Development of a web application for an automated user assistant

Vasiliki Giakoumakou

SID: 3306150003

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Mobile and Web Computing

DECEMBER 2017

THESSALONIKI – GREECE

Development of a web application for an automated user assistant

Vasiliki Giakoumakou

SID: 3306150003

Supervisor: Dr. Marios Gatzianas

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Mobile and Web Computing

DECEMBER 2017

THESSALONIKI – GREECE

Abstract

The emergence of web applications that provide automated user assistance as well as the exponential growth in using conversational applications, has led to the discovery of new paths in online customer support.

By taking into account the fact that quick and flexible online assistance is currently vital for engaging customers and by identifying the existing gaps in the automated customer support, we propose and develop a client-side conversational web assistant.

The proposed automated system aims to provide personalized communication through natural language interfaces. Since systems that simulate human conversations using Natural Language Processing techniques are more prevalent today than ever before, we utilize this emerging opportunity for automating a company's complex and time-consuming ordering process through a chatting assistant. The dissertation's conversational application provides work automation and suggests an alternative sales channel for the business.

Additionally, through the application, we introduce a purely client-side approach for developing automated user assistants, in order to enhance speed and independency, by avoiding server-side connections. The proposed architecture eliminates intermediate steps in the back-end data extractions and overcomes common web development struggles by handling all system logic in the front-end.

As a result, we discover the ability of avoiding server-side dependencies and we present the twofold contribution that is derived from adopting a client-side software architecture and from boosting customer experience through a conversational interface for automated assistance.

Vasiliki Giakoumakou

December 2017

Acknowledgments

First and foremost, I would like to thank my supervisor Dr. Marios Gatzianas for his useful guidance and constant availability to offer helpful advice while conducting my thesis project. Also, I would like to thank him for his kind support throughout my MSc studies at the International Hellenic University.

Moreover, I would like to thank my colleagues at the company I work for, since they have been helpful with the information extraction I needed for the dissertation project and they have contributed in the evaluation of my study.

Finally, I want to thank my family and friends for being supportive all these years in everything I choose to do, for believing in me and for strengthening my desire for continuous learning.

Contents

Chapter 1: Introduction

1.1. Background

Nowadays, customers are well-informed and highly demanding, requiring from companies to purchase their products and access their services quickly and efficiently. As technology is rapidly improving, customers want to acquire information the moment they need it, through their own research and real-time communication on online channels.

As a result of being part of such a competitive business world, companies have to adopt strategies that prioritize real-time online assistance, since the need for improving online customer experience has become acute. Through the online services they provide for customer support, businesses achieve to humanize their brands and boost their reputation. Additionally, they manage to eliminate time-consuming and error-prone processes from their employees' workload, while improving the response time in other important procedures.

Moreover, regarding the tremendous growth in developing systems that enhance online customer assistance, there is a significant proportion of them that rely on conversational interfaces. Based on the recent widespread use of messaging applications and the need of users for personalized communication, there is an emerging adoption of conversational agents in the online presence of businesses. These systems satisfy the desire of customers to contact companies through their natural language [1] and keeps customers focused to complete their intention of purchasing a product or service [2].

1.2. Motivation, aims and objectives

Currently, in the online channels of the general business world, the conversational customer assistance is conducted either by chatting live with a representative of a company or through an automated conversational system. However, the first method is cost-ineffective and prone to human errors and the second one lacks speed and independency, since systems rely on server-side connections, which add latency during the interaction with users.

Particularly in the lift industry, the most personalized kind of online assistance is provided through live chatting with a sales representative. However, this service is not as cost-effective as an automated user assistant that addresses the procedure of order creation. Thus, it can be concluded that companies in this industry have not foreseen until now to develop a fully automated conversational web agent.

Given the aforementioned facts, both in the business world in general, and in the lift industry, in particular, the motivation for developing an automated web assistant that would contribute in the online services of modern industries is very strong.

The initial scope of the dissertation refers to the case of a multinational lift company, impelling the business to an alternative sales channel. The system would mainly provide the key benefit of fully automating the lift order procedure of the company. This is a highly complex, time-consuming and error-prone procedure, which requires several subtasks to be accomplished in a specific order through the company's existing ordering system. Thus, there is a great incentive for the automation of the lift order creation through a conversational interface, so as to save valuable time in the company's employees' daily workload and provide work automation based on emerging technologies.

However, apart from aiming to contribute in the lift industry, we are also strongly motivated to develop a system that would address a general problem regarding the current online customer assistance. Through this dissertation, we propose the development of a conversational web application that is entirely based on client-side technologies as an alternative to the current design of web assistants. This system architecture aims to provide speed, independency from external server infrastructure, flexibility and simplicity.

Since software performance matters very much for customers, our proposed system will serve fast client-side interactions, contributing in the elimination of server-side dependencies and connections. The front-end design of the system will lead to the avoidance of the network latency bottleneck that modern applications are suffering from and will reinforce the self-executable capability of the system. Also, all the essential application logic and data storage

will be handled in the client-side, aiming to skip any intermediate step in the information extractions, which may add delay in the system's interactions.

Furthermore, in terms of our general contribution, the software will be usable in different business projects. After applying some parameterizations to adjust our project to the specific needs of other projects, it can be used for several other applications, while maintaining its existing capabilities.

Moreover, in order to fully design and develop our conversational and client-side web assistant, we state some more specific objectives as well.

First, the system will aim to replace human intervention in the procedure of the lift order creation, by adopting chatbot techniques, including the following:

- Natural Language Processing (NLP): The assistant will apply Natural Language Processing techniques and handle text by parsing it into sentences, tokenizing it into terms and Part-Of-Speech tagging it. The Part-Of-Speech taggers will be found through the use of lexicons, suffix regexes and sentence-level Markov Chain processing, by applying grammar rules.

- Spelling mistakes detection: The automated assistant will be capable of decisively handling typographic errors of users based on Levenshtein Distance calculation and anagramism detection.

The application will also provide a user-friendly, aesthetic and minimal interface, in order to enhance the user experience. It will also follow several Human Computer Interaction (HCI) principles, so as to reinforce a user-centric design. Through the design of the application's interface we will provide consistent guidance to users, avoidance of long responses, visibility of sent messages, minimization of the required effort from users and use of User Interface components as "guard rails" in the conversations [3]. Also, the system will ensure that all the veritably substantial information for the lift order will be gathered through the conversation and that the sequence of their collection will be efficient.

Regarding our development practices, we will aim to implement a maintainable and cross-platform system and we will also strengthen the system's security by applying a number of security measures and functionalities.

Finally, the system will focus only on effectively accomplishing specific tasks, instead of trying to carry out a wider range of tasks. By developing a domain-specific system, we will be able to successfully complete specific requests of customers, as well as boost the user experience, since customers will have clear expectations about the system's capabilities and the possible inputs.

1.3. Outline of the dissertation

In this Chapter we have addressed the motivation, goals and the scope of the research. The remainder of this dissertation is organized as follows.

In Chapter 2, we provide the literature review of the online web assistance field. The Chapter begins with describing the assistance types that are adopted by companies in both the general business world and in the lift industry. We also outline the existing online assistance through conversational interfaces, including all the major technical approaches that have been developed until now. Moreover, we present existing diversity in system domains, user types, information sources and channels in which current user assistants are hosted. Based on the foregoing research, we indicate and justify the need for a conversational client-side web assistant in the lift industry.

In Chapter 3 we go on to the design of the system, the methodology and the development process of it. After having discussed the theoretical knowledge in the previous Chapter, we now outline the design and high-level operations of the proposed system. We also describe the languages and technologies that are used in the application, justifying as well the reasons for their selection.

Chapter 4 investigates the implementation of the web application. It presents the User Interfaces and the major flow of the system, as well as the most significant toolkits and tools that are used for its development. Implementation information about the application are also

depicted, along with the major capabilities of the system, such as spelling mistakes detection and Natural Language Processing. Finally, we describe the development and security practices that are applied through the implementation process.

Chapter 5 presents and analyzes the results gathered from the survey performed in the study. In the context of analyzing the survey's results, we outline the questions of the questionnaire, the background of the respondents and the overall findings regarding the use of the application.

Finally, in Chapter 6, we summarize the contributions and conclusions of the thesis and we suggest some directions for future work and research.

Chapter 2: Literature Review

2.1. Business online assistance

In today's competitive world, which is constituted by highly demanding and well-informed customers, companies have to adopt business strategies that prioritize helping customers and quickly satisfying their needs. Customers compose every company's most significant asset and their online support and assistance is a major factor for any company's reputation [2]. Online services, such as a system for user assistance, are able to provide an improved customer experience and subsequently, help keeping customers satisfied.

In the following sections of this Chapter, we will provide information about the assistance types that are generally adopted by companies in the business world, as well as specifically by companies in the lift industry.

2.1.1. Online assistance by companies

In order to have a better reputation and become more competitive and profitable, companies invest in online technologies to provide state-of-the-art services to their most crucial asset, namely their customers. Online services for customer support and assistance are vital for engaging existing and prospective customers. They enhance the exertion of companies to make their customers loyal to their brand, by providing the capability of consistently meeting customers' needs through online channels [4]. Consequently, such online services have a great impact on the overall success of the companies.

There are several types of online customer assistance that can be adopted by companies and applied in their business strategies. The most significant types of online presence of companies are the following ones:

- **E-mail**: The use of e-mails for the communication between customers and companies is still a very wide-spread type of customer service. Furthermore, e-mails can be characterized as a safe and convenient way to contact the employees of a company,

since they can be delivered independently of the company's working hours and customers can be confident that their requests will be read by the customer service team sooner or later. This type of assistance is considered to be a personal service for customers, but unfortunately replies to customers' e-mails are not thoroughly ensured and their response time can take up to several days. As stated in the results of J. Murphy's and I. Tan's research that was conducted for two hundred (200) travel agency companies, only 57 of them truly responded to the received e-mails, which corresponds only to 28% of the total companies [5]. Regarding the response time, 68% of the companies that replied to their customers sent a response within a day [5].

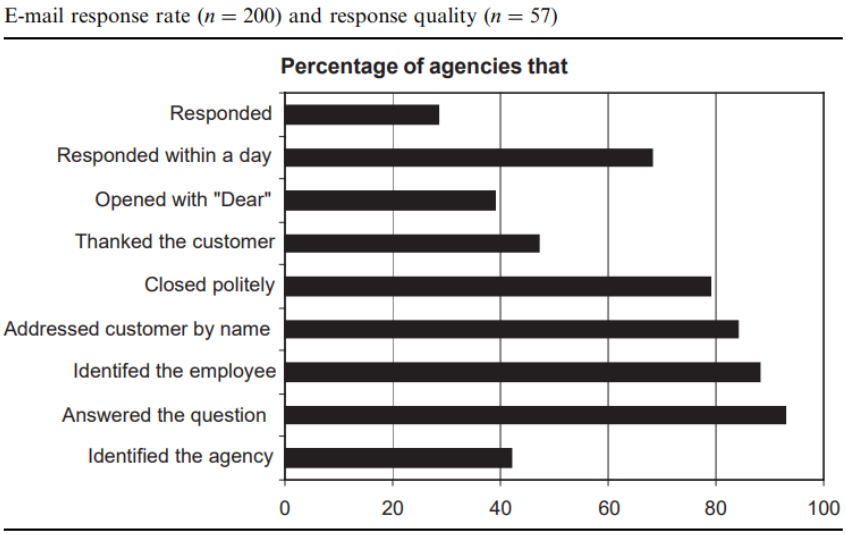


Figure 1. Research results for travel agency companies, indicating the rate and quality of responses (Source [5]*).*

- **FAQs** (Frequently Asked Questions): A FAQ page constitutes a commonly used section in web applications which helps customers get answers to commonly asked questions. Through the FAQ page, the number of received calls and e-mails can be effectively reduced, creating the opportunity of saving a lot of valuable time for both employees and customers. However, this type of assistance is not quite personalized and it often assists customers only for a limited number of issues.
- **Help videos**: Help videos, such as demos, tutorials and webinars, are another popular type of online customer assistance. Videos can successfully help companies share information clearly and quickly and can be very helpful especially for customers who

are visual learners [2]. Notwithstanding the fact that these kinds of services provide online assistance, they are impersonal and they may compel customers to watch a whole video that may not correspond to their real needs.

- **Social media support**: Social media are nowadays a substantial tool to reach out to customers and consequently, companies are able to provide help by communicating through social media channels [2]. By being publicly helpful, a company's commitment to its customers is advertised and its reputation can be uplifted. However, social media channels are not exclusively aimed to be used for customer support. Additionally, they are not part of the official websites of the companies and, thus, employees should additionally monitor their company's social accounts.
- **On-site chat**: On-site chat support is a powerful and rapidly-growing customer support tool that offers real-time assistance. This type of support can efficiently keep customers engaged with a company, resolve their problems or answer their questions quickly and most importantly, keep them focused to complete their initial intent of ordering a product [2]. However, in order for a company to fully take advantage of the benefits of this tool, it should invest either in hiring customer support representatives or in developing chatbot systems that would effectively support users in significant tasks. The opportunity that arises from the rapid growth in the use of messaging applications, facilitates the decision to invest in chatbot systems, providing an automated 24/7 assistance to existing and potential customers.

2.1.1. Online assistance by companies in the lift industry

Since the very first and long-established types of customer assistance in the lift industry, such as phone calls and personal appointments, the tremendous emergence of technology introduced several new and more effective types of assistance. Foreseeing the needs that arose by the technological expansion, the lift companies started providing online services.

The major types of customer assistance that are currently provided by companies in the lift industry are as follows:

- Through e-mail.
- Through online forms that can be either contact forms or order configurators.
- Through a chat dialog with a sales representative or customer support member of the company.

The aforementioned types of customer assistance are available from the lift companies in order to successfully satisfy their customers' needs. However, as we can perceive from the above enumeration of services, there is no type of conversational communication with consumers that is conducted automatically. Although companies offer the possibility of chatting with humans who are representatives of the company, they have not foreseen to adopt any kind of automated communication until now. Even for repeatedly asked and easily automated procedures, lift companies have not created any conversational web agent, such as a chatbot, that can handle these customer requests.

Regarding the current situation, some of the biggest companies in the lift industry invest money in human resources, so that their representatives will talk with their consumers online, through a chat dialog. But while this service is personal and helpful, it is nevertheless prone to human errors and moreover, it is not cost-effective. Furthermore, the customers of lift companies are often other companies (and not individuals) and according to a study that was conducted by Forrester.com in 2016 [6], it is stated that 59% of B2B consumers prefer to not communicate with a representative of a company as their major source of research.

Given the rapid growth of technology, companies are able to use automated conversation systems or chatbots, which would provide the ability of content discovery, through self-service research and communication with automated systems. These systems would simulate human conversations using natural language processing techniques and would provide direct solutions to the questions and needs of the customers, enhancing work automation and cost-effectiveness. Additionally, they could be accessible anytime, allowing thousands of customers to contact the company simultaneously and they could constitute an alternative sales channel which boosts customer experience [7].

2.2. Business online assistance through conversation

The emerging need for conversational agents and chatbots has become acute with the widespread and thorough use of messaging applications by humans, as well as with the desire of customers to contact companies through natural language interfaces [1]. It is stated that customers tend to prefer using their language in order to communicate with computers and that the facilitation of Human Computer Interaction (HCI) is achieved by allowing users to express their queries naturally and directly by typing or speaking [8].

2.2.1. Background of chatbots

In the recent years, there is a wide and rapid growth in the use of chatbots and conversational assistants. The rapidly growing market related to them is expanding its presence in many fields of our economy. In healthcare, there are systems that help people get a diagnosise about their medical condition [9]; in finance, there are systems that are able to make recommendations on asset holdings, whereas for commercial purposes, there are intelligent sales assistants that can offer assistance in the process of product selection and purchase [9].

In 2016, the tech-leading companies Facebook and Microsoft started supporting third-party development of chatbots on their platforms and surprisingly almost overnight after the announcement date, the potential audience for chatbot development has well surpassed the number of one billion individuals [9]. In July of the same year, Satya Nadella, Microsoft CEO, stated that “Chatbots will fundamentally revolutionize how computing is experienced by everybody” [9].

According to a report that was published by Mediapost.com in 2017, it is stated that 65% of customers would engage with a company through a chatbot [10]. This statement indicates that there is a great opportunity for companies to invest in a software that would automate many time-consuming processes of their employees’ daily workload. Moreover, another interesting finding of the same report is that, in 2017, a good chatbot system should aim to

successfully complete specific requests of customers [10] rather than try to handle a greater number of requests.

2.2.1.1. Definition of a chatbot

A chatbot system is a conversational agent which is able to interact with customers using natural language [11]. Chatbot systems ascribe their very first conceptualization to the question “Can machines think?” which was asked by Alan Turing in 1950 [12]. Since that decade, chatbots have been radically improved and have extremely expanded their usage by online users and customers. More precisely, their popularity increased so much that, by 2016, the social platform Facebook hosted around 30,000 bots and in parallel, it had around 34,000 developers to implement these automated assistants [13].

2.2.1.2. History of chatbots

The idea of creating communication between humans and computers goes back to the 1950’s, when the well-known British mathematician Alan Turing [14], posed the question “Can machines think?” in his paper “Computing Machinery and Intelligence” [15]. Alan Turing theorized that an intelligent machine should be indistinguishable from a human during a dialog [16]. Based on this theory he proposed a test with the name “the Imitation Game” (also known as “the Turing Test”) [14], which is considered to be the forerunner of modern chatterbots [14]. The Turing test was successfully passed if the interlocutors of the computer were not able to distinguish whether they were talking to a human or a computer machine [14]. Since that decade, a lot of effort has been made in order to develop a computer-based system that would be able to chat with humans and pass the Turing Test [17]. The effort of scientists regarding this problem has resulted in the development of a variety of approaches and systems that refer to the human-computer communication [17].

Starting in 1966, the very first chatterbot in history and earliest example of chatterbot design, named ELIZA, was developed by Joseph Weizenbaum [14]. The aim of ELIZA was to behave as a Rogerian psychologist [17], mimicking their responses and carrying on human conversations

[16], using simple pattern matching. Despite the low-level conversational ability of ELIZA, it was still able to confuse people at that time, since humans were not familiar with chatting with computers [17].

