FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Anssi Meisalmi

A CUSTOMISABLE CHATBOT AS A RESEARCH INSTRUMENT

Master's Thesis
Degree Programme in Computer Science and Engineering
June 2020

Meisalmi A. (2020) A customisable chatbot as a research instrument. University of Oulu, Degree Programme in Computer Science and Engineering, 63 p.

ABSTRACT

Chatbots are proliferating rapidly online for a variety of different purposes. This thesis presents a customisable chatbot that was designed and developed as a research instrument for online customer interaction research. The developed chatbot facilitates creation of different bot personas, data management tools, and a fully functional online chat user interface. Customer-facing bots in the system are rule-based, with basic input processing and text response selection based on best match. The system uses its own database to store user-chatbot dialogue history. Further, bots can be assigned unique dialogue scripts and their profiles can be customised concerning name, description and profile image.

In the presented validation studies, participants completed a task by taking part in a conversation with different bots, as hosted by the system and invoked through distinct URL parameters. Second, the participants filled in a questionnaire on their experience with the bot, designed to reveal differences in how the bots were perceived.

Our results suggest that the chatbot's personality impacted how customers experienced the interactions. Therefore, the developed system can facilitate research scenarios that deal with investigating participant responses to different chatbot personas. Future work is necessary for a wider range of applications and enhanced response control.

Keywords: chatbot tool, customer research, user test, chatbot persona.

Meisalmi A. (2020) Personoitava chatbot tutkimustyökaluna. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 63 s.

TIIVISTELMÄ

Chatbotit yleistyvät nopeasti Internetissä ja niitä käytetään enenevissä määrin useissa eri käyttötarkoituksissa. Tämä diplomityö esittelee personoitavan chatbotin, joka on kehitetty tutkimustyökaluksi verkon yli tapahtuvaan vuorovaikutustutkimukseen. Kehitetty chatbot sisältää erilaisten bottipersoonien luonnin, apuvälineitä datan käsittelyn, ja itse botin käyttöliittymän. Järjestelmän käyttäjille vastailevat bottipersoonat ovat sääntöihin perustuvia, niiden syötteet käsitellään suoraviivaisesti ja vastaukseksi valitaan vertailun mukaan paras ennaltamääritellyn skriptin mukaisesti. Järjestelmä käyttää omaa tietokantaa tallentamaan käyttäjä-botti keskusteluhistorian. Lisäksi boteille voidaan asettaa uniikki dialogimalli, ja niiden profiilista voidaan personoida URL-parametrillä nimi, botin kuvaus ja profiilikuva.

Chatbotin tekninen toiminta todettiin tutkimuksella, jossa osallistujat suorittivat annetun tehtävän seuraamalla osittain valmista käsikirjoitusta eri bottien kanssa. Tämän jälkeen osallistujat täyttivät käyttäjäkyselyn liittyen heidän kokemukseensa botin kanssa. Kysely oli suunniteltu paljastamaan mahdolliset eroavaisuudet siinä, kuinka botin käyttäytyminen miellettiin keskustelun aikana.

Käyttäjätestin tulokset viittaavat siihen, että chatbotin persoonalla oli vaikutus käyttäjien kokemukseen. Kehitetty järjestelmä siis pystyy mahdollistamaan tutkimusasetelmia, joissa tutkitaan osallistujien reaktioita erilaisten chattibottien persooniin. Jatkotyö kehitetyn chatbotin yhteydessä keskittyy monimutkaisempien käyttötarkoitusten lisäämiseen ja botin vastausten parantamiseen edistyksellisemmän luonnollisen kielen käsittelyn avulla.

Avainsanat: chatbot-työkalu, asiakastutkimus, käyttäjätutkimus, chatbotin persoona.

TABLE OF CONTENTS

FOREWORD

I would like to thank my supervisor Simo Hosio for his valuable advice and guidance. I would like to thank Denzil Ferreira aswell, for acting as the second examiner for my thesis.

I also would like to extend special thanks to my family and friends for their support during the writing process.

Oulu, 16th June, 2020

Anssi Meisalmi

LIST OF ABBREVIATIONS AND SYMBOLS

AI	artificial intelligence
UI	user interface
AIML	artificial intelligence markup language
SMS	short message service
NLP	natural language processing
STT	speech to text
NLU	natural language understanding
NLG	natural language generation
ECA	embodied conversational agent
NADiA	neurally animated dialog agent
TTS	text to speech
VUI	voice user interface
CLI	command line interface
GUI	graphical user interface
IoT	internet of things
CV	computer vision
SQL	structured query language
ML	machine learning
DNN	deep neural network
LSTM	long short-term memory
API	application programming interface
QnA	question and answer
FAQ	frequently asked questions
ITSM	information technology service management
VDMS	virtual diabetes management system
FR	functional requirement
NFR	non-functional requirement
HTML	hypertext markup language
CSS	cascading style sheets
JS	javascript
DOM	document object model
AWS	amazon web services
WWW	world wide web
URL	uniform resource locator
HTTP	hypertext transfer protocol
SDK	software development kit
HTTPS	hypertext transfer protocol secure
TLS	transport layer security

1. INTRODUCTION

1.1. Background

Customer services are constantly moving towards automation and the increasing capabilities of intelligent software agents are part of the cause. As intelligent agents like online chatbots and virtual assistants improve, the ability to help customers through capability, presentation, personality, and quality of services increases. In the future Artificial Intelligence (AI) assistants will have an even greater effect on customer experience than today.

