out of all the different features, a virtual assistant might have to offer, wayfinding, visitor registration, and greeting visitors were the top three features most organisations would like a virtual assistant to have.

Although not every organisation would prefer to use a virtual assistant over a staffed one, there is demand for it and depending on the complexity of the assistant, a virtual assistant will be able to replace the features of a staffed assistant.

Table of Contents

List of Figures

List of Tables

Table of Abbreviations

#	Abbreviation	Explanation
1	JS	JavaScript
2	API	Application Programming Interface
3	POC	Proof of Concept
4	VTT	Voice-to-Text
5	TTS	Text-to-Speech
6	AI	Artificial intelligence
7	Lo-fi	Low fidelity
8	Hi-fi	High fidelity
9	UI	User Interface
10	UX	User Experience
11	E2E	End-to-End

Glossary

#	Notation	Definition
1	On-premises software	A piece of software that is installed and runs on a computer locally rather than in a cloud environment.
2	Open-source software	A type of software of which the original source code is made freely available and may be modified or redistributed.
3	WaveNet	Deep neural network for generating raw audio which is used to generate natural-sounding speech.
4	Text-to-Speech	Form of speech synthesis that converts text into a voice output (speech)
5	Voice-to-Text	Type of speech recognition that converts speech to written text.

1. Introduction

Technology plays a much more significant role these days. From new smartphones to artificial intelligence, every digital innovation is developed to solve a problem or to automate an action. An example of this is a reception. Where organisations such as multi-company buildings or office buildings have cut back on receptions, new solutions appear, such as digital doorbells or digital receptions. One step further would be a virtual reception where you will not just have to use a touch screen or hit keys but where you can explain your reason for arrival by just using your voice.

This thesis aims to provide information about the concept "virtual reception" and how this can be realised in combination with certain concepts such as Voice-to-Text, Text-to-Voice and virtual avatars.

This report will conclude the graduation internship done at Brease. The internship period started from February 2021 till the start of July 2021 and was part of year four of the education HBO-IT, at Saxion under the guidance of my internship coordinator Dick Heijink and company supervisor Colin Bosman.

1.1 Project structure

The structure of this project consists of a total of 10 chapters. These chapters are as follows:

- Chapter 1 serves as the introduction to this report and gives some information about the project and company background.
- Chapter 2 contains information about the problem that is being solved, the assignment, a list of requirements and the research questions that are to be answered.
- Chapter 3 contains information about the different criteria that have been researched during the graduation period.
- Chapter 4 contains the Methodology chapter and aims to inform how specific procedures or techniques have been approached, analysed, and processed during this project.
- Chapter 5 contains all the results gathered during the time working on this project, such as research results, designs, project results and the overall Scrum process
- Chapter 6 shows information about each software competence and how these have been met during the graduation period.
- Chapter 7 contains answers to the previously mentioned research questions and some more information regarding the finished requirements of the project.
- Chapter 8 contains information about the continuation of the project and how it can be picked up for future development.
- Chapter 9 contains references to all the links that have been cited in this report by following the APA standards.
- Chapter 10 contains all the links to information that has been used during this project and creation of the report, but has not been cited in this report.

1.1 Company background

Brease is a company established in 2019 by Colin Bosman and Edwin Swaak located in Hengelo. They are a software company which develop software solutions for mainly the health-care sector and the social domain ("Sociaal Domein") but also for parties in (partly) other markets, such as Innivo, Intelligent building solutions.

One of these solutions of Brease is their application" Dialoog". This application handles the communication and administration around the registration- and intake phase of a health care organisation.

2. Problem analysis

This chapter is aimed to determine the problem that is trying to be solved. Next to this, an overview of the assignment is given, combined with a set of requirements created from this assignment. Finally, this chapter includes a set of research questions which my graduation will focus on.

2.1 Problem context

The world around us gets more and more digital which leads to more efficient services. An example of this is the introduction of a digital or virtual reception. Where organisations such as multi-company buildings and large office buildings have had their receptions in recent years cut back, new solutions such as a digital doorbell or a digital reception offers an alternative. A step further is a virtual receptionist, where you can not only choose with keys or touch but where you can:

- See a visual representation of a 2D or 3D' avatar'- receptionist.
- Be welcomed through a voice.
- Make use of your voice to let the application know what you are coming for.
- Retrieve instructions on how to navigate to the location of your appointment.

Besides this, there are already certain solutions to this problem in terms of possible competition, for example, the reception robot. Besides greeting the visitor, they can also guide him/her to their destination.

2.2 Problem statement

As of right now, a partner of Brease, Innivo, is interested in research regarding a virtual assistant and how this would work in combination with their current solutions. Innivo themselves already has solutions for floorplans, door signs and big screens containing lists of meetings. A digital reception could be an excellent addition.

2.3 Assignment

A partner of Brease, Innivo, is active in the fields of big buildings and hospitals and is interested in research regarding a virtual assistant and how this would work in combination with their current situation concerning reservations, smart building use and "wayfinding" within a company.

One of the components that will be used for this is a previously researched chatbot framework. My task is to give this chatbot framework a voice and face, in combination with voice input.

This application will initially be written in Vue.JS, but depending on hardware compatibility and existing libraries combined with the software, there is the possibility of using a different framework or a native solution.

The application will contain the following components:

- Text-to-Speech
- Voice-to-Text
- Visual representation of a "face"
- Handling of not recognized voice inputs.
- User detection
- Navigation through the building

As of right now, there are no designs for the application. These will also have to be made.

Besides the main points which are mentioned above, there is also the following additional functionality.

- Initial screen: video or screensaver of a static image or logos of the different companies within the building. Next to this, there should be a button to manually start the flow if the virtual assistant does not recognize the user.
- Assistant screen showing a visual representation with subtitles of the spoken message. Besides this, there should be a button to switch to a touch version for hearing impaired users.
- Possibility of the avatar pointing the right way to the user's destination.
- A way to show the route on a floor plan.
- Handling actions of different users. Based on their interaction with the application, different services will be called, making the result different.
- Information screen: The application will make use of external services. When there is no internet connection available, information should be shown to help the visitor. An example of this could be a floorplan of the building.
- The ability to show a user-specified floorplan by scanning an NFC or QR code.

Because of the complexity of this project, not all features are expected to be completed in time. Because of this, features will be prioritized and implemented based on their priority.

2.4 Requirements

2.4.1 Requirement prioritization

This project is aimed to finish as many requirements as possible. By prioritizing the most important requirements first, we can ensure that these will be met.

The requirement prioritization method is used to determine which requirements should be implemented first. Due to the total time of this project in combination with the number of requirements, there is a chance that not every requirement will be finished in time. Since every requirement is aimed to be finished before the end of this project, there is no "Won't" section.

Must

The application must need:

- A start screen showing an overview of all of the existing companies or departments.
- An avatar representing the virtual assistant.
- A Voice-To-Text implementation.
- A Text-To-Speech implementation
- A button to manually start the application.
- The virtual assistant to handle specific actions, for instance, a route description to a room where the user knows the room number, handling appointments or helping visitors without any planned appointment.

Should

The application should:

- Be able to restart the process of the application.
- Automatically detect users.
- Show an information screen on connection loss.

- Have a route description of the user's destination.

Could

The application could:

- Have a personal route through a scannable NFC/QR code.
- Have on/off times.
- Have a screen to call an employee.
- Have a setting to switch the active language of the application.

Non-functional requirements

In addition to the functional requirements mentioned above, given the context of the application, the following non-functional requirements can be created:

Must

The application must:

- The application must interact with the Rasa chatbot environment.
- The application must follow the code conventions from Brease.

Should

The application should:

- Work completely hands-free.

2.4.2 User stories

To get a better overview of how these requirements can be met, they are turned into user stories. Please check out the chapter "User Stories" inside the functional documentation for a complete overview of all the user stories.

2.5 Research questions

Based on the information mentioned above, the project assignment can be summarized as the following main research question (***MQ***):

MQ. *What is the added value of a virtual assistant to a digital reception that greets and guides visitors?*

To answer this question and to get a better understanding, the main research question can be split up into the following sub-questions:

SQ1*. How well does the solution work compared to a touchscreen solution?*

SQ2*. Is the solution suited for people with less knowledge of technology?*

SQ3*. How can an incoming person be detected?*

SQ4*. How can the virtual assistant be visualised?*

SQ5*. Which parts should be developed or created with existing solutions?*

SQ6. *Is a virtual assistant of added value to big buildings or hospitals?*

SQ7. *Which features of a virtual assistant are of added value to a possible end-user?*

3. Theoretical Framework

This chapter will include all the different subjects that have been researched throughout the entire graduation period. To get a better indication of how these new technologies work, Proofs-of-Concept have been made. This has been done by following several tutorials and essential start guides. Additional and more extended research can be found in the research report.

3.1 Rasa(x)

In a previous project, a chatbot has been researched and developed using Rasa. Rasa is an open-source machine learning framework for automated text and voice-based conversations (*Introduction to rasa open source* 2021). For a more in-depth explanation about Rasa, please check out the research document.

The virtual assistant will serve as a wrapper around the chatbot and call specific actions based on each question. In terms of this part, research has been conducted about what Rasa is, in combination with setting up a proof of concept to figure out how it works. This proof-of-concept gave more insight into how Rasa works and what you can do with it, and how it can be connected to the component that will be researched.

Besides Rasa, there is also Rasa X. Rasa X is a deployable UI tool which can be used to build and test the chatbot assistant and to improve and refine answers and conversations.

3.2 Voice-to-Text

Voice-to-Text is a type of speech recognition that converts speech into written text. This text can then be sent to the Rasa chatbot, which will return an action based on the given text. As part of this internship, the different possibilities of this concept have been looked into. Existing solutions have been compared to find the best possible solution.

3.3 Text-to-Speech

Text-to-Speech, also known as TTS, is a type of technology that reads digital text out loud. TTS is regularly used as an accessibility feature by people with visual impairments.

In this project, TTS will be used to convert the given text into a speech file. This speech file will then be played to give the virtual assistant a voice. To get an idea of which technology offers the best solution, several candidates have been compared in combination with different criteria. Most of the candidates offered an online demo environment to test their Text-to-Speech product. The companies where this was not the case were contacted. These companies offered a time period where their solution could be tested in the form of a sandbox mode. To get a precise indication of how well their solution would solve our problem, proof of concepts has been made.

3.3.1 WaveNet

A WaveNet generates speech that sounds more natural than other text-to-speech systems. It synthesizes speech with a more human-like emphasis and inflexion on syllables, phonemes, and words. On average, a WaveNet produces speech audio that people prefer over other text-to-speech technologies. *(Standard and WaveNet voices | cloud Text-to-Speech Documentation)*

Unlike most other text-to-speech systems, a WaveNet model creates raw audio waveforms from scratch. The model uses a neural network that has been trained using a large volume of speech samples. During training, the network extracts the underlying structure of the speech, such as which tones follow each other and what a realistic speech waveform looks like. *(Standard and WaveNet voices | Cloud Text-to-Speech Documentation)*

3.4 Vue

This front-end application of this project is written in Vue. Vue is a front-end JavaScript framework used to build one-page applications and web interfaces.

3.4.1 Vue 2 vs Vue 3

One of the desires of Brease was that this project would be written in Vue 3. Vue 3 is a successor on Vue 2, released at the end of last year and rewritten to operate more smoothly and powerfully. Significant changes and updates between Vue 2 and Vue 3 will be discussed in this subchapter and compared.

3.4.1.1 Creating independent instances

Vue instances are now created with "createApp()". This will return a fresh instance that is not affected/changed by any global configurations such as plugins, mixing, etc. The new architecture allows us to have two or more isolated Vue instances that do not share any features by default, even if they are created in one file. (*The Benefits of the New Vue 3 App Initialization Code* 2021). Because of this, you can ensure that these instances are isolated from each other, making it easier to perform actions such as unit testing.

3.4.1.2 Composition API

One of the essential features of Vue 3 is the composition API. Vue 2 makes use of properties such as data, computed, methods and watchers. Although this organizing logic can still be used and will work in most cases, as the component gets bigger, so will this. This can lead to components being hard to understand and maintain since different pieces of code that work together are scattered throughout the entire component.

This is where the Composition API comes in. This API defines components using the "setup()" function rather than using the properties mentioned above. The Composition API allows us to combine code related to the same logical concern, allowing for a better structure and overall organization in components. Besides this, more advantages of the composition API are better code completion, easier to re-use code and no longer having to use "this" inside of components.

To better indicate how the Composition API structure would look, please check out "Figure 1: Vue3 Options API vs Composition API**Error! Reference source not found.**". Although the code is not supposed to be readable, the main focus is laid on the coloured sections which displays how code is related with each other.

Figure 1: Vue3 Options API vs Composition API

3.4.1.3 Lifecycle hooks

Each Lifecycle event is separated into two different hooks. These hooks are called before and after the event. Vue has the following four main events:

- Creation – Runs when a component is created.
- Mounting – Runs when the DOM is mounted.
- Updates – Runs when reactive data is updated.
- Destruction – runs before an element is destroyed.

In the composition API, lifecycle hooks will have to be imported into the project before being used. This will help to keep the project lightweight. All of the lifecycle hooks will have to be invoked inside the "setup()" function. Previously, all the lifecycle hooks were only possible inside components, and now they can be used everywhere, making the code very re-usable. In Vue 3, the updated lifecycle hooks look as follows:

Vue 2	Vue3
beforeCreate()	use setup()
created()	use setup()
beforeMount()	onBeforeMount()
mounted()	onMounted()

beforeUpdate()	onBeforeUpdate()
updated()	onUpdated()
beforeDestroy()	onBeforeDestroy()
destroyed()	onUnmounted()
errorCaptured()	onErrorCaptured()

Table 1:Vue 2 vs Vue 3 Lifecycle Hooks

As can be seen in Table 1:Vue 2 vs Vue 3 Lifecycle Hooks, both the "beforeCreate()" and "created()" hook is no longer needed to be explicitly defined. This is because the "setup()" function is run around both lifecycle hooks. This means that any code that would normally be written in these hooks should now be written directly in the "setup()" function.

3.4.1.4 Ref variable

In Vue 3, any variable can be made reactive by using the new "ref" function as follows.

```
import { ref } from 'vue'

const counter = ref(0)

console.log(counter) // { value: 0 }
console.log(counter.value) // 0

counter.value++
console.log(counter.value) // 1
```

Figure 2: Vue 3 - Ref code snippet

The "ref" function takes the argument and returns it wrapped in an object. This object has a value property, which can be used to access or mutate the value. Doing this is required to keep the behaviour unified across different data types in JavaScript. This is because, in JavaScript, primitive types such as Strings or Numbers are passed by value and not by reference. Having a wrapper object around any value allows us to safely pass it across the entire application without worrying about losing its reactivity. (*# introduction - Vue.js*)

3.4.1.5 Fragments

In Vue 2, multi-root components were not supported, resulting in a warning when a user created one. To fix this error, components were wrapped in a single <div>. In Vue 3, components can have multiple root nodes. This means that component functionality can be bound into a single element without creating a redundant DOM node.