In 1972, a chatbot named PARRY was the first system that was evaluated by humans based on the Turing Test [17]. PARRY was developed in order to communicate as a person with paranoid schizophrenia. Its transcripts were given to psychiatrists for comparison with transcripts of really affected humans and psychiatrists were able to make correct identifications only 48% of the time [17].

Furthermore, Jabberwacky, a chatterbot that was developed in 1988 (and released online in 1997), was designed and developed in order to communicate with humans in a humorous and entertaining way [16].

Moving forward to more advanced systems, in 1995, a new chatbot named A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) was created by Richard Wallace. This chatbot utilizes AIML (Artificial Intelligence Markup Language) to form responses to users' questions and inputs [18]. The open-source chatbot ALICE was able to use more advanced pattern recognition techniques than the aforementioned chatbots [14] and it became an award-winning free natural language artificial intelligence chat robot [18]. ALICE used AIML (Artificial Intelligence Markup Language) as its main language to form replies [19].

Moreover, in 2001, a chatbot named "SmarterChild" was released, achieving to be the first one with a widespread adoption [16]. Using an approach of building chatbots for popular messaging platforms, SmarterChild managed to be the archetype that modern chatbot builders are following today [16].

Since the previous decade, chatbots are gaining popularity again due to the success of messaging applications, the rapid growth of web and mobile applications that use artificial intelligence techniques and the popularity of virtual assistants, such as Apple Siri [17].

2.2.2. Technical approaches, languages and tools

Regardless of the approaches and design techniques that are used for existing chatbots, the main purpose of them overall is to perform a conversation with humans, which would be in natural language form and human-like [17]. Simultaneously, the major challenge faced by existing chatbot systems is understanding human inputs and responses, while maintaining the context of a conversation [17].

Pattern Matching

The method of pattern matching is a common technique used in several chatbot systems, in order to classify users' text and produce suitable responses for them [20].

Since earlier chatbots, such as ELIZA, pattern matching approaches have been widely used and adopted for bot development [17]. All of these approaches are based on a basic common idea, but simultaneously, they can vary in their complexity [17]. Actually, the majority of the existing chatbot systems works on the pattern matching of inputs that are matched with a scripted response [17].

Based on the approach of using scripted responses, scientists keep adding new functionalities to existing chatbot development techniques and apply improvements to them, so that chatbots can become more successful. As a result, computer scientists have introduced a number of different systems and approaches regarding the same problem [17].

Textual Parsing

Textual parsing is an approach which refers to the conversion of an original text into a set of words (lexical parsing) [17]. Through this method, the grammatical structure of words is clearly determined and based on that, a check can be applied indicating whether this structure forms a syntactically valid expression or not (syntactical parsing) [17]. Figure 2 presents an example of a parse tree, referring to the grammatical structure of the textual parsing approach.

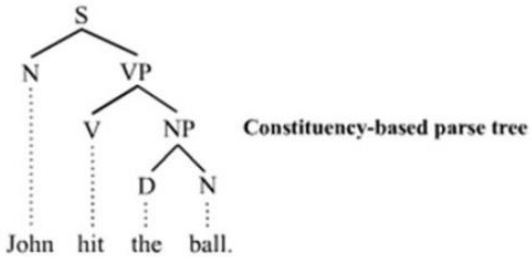


Figure 2. Example of a parse tree (Source: [17]*)*

Regarding the appliance of this approach in existing chatbots, the earlier parsers were looking for identifiable keywords in an acceptable order [17]. For instance, both the sentences "please order a laptop" and "can you order a laptop" would be parsed into the keyword phrase "order laptop", thus a restricted set of patterns would be able to cover numerous input sentences [17]. In modern chatbot systems, the parsers that are used are more advanced, covering the complete grammatical parsing of natural language sentences.

Markov Chain Models

Markov Chain Models refer to the notion that the occurrence of a word in a sentence probabilistically depends on other accompanying words, i.e. there exist conditional probabilities of words given that other words also appear in the same sentence. In chatbot design and development, they are used in order to generate responses that are probabilistically more viable and applicable and consequently more accurate [21]. Moreover, Markov Chain Models are even used to create non-sense replies that sound right, as fallback techniques, in chatbot systems [17].

Ontologies (semantic networks)

In chatbot systems, an ontology or semantic network is "a set of hierarchically and relationally interconnected concepts", that can be used in order to determine synonyms and other associations between these concepts [17]. The benefit of ontologies (or semantic nets) is that they provide the ability of a graphic interconnection of the concepts, enabling computers to conduct searches using special reasoning rules [17].

Artificial Intelligence Markup Language (AIML)

Based on the Pattern Recognition technology, the language AIML (Artificial Intelligence Markup Language) is adopted by computer scientists for the development of chatbot systems [14]. AIML is an XML-based interpreted language and it constitutes the major technology that is implemented in well-known bots, such as ALICE [17]. It mainly consists of input rules with respective outputs for each one of them [17] and it can be referred to as a tag-based language, which uses tags as identifiers that generate the chatbot's code snippets and commands. AIML models patterns of dialogs and enables chatbots to respond to anything that is related to an associated pattern. However, in this case, chatbot systems do not have the ability to handle inputs that are beyond the associated patterns.

Regarding the use of AIML code in chatbot systems, Figure 3 presents an example of an actual use of the language.

```
<aiml version = "1.0.1" encoding = "UTF-8"?>
  <category>
    <pattern> WHO IS ABRAHAM LINCOLN </pattern>
    <template>Abraham Lincoln was the US President during American civil war.</template>
  </category>

  <category>
    <pattern>DO YOU KNOW WHO * IS</pattern>
    <template>
      <srai>WHO IS <star/></srai>
    </template>
  </category>
</aiml>
```

Figure 3. Example of AIML code (Source: [20]*)*

Thus, based on the aforementioned code snippet, in the question "Do you know who Abraham Lincoln is?" the system would answer "Abraham Lincoln was the US President during American civil war.", since their name is included in the associated pattern [20].

ChatScript

ChatScript is a language that is intended to be a successor of the Artificial Intelligence Markup Language (AIML). ChatScript's syntax is more advanced than AIML's and the language is also considered to be more easily maintainable. Furthermore, ChatScript introduces more

functionalities than AIML, including fixations to matching problems and additions of concepts, variables and functions, in order to cover the need for ontologies inside the script [17].

Natural Language Processing

Natural language processing (NLP) aims to take users' unstructured phrases and convert them into a structured output, containing natural language understanding (NLU) [12]. The structured data that is generated can be used in order to select the related answer [20]. Natural Language Processing includes some major steps that are presented below:

- Tokenization: Through NLP, tokenization is the process of breaking down a sentence into tokens that represent each of its parts (such as words, numbers, and punctuation marks) [12].

- Named Entity Recognition (NER): In this step, the chatbot system searches for categories of words (for instance, the name of the product or the user's name) [20]. In the process of NER, the names of people and products will be extracted and labeled accordingly. NER-name pairs will be stored in the conversation's history so as to keep track of the context of the conversation [12].

- Normalization: Through the normalization process, text is broken up into its component parts; sentences are broken into words and punctuation marks and words are broken into their phonemes (the smallest unit of sound in speech [22]) [12]. A chatbot system may process user's text in order to find ordinary spelling mistakes, which would result in a more human-like conversation when they are used by the chatbot inside the dialog [20].

Figure 4. Sentence labeled with Information Extraction (Source: [12]*)*

- Part of Speech (POS) tagging: POS tagging aims to label each word of the users' sentences with its part of speech (for instance, "noun", "adjective" etc.). Also, a word's

label can be rule-based, which refers to the ability to create a manually-generated set of rules for ambiguous words.

Figure 5. Sentence labeled with Part-Of-Speech (Source: [23]*)*

- Dependency Parsing: In the dependency parsing step, a chatbot system searches in the user's text in order to find objects and subjects which may have some kind of dependency or relation [20].

Natural Language Toolkit (NLTK)

NLTK is an open-source toolkit for Python, which uses natural language technologies in order to handle text conversions [21]. More specifically, the widely used toolkit NLTK organizes the inserted text into sentences and then splits them into words to make the meaning extraction easier [21]. NLTK is able to define parts of speech by tagging word labels, as well as, by using grammar rules to categorize the tagged words into groups based on their positions and neighbor words [21].

2.2.3. Domain diversity of the web assistants

The online assistants that aim to provide help to users can be classified, based on their range of help, into general-purpose and specific-purpose systems. The general-purpose assistants aim to understand user's inputs regardless of the type of the question they ask or the topic their question is related to. For instance, a question that a user may ask can be either about the weather forecast or about real-time transportation information. In this type of assistants, there is not a specific domain in which the assistant corresponds, but the system should try to continue the conversation with the user regardless of the context of their dialog.

Concerning the specific-purpose systems (or consumer domain-specific systems [12]), which constitute the most common type of web assistants, their aim is to carry out specific tasks of

a business. As we have already mentioned, according to several reports, such as in Mediapost.com, a good online conversational agent pertains in this category; it should successfully conduct specific tasks in serving a company's customers [10] rather than try to handle a greater number of their requests. Consumer domain-specific assistants are designed to manage requests for companies in several different industries. For example, in flight transportation, an assistant's aim could be to provide real-time flight tracking to the customers, while in medicine, an assistant could provide accurate diagnosis of an illness, based on the user's symptoms [12].

2.2.4. Users of the web assistants

The types of users that communicate with automated assistants can be classified into two major categories. The first one refers to users that are either non-experts or whose expertise is not important to the system. The second one refers to experts, with strong background and knowledge in the field that the assistant serves.

A system's type of users constitutes one of its most fundamental features, since the whole user interface design and the system's functionalities will be designed and developed based on them, in order to effectively serve their specific needs.
In the lift industry, most of the companies communicate with customers that are professionally related with the field and sell products as B2B companies; hence, the web assistants' users are engineers with expertise and background knowledge in lifts.

2.2.5. Information sources (client and server-side)

Regardless of its implementation techniques, every system needs access to data, in order to be able to carry out its tasks [24]. As part of a web assistant's strategy, the decision of what sources of information and knowledge will be used is substantial for its implementation [24]. The knowledge base of the assistant can be placed either in the client-side or in the server-side. The latter choice is the most common one among existing assistants. In every interaction with the user, the system communicates with the server-side application to get data and provide appropriate responses to the user. On the other hand, the use of an entirely client-

side system, reinforces the self-executable capability of the application and eliminates every kind of delay. Access to data, knowledge and definitions of processes are already provided in the client side of the system, and there are therefore no intermediate steps in the information extractions and any kind of delay can be avoided.

2.2.6. Channels in which the assistants are hosted

There are several channels in which automated user assistants are hosted and every company can decide based on its needs where to locate its assistant. Most frequently, these systems are included in the official website or web portal of a business [24].

Moreover, the automated assistant can be placed in a mobile application [24], constituting a substantial part of its functionalities. When the company's customers use the application, the assistant is at their disposal for chatting through the provided dialog interface.

Furthermore, automated assistants can be embedded in external platforms which are channels that customers already use [24]. For instance, the Facebook Messenger is a widely-used application that hosts a tremendous number of assistants through its interface. However, unless a company targets every individual in the market, it has to find out the most suitable channel for its existing and potential customers [24].

2.2.7. Need for a conversational client-side system

Summarizing and evaluating the most significant aspects of the existing systems, it is clearly evident that web assistants, in general, as well as conversational web assistants, in particular, rely for their functionality on the server-side. This widely-used back-end architecture lacks speed and independency, since the User Interfaces depended on a continuous communication with the server. The need for connecting with the server (i.e. sending and receiving data from it) adds a delay in each interaction of the user with the system.

Therefore, it can be assumed that the development of a web assistant which handles the system's logic and data storage in the client-side can overcome the foregoing disadvantages, since the interactions with the back-end are minimized. A web application with a client-side design can be thoroughly independent of the server-side and, consequently, it can accomplish any interaction with the user quickly, without any connection delays.

Also, regarding the lift industry particularly, we can easily identify the gap in the automated conversational customer support. Hence, there could be a system that automates online customer assistance, differentiates the company from the competition and humanizes its brand, without the need for an actual salesperson to interact with its customers.

Based on the aforementioned facts, we propose a conversational web assistant that aims to meet all the foregoing needs, as well as constitute a fast, independent and flexible software platform. Our system is mainly designed and developed using front-end technologies, which significantly boost its speed and independency and also give it more flexibility and power.

Additionally, the proposed system, which uses conversational interfaces and handles natural language, may also be used for other applications after the implementation of some code parameterizations. For example, other companies or development teams can potentially adjust our web assistant to their needs, by changing keywords and other parts of the software that are specifically related with the lift company, while keeping unchanged the rest of its already implemented functionalities.

In the following Chapters, we extensively describe the design and development process of our conversational web assistant and we also provide any needed implementation information about our application’s interfaces and functionality.

Chapter 3: Design, Methodology and development process

Nowadays, the number of web and mobile applications that rely on conversational interfaces to nurture relationships with customers is growing [25]. However, as descripted in the previous Chapter, the existing customer assistance in the lift industry has been less concerned with providing automated web solutions through chat interfaces.

With technology evolving and language recognition improving, interaction with digital systems through conversational interfaces can become more intuitive, efficient and accessible [25]. Through the design of the company's web assistant we aim to fully utilize this emerging opportunity for the business's ordering process.

After an extensive research on the online web assistants, several useful design techniques are chosen to be applied to our web application, aiming in the creation of an efficient and usable system. In this chapter, we will thoroughly discuss the system's design, outline its high-level operations, as well as depict its selected languages and technologies.

3.1. Tasks, general flow and output of the web application

The system aims to thoroughly assist the complex process of the order creation through a conversational web interface that creates orders for its users. It is responsible for submitting orders containing the user's inputs, finalizing them in the company's existing ordering system and sending to the user the respective files (invoice and drawing files) of each order.

The complex process of order creation is composed of several smaller tasks, which combined together provide a complete and efficient procedure to the users. The application's tasks are conducted in modern front-end technologies, which use complex algorithms, under a friendly and minimal conversational interface.

Regarding the conducted tasks of the application, it first collects from the user the essential data for the lift order. Thereafter, based on these substantial information, the web application

conducts its algorithmic processing in order to quickly offer to the user the available lift options that correspond to their needs for the specific order.

As a part of the general order procedure, the application, then, uses complex engineering know-how algorithms that create a realistic plan view of the building's shaft.

Moreover, the application provides a preview of the lift, including all of its design parts and allows the user to configure, through the web interface, the lift design with the materials of their preference.

The user is also able to indicate additional specifications and constraints related with the specific order, through free-form conversational input. The web assistant will include this information in the order and conduct any algorithmic calculations and adjustments needed for the information's insertion in the final invoice.

While communicating with the user, the web assistant is simultaneously responsible for handling the natural language input (including possible spelling mistakes), through natural language processing tools and keyword detections, validating user inputs and also, running several algorithms related to the lift engineering field. The system has to conduct these procedures so as to sufficiently and quickly accomplish its tasks through its interface.

Furthermore, in order to boost the Human Computer Interaction and provide a positive User Experience, every aforementioned process, validation, calculation and provided output by the web assistant is instantly accomplished while interacting with the user. Hence, although the procedure is complex and it is composed of several subtasks and algorithms, the speed and efficiency of the application constitutes one of its greatest benefits.

Finally, as a last part of the process before submitting the order, the user selects whether they prefer to receive only the invoice of the order or both the invoice and the drawing. Right after this selection the assistant is responsible for communicating with the existing back-end system of the company, creating and finalizing the order through web services and thereafter, sending the requested files to the user's e-mail.

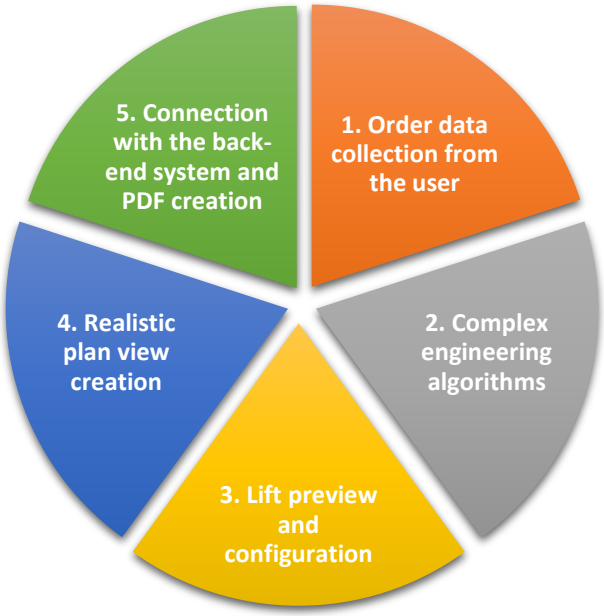


Figure 6. The tasks of a lift order

Moreover, as indicated in the next diagram (Figure 7), the orders can be classified in two pivotal cases: the first one is based on the lift load that the user prefers, referring to a building that is not built yet and the second one is based on the shaft constraints of an already existing building.

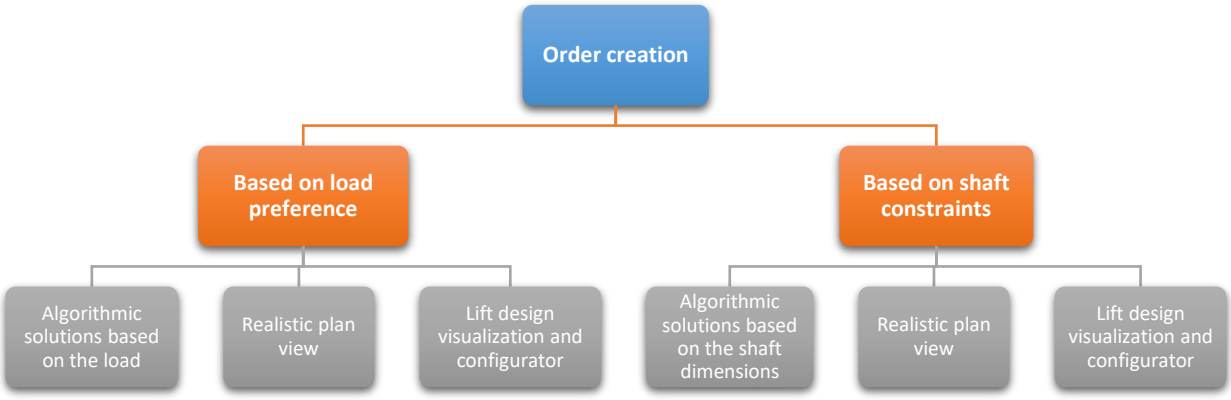


Figure 7. The two major cases of the lift orders

3.2. Question types of the application

For the design of the web application, different types of questions were selected to be used for a smooth conversational experience. The question types are outlined as follows:

- **Questions with open-ended answers**: Users can insert free-form conversational text.
- **Questions with close-ended answers**: Users have to insert their answers, based on specific restrictions. We can address our close-ended answers in two major categories:
  1. The expected answers are standardized and they consist of replies in the form of options (such as, “yes” and “no”).
  2. The expected inputs are referring to a specific type and range (e.g., a number between 2 and 10).