The chatbot system implemented in this thesis is called Custom Chatbot Maker. It provides tools to create, manage, and use online chatbots. Bots in the system are rule-based with basic input processing and best match response selection. Cloud database is used to store chatbot data and chat history. Bots can be given shared or unique dialogue options. Each bot has a customisable profile with name, description, avatar. The system has an interface for managing bots in the database, and a fully functional online chat User Interface (UI).

1.2. Scope and Objectives

The Custom Chatbot Maker is evaluated by analysing the results of a pilot study. During the pilot study participants completed tasks by interacting with one of four chatbots. Each bot used in the study is designed with different agency unknown to participants. Quantitative data was gathered with a survey after the interaction is finished. Data analysis was done using descriptive statistics, and the plots showcase different experiences study participants had. How the different identities of bots affect user experience are measured with changes in perceived agency, opinions, and morality of bots.

Chatbots created with the tool are limited to scripting based responses. They are useful when studying guided customer experiences like the online bank support agent used in the pilot study. The customer experience is controlled by giving the bots highly accentuated personalities. Every bot still has the same objective: to help customer, but each one of them also serves a different secondary interest. This is key contribution of the tool: it allows for the same site to host an unlimited amount of different chatbot personalities and respond differently to the same inputs by user depending on which persona is currently being selected. The chatbot personas are controlled via simple HTTP parameters upon page load.

This thesis has two main objectives it seeks to achieve. The objectives are selected with the scope of the work in mind and will be reviewed at the discussion chapter. Objectives are:

- O1: implement the customisable chatbot research tool.
- O2: validate the key functionalities of the tool in a realistic user study.

1.3. Thesis Outline

In this thesis work related to chatbots is explored, an easy to use and modify online chatbot tool for research is implemented, and the tool is evaluated using a pilot study.

Related work chapter is split into four sections related to chatbots. First section is a broad background description where also notable events are showcased from the history of chatbots. Then, conversational systems are given an overall examination. Third section presents development platforms for chatbots and virtual assistants. Last section examines the many applications of chatbots.

Third chapter contains the steps taken when implementing Custom Chatbot Maker research tool. Functional and non-functional requirements state the requirements for functions and qualities the final software should be able to achieve. Second section introduces the software design process consisting of used technologies, architecture, data structures, algorithm, security, and user interfaces. Third section gives description of how the software was tested to ensure requirements were being met. Software testing has four parts: unit testing, integration testing, system testing, and user acceptance testing.

The complete research tool is finally evaluated using a pilot study with real users. Study methodology section presents how environment, participants, and materials were used in the study. Study results and analysis section presents survey results that the participants had filled after interacting with a chatbot. Survey questions reveal subtle differences in agency of bots, opinions on bots, perceived morality of bots, and reflections on the four chatbot personalities demonstrated in study.

Fifth chapter discusses the work done in this thesis. Discussion sections are divided between software implementation, study circumstances and results, thesis objectives, and future work.

The final chapter summarises the thesis.

2. RELATED WORK

This chapter examines related topics on conversational systems. First section outlines a short history on background of chatbots and some significant milestones. Second section is a brief walk-through on different aspects needed to consider when developing a conversational system. Third section showcases some chatbot and virtual assistant development platforms and their main features. Fourth section lists applications for chatbots.

2.1. Brief History of Chatbots

Early concept of chatbots, also called chatterbots, began in 1950 with Alan Turing discussing the topic of machines thinking intelligently. He proposed a game named Turing test in which through written responses, a player must determine if the response comes from a machine or a person. [1]

1966 Joseph Weizenbaum creates ELIZA, a program that fools users in to believing they are conversing with a real human. It operates by matching keywords or phrases found in input message, and responding with pre-determined responses. If no keywords are found, a generic, or an earlier response is selected as response. [2]

1972 Kenneth Colby emulates ELIZA with PARRY, a program designed to simulate a person with paranoid schizophrenia. Colby used a variation of Turing test on a group of psychologists, fooling half of them that PARRY was a real patient. Later on PARRY spoke with ELIZA, which was using 'The Doctor' persona, several times. [3]

1988 Rollo Carpenter finishes first iteration of Jabberwacky. It was released online in 1997 to simulate natural human chat for entertainment. It adds user responses from previous conversations to database and uses them to increase vocabulary. It rejects certain types of responses and after manual clean-up accepts others. Currently Jabberwacky is set as a legacy site, with links to its evolved version: Cleverbot. [4]

1992 Dr.Sbaitso is an AI speech synthesis software made for Microsoft DOS personal computers. Made to showcase digitized voice that sound cards were able to synthesize. The software was bundled with some sound cards manufactured by Creative Labs.

1995 A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) is a free software chatbot created in AIML (Artificial Intelligence Markup Language), an open, minimalist, stimulus-response language for creating bot personalities like A.L.I.C.E. AIML uses two categories: pattern and template. Pattern is matched with input message, while templates are used to construct the response.

2001 SmarterChild, a popular chatbot that was available for Short Message Service (SMS) networks, AOL Instant Messenger, and MSN Messengers. Combines fast information delivery, like news, sports, weather, with entertaining answers based on personality.

2006 IBM's Watson is a system capable of answering questions in natural language. Originally the system was designed to compete and win in game show Jeopardy.

In 2010s many intelligent virtual assistants became available as voice activated assistants. Virtual assistants share many aspects with typical chatbots, only the focus is being placed more on smart-home and smart-phone applications. They are usually speech controlled with commands and requests, and can perform both goal-oriented and general conversation tasks.

Table 1 lists some notable virtual assistants, their release year, and the company behind the virtual assistant. Google now and S Voice were initially restricted to smartphone applications, but were later improved and expanded upon by being released as IoT applications as well.