4. Methodology

This chapter will give a broader explanation of how research has been conducted. Next to this, details about the different ways data has been collected are explained in combination with the different analysis methods. Finally, methodological choices are discussed and substantiated.

4.1 Methodological approach

Throughout the entire internship period, the following phases have been followed (in chronological order):

4.1.1 Analysis

The analysis phase was the first phase that was being worked on. Based on the assignment, a plan of approach has been created. Besides this, collected data such as libraries and new technologies that were going to be used were researched. This subchapter will explain the main parts of the Analysis phase, such as creating the requirements, problem analysis, and multi-criteria analysis.

4.1.1.1 Plan of Approach

The first thing that was done was the creation of the Plan of Approach. This played an essential role in the overall project since it helped to identify goals, analyse problems, identify requirements and the subjects that were important to complete the requirement analysis.

Next to this, a Gantt Chart was created, displaying a planning of the entire graduation period. This helped maintain planning and structure to the entire workflow of the project.

4.1.1.2 Multi-criteria analysis

To find the optimal option between different types of software, a multi-criteria analysis has been conducted.

Multi-criteria analysis is used to compare different candidates based on a set of criteria. Candidates for the multi-criteria analysis were chosen by making use of Google. The multi-criteria analysis consists of the following parts:

Criteria

To make sure the right software is chosen, it is essential to declare the proper criteria covering all the different essential aspects. The criteria have been created by looking at the fundamental concepts of Text-to-Speech. Besides, Brease has been asked which criteria they find essential to ensure all the criteria have been covered. Based on the feedback provided by Brease, a final set of criteria has been created.

Standardization

Since the scores of the different criteria can be expressed in different ways, it is essential to standardize these scores, thus comparing the different criteria easily. For example, these values have been estimated by the highest value is the lowest price (free) and the lowest value is the most expensive and thus highest price.

Weightage

To ensure the criterion we find the most important, has a higher priority compared to the less essential criteria, we can "weigh" each criterion. The weight of each criterion has been estimated and discussed several times with the company supervisor to make sure the criterion they find most important has a higher priority.

4.1.1.3 Survey
To determine the current state in which organisations fulfil their reception function, a survey has been conducted. The questions were initially created by the intern and thoroughly discussed with the company supervisor and owner. Based on their feedback, adjustments had been made. Besides the points mentioned above, participants were also asked to explain the different features they find important in a virtual reception. Based on their feedback, adjustments have been made and additional requirements have been applied during the development process which occurred during the realisation phase.

4.1.2 Design
The second phase which was being worked on was the design phase. Based on the requirements provided by the requirement analysis done in the analysis phase, architectural diagrams such as a flow diagram and sequence diagram were created. These diagrams displayed the overall architectural structure of the application and the external services that it communicates with. The creation of these diagrams helped to better understand how these technologies would function and could be applied in this project.

Besides diagrams, low fidelity (or Lo-Fi) wireframes were created to map out the interface's shell, screens, and basic information. The wireframes were made by analysing the gathered requirements from the previous phase and displaying these in the design by trial and error to determine which way looked good. The first draft of these designs was made by the intern and thoroughly discussed with the company supervisor. Based on the feedback, the designs were adjusted and updated.

After the Lo-Fi wireframes were finished, a Hi-Fi prototype was created. The Hi-Fi prototype serves as a representation of what the final product would look like. Just like during the creation of the Lo-Fi wireframes, the initial draft was created by the intern and based on the feedback from the company supervisor, adjustments were made.

4.1.3 Realise
The fourth phase is the realisation phase. Even though one developer created this project, the Scrum framework was used throughout the entire project. This is because it is the main project management framework used at Brease and because of experience gathered from school projects and previous internships.

Based on the previously created requirements, user stories, designs and survey results, tickets for each user story were created. Tickets were created on Jira, a platform used to manage an overview of the project and its user stories. Jira is mainly used as a scrum board/backlog. This backlog contained a list of all the existing user stories that have not been completed yet.

During the internship, work was done in sprints. Each sprint was two weeks and contained a set of user stories picked up from the backlog. Each user story consists of a name and acceptance criteria. Acceptance criteria was derived from the user story itself and discussed with the company supervisor to make sure the suspected outcome of the user story was clarified. Besides story points were determined for each user story. The intern estimated these story points and indicated how long it would take the developer to finish the story. Since not every user story contained the same amount of work, this indicated how many stories could be finished in one sprint. These story points were estimated during the sprint planning. This sprint planning occurred before the start of each new sprint.

4.2 Data collection

To gather the data required for my research, the following methods have been used:

Google/Google Scholar

The first and primary way of gathering data has been through using Google and Google Scholar. Information about different concepts that were unclear to me at the start of this period, such as Voice-to-Text and Text-to-Speech, helped to understand how these concepts exactly work and would be realised within an application.

Proof of Concept(POC)

To get a better idea of how certain solutions would work in practice, several Proofs of Concepts have been created. This way of gathering data helped get a better indication of how this solution would work in our context. Several examples in which the POC strategy was applied were Text-to-Speech and Voice-to-Text libraries, Rasa in combination with Rasa-X and Vue 3 combined with Vite.

Survey

A survey has been carried out to get a better indication of which core aspects are important for possible end-users. To keep the results measurable, the survey mainly consisted of multiple-choice questions. Questions were created by the intern and thoroughly discussed with the company supervisor and company owner. These discussions resulted in lots of feedback which made sure the right questions were asked. This survey was conducted by email and given to company complexes and municipalities in The Netherlands.

Interview

In the 4th week of the internship, an interview has been conducted with a possible partner, VidiNexus. Their take on digital screens at entrances, together with several available showcase products, gave a refreshing look at the many different possibilities a digital reception could offer.

4.3 Analysis methods

To indicate how data has been processed and analysed, methods from the DOT framework have been used. Over the entire internship period, the following methods have been used:

4.3.1 Library

4.3.1.1 Literature study

During the analysis phase, a literature study such as Google Scholar was used to research technologies such as Concatenative synthesis and Wavenet. They indicated what these terminologies meant and how these technologies could be applied in the application. Findings regarding these subjects can be found in the Research Report.

4.3.2 Field

4.3.2.1 Problem analysis

At the very start of this project, a problem analysis was done to determine the current situation of the problem. Besides the context of the problem, the problem statement was also determined. Finally, analysing which solutions already existed indicated how the problem was currently being dealt with.

4.3.2.2 Interview
During the analysis phase, an interview has been conducted with a possible partner. This interview gave more insight when it comes to possibilities and exciting concepts which a digital reception could offer.

4.3.2.3 Survey
During the analysis phase, a survey was given to figure out which aspects of the solution were most essential and interesting for the end-users. Next to this, based on the results, there was a better idea if companies were interested in a virtual assistant.

4.3.3 Workshop
4.3.3.1 Code review
Code reviews are part of the Scrum process. At the end of each user story, the code is being checked to ensure everything works correctly and no harmful code is being pushed to develop. When writing complex code, the code has been discussed with the company supervisor to see if there would be a better or "cleaner" way of doing it. Code reviews ensure that the code that is being pushed stays consistent and readable. Besides this, another code review at the end of graduation period has been done to ensure the code conformed to the Brease coding standards.

4.3.3.2 Prototyping
The research strategy, prototyping was done multiple times throughout the entire project. At the start of the project, during the analysis phase, specific libraries, and concepts such as Text-to-Speech, Voice-to-Text, Rasa and Vue were set up as proof of concept to discover technical limitations and possibilities to offer. Prototyping like this offered a clear indication of whether using these technologies would be a correct solution to solving the problem.

Next to the proofs of concepts made during the design phase, a low- and high-fi design was made to determine what the application would look like. Creating the low fidelity designs served as a skeleton of the application and helped indicate where certain functionalities would be and what they would look like. The high-fi design served as a prototype of the user interface and helped test the application's usability and identify issues in the workflow.

Because there were no designs in mind, there was much feedback from my company supervisor, company owner and other employees. The feedback helped create a design that both parties (the student and the company) were happy with.

The creation of architectural diagrams such as the sequence- and flow diagram gave a clear indication of how the application would function and in which order (or flow) specific actions would occur.

4.3.3.3 Requirements prioritization
To determine which requirement had a higher priority compared to other requirements and thus had to be implemented first, the requirements prioritization method was utilized. This was done by creating a list of requirements and indicating the priorities by using the MoSCoW method. These requirements were then converted into user stories which can be found in the attachment: "User Stories" or in the functional design.

4.3.4 Lab
4.3.4.1 Usability testing
During the development process, several employees at Brease were prepared to test the functionality of the application. Having people test these parts of the application in such an early development stage

helped find specific issues and bugs in time and helped to correct them before the application went live.

Besides testing the code, the High-Fi design was also tested. This indicated whether the created design was user-friendly and covered all the required functionalities. Besides the fact that the application should be used without any type of touch interaction, the option should still be available for people that would instead make use of it or if, for some reason, the voice option does not function properly. Since it is impossible to implement voice interaction into the prototype designs, just the touch interaction has been implemented. Using Hi-fi design tools such as Figma helped create a prototype that would be identical to the result and provided an easy way to create and edit the prototype/design.

4.3.5 Showroom

4.3.5.1 Product review

At the end of each sprint, a product review was done. During this review, user stories were discussed, and results were shown. Based on these reviews, the client and development team are aware of the entire project's status, met goals and requirements.

4.3.5.2 Static program analysis

To find vulnerabilities, weak spots, and an overall impression of how well the code was written, tests were written. Testing the code helped to find bugs and ensures that consistent and clean code gets pushed to the develop and master branch. This way, there is more attention to well-written code and vulnerabilities are found much quicker.

5. Results

This chapter presents an overview of all the different results, outcomes and deliverables that have been created during the internship.

5.1 Research

5.1.1 Voice-to-Text - Multi-criteria analysis

A multi-criteria analysis has been conducted to determine which library/plugin is considered the best options for a Voice-to-Text implementation.

5.1.1.1 Candidates

For this Voice-to-Text research, the following candidates have been compared with each other:

- **Google Cloud Speech to Text**
  The Google Cloud platform offers a wide variety of cloud computing services and uses their advanced deep learning neural network algorithms for automatic speech recognition (*Speech-to-Text: Automatic speech Recognition | Google Cloud*).
- **Microsoft Azure Speech to Text**
  Microsoft Azure is a cloud computing service created by Microsoft. Just like Google, they offer a Speech-to-Text service.
- **Web Speech API**
  The Web Speech API is a JavaScript API that allows for adding speech recognition to web pages. It is created by an open community of developers.
- **Vosk**
  Vosk is an offline open-source speech recognition toolkit. (Alphacep, *alphacep/vosk-api*) Vosk makes use of language models that run offline to recognize speech.
- **Annyang**
  Annyang is a Javascript speech recognition library that allows users to control the website with voice commands. Annyang relies on the browser's speech recognition engine, which performs recognition in the cloud (TalAter, *Talater/annyang* 2020).
- **IBM Watson Speech to Text**
  Watson is an IBM owned, open, multi-cloud platform that automates the AI lifecycle (*IBM Watson*). Watson is an AI, powered by machine learning that offers different services such as Speech-to-Text, which can be integrated into applications.

5.1.1.2 Criteria

The libraries mentioned above have been compared with each other based on a set of criteria. This criteria has been created by looking at the fundamental concepts of Voice-to-Text. Besides, Brease has been asked which criteria they find essential to ensure all the criteria have been covered. Based on the feedback provided by Brease, the following set of criteria has been created:

- **Accuracy**
  When the visitor speaks to the virtual assistant, the Voice-to-Text library should precisely detect which words exactly get spoken.
- **Documentation & Support**
  This criterion determines how well the library has been documented and how well it's being supported. The documentation should be easy to understand and thus allow for easy implementation.

- **Price per month**
  Most libraries offer a monthly payment based on the spoken audio per hour. Some libraries offer a free service for this.
- **On-premises**
  An on-premises solution means that the software/library will run locally on the system. Most libraries require a call to an external service to retrieve the data.
- **Custom word detection**
  When a visitor starts talking to the virtual assistant, the visitor might mention certain words that are not available in the dictionary. These words can for example be company names, locations, etc. These words should still be detected to guarantee that the exemplary service will be called.
- **Open-source**
  By this criterion, we define whether or not any user is granted the rights to use, distribute, study and change the software.

Next to this, the candidates have been chosen based on the following conditions being true:

- **Real-time speech recognition**
  When a visitor starts talking to the virtual assistant, the time it takes the library to detect the language must be happening in real-time. Having the library run and return text in real-time will result in the visitor being helped much quicker and makes for an overall smoother conversation and interaction.
- **Dutch language support**
  Since this application will be initially made for Dutch users, the voice-to-text library must support the Dutch language.

5.1.1.3 Standardization

To compare the different criteria with each other, the first thing that has to be done is to standardize the different measurements. In this instance, the standardization tables look as follows:

	Lowest value (0)	Highest value (1)
Number	$1,50/audio hour	$0 (Free)
Rating	--	++
Yes or No	No	Yes

Table 2: Voice-to-Text Standardization table

5.1.1.4 Weightage

Since not every criterion weighs the same, every criterion will be weighted with each other to determine which criterion is the most important. These weights have been thoroughly discussed with the company supervisor to ensure the criteria they find the most important have a bigger priority. In the end, the weights in the table for this Voice-to-Text research looks as follows:

	weightage	
	%	decimal
Accuracy	30%	0,30
Documentation & Support	10%	0,10
Price per month	15%	0,15

On-premises	13%	0,13
Custom word detection	25%	0,25
Open-source	7%	0,07

Table 3: Voice-to-Text Criteria weightage table

5.1.1.5 Accuracy testing

To get a better idea of how well the libraries work in terms of the accuracy of the word detection, an accuracy test was conducted. The candidates mentioned below all had a demo environment in which it was possible to test the library. The other candidates were tested based off the examples displayed on their website. The following candidates were taken into consideration when performing this test:

- Google cloud
- Microsoft Azure
- Web Speech API
- IBM Watson

These candidates are the top four of the candidates shown on the multi-criteria analysis list. Each of these candidates provided their demo environment to test their Voice-to-Text software. For these tests, the following demo environments were used:

- https://cloud.google.com/speech-to-text
- https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/#overview
- https://www.google.com/intl/en/chrome/demos/speech.html
- https://speech-to-text-demo.ng.bluemix.net/

To get a more objective view of the different libraries, different people were asked to participate in the accuracy test. Unfortunately, due to the Covid restrictions, the number of people that could test the different libraries was limited. In total, the libraries were tested by the following people:

- Colin Bosman
- Edwin Swaak
- Pascal Blikman
- Joey Teunissen
- Maria Lijding
- Remy Tapper

Each tester was asked to pronounce a set of questions. These random questions were asked with the intent of the library to convert the speech into the right words. The following questions were asked:

- Hallo, ik heb een afspraak om 12 uur met Colin Bosman.
- Hallo, ik ben op zoek naar het kantoor van Brease.
- Hoe laat is het?
- Dit is een uitgebreide vraag om te peilen hoe accuraat deze test is.
- Ik heb je niet goed verstaan. Kun je dat herhalen?