3.2.1. Questions with closed-ended answers

For the design and development of the web assistant, we consider the use of the aforementioned type of questions, which include close-ended answers. These questions are asked by the system and their allowed answers are a few given options, which are suggested by the assistant as options in the chat dialog. Furthermore, in the questions with closed-ended answers, the assistant may also prompt the user to insert a string (word or number) with a restricted type and range. In both cases, the user can either type the answer they prefer or click on an option (if it is provided in the dialog), in order to select it.

By using the aforementioned close-ended type of questions, we achieve the following beneficial results for the system’s usability and user experience:

1. We guide the user to insert some inputs that are veritably substantial for the lift ordering process. These inputs are so important, that a failure to provide them corresponds to incomplete order data, since they are required for every order, regardless of the customer’s preferences for the product. Hence, by designing the assistant to ask users for this information and guide them while they are providing

these data, we ensure that the most important order specifications are successfully collected.

2. We ensure that the most significant information is gathered from the user in the most efficient sequence for the flow of the ordering process and the application's algorithms. We choose to ask for these data in a proper sequence, resulting to a design that reinforces the user's experience and provides a clear and trustworthy data collection.

3. We avoid error-prone actions, since we provide questions whose allowable responses are of specific content or they are suitably proposed by the assistant. We generally aim to prevent a problem from occurring in the first place. Thus, users have to insert permitted options and values, so that their requirements will not lead to an infeasible lift order.

4. We enhance the user experience by giving users the key advantage of completing the whole ordering process with fewer keystrokes than regular applications [26]. By providing substantial questions for the ordering process along with some pre-defined answers, we help users submit their requirements time-efficiently and with the least possible effort.

5. We achieve to provide a concise conversational experience, since the web assistant presents users with their specific options [25]. This kind of conversation is composed of questions and answers which are short and to the point, avoiding irrelevant options.

3.3. Design principles

In traditional GUIs, long established and standardized design techniques can be adopted, in order to develop an efficient system. However, in conversational-driven interfaces, thinks get a little tricky and the developers have to carefully adopt specific design principles that will sufficiently satisfy the needs of the application.

The User Experience (UX) and usability principles which are applied to the graphical interface of our web assistant are presented as follows:

- Avoidance of long responses [3]: Each response that is generated by the system is composed of a few lines of text. There is a tendency of users avoiding to read long messages in applications, which in modern web development is widely-known with the phrase "*too long; didn't read*" (tl;dr) [3]. So, our system is designed to provide short and concise responses [3].
- Visibility of the responses: Along with using short responses, we also ensure that we provide visible replies. We do not want the user to have to scroll up/down in order to read a whole message, since this would be bad for their experience while interacting with the system [3].
- Personality that offers the user a more pleasant experience [27]: In our web application, the chosen communication style fits with the company's customers and brand, which keeps a balance between its professionalism and friendliness. The system is designed to interact with the company's clients and employees and consequently, its design aims to create an assistant with a polite personality. All of the answers that are provided by the web assistant are kind and cheerful, enhancing the professional image of the company, as well as customers' familiarity and engagement with the company.
- Aesthetic and minimalist design: Dialogues do not include information that is irrelevant to the context or seldom needed [28]. Each additional information in a conversation diminishes the relative visibility of the relevant units of information [28]. Although the application's conversation includes complex procedures, it maintains its minimal design, so as to provide an aesthetical interface to its users.
- Playful design: Moreover, the application's conversational interface is composed of images, icons, transitions and other modern web animation tricks when possible, so as to enhance the playfulness of the system and boost the user experience.
- Consistent guidance of the user and use of UI components as "guard rails" in the conversation [3]: The web application sufficiently guides its users to keep them

engaged [29]. This guidance is achieved through actionable buttons and quick replies, which indicate to the user what to do and which options are available for each question.

- Constant suggestion of the next steps of the conversation: The system is always alert to suggest the next steps, drive the conversation forward and even restrict it if it is needed [27]. It is significant for the system's design to constantly provide useful information to the users and maintain a continuous and positive experience for them.
- System with clear value proposition: The web application is focused on doing specific procedures correctly, rather than barely handling a greater number of processes. Since using, handling and validating natural language interactions is hard, the aim of the web assistant is to conduct specific tasks properly, providing a great experience to its users.
- Variety in the format of the messages that web assistant sends: There are always different ways in which the assistant asks a question or sends a statement. With this method, we achieve a more natural communication, since the assistant offers different forms of wording.
- Time-saving mentality for the application's users, in order to minimize the keystrokes that are needed from them: There is no need for the users to type their whole preferred answers, since the system instantly filters the user's inserted text and foresees their preferred lift options.

3.4. Programming languages and libraries

For the development of the web assistant there is a wide use of different programming languages and libraries. Thousands of lines of code were written, in order to create a fully-functional web assistant that facilitates the company's processes and enhances its customers' experience.

The emerging programming languages, libraries and frameworks that are chosen to be used in our web application are as follows:

- JavaScript
- jQuery
- HTML 5
- CSS 3
- PHP
- Bootstrap

The majority of the software is implemented in the programming language JavaScript. This language, along with the aforementioned languages HTML and CSS, constitute the triad of technologies that the vast majority web developers use, in order to specify the content of web pages (HTML), the presentation of web pages (CSS), and the behavior of web pages (JavaScript) [30].

3.4.1. JavaScript

JavaScript is the programming language of the Web [30]. It is considered as an interpreted programming language, which is mainly used for enhancing webpage functionality and interactivity [31]. Software written in JavaScript, which is deployed in HTML documents, provides useful client-side computation facilities and access to the client system, making web pages more interactive, engaging and responsive [32]. Nowadays, the overwhelming majority of modern websites use this language, and all major web browsers on tablets, smart phones and desktops, include JavaScript interpreters, making this language the most ubiquitous programming language in history [30].

Figure 8. JavaScript logo
(Source: https://www.computerhope.com/jargon/j/javascript.jpg)

3.4.2. Use of JavaScript and client-side development

As mentioned in the previous Chapter, web application functionality is rapidly migrating from the server-side to the client-side, making JavaScript front-end applications a standard in web development due to their speed, flexibility, power [33] and efficiency.

Also, as technology is improving, performance is becoming a factor that matters very much for the users, who expect fast interactions while using a software. With the adoption of client-side applications developed with JavaScript, high performance can be efficiently achieved [34].

Moreover, the most significant advantages of JavaScript web applications are their speed and immediacy, since they do not suffer from the common web application's frustrating delay associated with server-side programming languages [35]. This a key benefit, since one of the most crucial bottlenecks in web applications is network latency. The delay that stems from the time needed to make a request to the server and receive its response back can account for a huge portion of page load time [34].

The problem is magnified in cases of unstable Internet connections (especially, through mobile phones). Therefore, by developing a front-end web application, which includes limited interactions with the server-side, the application is fast and satisfies the demanding users' requests without delays [34]. Thus, JavaScript web applications enable the capability of providing instant responses to the user's requests and fundamentally improving the Human Computer Interaction (HCI).

Furthermore, front-end web applications do not need to be pre-compiled and do not require the installation of a plugin [33]. The whole HTML rendering is conducted in the web browser of the user, retaining the minimum interactions with the server [34]. The applications can be tested quickly and allow for easy programmatic manipulation of HTML Document Object Model elements [33].

Another crucial advantage of JavaScript front-end applications is the fact that they constitute universal (cross-platform) systems. Unlike back-end languages, such as PHP, web applications

that are developed with front-end technologies (JavaScript, HTML5, CSS3), can run into all types of web pages [36], web browsers, operating systems, devices and screen sizes.

Additionally, regardless of the technology that is used on the server-side of a system, front-end languages can be used to generate a rich and fully-functional software. This is a fact that enhances the reusability and the insulation of a web application against a changing landscape in server-side systems [34].

Finally, the industry momentum behind front-end technologies is also considerable, since JavaScript is currently the world's most popular web programming language [34] and has effectively supplanted other robust languages in web applications, such as Java. Nowadays, the vast majority of web applications are JavaScript front-end applications. Hence, our web assistant which is also based on front-end technologies, is a part of an enormous online community which offers great support and open-source solutions.

3.4.3. jQuery

jQuery is an open-source and cross-platform JavaScript library. According to W3Techs [37], jQuery is used by 96.2% of all the websites whose JavaScript library is known, which is a highly impressive percentage, corresponding to 72.9% of all currently existing websites.

Through jQuery, JavaScript commands are simplified and code lines are more readable and maintainable. The library comes with an easy-to-use API that can be properly interpreted by a multitude of browsers [38].

3.4.4. Use of jQuery

For our web assistant, we extensively use the library jQuery which has many advantages. According to D. S. McFarland, some of the library's significant benefits are as follows [39]:

- It has a relatively small size and can be quickly loaded by browsers.
- It is supported by a huge developer community, which facilitates the process of development and ensures that the library will be used by web applications for many years.

- It is open source and it has been already tried and improved by millions of web developers and web applications engineers.

Moreover, although the pure JavaScript language is simpler than other languages, software developers still face difficulties when writing code using its syntax. Another typical difficulty that can cost the developers many working hours, is the process of testing applications on different machines and browsers, in order to ensure that they run properly.

On the other hand, the well-known jQuery library can overcome the foregoing difficulties, since it both simplifies complex tasks and solves cross-browser problems [35]. More specifically, with jQuery, complex tasks can be accomplished even in a single line of code and much testing work can be avoided [35]. Thus, unsurprisingly, jQuery is extensively used not only for the needs of our web application, but for millions of websites as well.

Figure 9. Icon of jQuery library (Source: https://brand.jquery.org/logos/)

3.4.5. HTML5

HTML (Hypertext Markup Language) is the core language of the World Wide Web [40] for creating the structure and content of web pages. HTML5 refers to the fifth major version of HTML and constitutes a natural evolution of its earlier versions [41]. The cross-platform language provides plenty of web elements that aim to describe the content of Web pages [41], such as titles, paragraphs, tables and images.

Figure 10. Icon of HTML5 (Source: https://www.w3.org/html/logo/downloads/HTML5_Logo_256.png)

3.4.6. CSS3

CSS (Cascading Style Sheets) is a cross-platform language that describes the presentation of web pages [42] and how their elements will be displayed (such as their colors and sizes). CSS3, which is the third major revision of CSS, provides a more powerful way to style web pages through visual effects (such as shadows and gradients) [41] and it allows developers to create pages that adapt their layouts to different screen sizes.

Figure 11. Icon of CSS3 (Source: https://botw-pd.s3.amazonaws.com/styles/logo-thumbnail/s3/042015/css3.png?itok=OlYIVwA0)

3.4.7. Bootstrap

Bootstrap is one of the most popular front-end frameworks in the world [43]. It is an open-source toolkit for developing responsive, "mobile-first" web projects with the front-end languages HTML, CSS and JavaScript [44]. It was initially developed to standardize the frontend toolsets of engineers across the social-network company of "Twitter" [45] and nowadays, it has the file structure of Figure 12.

```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.min.css
├── js/
│   ├── bootstrap.js
│   ├── bootstrap.min.js
└── img/
    ├── glyphicons-halflings.png
    └── glyphicons-halflings-white.png
```

Figure 12. File structure of Bootstrap
(Source: http://getbootstrap.com/2.3.2/getting-started.html#file-structure)

Figure 13. Icon of Bootstrap (Source: [44])

Chapter 4: Implementation of the application

This Chapter presents the User Interfaces and the major flow of the web application, as well as the most significant toolkits and tools that are used for its development. Implementation information about the application is also provided, along with the major capabilities of the system, such as the spelling mistakes detection. Finally, the development and security practices that are applied through the implementation process are described.

The client-side application of the thesis, which is thoroughly described in this Chapter, is mainly developed in front-end technologies from scratch. Through its conversational interface, the web application assists the user in the process of order creation, accomplishing the whole process quickly due to its front-end driven design. The lightweight software also comes with a minimalistic and friendly interface, so as to boost the Human Computer Interaction (HCI).

4.1. Description of the web application's flow

When opening the web application in the web browser, the system initially asks the user to insert their credentials (i.e. their username and matching password), so as to gain access to the application's main interface. When the fields of the login form are submitted, our web application interacts with the company's server-side system through Restful APIs. The response of the server-side system, which is encrypted in order to enhance the application's security, indicates whether the submitted credentials are successfully validated or not. In case of failure, the system informs the user that either the username or password is invalid. Otherwise, in case of success, the authentication is successfully completed, the user's session token is initialized and the procedure can properly continue to the next screen of the web application.

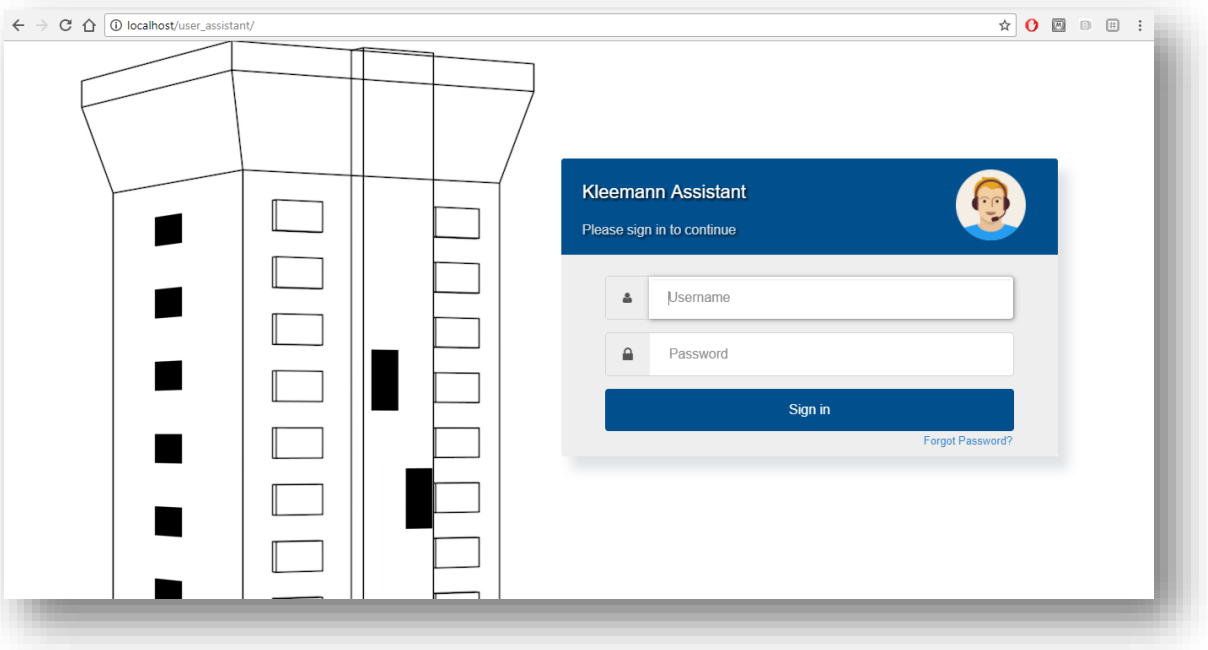


Figure 14. Login page

After the successful login of the user into the system, they are automatically redirected to the web assistant's main interface (Figure 15). The minimalist design of the interface, aims to avoid irrelevant context and focus on the system's purpose to provide help to the user. After the page is loaded, the assistant sends a message to the user asking them if they need any help, as Figure 15 depicts. Each time the assistant sends a message to the user, the phrasing of the question is randomly selected among a few possible available options. So, instead of asking "Can I help you?", the system could alternative send "Do you need any help?" instead.

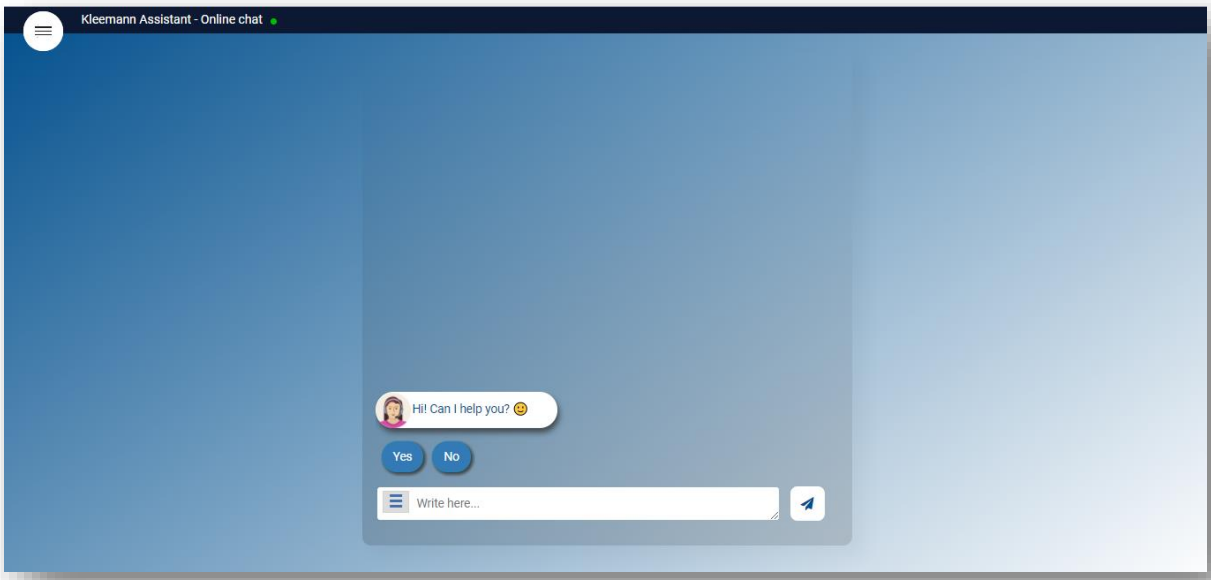


Figure 15. Main interface

The user can choose one of the available options (“Yes”, “No”), either by clicking on the option they prefer or by typing it. On each keystroke, the system checks the inserted character, in order to instantly filter and foresee the user’s preferred option, before they have to fully type the whole string (Figure 16). Thus, through the instant filtering, we manage to save valuable time from the user. In our example, the user can just type “y” to indicate that they want to say “Yes” and then, press enter or the submit button on the right to send their selection. The option “No” is automatically hidden from the system.