Table 1. Examples of some notable virtual assistants

Year	Name	Company
2010	Siri	Apple
2012	Google now	Google
2012	S Voice	Samsung
2014	Cortana	Microsoft
2015	Alexa	Amazon
2016	Google Assistant	Google
2017	Bixby	Samsung
2017	Alice	Yandex

2.2. Conversational Systems

This chapter gives an overview on conversational systems, their types, as well as the key tasks that are used in conversational systems: input handling, input analysis, dialogue management, output generation, and output rendering.

Conversational systems are computer programs that are designed to engage with humans using natural conversation. Conversation is communication between two or more people, so conversational systems mimic human communication as much as possible. They are powered with AI to interpret the meaning behind text messages, images, and speech. Generally used communication mediums are online text chat or clips of audio. Computer programs simulating natural conversation need to have at least some sense of context, flexibility, and personality. Conversational systems have many applications, but often can be seen in assistant or support roles. [5]

Computer programs can be made to understand written and spoken language using Natural Language Processing (NLP) and speech recognition. NLP is a field of study with parts of artificial intelligence, computer science, information engineering, and linguistics. NLP techniques are used to process syntax,

semantics, discourse, and speech. Main concepts in NLP are speech recognition, natural language understanding, and natural language generation. Speech recognition or Speech-To-Text (STT) is used to convert audio clips of human speech to text as accurately as possible. This is an essential task in conversational systems that use voice to converse. Natural Language Understanding (NLU) is a subset of NLP and its processes are used to find meaning in input. NLU processes include text categorisation, content analysis, and sentiment analysis. [6]

Natural Language Generation (NLG) is a process used to transform data to natural language. NLG can generally be template based or dynamically generated. In template based NLG a sentence has template with gaps for intents and entities. Dynamically generated can be considered more real natural language generation than template based. [7]

2.2.1. Conversational System Types

This subsection describes four types of conversational systems. The types include chatbots, Embodied Conversational Agents (ECA), spoken dialogue systems, and user interfaces.

Chatbot

Chatbots are computer programs that try to simulate conversation with humans using natural language. Chatbots generally communicate using a chat over the internet. Some bots try to appear as just another user in the chat and other times it is made obvious that the bot in chat has a specific function. Usually bots are designed with a specific role in mind, but can also be multipurpose. Chatbots can be placed in two categories: task oriented and general conversation chatbots [8]. Task oriented bots attempt to extract the intent of user message and fulfill the given task or request. General conversation chatbots try to entertain other chatters and keep the conversation flowing.

Chatbots can also be placed in to three categories: question answering agents, task-oriented dialogue agents, and chatbots [9]. Question answering agents provide direct answers to user queries based on knowledge drawn from multiple data sources. Task-oriented dialogue agents are required to complete tasks like reservation and scheduling. Chatbots need to converse seamlessly like a human would.

Embodied conversational agent

Embodied conversational agents are bots that use graphical representation in addition to natural language. They are able to enhance communication using animated facial expressions and gestures similar to those people are familiar with.

An embodied conversational agent has a body with which to perform gestures and other non-verbal communication like haze, posture, facial expressions. Greta is good example of a general purpose ECA [10]. Facial expressions, for example, can be recognized based on local binary patterns [11], geometry, or Gabor-wavelets [12].

Neurally Animated Dialog Agent (NADiA) uses both the users verbal input as well as their facial expressions in its responses. NADiA combines a neural language model that generates responses to user prompts using a convolutional neural network for facial expression analysis [13].

Research has been made on how embodied conversational agents affect peoples' social behavior during interaction [14].

Spoken dialogue system

Spoken dialogue systems only use voice for communication. They are suited for short service type conversations with clear steps or states that the system walks users through. Spoken dialogue systems can use online voice chat, phone calls, or voice based devices to communicate.

Examples of spoken dialogue systems generally include systems for guidance and introduction. ITSPOKE is an intelligent tutoring spoken dialogue system that gives students feedback on qualitative physics problems [15].

Spoken dialogue systems can be placed in tourist attractions to guide and give information on attractions and history. A study was made by changing neutral speech to expressive speech in synthesized Text-To-Speech (TTS) in a tourist attraction [16]. It indicates that expressivity in response generation can enhance spoken dialog systems.

Evolution and commercial success of spoken dialog systems have been studied in the developed world [17]. The study discusses how spoken dialogue systems have been evolved and how they can be used in the future, for example in developing worlds by translating local language.

User interface

Conversational systems have applications as text- and voice based user interfaces when voice recognition is included [18]. Voice User Interface (VUI) is similar to Command-Line Interface (CLI) and Graphical User Interface (GUI), but it uses voice commands instead of typing a command or clicking a button to complete a task. Applications exclusive for VUIs can commonly be found vehicles and smart home devices like smart speakers. Tasks conversational UIs can manage are for example radio, media player, messaging, telephone calls, scheduling, temperature and lighting control, and information search.

Virtual assistants often operate as the interface to control Internet of Things (IoT) devices. IoT is a network of smart devices in a small scale environment like a local home network. A smart home system has multiple devices connected and powered with AI, thus minimizing human interaction required to control the devices by centralizing the control panels of all devices in the network. Examples of virtual assistants capable of controlling smart home applications are Apple's Siri, Microsoft's Cortana, and Google Assistant. [19]

Sometimes an interface agent is necessary to help users navigate complex environments. Conversational systems of this kind can also called intelligent user interfaces. They have broad understanding of the environment, and will attempt to help users with their tasks by monitoring current activities and suggesting tips

and helpful resources. A well known intelligent user interface is Microsoft Office Assistant, perhaps better known as Clippy, was first introduced with Microsoft Office 97 as interactive interface helping users find help content. SPICE-A is a conversational user interface that can be used to help navigating an electronic program guide [20]. Another example of a conversational user interface is used to help activating metrics and possibilities of a visualisation tool user may not be aware. The visualisation tool in this case visualizes data about the structure of a large software [21]. JUPITER is an audio based conversational interface using telephones to share weather information [22]. It allows users to obtain worldwide weather forecast information over the telephone using spoken dialogue.