Each test was done based on the following conditions:

- Questions were spoken from the same distance.
- Every question was processed through the same computer (laptop).

- There was concurrent background noise (radio).
- Sentences were spoken in a normal understandable manner.
- There were small delays of 500ms between of every sentence.

Each accuracy rating is calculated in the following way:

- correct words / total words of sentence * 100
- Mispronunciation of non-existing words such as names of people/company names is graded based on how they sound rather than the way they are written.

Based on the criteria mentioned above, the following results were gathered:

Summarized results				
Questions	Google Cloud	Microsoft Azure	WebSpeech API	IBM Watson
Hallo, ik heb een afspraak om 12 uur met Colin Bosman.	93.8	92.2	98.3	80.9
Hallo, ik ben op zoek naar het kantoor van Brease.	84.3	98.0	98.0	89.3
Hoe laat is het?	95.8	87.5	100.0	83.3
Dit is een uitgebreide vraag om te peilen hoe accuraat deze test is.	17.9	100.0	94.8	50.3
Ik heb je niet goed verstaan. Kun je dat herhalen?	63.3	90.8	97.2	64.2
Average	71.1	93.7	97.7	73.6
Score	4	2	1	3

Table 4: Voice-to-Text Accuracy Testcase

For a total overview of all the test results, please check out the "VTT Accuracy TestCases" document and Appendix.

As can be seen in the table above, the Web Speech API placed first with an average accuracy rate of 97,7%. Next up, Microsoft Azure placed second with a score of 93,7%. IBM Watson placed third with an average score of 73,6%, and finally, Google Cloud placed fourth with an average accuracy score of 71.1%.

5.1.1.6 Results

Based on the researched criteria mentioned above, the following results have been gathered:

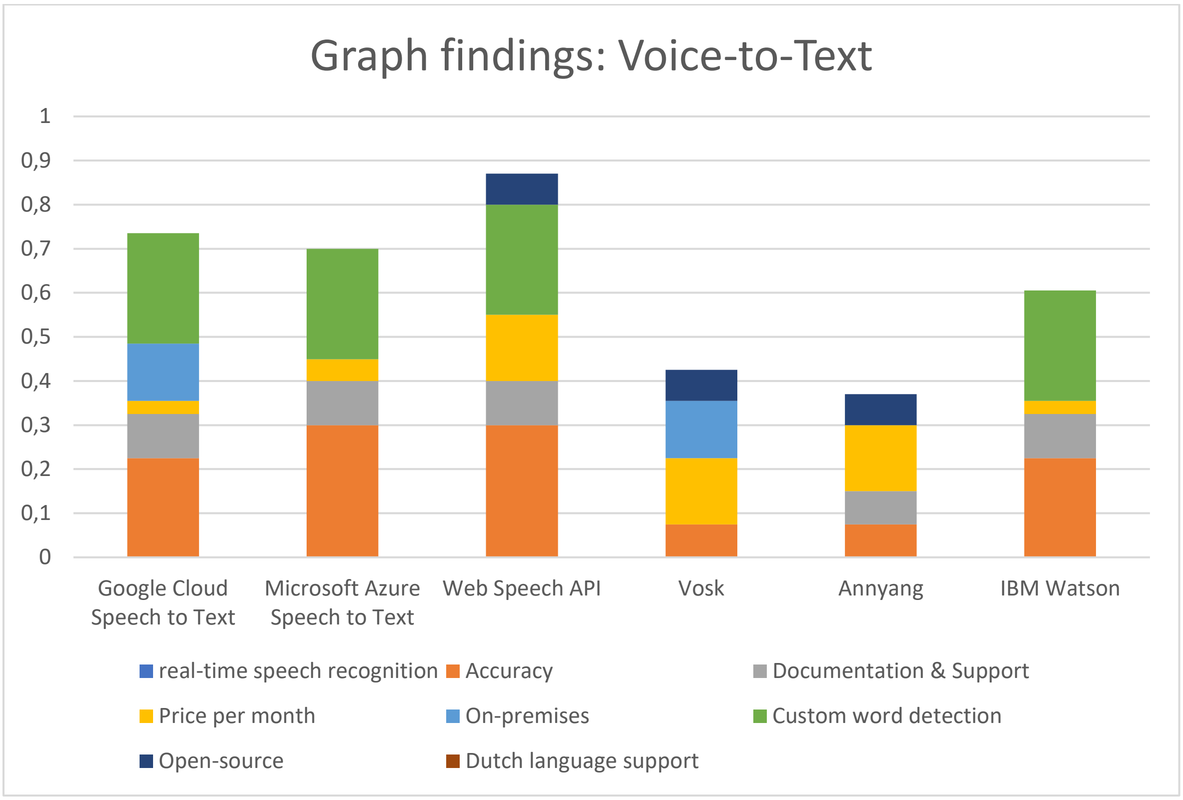


Figure 3: Voice-to-Text Findings

As shown in Figure 3: Voice-to-Text Findings, the Web Speech API came out as number one. The Web Speech API scored high on custom word detection and accuracy. Besides these two conditions, the Web Speech API is a free-to-use, open-source library, meaning that it also scored high on these aspects.

The Web Speech API and Microsoft Azure scored the highest on the accuracy test. Based on these results, in our instance, using the Web Speech API is the best option for this application.

5.1.2 Text-to-Speech - Multi-criteria analysis

Since there is a wide variety of different software choices to implement Text-to-Speech functionality in our project, a multi-criteria analysis has been created that contains a set of Text-to-Speech libraries in consideration. Based on the weight determination of each criterion, a conclusion can be drawn about the best outcome and solution to this problem.

5.1.2.1 Candidates

This Multi-criteria analysis will be comparing criteria between the following candidates:

- **Talkify**
  Talkify is a Text-to-Speech cloud service that made a custom JavaScript web app API.
- **Google Cloud TTS**
  Google Cloud is a platform that provides cloud computing services. One of these services is TTS (Text-to-Speech).
- **Microsoft Azure TTS**
  Microsoft Azure is a cloud computing service created by Microsoft. Just like Google, they offer a Text-to-Speech service.
- **Text-to-Speech-JS**

Text-to-Speech-JS is an open-source JavaScript library that provides a text to speech conversion using tts-api.com (*text-to-speech-js*)

- **Web Speech API**
  The Web Speech API is a JavaScript API that allows for adding speech recognition to web pages. It is created by an open community of developers.
- **ReadSpeaker**
  ReadSpeaker specialises in TTS (Text-to-Speech) solutions for websites, mobile applications, e-Books and more.
- **Amazon Polly**
  Amazon offers web services, also known as AWS. Amazon Polly is a service that turns text into lifelike speech by making use of deep learning technologies to synthesize a natural-sounding human speech (North, *Polly* 2012).

5.1.2.2 Criteria

The libraries mentioned above have been compared with each other based on a set of criteria. This criteria has been created by looking at the fundamental concepts of Text-to-Speech. Besides, Brease has been asked which criteria they find essential to ensure all the criteria have been covered. Based on the feedback provided by Brease, the following set of criteria has been created:

- **Natural Dutch Voice**
  By natural Dutch voice, we mean a more human-like sounding voice. This can be achieved by making use of a neural network technology called "WaveNet". A more detailed explanation of this concept can be found below.
- **Standard Dutch Voice**
  By this criterion, we mean the default Dutch TTS without any form of audio enhancement.
- **Documentation & Support**
  This criterion concludes how well the documentation of the used library is and how well this is being supported.
- **Price per month**
  Most of the premium software requires some form of payment. In our instance, some libraries require a monthly payment.
- **On-premises**
  One of the ways to ensure the confidentiality of sensitive user information can be achieved with on-premise software. On-premise software is installed and runs on the local machines of the person or organisation using the software. In this instance, this would mean that this library will be working locally without having to send data to online cloud environments.
- **Customizability**
  By customizability, we mean the grade of which the software can be changed (customized). Examples for this instance are the rate of speech, volume, voice, etc.
- **Open-source**
  Open-source software is made freely available to users and can be redistributed and modified.
- **Natural English voice**
  A natural, more human-like, English sounding voice. Like the "Natural Dutch Voice" criteria mentioned above, this criterion explains how well the voice sounds like a natural English voice by using "WaveNet" technology.

5.1.2.3 Standardization

In order to compare different criteria with each other, each criterion has been standardized. In this instance, the standardization tables look as follows:

	Lowest value (0)	Highest value (1)
Number	$8/mil. Char.	$0 (Free)
Rating	--	++
Yes or No	No	Yes

Table 5: Text-to-Speech Standardization table

For a more in-debt explanation of this table in the context of the criteria, please check out the Standardization chapter in the additional research document.

5.1.2.4 Weightage

To determine which criterion is the most important, each criterion has been weighted. These weights have been discussed with the company supervisors to ensure the most important criteria has the highest weighting. The weightage of the Text-to-Speech criteria looks as follows:

	weightage	
	%	decimal
Natural Dutch voice	25%	0.25
Standard Dutch voice	15%	0.15
Documentation & Support	25%	0.25
Price per month	12%	0.12
On-premises	10%	0.1
Customizability	6%	0.06
Open-source	5%	0.05
Natural English voice	2%	0.02

Figure 4: Text-to-Speech Criteria weightage table

For a more in-depth explanation of each weightage, please check out the Weightage chapter in the additional research document.

5.1.2.5 Results

Based on the researched criteria mentioned above, the following results have been gathered:

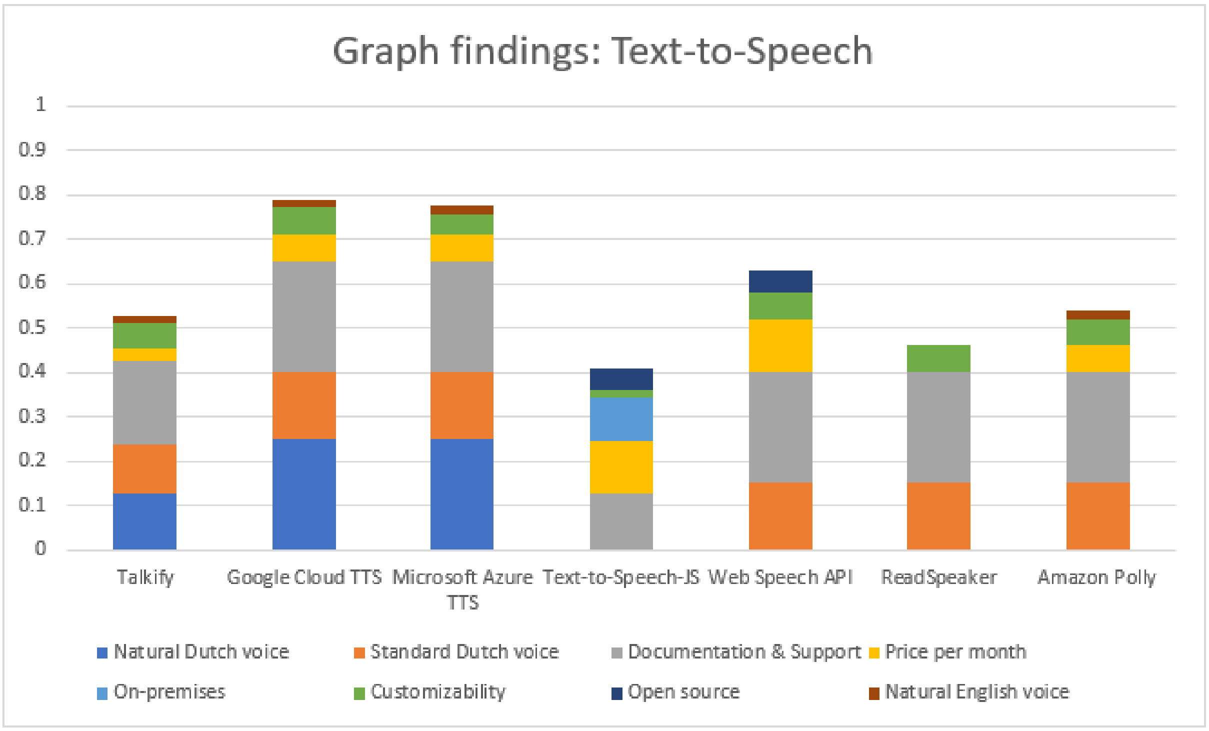


Figure 5: Text-to-Speech Findings

As shown in Figure 5: Text-to-Speech Findings, Google Cloud TTS came out as number one of the best Text-to-Speech library. Google Cloud TTS scored high on their implementation of the natural Dutch voice and their well-structured documentation & support. Overall, Microsoft Azure TTS scored almost the same number of points except for the customizability. Microsoft Azure TTS offered fewer ways to customize the returned speech (speed & pitch) compared to Google Cloud TTS.

Google Cloud TTS also offered a beta version of their timepoints feature. Based on the request, timepoints will be returned, which indicate when certain words are being spoken. This can be helpful when adding lip-sync support to the virtual assistant.

For a more detailed version of the results, in combination with a more in-depth explanation of the scores of each criterion, please check out the additional Research Document and the “Multi-criteria analysis” document.

5.1.3 Survey

A survey has been conducted to get a better indication of where the interest of a virtual reception lies. For a total overview of the survey, please check the document: "SurveyResults - reception.csv" for a total overview of the survey. Contact details such as names, phone numbers, etc., have been removed from the results for privacy reasons.

The purpose of the survey was to understand how organisations (and possible end-users) fulfil the needs of a reception function at that time. Next to this, the survey indicated which features might be interesting to these organisations and their opinion on a virtual reception compared to a digital reception or one operated by a person. Since the survey was given to Dutch participants, the questions and multiple-choice options were written in Dutch.