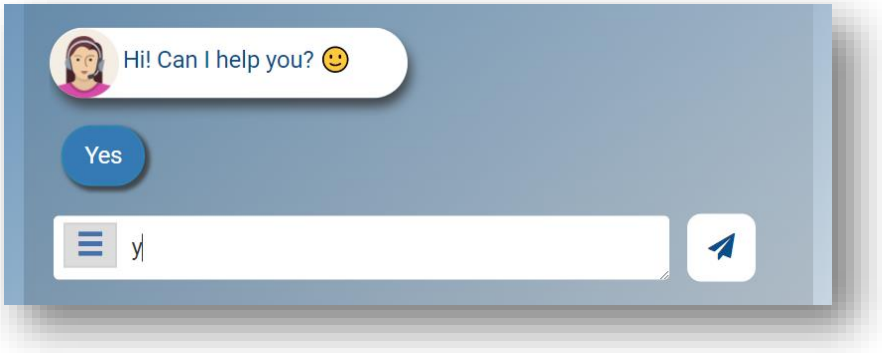


Figure 16. Instant filtering of an input

In case of answering “Yes”, the assistant asks the user if they want to create a lift offer based on their preferred load or based on existing shaft dimensions. This information is crucial for

continuing the dialog and it refers to one of the most substantial data of the order, which categorizes it in its appropriate order type.

Also, regarding the close-ended questions of the application, we aim to enhance the system's efficiency through them. This is achieved by ensuring that the most crucial data are collected, the sequence of their collection is the most efficient one and the available options are clearly presented to the users, so as to avoid errors and confusion.

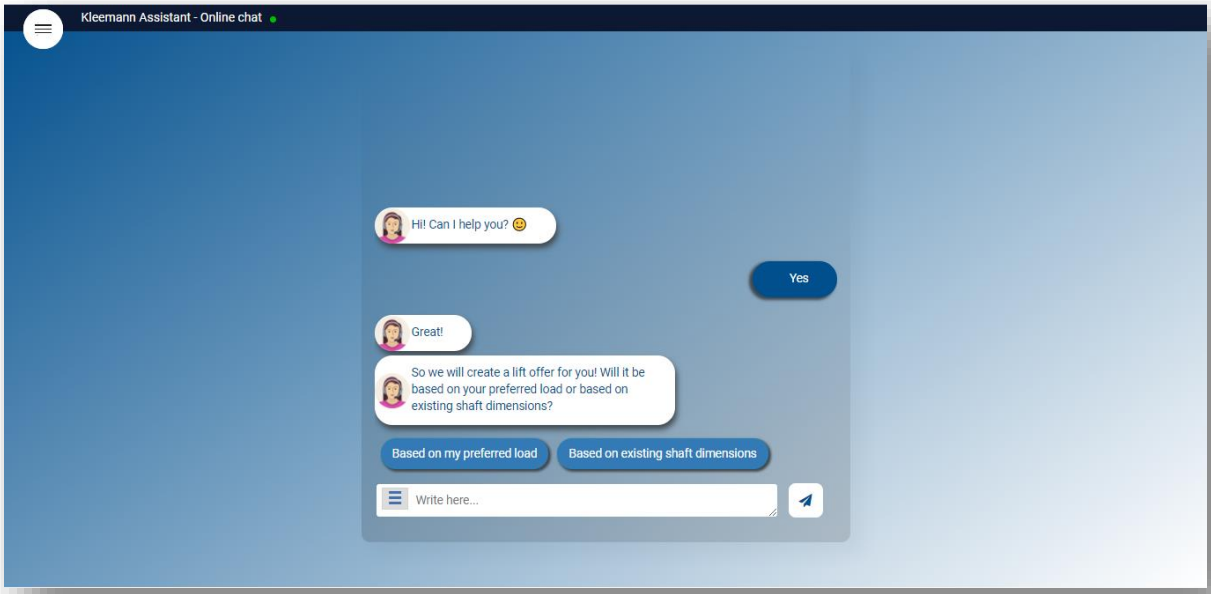


Figure 17. First input for the order

Based on the user's response, the web assistant continues the conversation, by prompting the user to insert the next required piece of information. The decision of how the conversation should continue is based on JavaScript arrays, which indicate, through tree-structure, the next questions that are appropriate for the user's answers.

If the user selects (or types) the option "Based on existing dimensions", the web assistant will next require the insertion of the shaft width of the building in millimetres. While the user types the answer they are specifically asked for, there is instant validation of their response. As depicted in the Figure 18, although the web assistant prompts the user to insert a value between 1600 and 3000, the user types the number 1400 which is below the allowed

minimum range and it is instantly marked as invalid. The number is in red color and neither the enter button nor the send button on the right can submit this data.

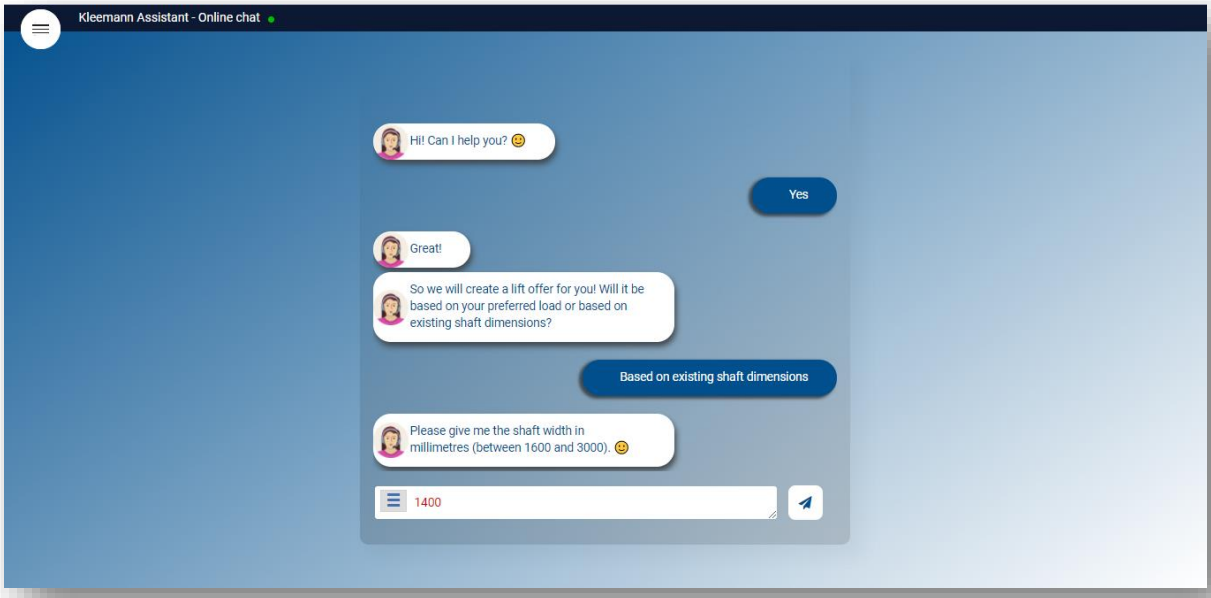


Figure 18. Instant validation of an input number

If the user selects (or types) the option "Based on my preferred load", the web assistant instead of asking for the shaft width of the building, prompts the user to insert the load of their preference (Figure 19).

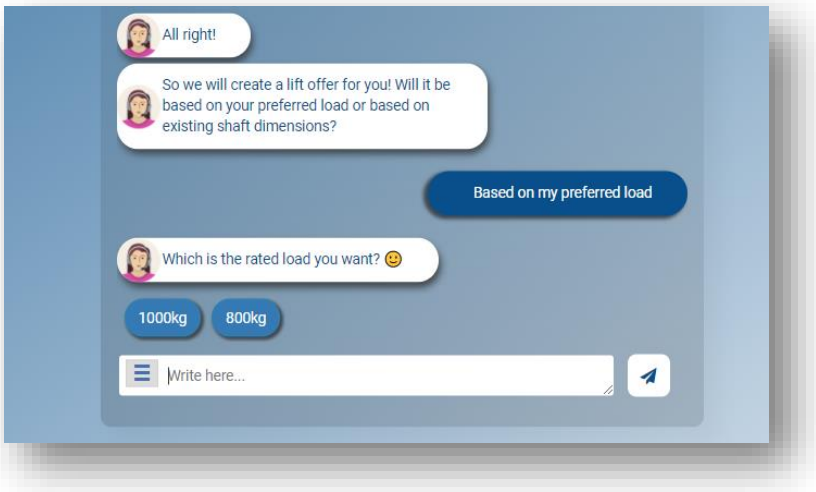


Figure 19. Question in case of selecting "Based on my preferred load"

The two aforementioned cases are distinct and every order can be classified only in one of these two categories. For our system description, we will continue our dialog flow as if the

user selects the option “Based on existing dimensions”. Hence, after the insertion of the shaft width’s value, the web assistant subsequently asks for the value of the shaft depth (Figure 20). Similarly, the user inputs this information, which is instantly validated, similarly to all provided values.

Moreover, the web assistant prompts the user to select if the car of the lift includes one or two entrances (which is referred to as “single” and “through” car type respectively in the lift industry’s terminology) and also, requires the input of the travel length and the stops of the lift. The above information is presented in Figure 20.

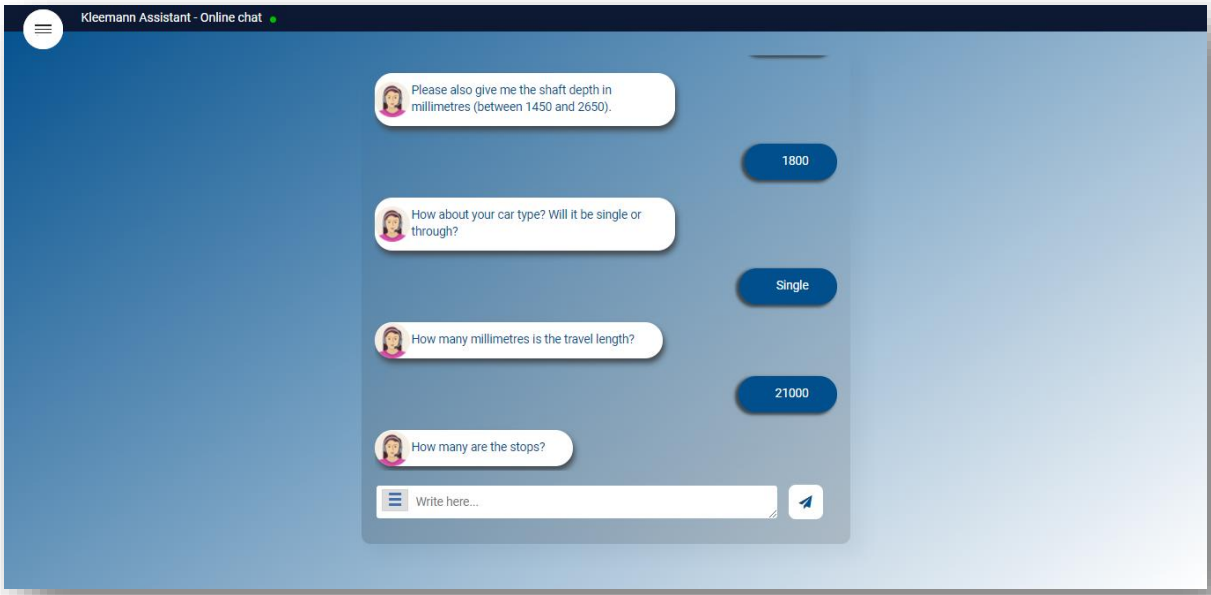


Figure 20. Data collection through application’s interface

Based on all the aforementioned data, the web assistant is thereupon able to provide the available cabin sizes for the user’s shaft dimensions. Although the cabin sizes are provided very quickly by the assistant, their calculation is a complex procedure which is based on algorithmic calculations. The algorithms take into account the constraints of the existing shaft and based on their outputs, the user is provided with the maximum cabin size, as well as the rest of the available cabin sizes, as depicted in Figure 21. The maximum cabin size is the one that makes the best use of the given shaft and it is selected beforehand by the assistant, as it corresponds to the vast majority of the cases for each user’s lift orders.

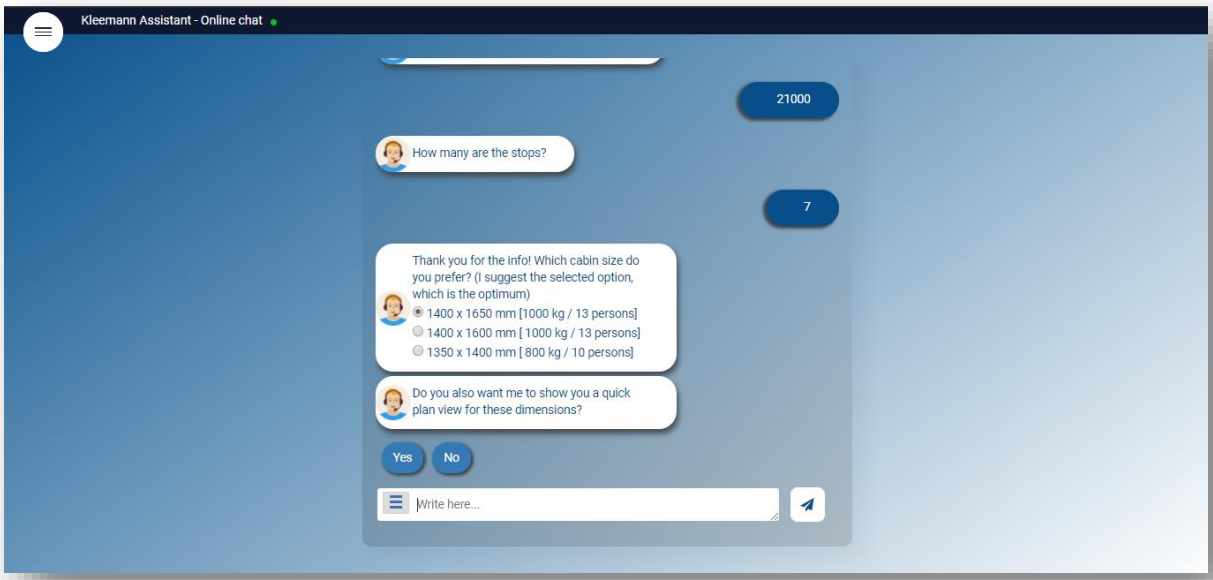


Figure 21. Cabin size options based on the constraints of a shaft

Furthermore, the system asks the user whether they want the web assistant to generate a plan view of the shaft, which will visualize the included dimensions. If the user wants to see the plan view, the web assistant runs JavaScript functions that combine the lift engineering know-how with creating a dynamic web element that is composed of a pure HTML5 and CSS3. The element that constitutes the plan view of the shaft is presented in Figure 22.

The plan view of Figure 22, is a realist depiction of the actual shaft of the building. Each lift part inside the shaft is accurately calculated and defined in millimetres by the algorithms of the application. Thereafter, these millimetres of every lift part are assigned to the pixels of the screen, and multiplied with a small scale of around ten per cent, in order to fit inside the conversation box.

Afterwards, the web assistant asks the user whether they want to proceed to some cabin design options, such as selecting the floor and walls of their preference.

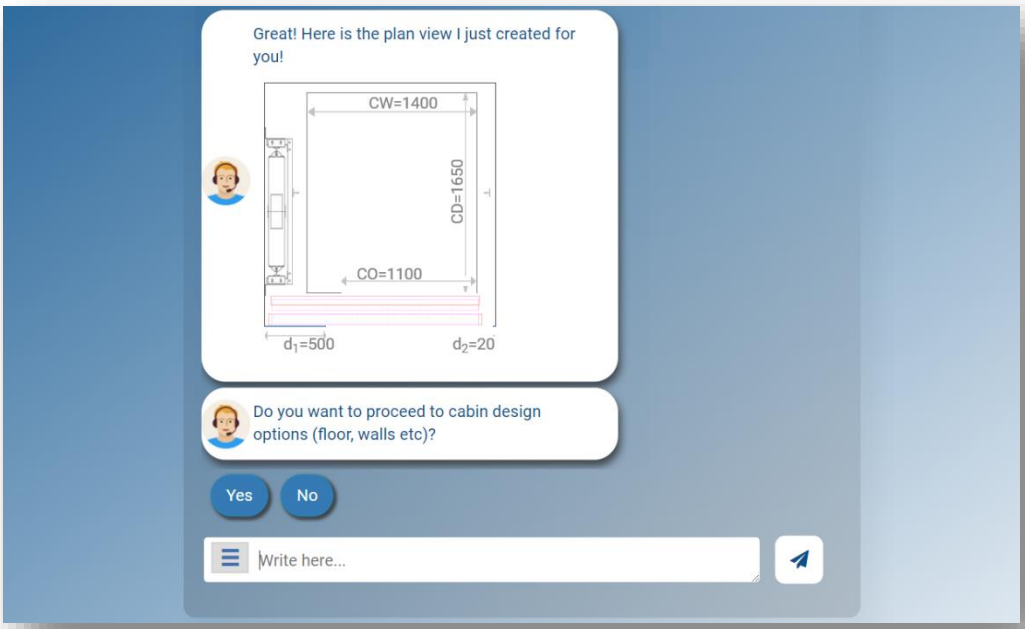


Figure 22. Realistic plan view of the shaft

If the user does not want to skip this step, but instead continue in customizing the design of their lift, they see the lift visualization and floor options that are presented in Figure 23. The web assistant, based on JavaScript algorithms, as well as using modern HTML5 and CSS3 commands, creates instantly and from scratch a realistic depiction of the lift cabin. The presented cabin is initially composed of the default and most popular options. Starting with the floor option, the web assistant asks the user to select a floor option among the given ones. By clicking on it and then, in the “Ok” button, their selection is taken into account by the system.