2.2.2. Input Handling

This subsection describes how voice, message, and image inputs can be handled in conversational systems. Handling input requires multiple processes before it can be easily analysed. Text and voice based inputs use NLP techniques. Digital image based inputs use computer vision techniques.

Natural language processing

Recognizing spoken or written language can cause problems, because each input has multiple variables affecting every word and sentence. Variables include language proficiency, grammar mistakes, environment noise, and size of vocabulary. Each variable significantly complicate finding the true meaning behind words. Context and sentiment can also completely change the meaning of sentences. [23]

Common tasks in NLP are [24]:

1. Tokenization. Process of segmenting a large string of words to smaller components.
2. Part-of-speech tagging. Based on grammatical context decide the meaning of words that otherwise can have multiple meanings.
3. Parsing. A parse tree, or some other data-structure, is created based on the syntax of used language, where tokens are divided to constituents.
4. Stemming. Words are shortened to their root forms.

Computer vision

Techniques of Computer Vision (CV) are essential in tracking and recognizing handwriting and gestures [25], facial expressions [26], or other patterns [27] in input of images and videos. Including object recognition increases ECA's and chatbots' possible applications and scope. Recognizing objects [28], facial expressions [29], and hand gestures [14] are some examples of what object recognition can enable in conversational systems. The gestures or facial expressions can be copied or further analysed to increase interactivity of the agents.

Even only by analysing human motion [30], can the activity and posture of interacting person be conveyed to a conversational system. For example a smart home system could dim the lights when it recognizes a human body laying on a couch, and turn the lights off when no movement is detected.

A conversational system that has an AI agent analyse images and answer questions regarding it in natural language is another good example of what CV makes possible [28].

2.2.3. Input Analysis and Understanding

Input analysis is needed to find the meaning behind words or expressions and can be done using NLU methods. Generally, input can be analysed by using rule-based or machine learning based methods [31].

User intent analysis benefits from grouping input to categories. Chatbots often have common concepts like intent, entity, confidence score, lifespan, and context. Intent can mean methods or skills like turning on the light or searching for data. Entities can be locations, objects, dates, currency, or device states. Confidence score is a decimal value from zero to one. It assists in selecting right intent and therefore the next response or task. Lifespan is the value of consecutive unrecognized inputs, in other words the bot can not help with an issue if lifespan accumulates. When certain threshold is met, the conversation is transferred to a real human support. Realizing the context becomes is relevant with words that have multiple meanings. Sometimes follow-up questions are needed before the original intent can be selected with certain confidence. [32]

Rule-based

In rule-based systems the analysis process goes through handcrafted set of rules, where certain keywords are tied to a response. Rule-based systems most often use pattern matching to analyse input and compare against knowledge base. Pattern matching means looking for a same or similar pattern in between sequences, like when looking for a word in a sentence [33]. The input word or phrase must match a keyword by a certain margin in order to be accepted as a match. Informal language and typographical errors are common in natural conversation, and therefore are a big problem for rule-based systems. A strict matcher will have to disqualify response if even one character in input does not match a keyword or a pattern. Processing typographical errors gives some leniency to the pattern matching, by for example allowing certain margin of error. If the system suspects and error based on the margin, it can sometimes request clarification from user. [34]

Pattern matching can also be implemented by using Structured Query Language (SQL) to query database with input words [35].

Machine learning based

Machine Learning (ML) is the study of algorithms and statistical models that computer systems use to complete a task without using exact instructions. ML

algorithms form a model based on learned behavior. Experience for model is first gained through training data, and can often later be improved by processing user responses during normal use. ML based system will usually respond with the most often used response to the input in its experience. ML is useful for training enormous amount of intents and entities data. Because ML based systems can learn vocabulary automatically, they often have bigger vocabularies and wider area of topics and tasks than rule-based chatbots. [36]

In order to form the model for ML based system, the learning data can be processed trough supervised or unsupervised learning. In supervised learning the training data is labeled so the system will have accurate comparisons between desired input and output. Unsupervised learning is done with unlabeled data, where algorithms draw conclusions based on the patterns in input data. [37] Actual chat conversations or for example movie dialogue subtitles are examples of training data for unsupervised learning.

Deep learning is subset of machine learning based on artificial neural networks. Deep learning models are composed of multiple processing layers. They have multiple processes, which each specialize in recognizing few types of features. Deep learning aims to predict future using trained models. [38]

Machine learning based input analysis can be retrieval-based or generation-based. Retrieval-based conversational systems use large database of pre-defined responses, while generation-based conversational AI does not use databases, but instead they use a trained model to generate response [39].