In total, the survey was sent to 141 participants. Out of these participants, 72 participants (51.1%) filled in the survey. The survey results showed that the participants worked at the following organisations:

Municipality	86.1%
multi-company building	11.1%
other	2.8%

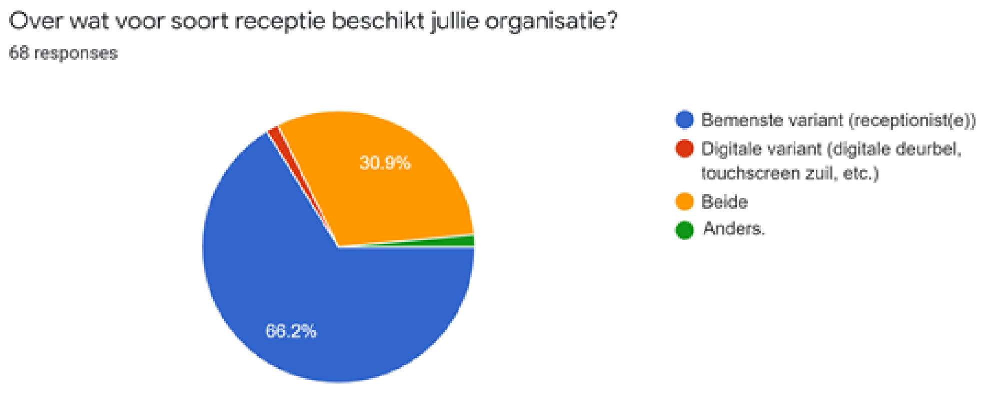


Figure 6: Survey result - reception types

Out of the 72 participants, 68 of them made already use of a type of reception. Out of these participants, 45(66.2%) and thus the majority made use of just a staffed reception. Only 1 participant makes use of just a digital reception, and 21(30.9%) makes use of both a staffed and digital reception.

We also asked what kind of features they would expect a virtual assistant to have. This question gave the following results:

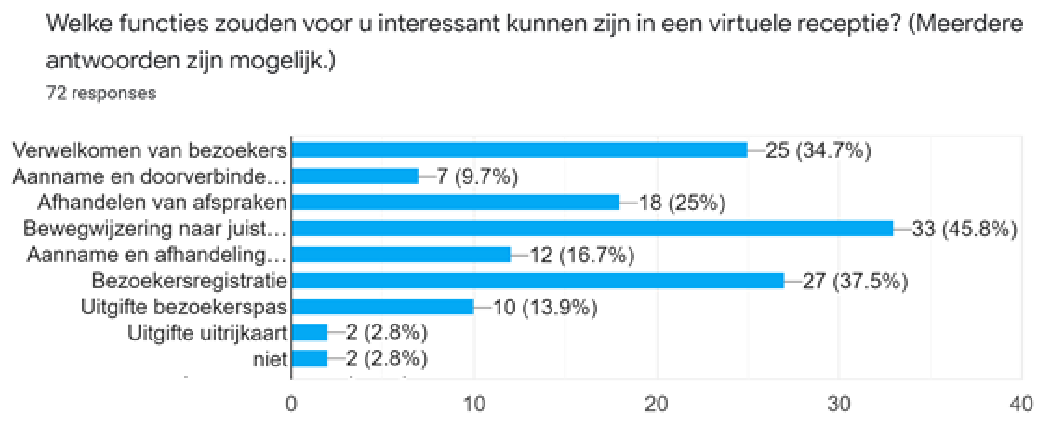


Figure 7: Survey result - desired reception features

As can be seen in the figure above, the following features would be exciting to organisations:

- Signage to the destination of the visitor
- Greeting visitors
- Visitor registration

Based on these results, it is clear which features would have to be focussed on first since these are considered core features of a virtual assistant to the end-user.

Next to this, we asked if a virtual reception would be a good addition to their organisation. This question gave the following results:

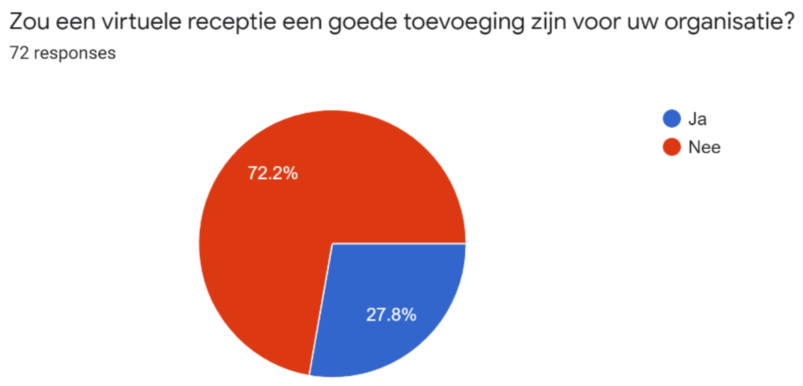


Figure 8: Survey result - Additional virtual reception

52 participants (72,2%) chose for no and 20(27.8%) chose yes.

As a follow-up question, we asked why they would think a virtual assistant would (not) be a good addition to their organisation:

- Yes:
  - Work is taken out of hands.
  - More efficient and faster for visitors
- No:
  - Human interaction and contact are important
  - Impersonal

Finally, each participant was asked whether or not their organisation would like to test a prototype of the application.

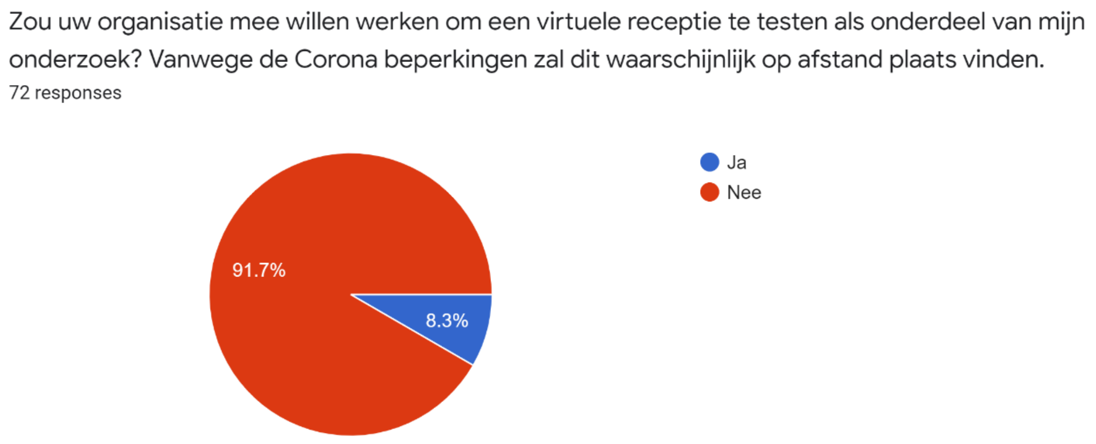


Figure 9: Survey result - Prototype testing

Out of all the participants, six participants are prepared to test a prototype of the application.

Even though not every person prefers a virtual assistant over a staffed one, there is demand. Although the statement many people made about a virtual assistant being impersonal, if executed well, a virtual assistant can be made in such a way, it could look and talk like a human, making the conversation more human- and life-like than a standard kiosk.

5.2 Design

This subchapter shows the overall architectural structure of the application and the external services that it communicates with, in the form of diagrams and wireframe designs.

5.2.1 Architecture

To explain the structure and flow of the application and certain concepts within the application, the following architectural diagrams and designs have been made:

5.2.1.1 Flow diagram

The following flow diagram has been created to replicate a default scenario of the application:

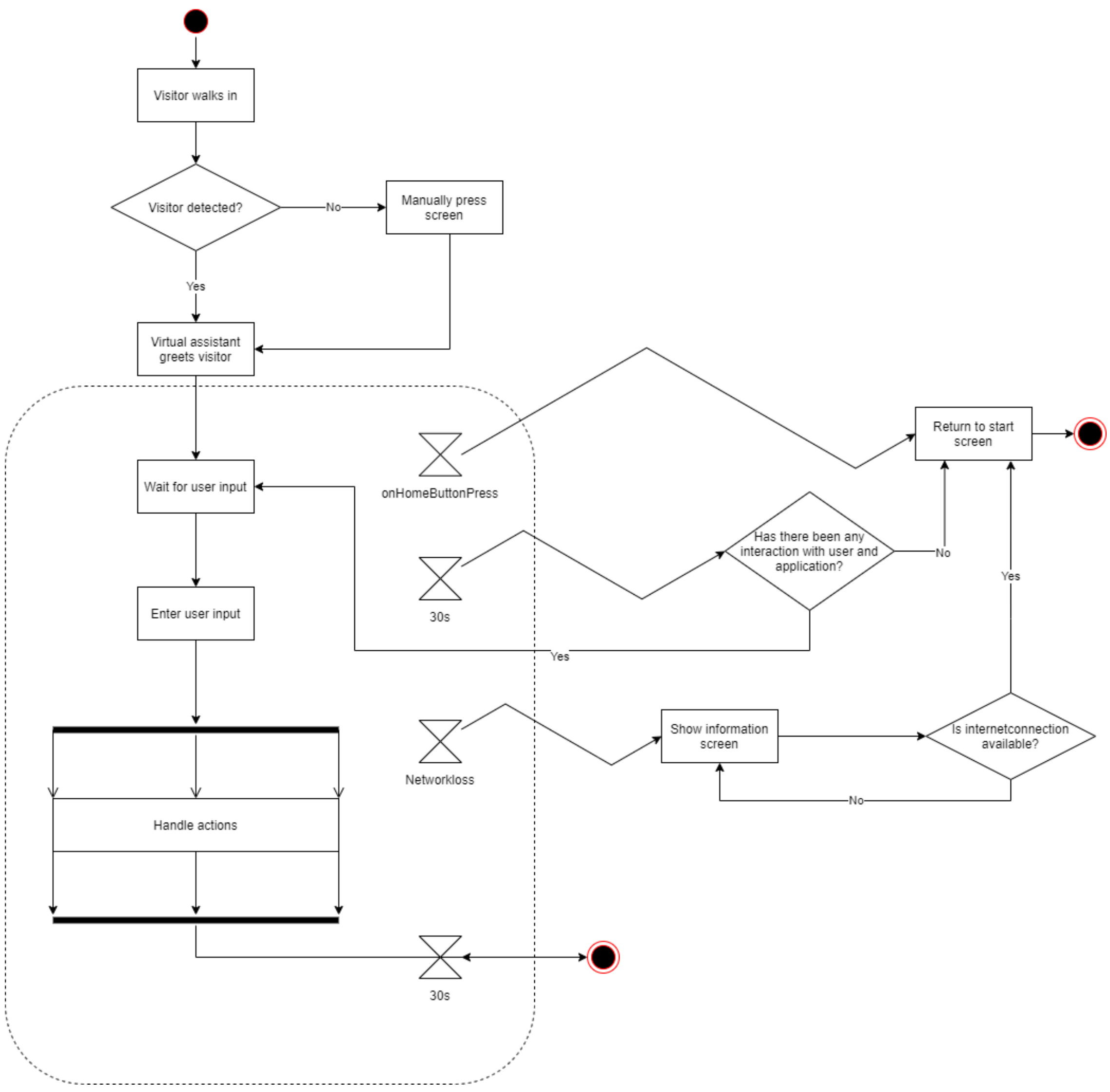


Figure 10: Flow diagram - Default scenario

Main flow

The flow starts the moment a visitor walks in. This is where the application should detect the visitor. If this isn't the case, the visitor can manually press the screen which starts the application and results in the virtual assistant greeting the visitor.

In this state, the visitor is asked for their reason for arrival, and the application will wait on the visitor's input. Based on the input, different actions will be handled. After this is completed, the visitor may ask more questions which will restart the flow. If the application does not receive any input, the application will restart, and the flow will stop.

Timed events

Besides the main flow of the application, there are several timed events. These events are active while the virtual assistant is active. In total, the following timed events will be active:

- **Inactivity**
  While the application is listening for user input, a timer will start counting. By detecting if there has been any user input while the timer is counting, we can state whether the application is being used. If there has not been any interaction between the application and the user, the application will navigate to the Start screen, and the flow will stop. However, if the application is still being used, nothing will happen, and the application will keep listening for any user input.
- **Network loss**
  The application could lose connection. If this occurs, the application will navigate to the information screen in which the visitor would see a static floorplan of the building.
- **Home button press**
  At any time throughout the application, the visitor may press the home button. This button will restart the process and navigate the visitor back to the start screen.

5.2.1.2 Sequence diagram

To replicate the default conversation flow of the assistant, a sequence diagram has been made. This diagram contains a list of different participants and how they communicate with each other and external services. Based on the sequence diagram below, a list of the following participants has been created:

- **Visitor**
  This is the user of the application. The visitor is the person that will be interacting with the application and virtual assistant.
- **Front-end Application**
  This application handles the interaction between the user and the back-end services through a user interface.
- **Web Speech API**
  This is an external service used to handle the Speech recognition part of the application.
- **Rasa**
  This is a chatbot that is used to retrieve actions based on the given intent.
- **Back-end**
  The back-end is used to gather additional data which is stored in the database, such as names. Besides this, the back-end is used to communicate with Google TTS.
- **Google TTS**

Google TTS (or Text-to-Speech) is a cloud service created by Google that handles a speech file generation based on a given string.