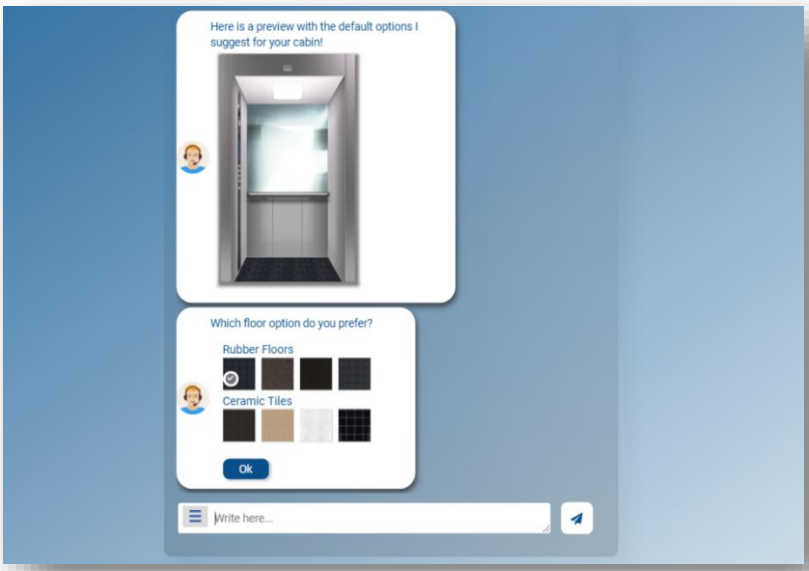


Figure 23. Lift design preview and floor options

Moreover, as depicted in the Figure 24, the web assistant prompts the user to select the ceiling of their preference. Similarly, the system presents the user with the possible options, in order to facilitate the process, as well as avoid errors and misunderstandings about the expected options. Next, the user is able to select the lift walls they prefer among the available options that the company provides. Again, the user can click on the options they want and select them for their lift.

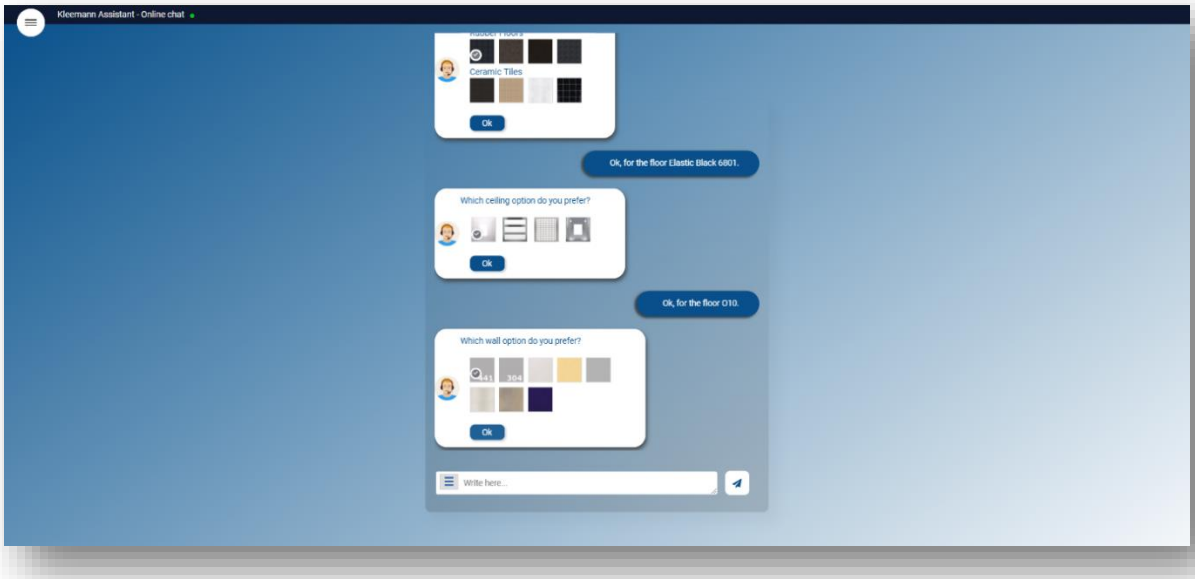


Figure 24. Ceiling and wall options

Thereafter, based on the selected ceiling, floor and walls, the final cabin preview is created again by the web assistant, aiming to sufficiently inform the user for the result of their selections (Figure 25). The cabin preview, similarly to previous previews, is created purely with front-end technologies code. In general, all of our outputs and visualizations are instantly and dynamically created for the user, in order to provide a more realistic experience for the customers and enhance the Human Computer Interaction (HCI).

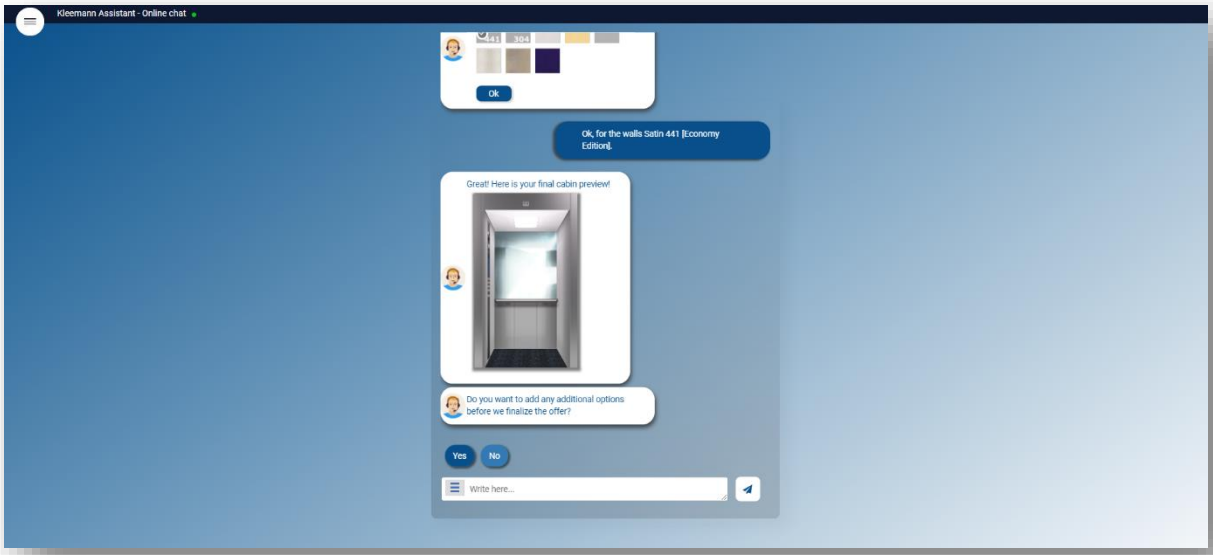


Figure 25. Final cabin preview

Moreover, the web assistant asks the user whether they want to input their additional preferences and even constraints about the lift order. In this open-ended question, the user can type in free-form conversational text to provide their additional preferences and then, the web assistant has to correctly interpret the given information and include it in the order's specifications, just like a salesman would do in a chat dialog with a client. For instance, if the user types "I want a lift with speed of 1.6 m/s, code standard en81.20, headroom 3830 and controller Lisa, please", the web assistant is able to interpret the content of the request and include the foregoing specifications in the order (Figure 26).

In the context of free-form text interpretation, the system conducts several complex operations, including natural language processing, finding Part-of-Speech taggers for each

inserted word, detection of the keywords that are related with the lift order and detection and handling of possible spelling mistakes from users.

After inputting the free-form text specifications for the order, the system informs the user about the new options that are included in the order, aiming to keep them aware of everything that is related with their order.

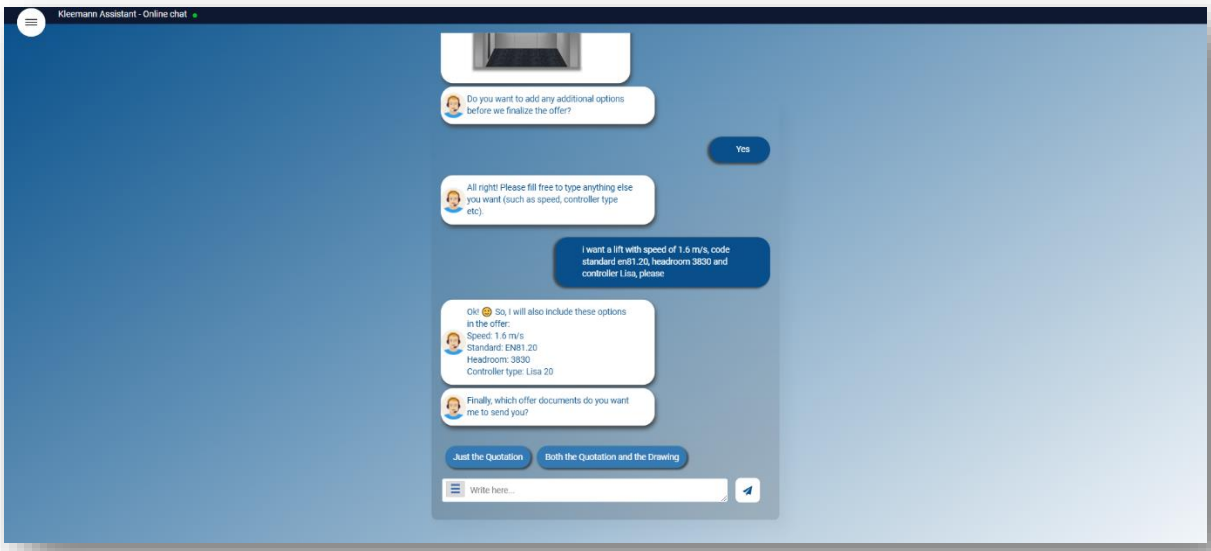


Figure 26. Free-form text interpretation and output selection

Finally, the web assistant requires the user to indicate whether they want to receive only the order quotation (invoice) or both the order quotation and the drawing of the lift. Based on this last answer, the assistant submits all the aforementioned options and constraints that the user has provided. The submission includes communication with the back-end system of the company, which is conducted through encrypted web services.

After sending the data to the back-end system, the web assistant receives almost instantly the order number that has been opened and informs the user that they should wait for a little while, in order to finalize the order in the company's system (Figure 27). In this waiting time, our web application completes all the interactions with the back-end system of the company and performs the respective AJAX calls that are taking place between the two systems.

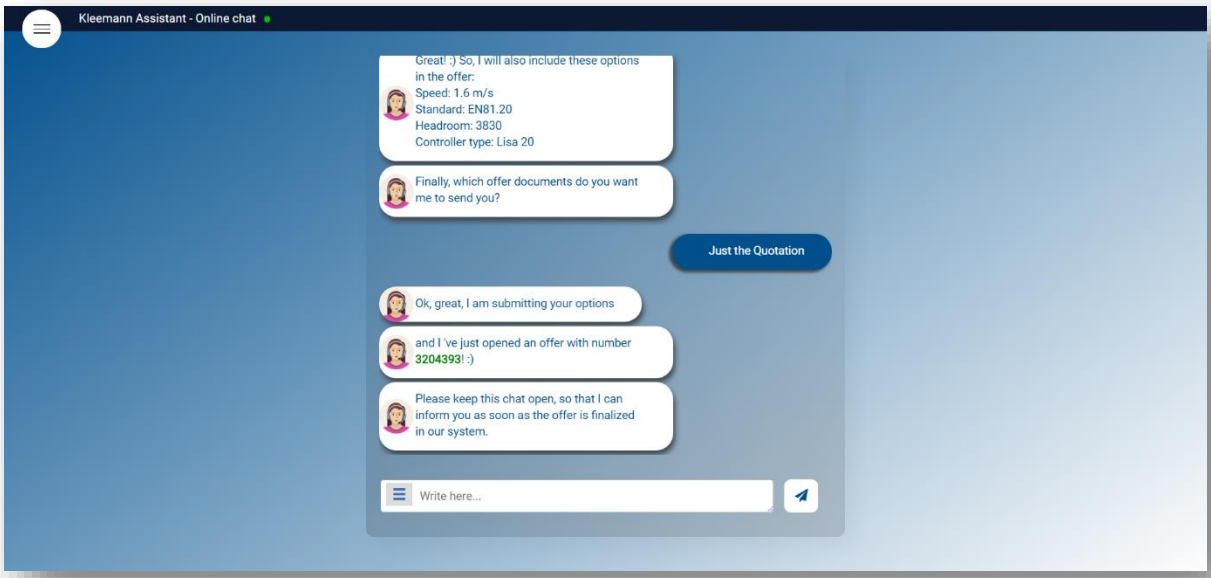


Figure 27. Order creation

As soon as the procedure is completed, the web assistant informs the user that the order is finalized and soon after, an e-mail is sent to the user.

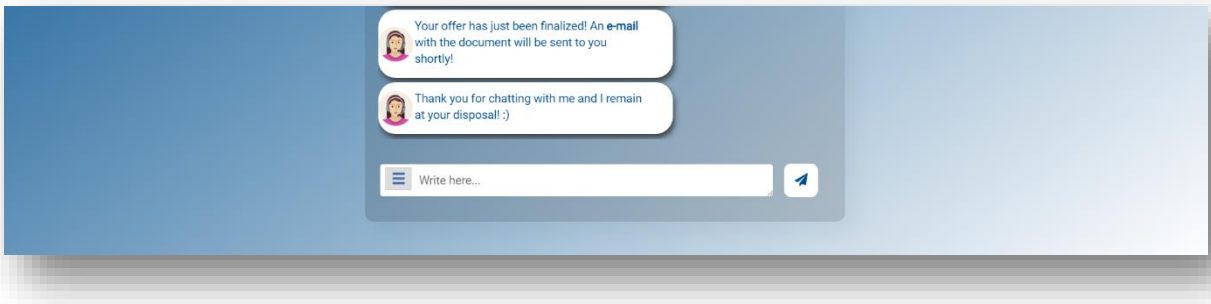


Figure 28. Order finalization and e-mail sending

The e-mail that is sent to them includes the PDF with the lift quotation, as well as the drawing, if it is also requested by the user. In Figure 29, we can see the received e-mail, with the attached PDF, which is the quotation of the order that the web assistant has created in the company's ordering system.

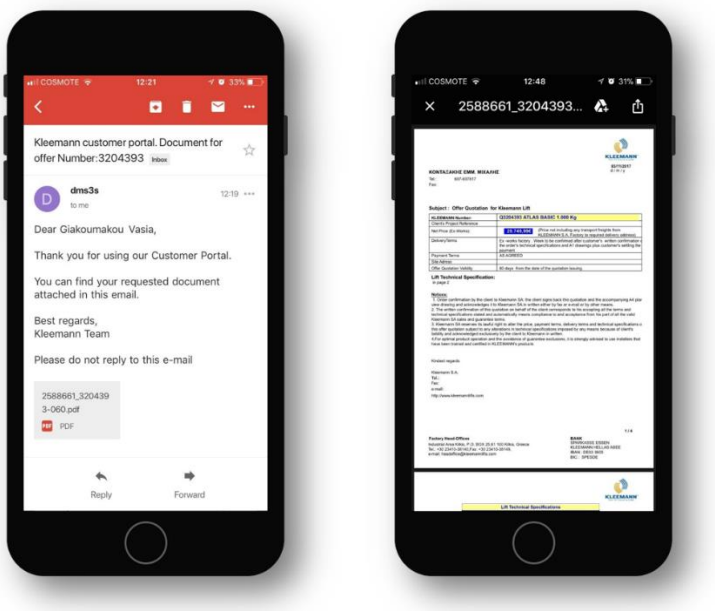


Figure 29. The received e-mail with the invoice file.

4.2. Responsiveness in the web application's User Interface

The User Interface of the web application is responsive, aiming to provide an optimal viewing of the software across multiple devices and screen sizes. With the use of modern front-end technologies, including Bootstrap and CSS3 media queries, the web assistant achieves to enhance the user experience.

Figure 30. The application on mobile phone and tablet.
(Sources: https://mockuphone.com/static/images/devices/apple-iphone7-silver-portrait.png, https://images.contentful.com/q1nb8mvzrssh/VMgz3iRnEsY2QImg2yQ0q/ee7ae55ea7bc40a924a8e2e5758f6274/ipad_base.png)

4.3. Implementation and use of open-source toolkits

The web application of this dissertation implements an intelligent web assistant which is developed from scratch, including modern web languages and technologies, such as JavaScript, HTML5, CSS3, Bootstrap and PHP.

For the development of the project, the following open-source toolkits (licensed under the MIT license) were used:

- **nlp-compromise**: Natural language processing toolkit in JavaScript, developed by Spencer Kelly and other contributors.
- **conv-form**: jQuery plugin that relates an HTML form to a chatbot interface. The plugin is developed by Eduardo Thomas Koller.
- **levenshtein.js**: jQuery plugin that calculates the Levenshtein Distance between two given strings. The plugin is originally created by Andrei Mackenzie.

The aforementioned web plugins include useful functionalities that are included in the web application, in order to enhance the capabilities of the automated web assistant and enrich the web application's capabilities with some already developed features, freely available online as open-source plugins.

4.3.1. NLP-Compromise toolkit

4.3.1.1. Description and capabilities

Nlp-compromise is a lightweight Natural Language processing engine for the client-side of web development. It is thoroughly built in JavaScript and compiled down to a small file, which can run in all systems and devices [46]. The toolkit handles and understands English text in an efficient and quite simple way, enabling the development of interfaces that are based on the English language [47].

On the basis of handling natural language, any inserted text in the nlp-compromise toolkit can be quickly [48]:

- parsed into sentences,
- tokenized into terms,
- part-of-speech tagged

Parse text into sentences

In the initial step, the inserted text is parsed into separate sentences. These sentences will subsequently be treated as distinct parts of the initial text, in order to be efficiently queried and tagged by our software.

Tokenization into terms

As descripted in Chapter 2, in NLP, tokenization is the process of breaking down a sentence into tokens that represent words or other types of its parts [12]. The JavaScript toolkit can efficiently handle non-sentence uses of periods, exclamation marks and question marks [49].