Retrieval based systems can learn by using Deep Neural Networks (DNN). DNNs have multiple layers between the input and output layers. Results are improved by using vast amounts of training data. DNNs are machine learning models that work well on speech recognition. DNN model can be trained for example by using vocabulary business search data set collected from Bing mobile voice search application. [40]

Sequence-to-sequence learning is used with Long Short-Term Memory (LSTM) neural networks. LSTM is an artificial RNN architecture. RNNs are useful in NLP, because each word in the sentence are recursively analysed. [41]

Models for customer service can be generated using LSTM networks. With LSTM the system takes a request as the input, computes its vector representations, feeds it to LSTM, and then outputs response. Such a system was trained on nearly 1 million Twitter conversations between users and agents from over sixty brands. [42]

2.2.4. Dialogue Management and Output Generation

Dialogue manager manages the general flow of the conversation. It chooses how the output is generated based on the results of analysis [31]. Dialogue management is important if a conversational system aims to be as natural as possible, as it can minimize bad responses in uncertain situations.

In some conversational systems the dialogue manager can strictly direct user from one point to the other, like in simple telephone based spoken dialogue systems. A dialog manager capable of recursive definition of dialog flow is

preferable for example in banking applications, where user can jump back and forth dialogue flow using shortcuts. [43]

Dialogue manager benefits from saving the conversation history. By analysing previous messages can a response be generated if current topic of discussion has ended. Dialogue manager can save and update state of conversation using a dialog state tracker. Dialog state tracker saves all observable elements of the conversation. It can save confirmed and unconfirmed statements related to intents or entities, for example. [44]

In some systems, dialogue manager needs to choose a communication strategy in order to proceed with the conversation. The strategy depends on conversation leader, which can be the user, bot, or a script. Language tricks are needed to generate a response when good response was failed to be generated based on user's input. Language tricks include topic switching, open ended questions, jokes, or asking for more information. [31]

2.2.5. Output Rendering

Generating output in natural language is often the final step for text-based conversational systems, but output can be further rendered using images, audio, or other actions.

Text-To-Speech is converting natural language to speech. Tone and quality affect how the output is understood and received. TTS is a necessary part in many conversational systems that use voice communication.

Goal or task oriented conversational systems can provide information or advertise by providing pictures or videos of products and services. Some systems are capable of immediately ordering a product or making reservations, others can provide links or other ways to contact a company. Rendering additional music to audible responses can enhance the message or mood.

Avatars are a visual representation of a character and can make chatbots more relatable. The mood of the output can be transformed or even animated to further enhance the conversation experience.

Rendering eye gaze [45] in real-time and simulating natural eye contact during conversation makes the system even more relatable.

An emotionally aware embodied conversational agent has multiple channels to render. Its behavior is realized through speech, facial expressions, and body language in the form of audio and animations. [46]

Embodied conversational agents can use two- or three-dimensional representations as avatar images on a display device, but some agents require a physical body as well. Issues with physics, visual and speech recognition are more pronounced in a real physical space with uncertain sound levels and disruptions.

2.3. Chatbot Development Frameworks

This section lists development frameworks marketed for both business and casual chatbot applications.

Chatbots and other conversational systems can be developed for most platforms with several options to choose from in terms of development frameworks. The offered functionality and applications vary from one framework to the other, so when choosing whether to use one, it is a good idea to take a look at the multiple options. The frameworks often also have differences in NLU performance in similar tasks [47], so acknowledging the differences is good practise.

The list of featured development frameworks are:

1. Microsoft bot framework,
2. Botpress,
3. Wit.ai,
4. Dialogflow,
5. Pandorabots,
6. Amazon Lex,
7. SiriKit, and
8. IBM Watson.

2.3.1. Microsoft Bot Framework

Microsoft Bot Framework is a comprehensive framework for building enterprise-grade conversational AI experiences. It enables developing bots from simple QnA bots up to smart virtual assistants.

Data input type options are touchscreen buttons, text, speech, and adaptive cards. Adaptive cards are condensed information pages rendered to chat screen space. Communication channels available include the most used, like websites, Skype, Slack, e-mail, SMS, and phone calls. Along with personal computers, other supported devices are smartphones, smart home systems, IoT devices, and cars to name a few.

Cognitive services that are supported for virtual assistant bots are speech, QnA, language understanding, vision, and search functionalities. Knowledge that are supported for search are general knowledge, data sources, and manuals. Skills that can be interacted with include mail, tasks, calendar, other bots, and custom skills. [48]

2.3.2. Botpress

Botpress is an open-source bot building framework advertised for businesses to make conversational assistants. Main features are NLU, dialogue manager, and communication channels that are all made easy for non-developers.

With Botpress, it is possible to manage conversations with a flexible flow editor. NLU can understand meaning, intents, and entities found in input messages. It also has built-in analytics on bot operation to allow easier optimisation and tweaking. Most communication channels are available like Messenger, Skype, SMS, and websites. Botpress has complete GUI for making and analysing bots.

It also supports 3rd-party applications and Application Programming Interfaces (API). [49]

2.3.3. Wit.ai

Wit.ai is an open and extensible natural language framework for developers to build automation and interfaces for mobile applications, wearable devices, and other home electronics.

Wit.ai can support text or voice based bots on most platforms. It is possible, for instance to make hands free mobile apps controlled with voice commands. Supported devices include home devices for home automation, hands free wearables with tiny screens, and other hardware like robots.

The framework can be integrated to mobile apps, Facebook Messenger, and websites by using node.js, python, or ruby clients. On other platforms HTTP API is usable.

Intents and entities are analysed from commands and are used to call methods on devices or apps. [50]

2.3.4. Dialogflow

Dialogflow is a natural conversational interface development suite by Google. It is build on Google infrastructure and optimized to be used with Google Assistant devices.

NLU for Dialogflow is powered by Google's AI and machine learning. Sentiment analysis considers concepts like: intents, entities, and contexts. Agents made with Dialogflow support events for actions not related to discussion, fulfillments to handle responses of intents, and it can integrate to most other conversation platforms. Dialogflow can integrate with communication channels like Google Assistant, Slack, and Facebook Messenger.