Based on the list of different participants mentioned above, the following sequence diagrams have been made:

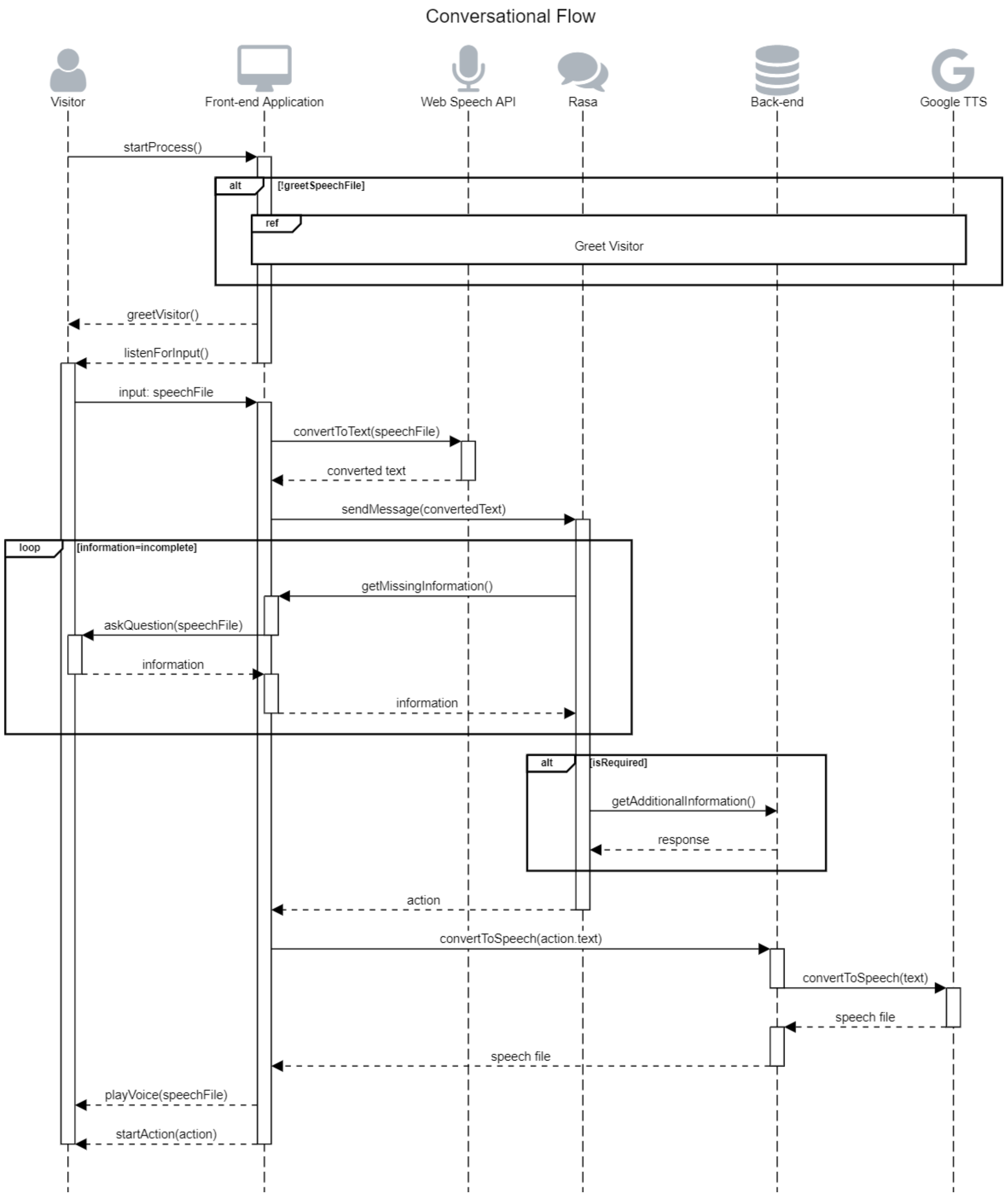


Figure 11: Sequence diagram - Default conversational flow

When a visitor walks towards the application, the arrival of the visitor will be detected. This can be done through a simple sensor which detects movement (More information about sensors can be found in the Research Report). If this is for some reason not working, the user will be able to click on the screen. The application will be started in both ways, and the "startProcess()" function will be called.

If there is a speech file available that greets the user, the "greetVisitor()" function will be returned. If this is not the case, the "Greet Visitor Flow" ref will be called. This part of the sequence diagram can be found below underneath Figure 12: Sequence diagram - Greet visitor flow.

Besides the previous return function, the "listenForInput()" function will also be called. This function will listen for any type of speech input and returns this in the form of a speech file.

After the application receives speech, it will be sent to the Web Speech API by calling the "convertToText()" function. This function will based on the given speech file, return the converted text.

After the application receives the text from the "convertToText()" function, the "sendMessage()" function will be called. This function will use the converted text gathered from the previous function and sends it to the Rasa chatbot. Here, based on the input, the rasa chatbot will activate the right intent. Based on this intent, Rasa requires additional information. If this is the case, Rasa will ask for the missing information until it has all the required information. Then it will return the action to the application. (To remove repetitive information, Voice-to-Text translation has been removed from the loop section of the diagram)

Besides asking the user for information, the chatbot might also have to retrieve information from the back-end. This is done through an API call inside a custom action. Based on the information returned from the back-end, the proper response will be sent back to the front-end application.

Every time the chatbot sends messages back to the front-end, the message will be parsed inside of the "convertToSpeech()" function and sent to the back-end. From here, the back-end will make a call to the Google TTS API and return data containing a base64 string. This string will then be returned to the front-end application, where it will be played as audio.

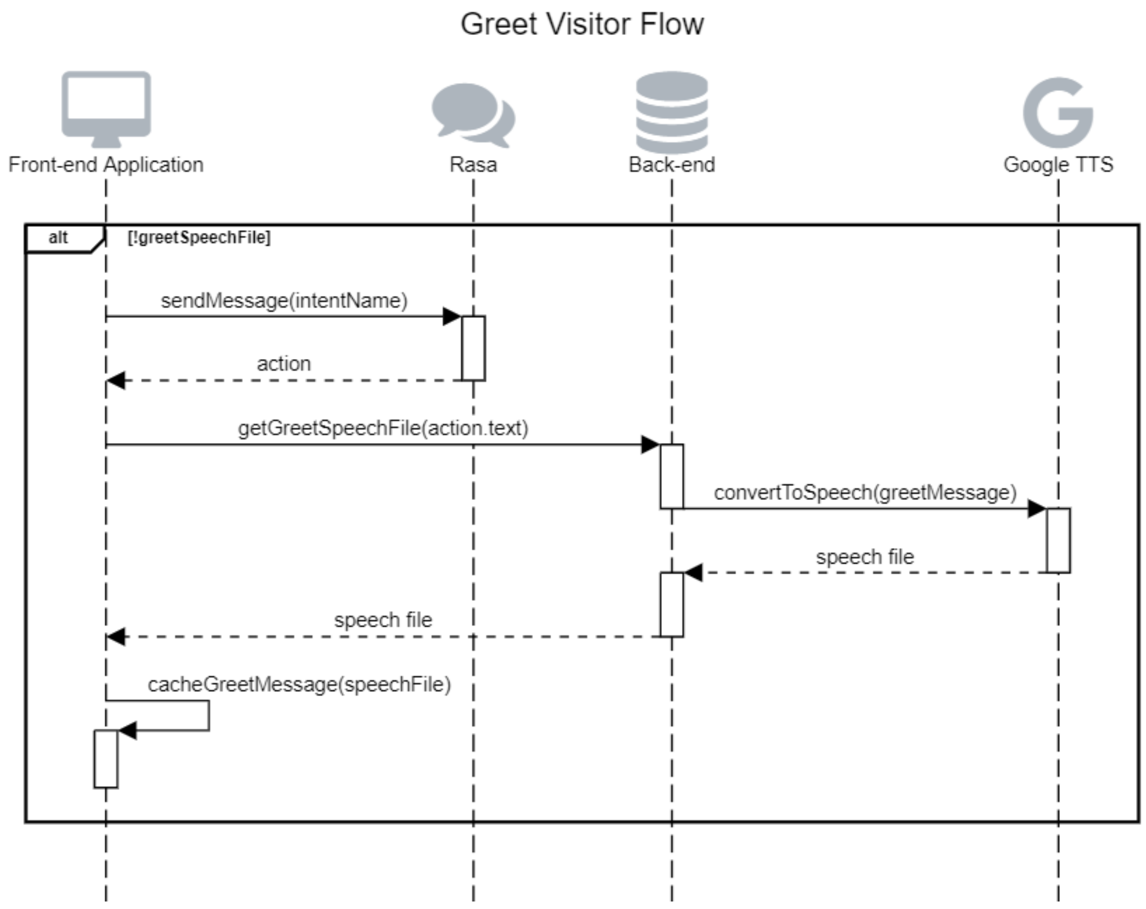


Figure 12: Sequence diagram - Greet visitor flow

If there is no "greetSpeechFile" available, a new call will be done to Rasa ("sendMessage()"). This function will return the action containing the greetings text to which the virtual assistant should speak. To convert the text into lifelike speech, the text will be sent to the back-end. From here, the text will be parsed in the "convertToSpeech()" function to the Google TTS service. This service will return a speech file which will then be returned to the front-end application. Every communication between the application and external service will happen in real-time. Since the files that are being sent back and forth will be small, the travel duration will be brought to a minimum, resulting in a natural conversation without any significant waiting times between question and answer (provided that there is a stable connection to the internet).

Finally, the returned speech file will be cached locally to make sure it can be used for future use and to minimize the number of different calls that have to be done.

5.2.1.3 Class diagram

For this project, a dummy database has been created. This database served as a way to show the possibilities of communicating between Rasa and the back-end. As of right now, the database contains dummy data to test the functionality between front-end, back-end and Rasa chatbot. A class diagram has been created and looks as follows:

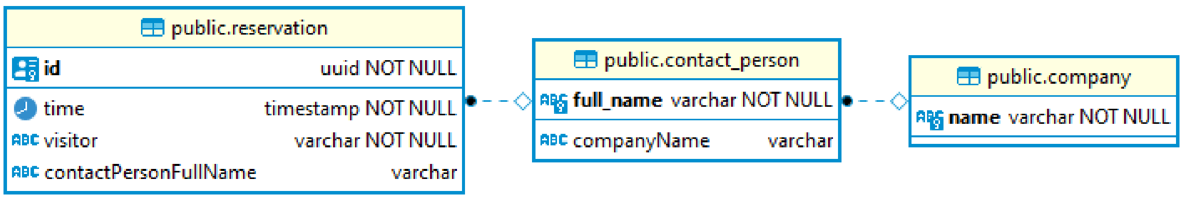


Figure 13: Class diagram

As can be seen in the figure above, there are three tables: "reservation", "contact_person" and "company". For a total overview of all the existing calls, please checkout chapter: 6.3 Realisation.

5.2.2 Wireframes

5.2.2.1 Low fidelity wireframe

The first design that was created was a Low fidelity(a.k.a Lo-Fi) wireframe. This wireframe served as a way to design and display an unfinished presentation of the different features present within the application. It indicates where certain functionalities would be placed and result in a better understanding of how the interaction between user and application would work.

The Low-Fi wireframe was created using "WireframePro", an online, free-to-use wireframe tool that can be accessed through the web browser.

The low-fidelity designs combined with an explanation about each screen and design choices can be found inside the external Functional Documentation.

5.2.2.2 High fidelity prototype

After creating the Lo-Fi wireframe, there was a better indication of where certain functionality would be displayed. To make sure the design decisions align with the end-product, a High-Fidelity (or Hi-Fi) prototype has been created. This prototype displays the different functionalities combined with a design that makes it feel like an actual application.

The Hi-Fi prototype has been created by using "Figma", a digital design and prototyping tool. Just like the previously used Lo-Fi tool, Figma is a tool that can be accessed through the web browser. Figma adds the possibility of creating a complete UI and UX design of the entire application, including flow animation and styling. This prototype can be accessed by clicking on the following link:

https://www.figma.com/file/bcHsTZ8OFCkp1GofFJ6opc/Virtual-Assistant-Hi-FI?node-id=0%3A1

Based on the previously created Lo-Fi wireframe in combination with the existing functionalities that have been determined during the requirement analysation phase, the following Hi-Fi prototype has been created:

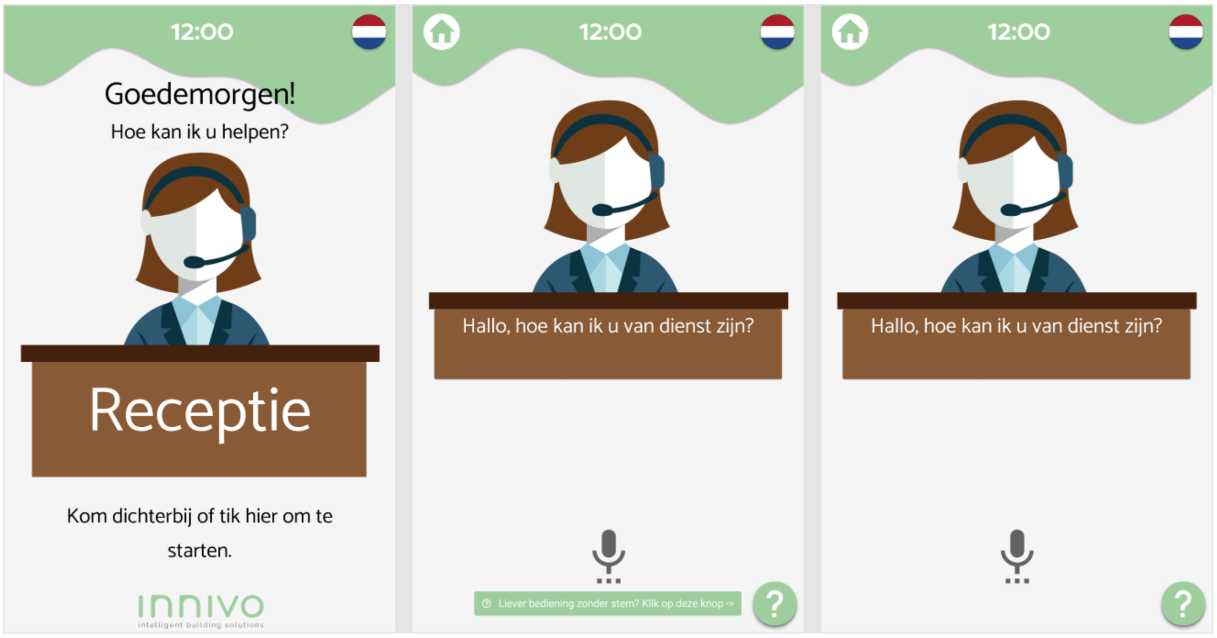

12:00
Goedemorgen!
Hoe kan ik u helpen?
Receptie
Kom dichterbij of tik hier om te starten.
innivo
intelligent building solutions
12:00
Hallo, hoe kan ik u van dienst zijn?
Liever bediening zonder stem? Klik op deze knop ->
?
12:00
Hallo, hoe kan ik u van dienst zijn?
?

Figure 14: Hi-Fi Prototype 1/3

The figure above consists of the following screens:

Start screen

This is the first screen the visitor sees. This screen shows the same information compared to the Lo-Fi version of this screen. The current time has been moved to the centre of the top part of the screen. In combination with the picture of a reception desk with the word "receptie", the virtual assistant aims to tip the visitor that this application is a virtual reception. When the visitor gets closer to the application, the sensor should detect and move to the next screen. If this is not the case, the visitor can tap anywhere on the screen, navigating the visitor to the next screen.

Virtual assistant screen – Voice input

On this screen, the visitor will be able to ask a question to the assistant. The microphone at the bottom of the screen indicates that the microphone is active and can be used to ask a question. Text spoken by the virtual assistant will be displayed on the reception desk as a form of subtitles.