Part-Of-Speech (POS) tagging

Part-Of-Speech tagging is conducted in the following sequence (from first to last):

1. **Lexicon**: we establish a really big list of words and if a word happens to be in that list, we know the part of speech [50].
2. **Suffix regexes**: suffix rules, for instance if a word ends in "ly", it is most likely to be an adverb [50]. Of course, there are exceptions, but the vast majority of these kind of words will be adverbs (“quickly”, “immediately”, just to name a few). So, if you have really well-crafted regular expressions, you can actually identify words by their suffix as well.

```
4098
4099  var tag_mapping = _dereq_('../../parts_of_speech.js').tag_mapping;
4100  //regex patterns and parts of speech],
4101  module.exports = [
4102  ['^[0-9]+ ?(am|pm)$', 'DA'],
4103  ['^[0-9]+(st|nd|rd)?$', 'CD'],
4104  ['^[a-z]et$', 'VB'],
4105  ['cede$', 'VB'],
4106  ['.[cts]hy$', 'JJ'],
4107  ['.[st]ty$', 'JJ'],
4108  ['.[lnr]ize$', 'VB'],
4109  ['.[gk]y$', 'JJ'],
4110  ['.fies$', 'VB'],
4111  ['.some$', 'JJ'],
4112  ['.[nrtumcd]al$', 'JJ'],
4113  ['.que$', 'JJ'],
4114  ['.[tnl]ary$', 'JJ'],
4115  ['.[di]est$', 'JJS'],
4116  ['^(un|de|re)\\-[a-z]..', 'VB'],
4117  ['.lar$', 'JJ'],
4118  ['[bszmp]{2}y', 'JJ'],
4119  ['.zes$', 'VB'],
4120  ['.[icldtgrv]ent$', 'JJ'],
4121  ['.[rln]ates$', 'VBZ'],
4122  ['.[oe]ry$', 'NN'],
4123  ['[rdntkdhs]ly$', 'RB'],
```

Figure 31. Suffix rules in Part-Of-speech tagging.

Through this implementation, in less than a millisecond, we can define the part-of-speech tag of a word with great accuracy [50].

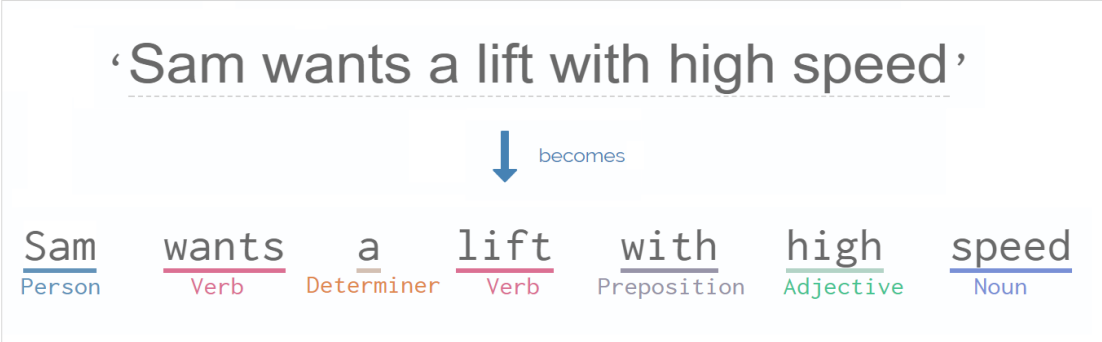


Figure 32. Part-Of-speech tagging in nlp-compromise (Source: http://compromise.cool/)

3. **Sentence-level Markov Chain**: In this step of our tagging, we conduct the sentence-level processing, by applying grammar rules. For instance in the sentence of Figure 32, we would initially interpret "walk the walk" part of the sentence as "verb – determiner – verb", but according to the English grammar rules, there is a misinterpretation in this assumption [50]. Thus, our software will conduct a second assumption and correctly assign the tags "verb – determiner – noun" [50].

Figure 33. Sentence-level Markov Chain. (Source: [50]*)*

Performance

Using the toolkit, even a whole document can be fully parsed into a few seconds. Furthermore, on a sentence-level, which is the most common level of string- length that is expected to be inserted in our application, the sentence can be parsed and queried in just 15 milliseconds [51]. Consequently, the speed performance of our nlp-compromise toolkit is very satisfying and it effectively meets the aims and aspects of our web application.

4.3.1.2. Reasons to use the NLP-compromise

There are several reasons that make nlp-compromise a great fit for the needs of our web application. The major benefits of its use are presented as follows:

- It is a very small and lightweight JavaScript library, which is only around 200 kb.
- It is very fast, capable of assigning Part-Of-Speech Tags to words in less than one millisecond.
- It is easily imported to any front-end web application, in order to constitute a part of its operations.
- It offers high precision and accuracy in the results it provides, after handling natural language inputs.
- With the use of the toolkit, there is no need for server-side code.
- There is a sufficient community of developers that work on the toolkit and there are continuous upgrades and new features that are added to it.
- Likewise JavaScript, it is a cross-platform tool that can run in every system and device.

4.3.2. conv-form jQuery plugin

The conv-form plugin aims to help transform a web form into an interactive chat dialog between two interlocutors. The plugin uses the languages JavaScript (jQuery), HTML5 and CSS3, in order to provide its useful features. The most significant of them are presented as follows:

- Transformation of HTML form elements, such as inputs and select elements, to interactive questions that appear in a chat interface.
- Support of regular expression patterns and messages that does not expect answer.
- Access on previous answers, which can be helpful in order to fork conversation based on them.

To build the chat dialog, we wrap the web form inside an element, and afterwards, call the function .convform() on it's jquery selector. The function requires a placeholder for the user input.

```
494 $(function(){
495     //instantiate conversation form on .conv-form-wrapper (default class for plugin);
496     var convForm = $('.conv-form-wrapper').convform("Write here...");
497
498
```

Figure 34. Build of the chat dialog through JavaScript

Also, based on the plugin, we dynamically build our own questions for each form field using conv-* attributes and we extend the capabilities by generating additional functionalities.

In order to use the conv-form jQuery plugin in our web application, we import its JavaScript and CSS files into our application's files.

```
26
27     <!-- convForm -->
28     <link rel="stylesheet" type="text/css" href="dist/jquery.convform.css">
29     <script type="text/javascript" src="dist/jquery.convform.js"></script>
30
```

Figure 35. Import of the conv-form plugin in the assistant's page

4.4. Detection of spelling mistakes

In order to detect spelling mistakes, the assistant accomplishes the following procedures:

- Levenshtein distance calculation
- Anagramism detection

- Combination of the foregoing methods (both the Levenshtein distance calculation and the anagramism detection), in order to decisively handle every typographic error.

4.4.1. Levenshtein Distance

Levenshtein distance (LD) is a measure of the similarity between two given strings [52]. It is also known as edit distance and it is named after the Russian scientist Vladimir Levenshtein, who originally formulated the concept in 1965 [52]. The calculated Levenshtein distance indicates the number of deletions, insertions, or substitutions needed to transform the first string into the second one [52] or, in other words, the cost of changing one string into another in the easiest way [53].

For example, if the two given strings are s_1="shaft" and s_2="shaft", then the LD(s_1, s_2) = 0, since no transformations are required to make the strings identical [52]. However, if the two given strings are s_1="shaft" and s_2="sheft", then the LD(s_1, s_2) = 1, since one substitution is required in order to transform s_1 into s_2. Thus, as it can be easily assumed, the greater the Levenshtein distance, the more dissimilar the two strings are [52].

Until now, there is a wide usage of the algorithm in several scientific topics, such as spell checking, speech recognition, DNA analysis and plagiarism detection [52].

4.4.2. Anagramism detection

The web application also applies anagramism detection. For example, if the user inserts the word "complaince" instead of "compliance" (which is a keyword), the assistant detects this typographic error, proceeds to the appropriate corrections and triggers the events that are related to the keyword.

In Figure 36, we can see the implementation of the function *checkAnagramism()*, in which we detect in a very compact and quick way the existence of an anagramism. In figure 37, we can also see an example of the output of the function when it runs in the web browser's console.

```
function checkAnagramism(currentKeyword, currentNormal) {
    return currentKeyword.split("").sort().join("") === currentNormal.split("").sort().join("");
}
```

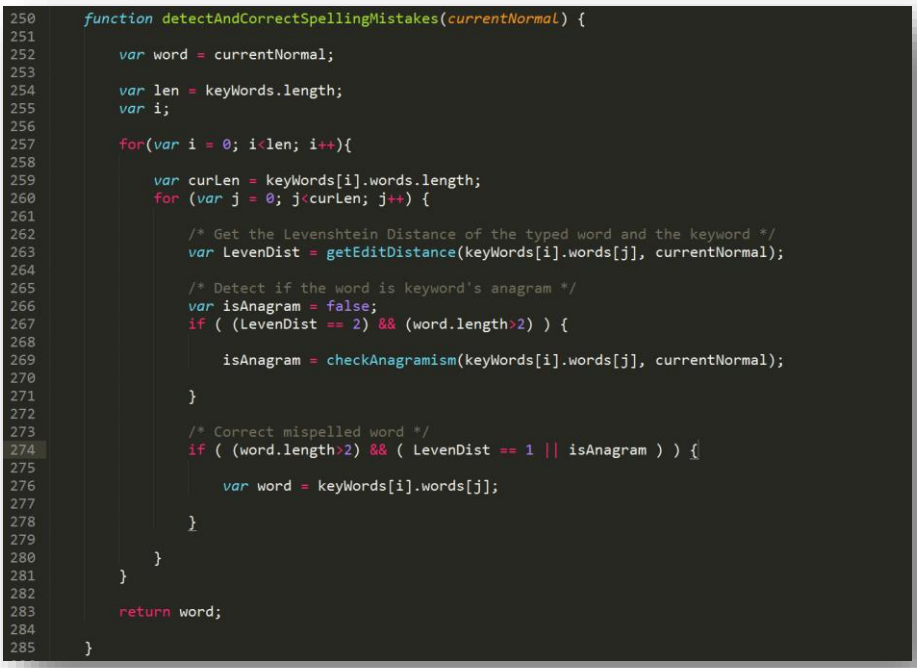
Figure 36. Implementation of the function checkAnagramism()

```
> checkAnagramism('compliance', 'complaince');
< true
>
```

Figure 37. Output of the function in browser's console

4.4.3. Combination of Leveshtein Distance and anagramism detection

In our web assistant, we apply a combination of the Levenshtein distance calculation and the anagramism detection, so as to handle and interpret efficiently every sentence that our users type. For instance, the word "shaft" may be wrongly written as "shft" or shfat". Both of these words will be correctly interpreted by the assistant as spelling mistakes of the word "shaft". In the first case the Levenshtein distance is 1 and in the latter, there is anagramism and the Levenshtein distance is 2 (which is the maximum allowed Levenshtein distance, only accepted for anagrams of keywords). The implementation of the core function of our web application *detectAndCorrectSpellingMistakes()* is presented in Figure 38.



```
function detectAndCorrectSpellingMistakes(currentNormal) {

    var word = currentNormal;

    var len = keyWords.length;
    var i;

    for(var i = 0; i<len; i++){

        var curLen = keyWords[i].words.length;
        for (var j = 0; j<curLen; j++) {

            /* Get the Levenshtein Distance of the typed word and the keyword */
            var LevenDist = getEditDistance(keyWords[i].words[j], currentNormal);

            /* Detect if the word is keyword's anagram */
            var isAnagram = false;
            if ( (LevenDist == 2) && (word.length>2) ) {

                isAnagram = checkAnagramism(keyWords[i].words[j], currentNormal);

            }

            /* Correct mispelled word */
            if ( (word.length>2) && ( LevenDist == 1 || isAnagram ) ) {

                var word = keyWords[i].words[j];

            }

        }
    }

    return word;

}
```

Figure 38. Implementation of the function detectAndCorrectSpellingMistakes()

4.5. Development practices of the web application

The process of creating clean, maintainable and cross-browser code is a significant aspect of the software development cycle. For the development of our web assistant, we aimed to create quality code, which would be easily readable and maintainable. For the achievement of this goal, we followed several good practices, which are presented as follows:

- Use of minified HTML, CSS and JavaScript code: One of our main aims throughout the development process was to reduce the size of files, by removing all necessary code lines, comments and white spaces, in order to improve the application's performance.

- Use of external JavaScript and CSS files: The practice of using external files enhances the maintainability and readability of the web application's code. Moreover, it makes our web pages load faster, because the external files can be cached by the browser of the user. Inline styles are considered as a bad practice in web development, thus we ensure that whenever possible, external files are used in our software instead.

- In case of using internal CSS and JavaScript code snippets (which are placed in our HTML/PHP files in limited cases), we ensure that they are placed in the most proper position inside our file. More precisely, CSS code is placed at the head of the file, whereas JavaScript code is placed at the bottom of our file. By adopting this web development practice, our web pages appear to load faster and we ensure that our JavaScript code will be loaded after the content of our web page has been displayed.

- Use of optimized images: we ensure that our images are in a correct size and format. Thus, for our web assistant, we ensure that the size of the images corresponds to their usage, instead of being oversized. Also, their file format is either .jpg or .png, in order to keep a reduced file size.

- Minimization of HTTP requests: According to Yahoo [54], 80% of the loading time of a Web page is spent in downloading the different files that compose the page, such as images and scripts. Thus, we ensure that we do not separate JavaScript or CSS in too many small files, but we group the code into bigger and logically separated files instead. Furthermore, whenever possible, we apply CSS styling in our web pages rather than using images for the same purpose.

4.6. Security practices of the web application

Modern web development includes plenty of challenges, and of those security is both crucial and frequently under-emphasized by web developers and engineers. For our project we apply a number of security measures and functionalities that can efficiently strengthen the application's security. The measures we apply are the following ones:

- Authorized login to the system is required, in order to gain access the web application's interface.

- There is use of encrypted AJAX calls, in order to communicate with the back-end system and request data from it or send data for the order creation. Thus, the communication with the server-side systems of the company is limited and any kind of non-encrypted connection with them is unpermitted.

- User's session times out after a certain time limit, forcing users to logout of the system if they do not want to continue using the assistant after a logical amount of time.
- If a user wants to terminate their session, they can easily logout from the system through the menu "Logout" option.
- Although our web assistant refers to a client-side application, every code line that needs to be hidden from the end-users is written in the language PHP, so that it is not visible in the browser's console. PHP is mainly used to handle sensitive information, including user's session information and data related to the connection with the company's back-end system.

4.7. Other tools used for the development process

4.7.1. Version Control System

For the development of the web application there was a need for the use of a Version Control System, in order to record changes in the software's files, so that we are able to recall specific versions if needed [55]. For our web assistant, we use the Git hosting website called "Github", which is based on Git. Through the use of Github, we manage to efficiently keep track of changes that are conducted in the software, and in general, to handle and maintain the web application in a secure way. Thus, apart from facilitating the process of source code management, the use of Github also provides safety to the software from unexpected incidents, such as file losses and other unwanted actions.

Figure 39. GitHub logo. (Source: https://github.com/logos)

Our dissertation project is accessed and updated using the Git command-line interface on the following online address of GitHub: https://github.com/vgiakoumakou/userAssistant

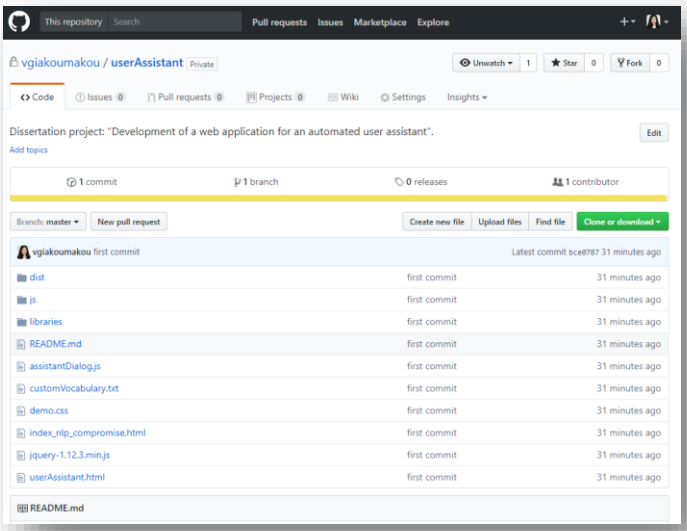


Figure 40. GitHub repository of the application

4.7.2. Task tracking and management

The “Trello” task management tool was used in order to organize the tasks of the dissertation project. Through its tools, we set due dates, organize our tasks in categories, set priorities and keep track of all the tasks that are related to the project.

Chapter 5: Evaluation and results

This chapter presents, analyses and interprets findings gathered from the application survey performed in the study, in the form of an online questionnaire in Google Forms.

The online questionnaire was composed of eleven questions, starting with a few significant background information of the respondents and continuing with their evaluation and rating of the application.

Before filling in the questionnaire, respondents were briefly informed about the objective of the web application and they were given as much time as they wanted to interact with the system on their own. Each respondent voluntarily used the application to thoroughly create an order through its interface, so that they could efficiently compare it with the existing ordering system of the company. After using the web application, the respondents filled in the questionnaire anonymously and voluntarily, as well.

5.1. Question types

In the questionnaire of our study, the context of the questions was carefully formulated, in order to get useful and efficient results, as well as facilitate the submission process for the respondents. All of our eleven questions were closed-ended, meaning that respondents were not asked to supply their own answers, but there were given pre-defined answers instead.

By following this design in our questionnaire, we ensure that the process is not time consuming for the respondents and thus, we achieve to keep them interested in filling in the form. Furthermore, by providing closed-ended questions, we obtain reliable results about leading questions and we enhance our efficiency in analysing results [56].

The provided choices in the questions are including either rating-scales or distinct options, covering all the possible cases that the respondents may want to choose. In the provided rating-scales we ensure that users can identify even the “worst” and “best” opinion, setting their honest responses as a priority.

5.2. Background and experience of the respondents

The respondents of the questionnaire are individuals who work as professionals in the lift industry and more specifically, they constitute employees of the company. Moreover, they are engineers who are familiar with the company's ordering system as well as with the use of web applications. The selection of the survey respondents addressed individuals that are representative of the total potential users of the application.

As we can observe in the following pie chart, almost three out of four of the respondents are male (in percentage accuracy, men correspond to the 73.1%, whereas women to the 26.9%). Based on the general underrepresentation of women in engineering [57] and since the majority of the engineers in the company are male, these results are quite expected.