Input can be in text form or extracted from Google Assistant. Audio input and output are also supported with Google's speech-to-text and text-to-speech APIs. [51]

2.3.5. Pandorabots

Pandorabots is an online web service for building and deploying chatbots. It is useful for both hobbyists and big brands. Chatbots can be made with or without writing code, so anyone is encouraged to use the framework.

There is support for most communication channels like Messenger, Whatsapp, web, SMS, Skype, and Twitter in either voice or text form. Text-to-speech and speech-to-text functionalities are available for voice applications.

AIML is the primary language currently used by the Pandorabots framework. It is an extension of XML and can be used to generate content for any spoken language.

Pandorabots is relatively old current framework, as it has been in business since 2008. It offers possible range of solutions from a free DIY platform to fully turnkey chatbot and application development. Pandorabots is a flexible, extensible open standard with a large community backing. By being a scripting based chatbot system, it does not necessarily suffer from some of the drawbacks of machine learning based systems. [52]

2.3.6. Amazon Lex

Part of Amazon Web Services, Amazon Lex is a service for making conversational interfaces using voice and text inputs.

Amazon Lex has easy to use console with extensive guides along with integration with AWS. It is possible to deploy chatbots to mobile devices, web apps, and chat services. As with other cloud services offered by Amazon, Lex charges only for text or speech requests made.

Lex uses same technology as Alexa with deep learning solutions to speech recognition and language understanding tasks. Utterances invoke intents, which require entities called slots on the framework.

It is suitable for chatbots used in call centers with features like 8 kHz telephony audio processing, extracting intent and option to query other applications. Informational chatbots can be give access to latest news updates, game scores, or weather. Possible applications include bank account management, booking tickets, ordering food, or calling a cab or a ride-share services. Another service offered is an enterprise productivity analysis with fast access to sales data, marketing performance, and customer service status. Internet of Things applicability is supported when chatbot needs to connect to IoT devices and forward commands or data. [53]

2.3.7. SiriKit

SiriKit can be used to handle user requests for third-party apps using Apple's Siri or Maps services. Siri is an intelligent assistant that comes with many Apple devices. The devices include iOS, watchOS, tvOS and macOS devices.

SiriKit has Intents and Intents UI frameworks that are used for third party services with Apple apps like Siri and Maps. Intents app extension receives requests from Siri, and accesses third-party app services. Intents UI app extension for Siri and Maps enable customizing how the data received from app extensions is presented.

By using Siri, it is possible to control applications like messaging, payments, media, booking, or reservation.

SiriKit has support for the following domains: VoIP calling, Messaging, Payments, Photos, Workouts, Ride booking, Car commands, CarPlay, and Restaurant reservations. A domain is a category of tasks that Siri already has knowledge to talk about. [54]

2.3.8. IBM Watson

IBM's Watson allows developers to build, deploy, and optimize chatbots for businesses by using Watson Assistant. Watson offers building of conversational interfaces for any application, device, or channel. It can be taught when to search for an answer from a knowledge base, when to ask for clarity, and when to direct user to a human for more support.

Watson is pre-trained with industry-relevant content. It can learn from historical chat and call logs. With a visual dialog editor, custom dialogue can be constructed. Watson has industry leading AI to power NLU as well as NLP, with support for up to 13 languages in 2019. It also supports IBM Cloud, which offers an open and secure public cloud for businesses.

Watson APIs include AI solutions for language, speech, vision, and empathy APIs. Each API can be useful when building chatbots. [55]

2.4. Chatbot Application Areas

This section offers a look at categories of applications suited for chatbots. Categories of chatbot applications include marketing, customer support, health care, and social chatting. Some roles chatbots can automate work are as shopping helpers, customer support agents, scheduling and booking services, or a human-like chatters. [56]

Chatbots can be used on most communication platforms like email, SMS, messenger apps, websites, forums, social media, and standalone devices. The chatbots use the same channel for communication, as other users in the system, and sometimes can even be indistinguishable from other users. Other times the bots' appearance and role is made obvious, and users are recommended to use many functionalities the bot can perform to improve their experience. Loebner prize competitions are held each year to award prizes to computer programs that present the most human-like chatting experience [57].

2.4.1. Marketing

Chatbots can be used to advertise products or services to users on mobile and online platforms. Bots are suitable for direct advertisement on communication channels like social media used by groups and individuals [58].

E-commerce chatbots often accompany marketplaces, where people are browsing and shopping. The bot may help find items from the catalogue and give suggestions based on users actions. An e-commerce bot can even act as a shopping basket capable of storing shopping lists or one-off purchases. The bot may also point out similar products the user could be interested in based on viewing and purchase histories. SuperAgent is one such example of chatbot using natural language to help customers of e-commerce websites to gather more information on products [59]. SuperAgent also takes advantage of previously gathered customer data when generating responses.

Chatbots in marketing can provide improved customer service, data-driven product recommendations, automated lead generation and qualification, a mine of customer data, rewarding customer loyalty. They can proficiently be used to gauge customer opinions by passively observing or by proposing questions and analysing the mood for services and product being discussed. Chatbots like SamBot can be used to improve marketing aspect of a corporate website as well as its interactivity [60].

Chatbots' role in marketing also bring in a risk of doing more harm than good. They can be very obtrusive, for instance if they invade public chat spaces with spam or irrelevant ads. Understanding customers' expectations is important for companies adopting automated customer service. Chatbots' effect on customer satisfaction regarding luxury brands has been researched [61]. The research compares traditional face-to-face interactions against interactions with online chatbots.