At the top of the screen, a home icon can be found. This icon is displayed throughout the entire application and is used to navigate back to the home screen. When multiple visitors would like to use the application after each other, the home button can be tapped to restart the process.

At the bottom of the screen, a small tooltip is shown, asking the visitor whether they prefer to use a hands-free version or choose a touch version of the application. If they decide to choose a touchable version, the visitor can press the "?" button at the bottom-right of the screen. This button will expand into two smaller buttons presented above the current button. The button at the bottom changes between the different user-input types (hands-free/touch). The button at the top gives the option to call for external help.

External help dialogue

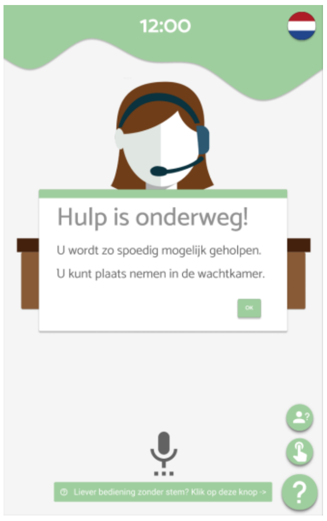

12:00
Hulp is onderweg!
U wordt zo spoedig mogelijk geholpen.
U kunt plaats nemen in de wachtkamer.
OK
Liever bediening zonder stem? Klik op deze knop

Figure 15: Hi-Fi Prototype 2/3

When the visitor clicks on this button, a dialogue will explain to the user that help is on the way. (see Figure 15: Hi-Fi Prototype 2/3 for a visual representation)

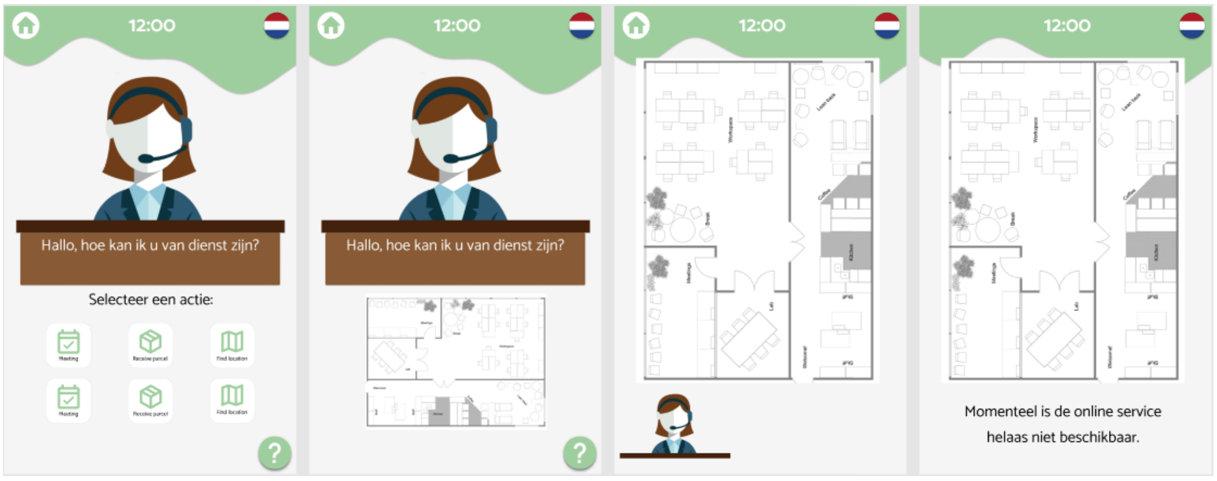

12:00
Hallo, hoe kan ik u van dienst zijn?
Selecteer een actie:
Meeting
Receive parcel
Find location
12:00
Hallo, hoe kan ik u van dienst zijn?
12:00
12:00
Momenteel is de online service helaas niet beschikbaar.

Figure 16: Hi-Fi Prototype 3/3

Virtual assistant screen – Touch input

In the previous figure, an explanation was given about the "?" button at the bottom of the screen and how this serves as a way to switch to different user input states (hands-free or touch). The first screen of Figure 16: Hi-Fi Prototype 3/3 shows a touch version of the application. The place where the microphone icon used to be, is now replaced with a set of buttons that the user can click on. Each button calls a different action. Depending on the variation of actions, different screens will show.

One of these screens contains the floorplan of the building. On this screen, the virtual assistant will be moved to the bottom-left of the screen and will explain to the visitor how he/she will reach his/her destination.

Network connection lost screen

If there are any network problems and the online services are unavailable, the last screen of Figure 16: Hi-Fi Prototype 3/3 will be shown. This screen will show a static image of the floorplan of the building. This way, the visitor will still have some sort of indication of where he/she can find his/her destination without the help of the virtual assistant.

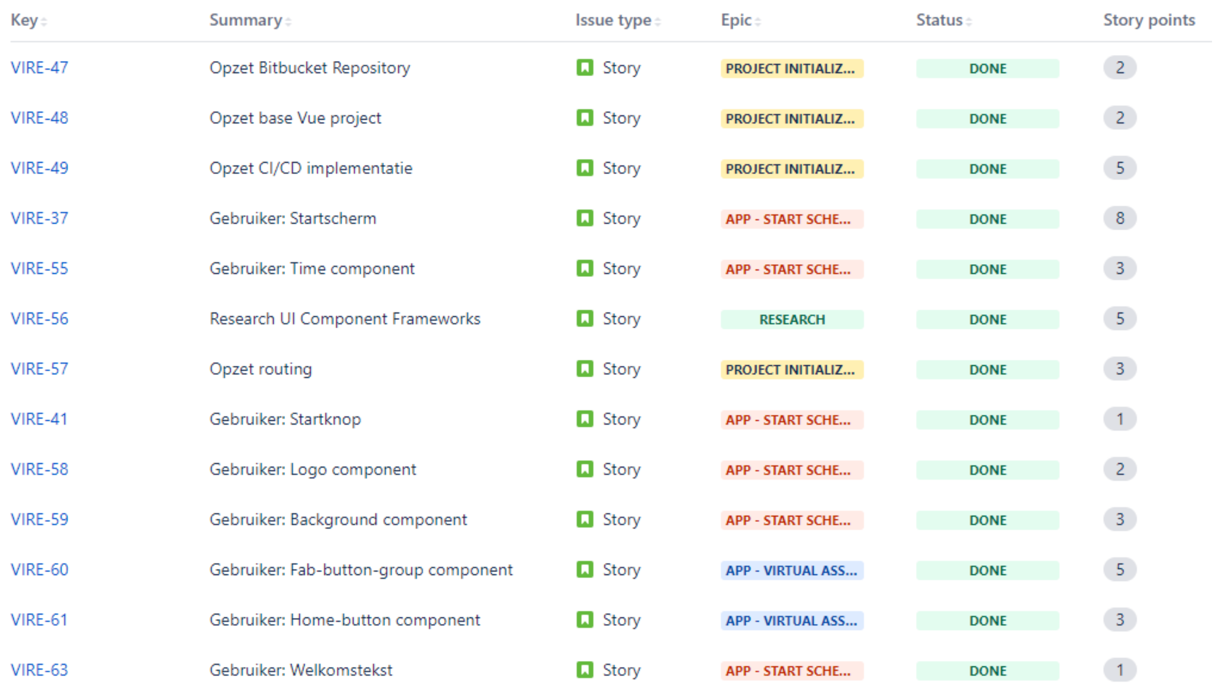
5.3 Implementation

This subchapter will include the implementation process of the project. A short description of the implemented key-functionalities of each sprint will be mentioned, together with a burndown chart about one of the sprints to explain the general development process. Besides this, there will also be talking about how testing of the application has been done and the different problems that arose during the graduation period.

5.3.1 Sprints

5.3.1.1 Sprint 1

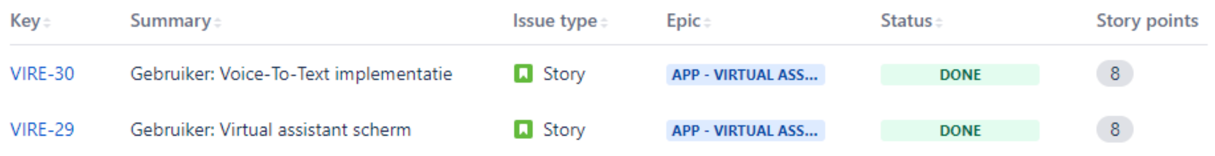
Sprint one was merely used as an initialization sprint. A big part of this sprint consisted of setting up the different project environments and installing required plugins, repositories and routing. In total, the following user stories were burned during this sprint.



Key	Summary	Issue type	Epic	Status	Story points
VIRE-47	Opzet Bitbucket Repository	Story	PROJECT INITIALIZ...	DONE	2
VIRE-48	Opzet base Vue project	Story	PROJECT INITIALIZ...	DONE	2
VIRE-49	Opzet CI/CD implementatie	Story	PROJECT INITIALIZ...	DONE	5
VIRE-37	Gebruiker: Startscherm	Story	APP - START SCHE...	DONE	8
VIRE-55	Gebruiker: Time component	Story	APP - START SCHE...	DONE	3
VIRE-56	Research UI Component Frameworks	Story	RESEARCH	DONE	5
VIRE-57	Opzet routing	Story	PROJECT INITIALIZ...	DONE	3
VIRE-41	Gebruiker: Startknop	Story	APP - START SCHE...	DONE	1
VIRE-58	Gebruiker: Logo component	Story	APP - START SCHE...	DONE	2
VIRE-59	Gebruiker: Background component	Story	APP - START SCHE...	DONE	3
VIRE-60	Gebruiker: Fab-button-group component	Story	APP - VIRTUAL ASS...	DONE	5
VIRE-61	Gebruiker: Home-button component	Story	APP - VIRTUAL ASS...	DONE	3
VIRE-63	Gebruiker: Welkomstekst	Story	APP - START SCHE...	DONE	1

Figure 17: Sprint 1 - User stories

The estimated amount of story points for sprint 1 were 43 story points. These story points were finished before the end of the sprint and because of this, two additional stories were picked up:



Key	Summary	Issue type	Epic	Status	Story points
VIRE-30	Gebruiker: Voice-To-Text implementatie	Story	APP - VIRTUAL ASS...	DONE	8
VIRE-29	Gebruiker: Virtual assistant scherm	Story	APP - VIRTUAL ASS...	DONE	8

Figure 18: Sprint 1 - Additional user stories

Out of these two stories mentioned above, one story was finished in time, namely the Voice-to-Text implementation (8 story points). The remaining story was put back into the backlog and planned to be picked up again in sprint two.

5.3.1.2 Sprint 2

The main focus of sprint two was the implementation of the Virtual Assistant screen in combination with Text-to-Voice. Next to this, a dummy back-end has been set up containing the call to the Google API to convert text into life-like speech. During this sprint, a connection has also been made between the front-end and the Rasa chatbot.

In total, sprint two was estimated to be 50 story points with eight different stories. At the end of this sprint, 47 story points were burned. Due to bugs, the setup of CI (Continuous Integration) used for automated tests was not finished in time and moved to the next sprint.

5.3.1.3 Sprint 3

The main focus of sprint 3 was getting the Rasa Actions to work. Unfortunately, this took much time out of the sprint due to problems setting up and communicating with the Rasa action server. In the end, this story was burned by recreating a new Docker environment and Rasa instance. Besides this, a touch version of the custom action to check a reservation has been created in combination with properly displaying subtitles when being spoken by the virtual assistant.

5.3.1.4 Sprint 4

At the time of writing this, Sprint 4 is still ongoing. The main focus of this sprint was processing feedback, creating the touch version of checking a reservation, creating tests, and implementing additional name validation on the chatbot. Besides this, since this is the final Sprint before the end of the graduation, additional feedback on the documentation is still being processed which took quite some time out of the sprint. Because of the this, the results of this sprint are still uncertain.

5.3.2 Burndown chart

During each sprint, a burndown chart was used to keep track of the remaining amount of story points in combination with the remaining amount of time. For example, for the first sprint, the number of story points estimated to be burned at the start of the sprint was 43. Since this was met before the end of the sprint, two more stories were picked up from the backlog and put into the sprint, resulting in 59 story points. In the end, 51 story points were burned, resulting in the following burndown chart:

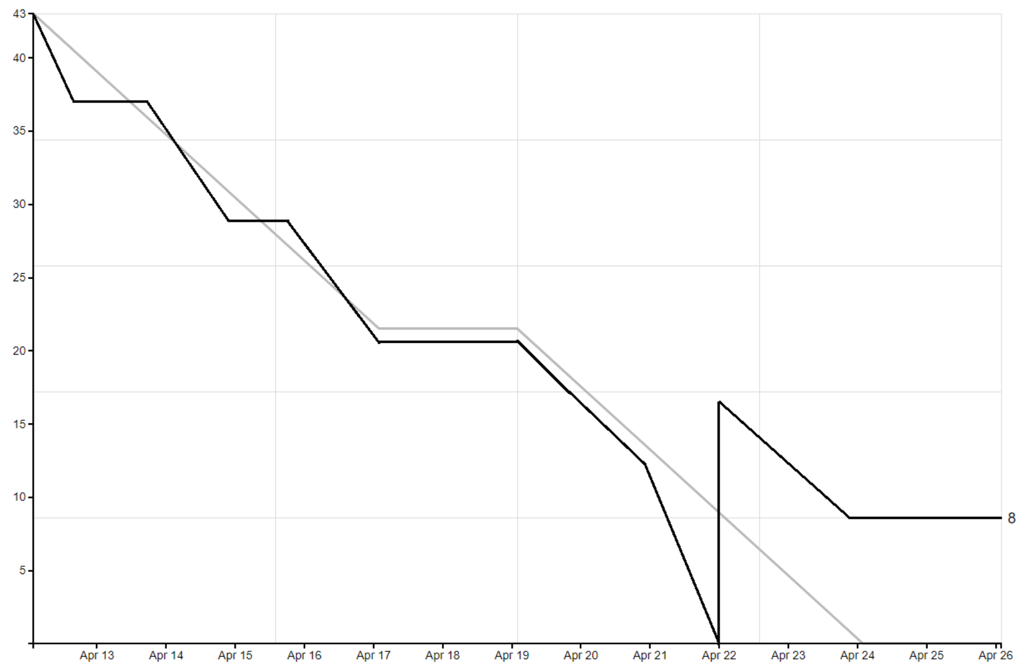


Figure 19: Sprint 1 - Burndown chart

5.3.3 Testing

To make sure code runs correctly (both old and new code), tests are made. At first, the plan was to test the application automatically through the use of Continues Integration(CI). This was planned to be implemented in an early stage of development but due to problems with the configuration and time constraints, this is yet to be done. Eventually, by discussing the problems with my company supervisor, there was decided to test functions and code where necessary and to do this locally rather then through CI.