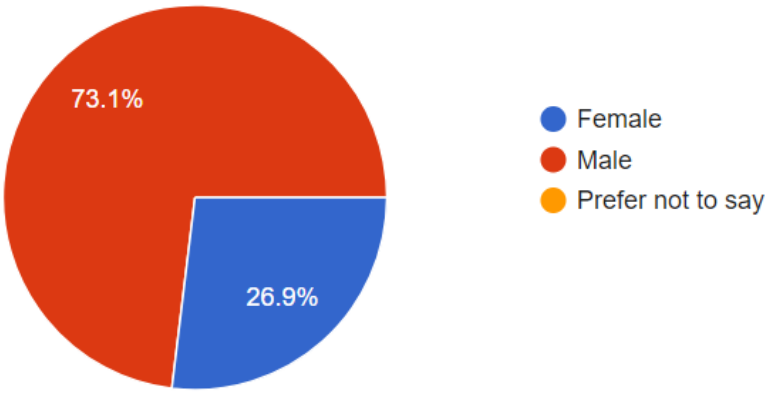


Figure 41. Responses by gender.

Moving on the highest educational qualification of the respondents, the majority of those who were surveyed indicated that they hold a Master's degree in their field of expertise. More precisely, 76.9% of the respondents belong to this educational background, whereas the rest 23.1% holds a Barchelor's degree. None of them was either a Ph.D holder or an individual who completed their high school education without any further studies.

Thus, based on this question, it is concluded that the respondents of the survey are well-educated people.

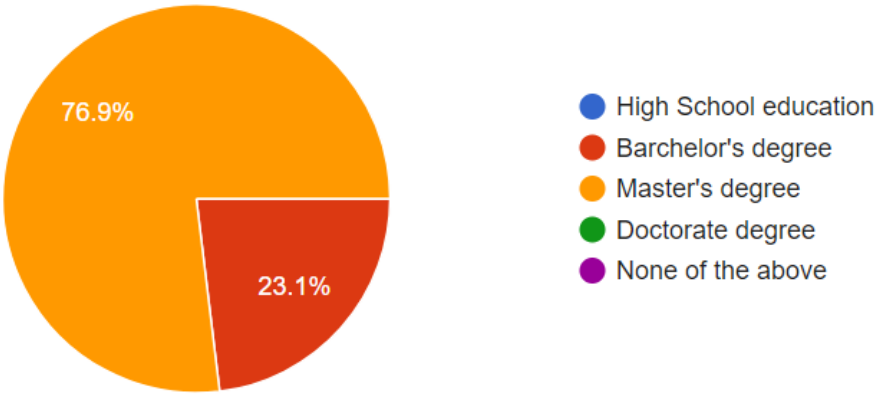


Figure 42. Responses by educational background.

Regarding the question "How long have you held a position in the lift industry?", it is reported that most of the respondents have been working in the company for 2-3 years. The next most common answers correspond to people who have been working in the company for 1-2 years (26.9%) and to those who hold 4-5 years of experience and can be characterized as very experienced engineers in the ordering process. Cumulatively, these three categories constitute 84,6% of the total people who participated in the survey. Also, 7.7% of the respondents are highly skilled, with more than five years of working experience in the company. The remaining 7.7% relies on recently employed engineers, with less than one year of experience in the company and the ordering process.

Thus, it is noticable that the majority of the respondents has a considerable experience of working in the lift industry and around 65% of them has been creating lift orders for more than 2 years. This fact about the survey's respondents confirms their critical ability on evaluating ordering systems.

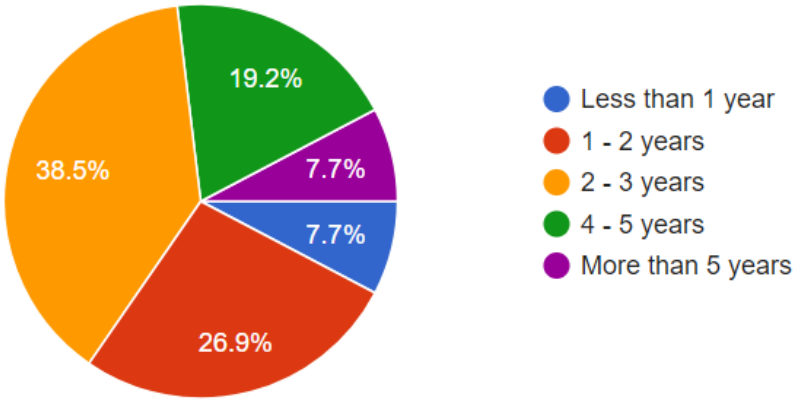


Figure 43. Responses by years of experience in the lift industry.

5.3. Findings for the use of the web application

After the findings of the questionnaire which were related to the background of the respondents, our evaluation moves on questions related to the use of the web application.

First, in the question “How easy was it for you to use the web assistant?” there is a very positive feedback by the respondents for the application. Of the 26 employees who were surveyed, 24 claimed that it was very easy for them to use the web assistant, giving the highest possible answer (5) in the scale between 1 and 5. The other two respondents found the application very easy to use, providing a rating of 4 to ease of use. Hence, it is clearly indicated by the respondents that the system provides great User Experience to the users, offering an impressively easy interface for them.

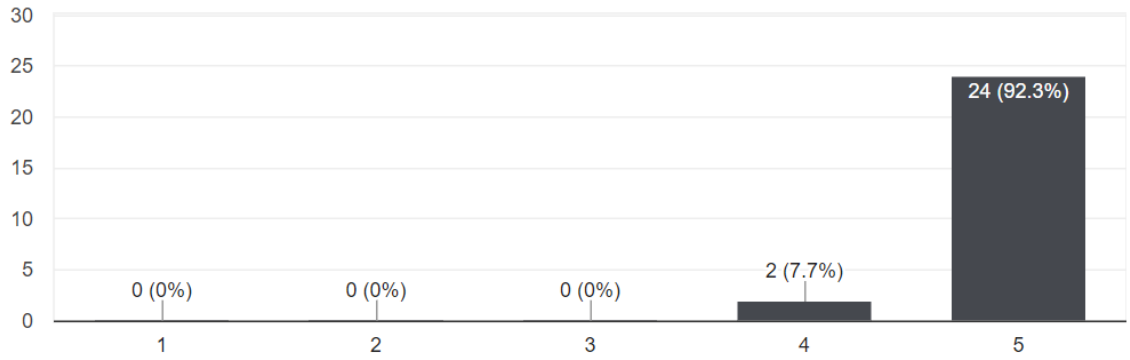


Figure 44. Results about the ease of use of the application.

Furthermore, on the question "Approximately how much time did you spend to complete the offer process through the web assistant?", the vast majority of the respondents (96.2%) claimed that they needed less than 5 minutes to complete the whole ordering process through the web assistant. For the remaining 4.8% of the people who were surveyed, it also took them a small amount of time, which was less than 10 minutes. None of the respondents needed more than 10 minutes to complete the process through the conversational application. A significant finding of this question is that the process completion through the web assistant is very quick for all the respondents and the system efficiently fastens the conduction of such a complex process.

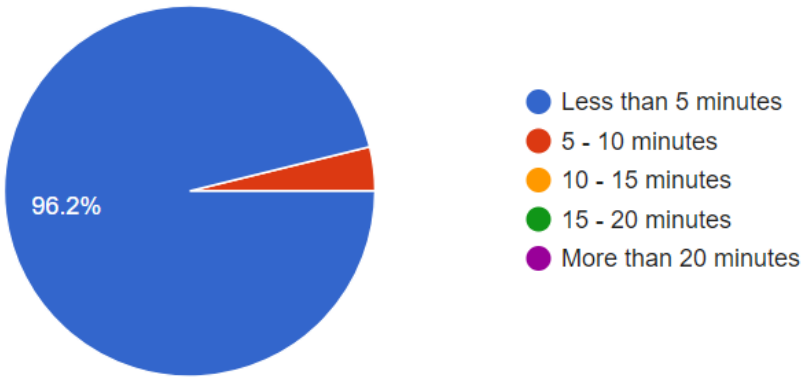


Figure 45. Results about the time needed to complete the process through the web assistant.

Comparing the time spent for the web assistant with the time that would be respectively spent for the same procedure through the existing ordering system of the company, the respondents expressed that the time that will be saved is very important. More precisely, the majority of them (46.2%) estimates that they will save more than 3 hours for each order submission. Similarly, the 34.6% of respondents stated that they would save 2-3 hours for each order. Regarding the remaining 19%, it is expressed that the time saved corresponds to less than two hours, with only one person claiming that they would save at most half an hour. Thus, it is clear that 80.8% of the respondents indicate that they would save more than 2 hours for each order submitted through the web assistant instead of the existing ordering system.

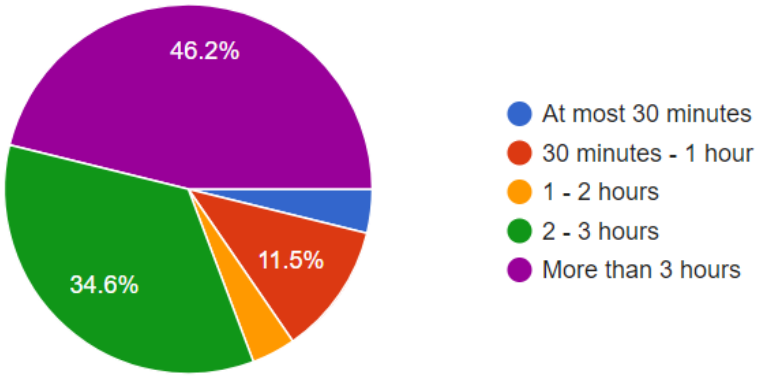


Figure 46. Results about the time saved when creating an offer through the web assistant.

Moreover, apart from being faster and time-saving, the process through the web assistant was also quite easier for the users.

As was claimed by the 88.5% of those who were surveyed, using the web assistant for completing the whole process was a lot easier for them than using the existing ordering system. Similarly, the other 11.5% of the respondents also found it easier to create the order through the web assistant, enhancing the usability and the potential usefulness of the proposed system. Surprisingly, none of them voted that there is either the same level of

easiness in the two systems or the existing system is easier for them to use in order to conduct process.

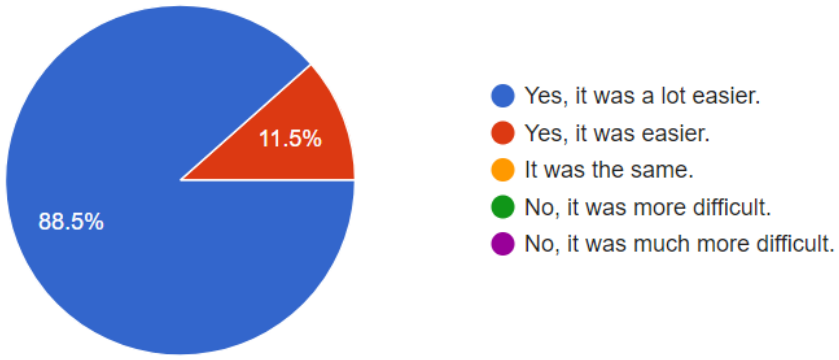


Figure 47. Results about the ease of use of creating an order through the web assistant in comparison with the company's ordering system.

Moving to the next questions of the survey, their context is related to the respondents' ratings for the web application. The findings of these questions are analyzed as follows:

As depicted in the next bar diagram (Figure 48), all of the respondents provided a very high rating for the web assistant's responses. More precisely, in a rank between 1 and 5, referring to "very bad" and "very good" responses respectively, the impressive 92.3% of those who were surveyed rated with a "5" the responses, which corresponds to the highest ranking in the scale, whereas the remaining 7.7% chose to rate the responses with 4. The fact that none of the respondents rated the web assistant's replies with bad or even moderate ratings, significantly boosts the system's efficiency in providing useful and effective results in the conversation with its users.

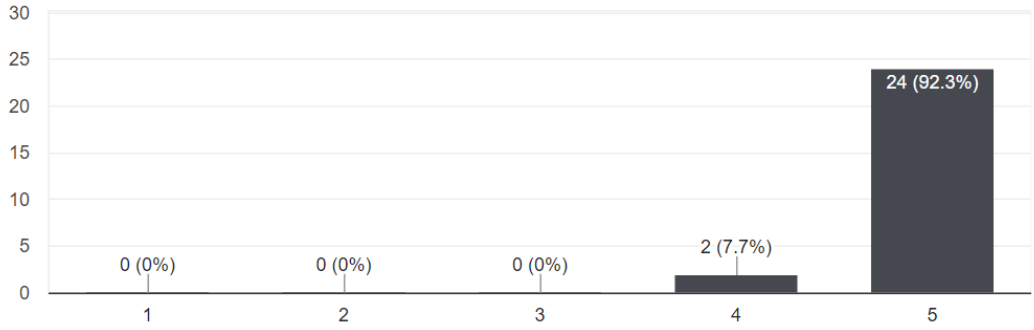


Figure 48. Results about the rating of the web assistant's responses.

Moreover, respondents are asked to provide an overall rating for the web assistant, taking into account all possible aspects that may affect their overall opinion about the system. The findings of this question are likewise favorable, enhancing the positive feedback that was already provided. In the next bar diagram (Figure 48), the outcome of this question is visualized where 24 out of 26 respondents provided the highest rank in the web application (they gave a "5" in a scale between 1 and 5). Similarly, the remaining 2 respondents of those who were surveyed rated the assistant with 4 out of 5. Once again, the respondents of the survey did not indicate any bad experience about the web application and through their high ratings they expressed their satisfaction from using the system.

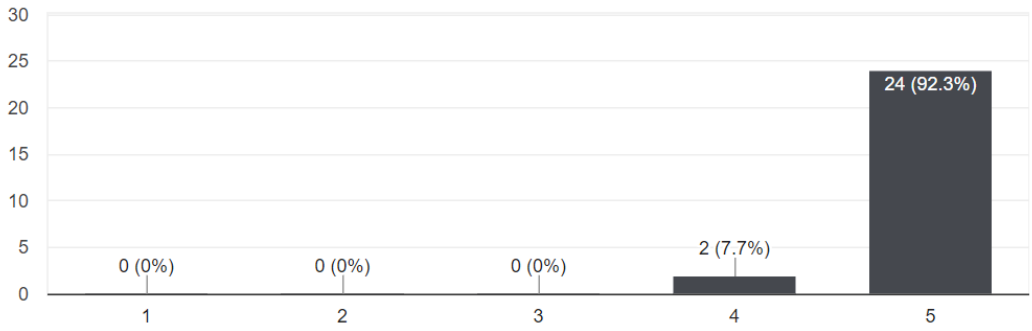


Figure 49. Results about the overall rating of the web assistant.

Notwithstanding the foregoing great findings about the system's ease of use and the respondent's ratings, it is also very significant to outline the employees' willingness to recommend the system to others as well as use it as individuals.

As shown in the following diagram (Figure 50), in the question “How likely would it be for you to recommend the web assistant to the company's clients and employees?” for the vast majority it would be very likely. Of the 26 respondents who were surveyed, 22 claimed that it is very likely that they would recommend the system to their colleagues and the company’s clients and for the rest 15.4% of the respondents it was likely as well. Thus, there is a strong and common trend by the respondents to support the web application and extend its reputation by recommending it to potential users.

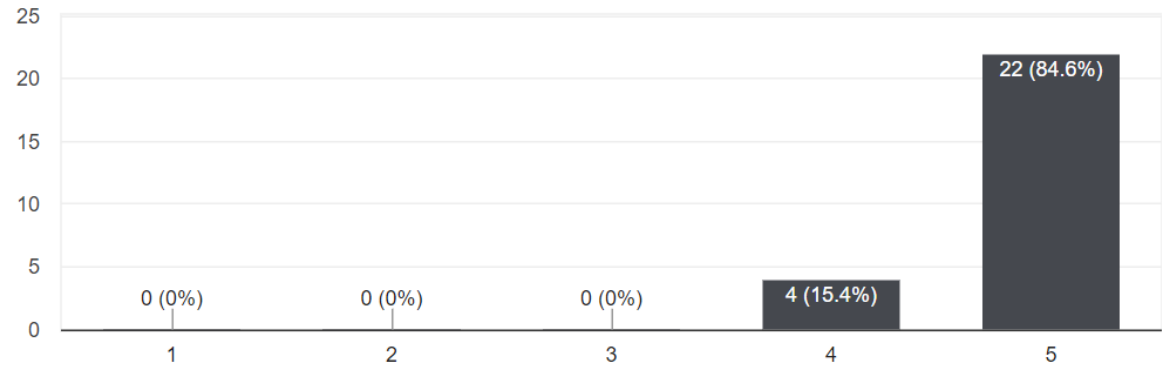


Figure 49. Results about the likeliness of recommending the web assistant to the company’s clients and employees.

Lastly, in the question “How likely would it be for you to use the web assistant again?” the positive viewpoint of the respondents appears to remain unchanged. Examining the data, we can see that for all the respondents it was either likely or very likely to use the system again, with the vast majority of them belonging to the latter case (84.6%). This question of the survey constitutes a significant indicator of the usefulness of the system for the respondents’ workload and it points out, as well, that their first impression was positive enough to get them quite triggered to use it again for the ordering process.

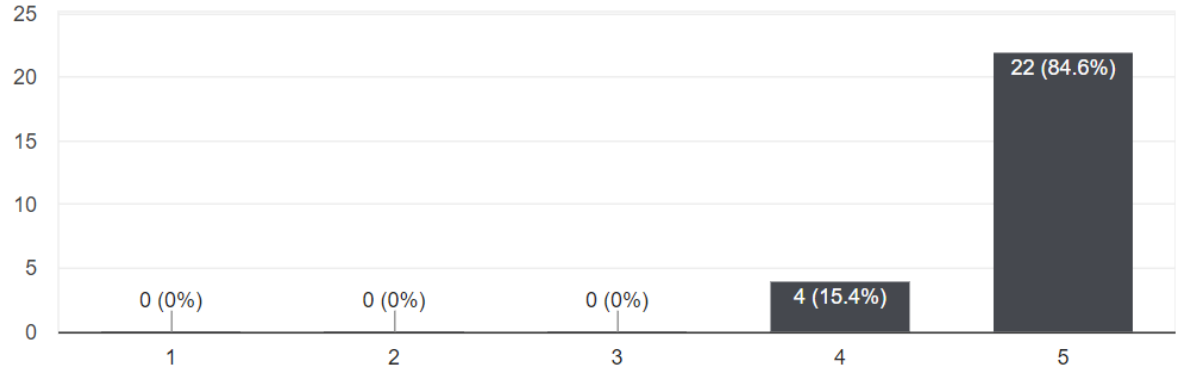


Figure 50. Results about the likeliness of using the web assistant again.