2.4.2. Customer Support

Chatbots can effectively be used as support in customer service. They will ease the burden of support by either solving simple issues first, or transferring the conversation to human support for more difficult issues. Support mediums include phone calls, private chat, and social media like Twitter or Facebook.

Support chatbots ease logging work schedules and appointments on the go at any time. Specifically managing time sheets or calendars is possible with setting the bot to operate on nearly any communication platform in use.

Chatbots perform incredibly well as automated Question and Answer (QnA) services. An even more straightforward solution is a Frequently Asked Questions (FAQ) service as chatbot. Both options can provide better interactivity between a company and its users. Transferring conversation to human support also happens seamlessly without user having to wade through layers of topics in worst cases. Still, it would be best to offer the option for users to choose between a traditional QnA or FAQ services and ones with chatbot support. A specific example of a chatbot designed to help search and return solutions is an Information Technology Service Management (ITSM) application that helps users search solutions using natural language [62].

Similarly to FAQ services, implementing a search engine feature to chatbots in order to search web or databases can be a helpful feature. The same point applies to other functions applicable to chatbots like text or articles summarisation. Chatbots have an application as computer-based tutorial for software. They are also suited to help in technical issues by debugging and analysing problems in software frameworks.

Another example of a support role is a mediator chatbot making requests for expert chatbots on a group chat. Mediator chatbot analyses users utterance, invites expert bot to group chat, and forwards relevant data to it. The expert bot also uses natural language for communication and is knowledgeable on a limited area of expertise. [63]

Chatbots can help in education as well. They can be designed as either singular structured lessons, or as teacher's assistant during lessons by helping with simple concepts and questions. Chatbots may play a similar role to coursebooks and exercise books in an interactive form. They can analyse exercises and provide feedback for example with math problems, language lessons, and history. ITSPOKE, for example, is an intelligent tutoring spoken dialogue system that gives students feedback on qualitative physics problems [15]. Web based chatbots have also been studied in order to teach foreign languages through chat [64]. Another education application is a support system for students, that can detect questions and give answers to students [65].

2.4.3. Health Care

Extension of support chatbots are bots specialised in health care topics. These types of bots are suited to support-, motivate- and educate patients. They can also remind patients of upcoming appointments and when to take a dose of medicine, like for example PharmaBot, a conversational chatbot that is designed to prescribe, suggest and give information on generic medicines for children [66]. Bots in health care need to be easily accessible for patients like, for instance in voice based conversational systems. [67]

At hospital locations chatbots can automate roles as hospital administrator, health care consultant, self-care coach, elderly care assistant, and even a red panic button for emergencies. These roles can automate smaller tasks that help with work load. A mobile chatbot can help identify medical conditions by analysing images or descriptions of symptoms through mobile app [67].

Issues of chatbots in healthcare arise when they need to be fully reliable in understanding input and doing activities. Otherwise patients' health can be compromised from errors in language understanding or generation. Building a robust corpora for medical terminology is important [68].

Example of a conventional chatbot extended with external knowledge source Wikipedia by using Media Wiki API. It is an open source AIML web based chatbot modified and programmed to educate and help manage diabetes patients and general community. The chatbot is called Virtual Diabetes Management System (VDMS).[69]

2.4.4. Social

Social chatbots are mainly used for general conversations or entertainment. These chatbot often are developed with personality or an acting role. Unique personalities can be crafted from scratch, or known characters can be replicated from movies or books. With text-to-speech functionality enabled, chatbots have applications in reciting audiobooks or articles on request. Chatbots can also recite jokes or even attempt to generate original ones [70]. Public or private chat rooms, or a virtual assistants are great platforms for social chatbots.

One chatbot is used to analyse users' affective states using natural language [71]. The system's application scenarios include its use for querying individual users about their affective states in relation to various entities, events and processes. This type of social application helps understanding of the affective states of individuals, groups of people, relations between the occurrence of external events and collective group feelings, and how those form and spontaneously evolve over time. Other existing application for social chatbots are speech-enabled conversational agents with application to voice enabled chatbots in a virtual storytelling environment [72]. The system extracts information from RDF/XML files.

Chatbots can be programmed to be able to play games as a host or as a player. Suitable game genres include guessing and trivia games, some classic games like chess and battleship -type board games, or the bot may only be used to launch external game executables on command. Guidelines written for believable bots in video games can be applied to chatbots as well [73].

3. CUSTOM CHATBOT MAKER IMPLEMENTATION

This chapter describes the implementation of an online customisable chatbot designed to be used as research tool. Featured sections are requirements, design, and testing. The resulting software is called Custom Chatbot Maker, available as open source on GitHub [74]. It was developed in 2019-2020 by Anssi Meisalmi. Development process was iterative with regular meetings with the supervisor of this thesis.

3.1. Requirements

Requirements are divided to functional and non-functional requirements. The following requirements documentation defines the necessary functions and qualities the software needs to be able to fulfill. The key functionalities set for the chatbot tool are customisability and usability. Requirements need to satisfy all users of the system. Users include researchers who set up and manage the study environment, and research participants who interact with chatbots in the study environment.

3.1.1. Functional Requirements

Functional Requirements (FR) describe functions and tasks the final software needs to be able to perform. These are features that are at minimum necessary for the complete software.