At the time of writing this document, these tests are still being implemented and are therefore not fully finished yet. However, these tests will be finished before the end of the graduation period. Both Unit tests and end-to-end (e2e) test will be written. Unit tests will test custom functions containing complex code. End-to-end tests will test the overall workflow of the application.

At the end of the graduation, the entire codebase has also been reviewed by the company supervisor. Based on this feedback, adjustments were made to ensure the code conformed to the Brease coding standards. In most times, this resulted in renaming variable names and generalizing existing code.

5.3.4 Retrospective

During sprints, several problems arose, which led to slight adjustments in terms of the development progress. Although most of these problems were relatively small and easy to solve, the following problems were more significant, which took quite some time out of the sprint:

Outdated UI framework components

Since the project is written in Vue3, a relatively new and updated version of Vue, specific UI components provided by Vuetify (Vue UI library) were not supported at this time. As an alternative, a different component library was used, namely "Element-Plus". Unfortunately, further in the development, specific additional components required for the project, namely the speed dial component, were not supported by Element-Plus. This problem could have been prevented by having done more comprehensive research on the different components this UI library had to offer.

Finally, the decision was made by the intern and company supervisor to stick with Vuetify as the primary UI component library of this project and to make use of alternative libraries or CSS where necessary. Since Vuetify releases new versions regularly, there will be more support for different components. This means that existing custom solutions can be replaced with updated Vuetify components in time, allowing for a more compact and standardized approach.

Rasa configuration & Action server

Rasa is a big part of the application, it handles all the messages between the user and application. Setting up Rasa came with several problems. The main problem was that, for some reason, Rasa didn't save any data locally. Next to this, several Docker containers kept restarting and throwing errors. After contacting Rasa admins and the Rasa forums, the problem was eventually solved by recreating a new Dockerfile from scratch.

Besides the Rasa server, an action server is also run in parallel. This action server handles all the custom actions used in the application to check whether a reservation is available in the back-end. Like Rasa, the action server had connection problems, but this was also fixed when recreating the Dockerfile. Next to this, writing the custom action took some time to figure out what exactly are and how to use Rasa concepts such as forms, slots, default actions and dispatchers.

Rasa name validation

One of the tickets was about creating a custom Rasa action that checks whether a visitor is available based on the input. Even though this might look like a simple concept, it is a lot of work to get this fully functional. The flow of failing paths was not thought about that well and therefore underestimated. What if the chatbot receives the wrong information because the Voice-to-Text engine misheard the user (Remy > Amy). This would result in the call failing. Additional validation had to be created to check whether the given input is correct to catch these possible failures. Because of these small additions that were overlooked, the feature was not fully complete in time. This could have been prevented by having done a more in-depth investigation of possible failures. Based on these results, the ticket might have been worth a lot more story points.

Ticket size

At the start of each sprint, there is a sprint planning. During sprint plannings, story points are being assigned to existing user stories tickets that are being worked on will be discussed with the company supervisor. During these sprints, there have been several times where tickets have been too large. An example of this is the implementation of the virtual assistant. By not thinking about all the extra stuff that has to be implemented besides the main functionality of the ticket, a ticket can take up a lot more time than initially thought. A better option would've been to split up stories as much as possible. This way, story point estimation will be a lot closer to the end result. Besides this, since there was being worked with a new technology that was not very familiar, it was difficult to detect every tiny detail.

5.3.5 Application

When the application starts, the first page that the user will see is the home page. This page will show the current time(responsive) and text explaining how to use the application. The screen can be clicked on, which will redirect the user to the assistant screen.

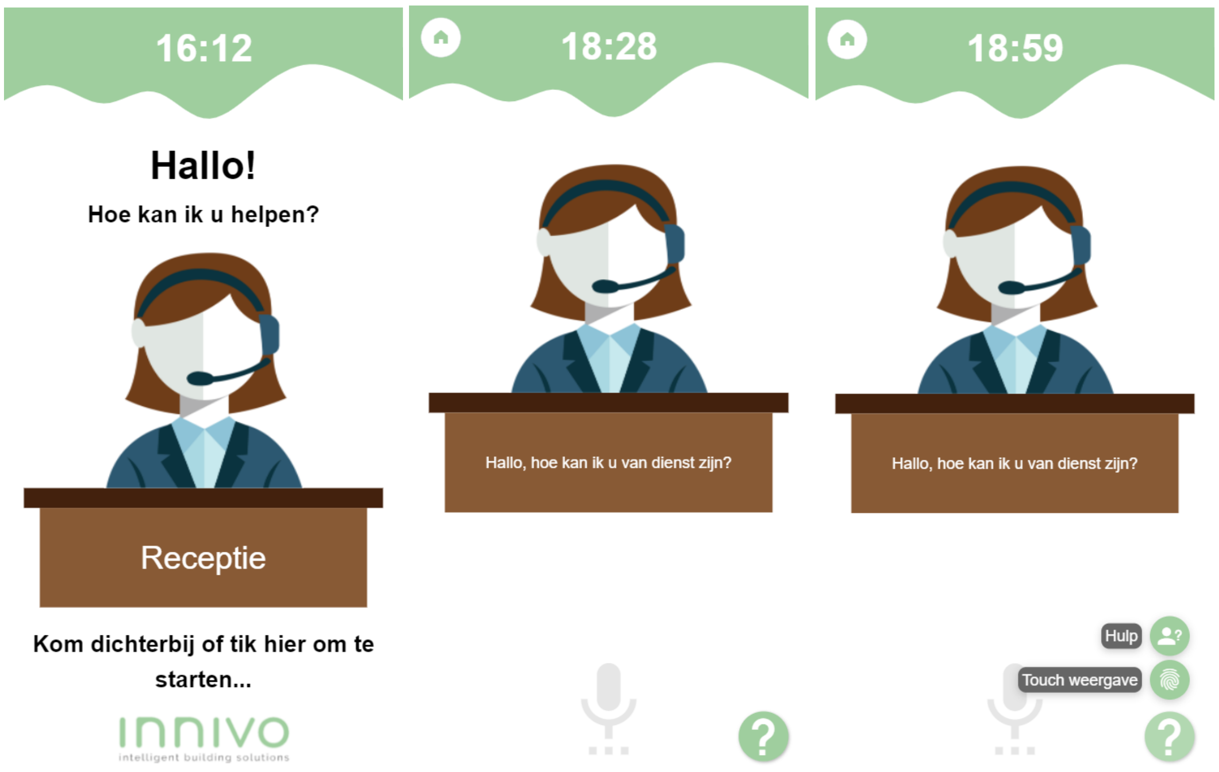

16:12
Hallo!
Hoe kan ik u helpen?
Receptie
Kom dichterbij of tik hier om te starten...
innivo
intelligent building solutions
18:28
Hallo, hoe kan ik u van dienst zijn?
18:59
Hallo, hoe kan ik u van dienst zijn?
Hulp
Touch weergave

Figure 20: Front-end application screens

When entering the assistant screen, audio will play, asking how the assistant may be of service followed by the microphone being enabled. The user will now be able to talk to the assistant, and the voice input will be translated to text. This text will then be sent to the Rasa Chatbot, and depending on the question, the chatbot will return a message. This message will then be converted to speech using the Google TTS service and the audio will be played to the user. Besides the audio being played, the subtitles will also be updated to the corresponding audio.

There are also two different buttons on the page. The home button, located at the top-left of the screen, redirects the user to the home screen. The "?" button at the bottom-right will open two more buttons. One of them will show a dialogue that will inform the user that help is on its way. The other button will allow the user to switch between touch and voice input. When clicking on the touch button, the microphone will disappear, and a set of action buttons will appear which can click on. As of right now, there is one action available which will be based on user information, check whether the user has an existing reservation.

6. Software competences

This chapter explains each of the software competencies and how these have been met during this graduation period.

6.1 Analysis

To show that the analysis competence was met, a list of requirements has been created. These requirements can be found in chapter 2.4 Requirements. Next to this, the following frameworks have been researched:

- Voice-to-Text
- Text-to-Speech
- Vue

For both Voice-to-Text and Text-to-Speech, a multi-criteria analysis has been conducted to find the best choice of library/plugin for this application.

Next to this, a survey has been conducted, which can be found in chapter "5.1.3 Survey". The information gathered from the survey helped indicate which features of a virtual reception are useful to end-users. Based on these points, additional requirements were realised and developed during the realisation process.

6.2 Design

To show that the design competence was met, architectural diagrams such as two flow diagrams and one sequence diagram have been created. These diagrams display the generic flow of the product and the architecture of the application. Next to this, several wireframes such as the Lo-Fi wireframe and Hi-Fi prototype have been created to sketch the application's design.

6.3 Realisation

Docker

A docker environment has been created for all the Rasa services. These different services such as Rasa-X and the Rasa Action server can be run through a single command. More information about how to build and run these docker environments can be found in the Technical documentation

Rasa

A Rasa environment has been created with a set of actions and responses. A custom action has been created, which runs on its own action server. Next to this, Rasa X has been installed and configured, allowing the user to easily add or update the existing chatbot model through a simple to use UI.

Back-end

A back-end has been created using Nestjs, a progressive Node.js framework. Besides this, a PostgreSQL database has been connected to the back-end, containing dummy data about reservations

Next to this, Open API specification(Swagger docs) has been created for an accessible overview of the existing calls and expected return types. Swagger docs can be accessed by running the back-end. Alternatively, the OpenAPI specification has been exported to JSON and can found in the attachments.

Front-end

The front-end is written in Vue3. Besides CSS, Vue also makes use of Vuetify as it’s main UI Component library. The application has been build through components and allows us to create a scalable vue application.

At the time of writing this, both e2e and unit tests Tests are still being implemented. Both these testing frameworks are planned on being finished before the end of the graduation period.

For more information about the workprogress, testing, restrospectives and an overview of the application, please check out chapter: 5.3 Implementation.

7. Conclusion

This chapter contains answers to the earlier mentioned research questions and some more information regarding the finished requirements of the project.

7.1 Research questions

In this sub-chapter, each research question will be answered.

> ***MQ.** What is the added value of a virtual assistant to a digital reception that greets and guides visitors?*

To answer this question and to get a better understanding, the main research question can be split up into the following sub-questions:

> ***SQ1**. How well does the solution work compared to a touchscreen solution?*

A voice-driven application such as a virtual assistant will take away most of the work from a user. Although the implementation of each specific functionality may take time, in practice, users will have a simpler time controlling the application by just making use of their voice. This makes interacting with the application feel like more of a conversation than a screen containing a lot of information and buttons requiring user interaction.

> ***SQ2**. Is the solution suited for people with less knowledge of technology?*

At first, it could be strange for a person to talk to a virtual assistant because they are not used to it. Even though this could be the case, this solution is suited for people with less knowledge of technology since it takes away the technical aspects a digital system has and replaces it with a simplified voice input implementation. This makes the overall process feel more like a normal conversation without any required, complex actions from the user.

> ***SQ3**. How can an incoming person be detected?*

An incoming person can be detected through a simple sensor such as proximity, infrared, or light sensors. Besides existing suppliers with these features built-in, these features can be realised by connecting sensors to the application through serial port or USB port. Besides this, an incoming person can also just be detected by a tap on the screen. Although face-detection through a camera could also be an option, due to invasion of privacy, this would not be a suited solution.

> ***SQ4**. How can the virtual assistant be visualised?*

The virtual assistant can be visualised in different ways. The simplest solution would be a static avatar with a speech bubble or for example subtitles. A step further would be the implementation of a 2D or even 3D model. Currently, the visual presentation of the avatar consists of a static image with subtitles. Due to time constraints, the plans of creating a talking 3d model has not been implemented. However, an example of a lip-sync prototype has been created to mimic the possibilities. Besides this, research has been done regarding these technologies and how these can or may be implemented in the future.

***SQ5**. Which parts should be developed or created with existing solutions?*

Based on the conducted research during the analysation phase, the following parts have been created with existing solutions:

- Text-to-Speech
- Voice-to-Text

Although these features offered were created with existing solutions, they still had to be configured and built into the application. For these features to work together, a front-end, back-end, and chatbot environment has been developed. Since the chatbot requires specific input to run custom actions, this also had to be developed. Custom chatbot actions had to be created and called by communication between the front- and back-end.

Detection of a person could be solved with an existing solution such as a sensor and a developed solution such as a simple button or click listener.

***SQ6.** Is a virtual assistant of added value to big buildings or hospitals?*

Yes, a virtual assistant can be of added value to big buildings since it takes away tasks of a reception and can be used as an addition to or replacement of a staffed reception. Because of a higher complexity and privacy rules surrounding medical appointments for hospitals, a different target was selected, namely municipalities. To confirm this statement, a survey was conducted and questions were asked to business complexes and municipalities. These results can be found in chapter 5.1.3 Survey.

***SQ7.** Which features of a virtual assistant are of added value to a possible end-user?*

Based on the results gathered from the survey, the main features that are of added value to possible end-users are the following:

- Signage to the correct destination of the visitor.
- Visitor registration
- Welcoming visitors

Besides these main features, additional essential features gathered from the survey can be found in chapter 5.1.3 Survey.

When combining all these sub-questions, the following main research question can be answered.

***MQ.** What is the added value of a virtual assistant to a digital reception that greets and guides visitors?*

Based on the gathered results, we can state that it is possible to gather different types of user information through asking a question and use this to handle specific actions such as checking for reservations or obtaining additional information from a database.

Besides just greeting and guiding visitors, the virtual assistant can solve a receptionist's features through a simple, voice-driven conversation or touchscreen alternative. Based on this, we can affirm that a virtual assistant can handle more complex features and present this to the user in a very simple way. Thus, allowing for a user-friendly way of interacting between user and application. With this being said, a virtual assistant is a solid addition to a staffed or digital reception.

All in all, this project was a success. Each sub-question and main research question have been answered. Besides this, all the relevant **Must** requirements were completed except for the first one. (A start screen showing an overview of all the existing companies or departments.) Due to different insights (different primary target market), this requirement changed into the start screen, indicating that the application can be used as a reception.

A good base of the application containing all the essential key features has been developed. Code has been documented, commented, and written in such a way that implementation of additional features for future development will be without any problems.

8. Discussion

This chapter contains information about the continuation of the project and how it can be picked up for future development.

Although the base of the application already exists, there is room for upgrades and implementation of additional features. This chapter concludes all the provided information and additional measurements that have been taken to make sure the future development of this project will be smooth and easily expendable.