Summarizing our findings and on the basis of the survey results, it can be concluded that the overall opinion of the respondents about the web application is very good. Our survey findings confirmed our hypothesis that a conversational web application that automates the ordering process can efficiently save time and facilitate the most major process of the employees' daily workload. Also, regarding the interface of the system, it would be fair to conclude that the users found it easy to use, so as to create an order and there was an overall positive tendency by them to use the application again.

All in all, the conclusion that can be drawn from the aforementioned findings, indicates that the web application seems to be very time-saving, useful and easy to use for the respondents and it also has the potential to efficiently facilitate the complex ordering process.

Chapter 6: Conclusions and future work

This chapter summarizes our thesis's contributions and conclusions and outlines the directions for future work and research. The chapter is divided in two sections; the first one depicts the contribution of the current work and the second one, indicates the potential future research that can be conducted as an extension of the current work.

6.1. Conclusions and contributions

In this thesis, we addressed the problem of creating an automated web assistant for a lift company having a conversational interface and a client-side architecture. Our work presents the complex lift order creation task as an automated and simplified procedure by proposing a web application that is based on front-end technologies that boost its speed and independency.

We have provided an extensive discussion on the existing web assistance solutions, as well as a review of the most significant conversational assistants that have been developed until now. Based on the research that has been conducted in the field, the contribution of the proposed automated system is twofold.

Firstly, we have automated the lift order procedure through a conversational interface, achieving to minimize the time needed for the whole process. Moreover, we have simplified the procedure by creating a friendly interface that includes all the subtasks of an order, such as the plan view creation and the lift design configuration. In this way, we manage to save valuable time in the company's employees' daily workload, to facilitate a very time-consuming process and to provide work automation, avoiding possible human errors. These advancements constitute key contributions for the lift company's order procedure and impel the business to alternate its sales channel in order to strengthen its brand image.

Secondly, we have proposed a different implementation technique for conversational web assistants, in order to interact with the users more quickly. This is achieved by designing and developing the web application thoroughly with front-end technologies, such as JavaScript,

HTML5 and CSS3. In order to interact with the customers quickly and present complex algorithmic outputs to them without any server-side delays, we handle all the essential system logic and data storage in the client-side. While commonly-used web assistants heavily depend on continuous interactions with servers leading to lower speed when communicating with the users, our system overcomes the foregoing obstacles by using client-side languages. Thus, we have developed a fast, independent and flexible software as a contrariwise suggestion of how conversational web assistants could be developed.

6.2. Future work

Notwithstanding the foregoing contributions, opportunities for extending the thesis's work remain. In this section, we present the possible future work on the automated web assistant, which can extend the existing scope of the application.

Firstly, since speech recognition constitutes an emerging trend in modern applications and virtual assistants, it would be useful for the Human Computer Interaction (HCI) of the application to also recognize user's speech. As technology is improving, toolkits that are related to speech recognition are becoming more efficient and robust. Hence, a potential addition of speech recognition, would further boost the user experience and fasten the time needed to accomplish the whole order process.

Moreover, a further line of work on the web assistant's operations is to extend the tasks that the system accomplishes. Following the logic of the assistant's existing tasks, the system can serve further support on sales-related procedures. For example, the web assistant can present to a customer their order history, the current status of specific orders that they have conducted or suggestions based on their previous options. Since we have assumed that web assistants should preferably do specific tasks (based on our research on Chapter 2), we propose to extend the tasks of the system with ones that are also related to the sales channel.

Finally, based on the emerging use of applications in mobile phones, we suggest the development of the automated assistant as a mobile application (both for Android and iOS devices), including the same tasks and functionalities as the web application. The

development of the Android and iOS mobile applications, would increase the number of the company's revenue channels and increase the customers' engagement.

Chapter 7: References and bibliography

[1] Y. Wilks, Machine Conversations (The Springer International Series in Engineering and Computer Science), 1999.

[2] F. Frichou, "6 types of online customer service you can implement today — Trustpilot.com," 2017. [Online]. Available: http://blog.trustpilot.com/blog/6-types-of-online-customer-service-you-can-implement-today. [Accessed 4 11 2017].

[3] E. Seo, "19 Best UX Practices for Building Chatbots - How do we make chatbots usable?," 2017. [Online]. Available: https://chatbotsmagazine.com/19-best-practices-for-building-chatbots-3c46274501b2. [Accessed 2 10 2017].

[4] A. Steenkamp, "6 tips for the best online customer support," 2014. [Online]. Available: 6 tips for the best online customer support. [Accessed 8 11 2017].

[5] J. Murphy and I. Tan, "Journey to nowhere? E-mail customer service by travel," 2002.

[6] S. Casey, P. O'Neill, M. Camuso and T. Thurston, "How Self-Service Research Changes B2B Marketing -," 2016.

[7] "What are the benefits of using a Chatbot?," Maruti Techlabs, 2017. [Online]. Available: https://www.marutitech.com/benefits-chatbot/.

[8] B. A. Shawar and E. Atwell, "Chatbots: Are they Really Useful?," 2017.

[9] D. Withey, "Chatbot Trends," 2017. [Online]. Available: https://blog.ubisend.com/optimise-chatbots/chatbot-trends. [Accessed 2017 8 15].

[10] A. Campbell, "One Year Of Chatbots: What We Learned And What's Next," 2017.

[11] B. A. Shawar and E. Atwell, "Using dialogue corpora to train a chatbot.," 2003.

[12] J. Cahn, "CHATBOT: Architecture, Design, & Development," 2017.

[13] J. Constine and S. Perez, "Facebook Messenger now allows payments in its 30,000 chat bots.," 2016. [Online]. Available: https://techcrunch.com/2016/09/12/messenger-bot-payments/. [Accessed 2 11 2017].

[14] M. d. G. B. Marietto, R. V. d. Aguiar, G. d. O. Barbosa, W. T. Botelho, E. Pimentel, R. d. S. França and V. L. d. Silva, "ARTIFICIAL INTELLIGENCE MARKUP LANGUAGE: A BRIEF TUTORIAL," 2013.

[15] A. M. Turing, "Computing Machinery and Intelligence," 1950.

[16] "Erik Devaney," 2016. [Online]. Available: https://chatbotsmagazine.com/infographic-rise-of-the-chatbots-1e2c086aece4. [Accessed 24 8 2017].

[17] L. Bradeško and D. Mladenić, "A Survey of Chabot Systems through a Loebner Prize Competition," 2012.

[18] "A.L.I.C.E. Artificial Linguistic Internet Computer Entity," A.L.I.C.E. AI Foundation, Inc., [Online]. Available: http://www.alicebot.org/about.html. [Accessed 7 10 2017].

[19] J. Rahman, "Implementation of ALICE chatbot as domain specific knowledge bot for BRAC U (FAQ bot)," 2012.

[20] "How do Chatbots work? A Guide to the Chatbot Architecture - Maruti Techlabs," Maruti Techlabs Pvt. Ltd., [Online]. Available: https://www.marutitech.com/chatbots-work-guide-chatbot-architecture/. [Accessed 14 9 2017].

[21] S. A. Abdul-Kader and D. J. Woods, "Survey on Chatbot Design Techniques in Speech Conversation Systems," 2015.

[22] "What is a phoneme? - Phonic Books," Synthetic Phonics, 2011. [Online]. Available: https://www.phonicbooks.co.uk/2011/01/17/what-is-a-phoneme/. [Accessed 12 12 2017].

[23] D. Griol and M. F. McTear, The conversational interface: talking to smart, 2016.

[24] "Answer These 5 Basic Questions Before You Build A Chatbot - TOPBOTS," 2017. [Online]. Available: https://www.topbots.com/answer-5-basic-questions-build-chatbot/. [Accessed 17 10 2017].

[25] C. Mielke, "Conversational Interfaces: Where Are We Today? Where Are We Heading?," Smashing Magazine, 2016. [Online]. Available: https://www.smashingmagazine.com/2016/07/conversational-interfaces-where-are-we-today-where-are-we-heading/.

[26] M. Yao, "Chatbot UX – Does Conversation Hurt Or Help?," Smashing Magazine, 2016. [Online]. Available: https://www.smashingmagazine.com/2016/11/does-conversation-hurt-or-help-the-chatbot-ux/). [Accessed 2 10 2017].

[27] S. Krumhausen, "The Bot Playbook," Chatbots Magazine, 2016. [Online]. Available: https://chatbotsmagazine.com/the-bot-playbook-7bb6d181a6a9. [Accessed 28 11 2017].

[28] K. Scott, "Usability Heuristics for Bots," Chatbots Magazine, [Online]. Available: https://chatbotsmagazine.com/usability-heuristics-for-bots-7075132d2c92.

[29] "Complete guide on Chatbots - development to promotion," Maruti Techlabs, 2017. [Online]. Available: https://www.marutitech.com/complete-guide-chatbots/. [Accessed 12 11 2017].

[30] Flanagan and David, JavaScript: The Definitive Guide, 6th Edition, O'Reilly Media, 2011.

[31] C. Yue and H. Wang, "Characterizing Insecure JavaScript Practices on the Web," 2009.

[32] D. Yu, A. Chander, N. Islam and I. Serikov, "JavaScript Instrumentation for Browser Security," 2007.

[33] S. Guarnieri, M. Pistoia, O. Tripp, J. Dolby, S. Teilhet and R. Berg, "Saving the World Wide Web from Vulnerable JavaScript," 2011.

[34] T. Anglin, "5 Benefits of Developing JavaScript Sites & Apps," Progress Software Corporation, 2011. [Online]. Available: https://www.telerik.com/blogs/5-benefits-of-developing-javascript-sites-amp-apps.

[35] D. S. McFarland, JavaScript & jQuery - the missing manual (3rd Edition), O'Reilly Media Inc., 2014.

[36] "The Pros and Cons of JavaScript: Is it Still Necessary?," International Academy of Design & Technology, 2013. [Online]. Available: http://www.iadt.edu/student-life/iadt-buzz/december-2013/the-pros-and-cons-of-javascript-is-it-still-necessary. [Accessed 28 7 2017].

[37] "Web Technology Surveys - Usage statistics and market share of jQuery for websites," 2017. [Online]. Available: https://w3techs.com/technologies/details/js-jquery/all/all. [Accessed 30 10 2017].

[38] "jQuery," [Online]. Available: https://jquery.com/. [Accessed 10 30 2017].

[39] D. S. McFarland, JavaScript & JQuery: The Missing Manual (3rd Edition), O'Reilly Media, 2014.

[40] "W3C - HTML Design Principles," 2007. [Online]. Available: https://www.w3.org/TR/html-design-principles/. [Accessed 31 10 2017].

[41] E. Castro and B. Hyslop, HTML5 and CSS3, Seventh Edition: Visual QuickStart Quide, Peachpit Press, 2012.

[42] "HTML & CSS - W3C," 2016. [Online]. Available: https://www.w3.org/standards/webdesign/htmlcss. [Accessed 31 10 2017].

[43] "A brief overview of the history of Bootstrap.," 2017. [Online]. Available: http://getbootstrap.com/docs/4.0/about/history [Accessed 2/11/17]. [Accessed 2017 11 2].

[44] "Bootstrap - The most popular HTML, CSS, and JS library in the world.," [Online]. Available: http://getbootstrap.com/. [Accessed 1 11 2017].

[45] J. Spurlock, Bootstrap: Responsive Web Development, O'Reilly Media, 2013.

[46] S. Kelly, "Quickstart · nlp-compromise/compromise Wiki · GitHub," 2017. [Online]. Available: https://github.com/nlp-compromise/compromise/wiki/QuickStart. [Accessed 9 8 2017].

[47] S. Kelly, "Usage · nlp-compromise/compromise Wiki · GitHub," 2017. [Online]. Available: https://github.com/nlp-compromise/compromise/wiki/Usage. [Accessed 9 8 2017].

[48] S. Kelly, "Compromise Docs," [Online]. Available: http://compromise.cool/docs. [Accessed 20 7 2017].

[49] S. Kelly, "Tokenization · nlp-compromise/compromise Wiki · GitHub," [Online]. Available: https://github.com/nlp-compromise/compromise/wiki/Tokenization. [Accessed 21 11 2017].

[50] S. Kelley, "Language as an Interface," Presentation at "goto;" conference in Chicago (May 2016), [Online]. Available: https://gotocon.com/dl/goto-chicago-2016/slides/SpencerKelley_LanguageAsAnInterface.pdf.

[51] S. Kelly, "Speed · nlp-compromise/compromise Wiki · GitHub," 2017. [Online]. Available: https://github.com/nlp-compromise/compromise/wiki/Speed. [Accessed 21 7 2017].

[52] M. Gilleland and M. Park, "Levenshtein distance in three flavors," 2009. [Online]. Available: https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm.

[53] W. J. Heeringa, "Measuring Dialect Pronunciation Dierences using Levenshtein Distance," *Thesis/Research at the University of Groningen, Netherlands,* 2004.

[54] "Best Practices for Speeding Up Your Web Site - Yahoo Developer Network," [Online]. Available: https://developer.yahoo.com/performance/rules.html#num_http. [Accessed 12 10 2017].

[55] S. Chacon and B. Straub, Pro Git (Second Edition), Apress, 2014.

[56] "Types of survey questions," Canada Business Network, [Online]. Available: https://canadabusiness.ca/business-planning/market-research-and-statistics/conducting-market-research/types-of-survey-questions/. [Accessed 25 11 2017].

[57] C. Price, "It's time to bridge the engineering gender gap," The telegraph, 2017. [Online]. Available: http://www.telegraph.co.uk/technology/time-to-bridge-engineering-gender-gap/. [Accessed 19 11 2017].

[58] "Messaging apps are now bigger than social networks," Business Insider, 2016. [Online]. Available: http://www.businessinsider.com/the-messaging-app-report-2015-11. [Accessed 14 8 2017].

[59] A. R. C, C. Jacob and A. Mohanan, "A SURVEY ON WEB BASED CONVERSATIONAL BOT DESIGN," 2016.

[60] J. Jiyou, "The Study of the Application of a Keywords-based Chatbot System on the Teaching of Foreign Languages.," 2003.

Chapter 8: Appendices

Appendix A – Questionnaire of the dissertation

Dissertation Questionnaire

Dissertation Subject: "Development of a web application for an automated user assistant"

MSc in "Mobile and Web Computing"

You are being asked to participate as a volunteer in a research study conducted by Vasiliki Giakoumakou, a Master of Science student at the International Hellenic University. This questionnaire is designed to gather information about the student's thesis with title "Development of a web application for an automated user assistant".

The research is being conducted under the supervision of Dr. Marios Gkatzianas.

Your participation in this study is anonymous and your confidentiality as a participant in the questionnaire will remain secure.

* Required

What is your gender? *

- ○ Female
- ○ Male
- ○ Prefer not to say

What is your highest educational qualification? *

- ○ High School education
- ○ Barchelor's degree
- ○ Master's degree

- ○ Doctorate degree
- ○ None of the above

How long have you held a position related to the lift industry? *

- ○ Less than 1 year
- ○ 1 - 2 years
- ○ 2 - 3 years
- ○ 4 - 5 years
- ○ More than 5 years

How easy was it for you to use the web assistant? *

	1	2	3	4	5	
Very difficult	○	○	○	○	○	Very easy

Approximately how much time did you spend to complete the offer process through the web assistant? *

- ○ Less than 5 minutes
- ○ 5 - 10 minutes
- ○ 10 - 15 minutes
- ○ 15 - 20 minutes
- ○ More than 20 minutes

Approximately how much time will be saved by creating an offer through the web assistant in comparison with creating it through the company's ordering system? *

- ○ At most 30 minutes
- ○ 30 minutes - 1 hour

- ○ 1 - 2 hours
- ○ 2 - 3 hours
- ○ More than 3 hours

Was it easier to create an offer through the web assistant in comparison with creating it through the company's ordering system? *

- ○ Yes, it was a lot easier.
- ○ Yes, it was easier.
- ○ It was the same.
- ○ No, it was more difficult.
- ○ No, it was much more difficult.

How would you rate the web assistant's responses? *

	1	2	3	4	5	
Very bad	○	○	○	○	○	Very good

Please provide an overall rating of the web assistant. *

	1	2	3	4	5	
Very bad	○	○	○	○	○	Very good

How likely would it be for you to recommend the web assistant to the company's clients and employees? *

	1	2	3	4	5	
Very unlikey	○	○	○	○	○	Very likely

How likely would it be for **you** to use the web assistant again? *

	1	2	3	4	5	
Very unlikey	○	○	○	○	○	Very likely

Appendix B

In this part of the appendix we include information and code samples that have to be quoted after the main thesis text.

Icons for the web assistant

For the icon of the assistant that is shown in every message that the automated user assistant sends, there is a random choice between a female and a male customer support representative. Through a few lines of PHP code, we implement a random choice of the current assistant icon, as shown in the next code snippet.

```
<?php
 $bg = array('supportfemale.png', 'supportmale.png' ); // array of images

 $i = rand(0, 1); // generate random number size of the array
 $selectedBg = "$bg[$i]"; // set variable equal to which random image was chosen
?>
```

Based on the previous random image selection, the aforementioned PHP variables are used to apply the respective CSS code and show randomly the assistant's image.

```
<style>
div#chat.conv-form-wrapper  div#messages div.message.to {
        background-image: url(images/<?php echo $selectedBg; ?>);
}
</style>
```

The icons that are used for the web application were created by Anna Litviniuk and they are licensed under the "Free for commercial use" license. Figure 51 presents the two icons.

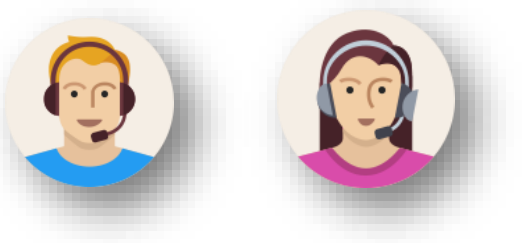


Figure 51. Icons of the web assistant created by Anna Litviniuk. (Source: https://www.iconfinder.com/iconsets/user-pictures)