Rasa chatbot

Rasa has been set up in a Docker container, meaning that it can be easily implemented and deployed between different environments. To provide more insight in how to add additions to the current chatbot, code has been commented such as the custom actions and further explained in the Technical documentation. A "README" file has been provided, which explains how the application can be set up and run. Next to this, all the code has been commented to ensure that the existing code is easier to understand for future programmers that will be adding and improving the chatbot.

One thing that was somewhat difficult to fully implement was form validation. Since values such as first names or organisation names are unique values, it is hard to map these values to the corresponding group. In the future, extended form validation could be implemented, which compares the name to a list of names (or organisations) gathered from the back-end.

Back-end

At the start of this project, both parties agreed (company & intern) that a dummy database will serve as a temporary back-end. Communication between the front- and back-end can be done through API calls which are documented in Swagger. Besides Swagger documentation, Back-end code has been commented and a "README" file has been provided to explain how this can be run.

Front-end

The focus of the front-end application was to build the application as modular as possible. Modular programming makes the code smaller, easier to read and understand for future developers. Complex code has been commented, and the file structure of the project has been set up in a scalable way, ensuring that when the code base gets larger, the project's structure will maintain clear. From the start of the project, the code conventions of Brease have been followed to make sure the code stays maintainable. Next to this, additional documentation and decisions have been documented and a "README" file is provided, giving instructions on how to run the application.

Scrum board

During the project, Scrum was used. All the requirements have been turned into user stories based on the story conventions of Brease. Existing stories remain on the backlog and can be picked up for future implementation. Each story has been attached to dependant stories (where necessary) and has been colour coded for each part of the application. Since Brease has their user stories written in Dutch, to meet the conventions of Brease, all the user stories have also been written in Dutch.

Virtual avatar

Unfortunately, due to time constraints, an animated visual avatar has not been implemented. However, a basic example of a lip-sync animation has been implemented to show the possibilities. Besides this, certain concepts such as WebGL, meshes, 3Dmodel and libraries such as Three.js that allow for the implementation of a virtual avatar have been documented in the research report.

9. References

The chapter gives an overview of all the references that have been cited (referenced) inside this document.

Introduction to Rasa X. (2021, February 18). Retrieved February 25, 2021, from https://rasa.com/docs/rasa-x/

Introduction to rasa open source. (2021, February 11). Retrieved February 25, 2021, from https://rasa.com/docs/rasa/

Gore, A. (2020, December 07). Vue.js developers-the-future-of-vite. Retrieved March 01, 2021, from https://vuejsdevelopers.com/2020/12/07/vite-vue-cli/#the-future-of-vite

Oord, A. V., & Dieleman, S. (n.d.). WaveNet: A generative model for raw audio. Retrieved March 02, 2021, from https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

Standard and Wavenet voices | cloud Text-to-Speech Documentation. (n.d.). Retrieved March 02, 2021, from https://cloud.google.com/text-to-speech/docs/wavenet

Pricing | cloud text-to-speech | google cloud. (n.d.). Retrieved March 08, 2021, from https://cloud.google.com/text-to-speech/pricing

CEFR levels. (n.d.). Retrieved March 08, 2021, from https://www.eur.nl/en/education/language-training-centre/cefr-levels

Zet U er op TIJD een punt achter? (n.d.). Retrieved March 08, 2021, from https://juistetaal.nl/zet-u-er-op-tijd-een-punt-achter/

Data logging | cloud speech-to-text documentation | google cloud. (n.d.). Retrieved March 08, 2021, from https://cloud.google.com/speech-to-text/docs/data-logging

Security, privacy, and cloud compliance | google cloud. (n.d.). Retrieved March 08, 2021, from https://cloud.google.com/security/

Speech-to-Text: Automatic speech Recognition | Google Cloud. (n.d.). Retrieved from https://cloud.google.com/speech-to-text

Alphacep. (n.d.). Alphacep/vosk-api. Retrieved from https://github.com/alphacep/vosk-api

TalAter. (2020, June 01). Talater/annyang. Retrieved from https://github.com/TalAter/annyang/blob/master/docs/FAQ.md

Text-to-speech-js. (n.d.). Retrieved from https://www.npmjs.com/package/text-to-speech-js

North, F. (2012). Polly. Retrieved from https://aws.amazon.com/polly/

Kuznetsov, The Benefits of the New Vue 3 App Initialization Code. Vue.js Tutorials. https://vueschool.io/articles/vuejs-tutorials/the-benefits-of-the-vue-3-app-initialization-code/.

10. Bibliography

This chapter gives an overview of all the sources that have been used to gather information but have not been referenced inside this document.

@google-cloud/text-to-speech. (n.d.). Retrieved March 08, 2021, from https://www.npmjs.com/package/@google-cloud/text-to-speech

Cognitive speech Services Pricing: Microsoft Azure. (n.d.). Retrieved March 08, 2021, from https://azure.microsoft.com/en-ca/pricing/details/cognitive-services/speech-services/

Hagsten. (n.d.). Hagsten/Talkify. Retrieved March 08, 2021, from https://github.com/Hagsten/Talkify

Make any website talk! (n.d.). Retrieved March 08, 2021, from https://talkify.net/

Neels, B. (2008). Polly. Retrieved March 08, 2021, from https://aws.amazon.com/polly/

Pricing | cloud text-to-speech | google cloud. (n.d.). Retrieved March 08, 2021, from https://cloud.google.com/text-to-speech/pricing

ReadSpeaker speechcloud API - text to speech Production API. (2019, April 23). Retrieved March 08, 2021, from https://www.readspeaker.com/solutions/speech-production/readspeaker-speechcloud-api/

Talkify text to speech services. (n.d.). Retrieved March 08, 2021, from https://manage.talkify.net/docs

Text to speech. (n.d.). Retrieved March 08, 2021, from https://azure.microsoft.com/en-us/services/cognitive-services/text-to-speech/#features

Text-to-speech-js. (n.d.). Retrieved March 08, 2021, from https://www.npmjs.com/package/text-to-speech-js

Text-to-speech: Lifelike speech synthesis | google cloud. (n.d.). Retrieved March 08, 2021, from https://cloud.google.com/text-to-speech

Web technology for developers. (n.d.). Retrieved March 08, 2021, from https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

Https://opensource.org/. (n.d.).

The dot framework. (n.d.). Retrieved from http://ictresearchmethods.nl/The_DOT_Framework

IBM Watson. (n.d.). Retrieved from https://www.ibm.com/nl-en/watson

Make any website talk! (n.d.). Retrieved from https://talkify.net/text-to-speech

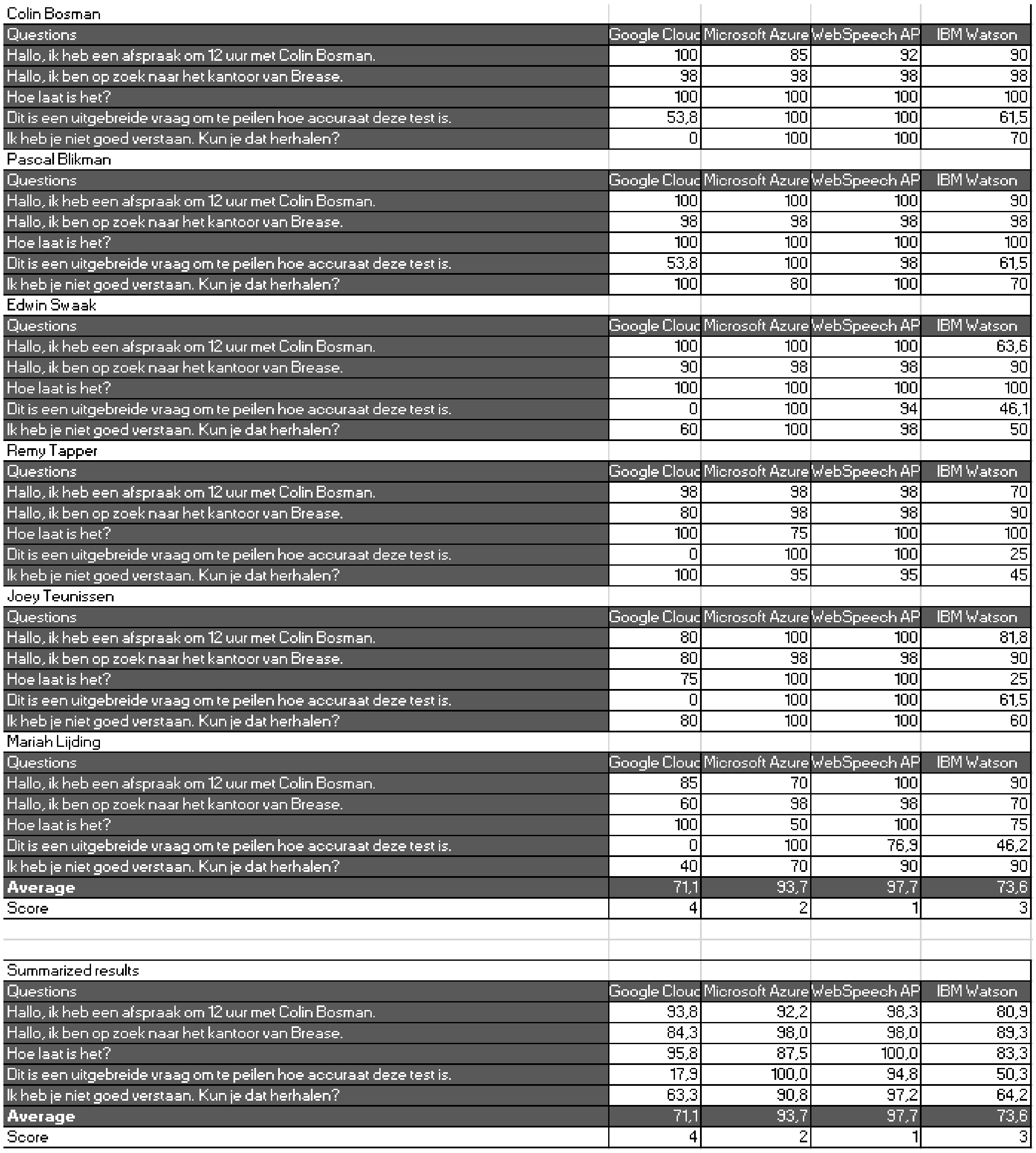
Pollack, G. (n.d.). Vue router: A tutorial for Vue 3. Retrieved April 01, 2021, from https://www.vuemastery.com/blog/vue-router-a-tutorial-for-vue-3/

https://docs.nestjs.com/. (n.d.).

Introduction to Rasa Open Source. Rasa. (2021, April 12). https://rasa.com/docs/rasa/.

Introduction to Rasa X. Rasa. (2021, April 22). https://rasa.com/docs/rasa-x/.

Appendix A



Colin Bosman

Questions	Google Clouc	Microsoft Azure	WebSpeech AP	IBM Watson
Hallo, ik heb een afspraak om 12 uur met Colin Bosman.	100	85	92	90
Hallo, ik ben op zoek naar het kantoor van Brease.	98	98	98	98
Hoe laat is het?	100	100	100	100
Dit is een uitgebreide vraag om te peilen hoe accuraat deze test is.	53,8	100	100	61,5
Ik heb je niet goed verstaan. Kun je dat herhalen?	0	100	100	70

Pascal Blikman

Questions	Google Clouc	Microsoft Azure	WebSpeech AP	IBM Watson
Hallo, ik heb een afspraak om 12 uur met Colin Bosman.	100	100	100	90
Hallo, ik ben op zoek naar het kantoor van Brease.	98	98	98	98
Hoe laat is het?	100	100	100	100
Dit is een uitgebreide vraag om te peilen hoe accuraat deze test is.	53,8	100	98	61,5
Ik heb je niet goed verstaan. Kun je dat herhalen?	100	80	100	70

Edwin Swaak

Questions	Google Clouc	Microsoft Azure	WebSpeech AP	IBM Watson
Hallo, ik heb een afspraak om 12 uur met Colin Bosman.	100	100	100	63,6
Hallo, ik ben op zoek naar het kantoor van Brease.	90	98	98	90
Hoe laat is het?	100	100	100	100
Dit is een uitgebreide vraag om te peilen hoe accuraat deze test is.	0	100	94	46,1
Ik heb je niet goed verstaan. Kun je dat herhalen?	60	100	98	50

Remy Tapper

Questions	Google Clouc	Microsoft Azure	WebSpeech AP	IBM Watson
Hallo, ik heb een afspraak om 12 uur met Colin Bosman.	98	98	98	70
Hallo, ik ben op zoek naar het kantoor van Brease.	80	98	98	90
Hoe laat is het?	100	75	100	100
Dit is een uitgebreide vraag om te peilen hoe accuraat deze test is.	0	100	100	25
Ik heb je niet goed verstaan. Kun je dat herhalen?	100	95	95	45

Joey Teunissen

Questions	Google Clouc	Microsoft Azure	WebSpeech AP	IBM Watson
Hallo, ik heb een afspraak om 12 uur met Colin Bosman.	80	100	100	81,8
Hallo, ik ben op zoek naar het kantoor van Brease.	80	98	98	90
Hoe laat is het?	75	100	100	25
Dit is een uitgebreide vraag om te peilen hoe accuraat deze test is.	0	100	100	61,5
Ik heb je niet goed verstaan. Kun je dat herhalen?	80	100	100	60

Mariah Lijding

Questions	Google Clouc	Microsoft Azure	WebSpeech AP	IBM Watson
Hallo, ik heb een afspraak om 12 uur met Colin Bosman.	85	70	100	90
Hallo, ik ben op zoek naar het kantoor van Brease.	60	98	98	70
Hoe laat is het?	100	50	100	75
Dit is een uitgebreide vraag om te peilen hoe accuraat deze test is.	0	100	76,9	46,2
Ik heb je niet goed verstaan. Kun je dat herhalen?	40	70	90	90
Average	71,1	93,7	97,7	73,6
Score	4	2	1	3

Summarized results

Questions	Google Clouc	Microsoft Azure	WebSpeech AP	IBM Watson
Hallo, ik heb een afspraak om 12 uur met Colin Bosman.	93,8	92,2	98,3	80,9
Hallo, ik ben op zoek naar het kantoor van Brease.	84,3	98,0	98,0	89,3
Hoe laat is het?	95,8	87,5	100,0	83,3
Dit is een uitgebreide vraag om te peilen hoe accuraat deze test is.	17,9	100,0	94,8	50,3
Ik heb je niet goed verstaan. Kun je dat herhalen?	63,3	90,8	97,2	64,2
Average	71,1	93,7	97,7	73,6
Score	4	2	1	3