Functional requirements are listed in Table 2. Requirements FR01 and FR02 state that new chatbots can be added to the system and modified afterwards. Main characteristics each chatbot have are name, description, and avatar. Chatbots need to also have an option to respond with custom dialogue different from other bots in the system. FR03 states that it should be possible to switch from talking with one bot to another. In other words to select which bot is responding to users' messages. FR04 chatbot can be embedded on a website, so that the end users can access the bot using a web browser. FR05 interaction between user and chatbot is through a typical chat interface most users are familiar with. FR06 and FR07 state that the system needs to be able to provide functioning chatbots with reasonable expectations on results. FR08 chat logs are saved in database for later analysis.

These requirements form the base functionality of what the chatbot maker needs to be able to execute on command.

3.1.2. Non-Functional Requirements

Non-Functional Requirements (NFR) are qualities of the software. While functional requirements describe what the software is meant to do, non-functional requirements describe how the system is supposed to be. NFR are qualities of

Table 2. Functional requirements

ID	Description
FR01	Create new chatbots
FR02	Customize chatbots' appearance and responses
FR03	Switch between chatbots
FR04	Can be embedded on a website
FR05	Basic chat functions
FR06	Basic AI
FR07	Basic NLP operations
FR08	Log user interaction

the system that describe things like accessibility, security, usability, extensibility, or scalability.

Non-functional requirements are listed in Table 3. NFR01 states the system needs to be extendable, meaning that changing and adding features should be easy and non-disruptive to other features. NFR02 means that, in case the tool is used and modified by other than the original developer, source code should be easy to read and understand. NFR03 the project may use and is to be released as open source. Software under open source license usually means that source code can be accessed, modified, and used for certain purposes. NFR04 signifies that talking to chatbots should be possible with most web browsers, including ones on mobile devices. NFR05 the graphical user interface should be modifiable with relative ease.

Table 3. Non-functional requirements

ID	Category	Description
NFR01	Extensibility	Easy to add new features
NFR02	Readability	Easy to understand source code
NFR03	Open source	Project uses and is open source
NFR04	Platform	Usable with most web browsers
NFR05	Modifiability	Appearance of interface modifiable

By meeting non-functional requirements, the system is made easy to use.

3.2. Design

This section describes the design process of Custom Chatbot Maker. Design description includes technologies, architecture, interface, and future improvements used in implementation. This section should give a clear view on how the Custom Chatbot Maker functions.

3.2.1. Technologies and Libraries

Technologies and libraries presented are the set of tools used to develop and host this software.

The technologies are divided to front-end and back-end. Front-end consists of frameworks used to create the chat UI. Back-end frameworks are required to run and host the server and chatbots.

The front-end is made with Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (JS) languages. Chatbots can be embedded to websites using JavaScript HTML Document Object Model (DOM) elements. To do this, the website HTML source has to include the chat UI HTML file.

The web server is hosted using Node.js and Express.js. Node.js is a open-source JavaScript run-time environment. Express.js is a minimal and flexible web application framework.

Amazon DynamoDB, a fast and flexible NoSQL database service, is used to host all data of all chatbots and the log of chat history. NoSQL databases do not use SQL interface to store and retrieve data.

AWS Elastic Beanstalk is used to deploy the application using Amazon Web Services (AWS) [75]. Elastic Beanstalk allows quick deployment and management of applications to AWS Cloud Computing Services. Cloud services are computing services delivered over the Internet. The services include computer system resources, data storage, and computing power. Cloud services were selected for the project for easy scalability and reliable availability.

Node.js package 'natural' is used to process the NLP parts of the system [76]. It is a general natural language facility for JavaScript. Word tokenization is handled using the packages tokenizer 'WordTokenizer'. Tokenization means splitting user messages to an array of tokens, excluding alphabetic characters, digits, and underscore symbol. The tokens are stemmed with Porter stemming algorithm [77]. Porter stemmer is old and much used for its basic level word stemming. Porter stemming algorithm removes the commoner morphological and inflexional endings from words in English.

Levenshtein distance is used to measure the distance between keywords chatbot knows and words in user messages. The calculation is made with JavaScript package 'js-levenshtein' [78]. It is an efficient JavaScript implementation for calculating the Levenshtein distance.

3.2.2. Architecture

Software architecture describes the inner structure of software system. Software architecture realise the requirements of functionality and quality attributes.

The project uses World Wide Web (WWW), also known as the web, on the Internet. Web is used to transfer data and documents between computers, in this project between client and server. Accessing the web requires the use of web browsers to handle Uniform Resource Locators (URL). URLs identify location web resources on the internet. Web browser then renders the document as a web page. The document is often in HTML, a markup language designed to

be displayed in web browsers. Data is transferred using Hypertext Transfer Protocol (HTTP) requests. Some basic HTTP requests include GET, POST, and DELETE. GET request is made any time URL address is entered in web browser. POST requests are used to add new data to a resource with HTML forms. DELETE requests specify which resources or data are to be removed.

Custom Chatbot Maker uses client-server architecture model. Client-server models have multiple clients connect and send requests to a server or several servers with the same address. Servers try to respond to all HTTP requests they receive from clients. Communication model is asynchronous request-response. Asynchronous requests do not block the client from functioning while waiting for response from server. After response is received, the client will process data attached to the response and display the results. In this system asynchronous functions are requesting chatbot data, waiting for chatbot response, and finally modifying chatbot data. There is always a slight delay when waiting for server responses to appear on client side. The delay is affected by the quality of connection, how busy the server and database are, and the amount of data being transferred.

Deployment diagram is in Figure 1. It shows the execution architecture of the system. A client device is running browser, which sends HTTP requests to server running on AWS cloud services. AWS Elastic Beanstalk is used to deploy the web server running on Node.js. Express.js handles routes and calls chatbot module to process data received from client. Chatbot module accesses DynamoDB database using AWS Software Development Kit (SDK).

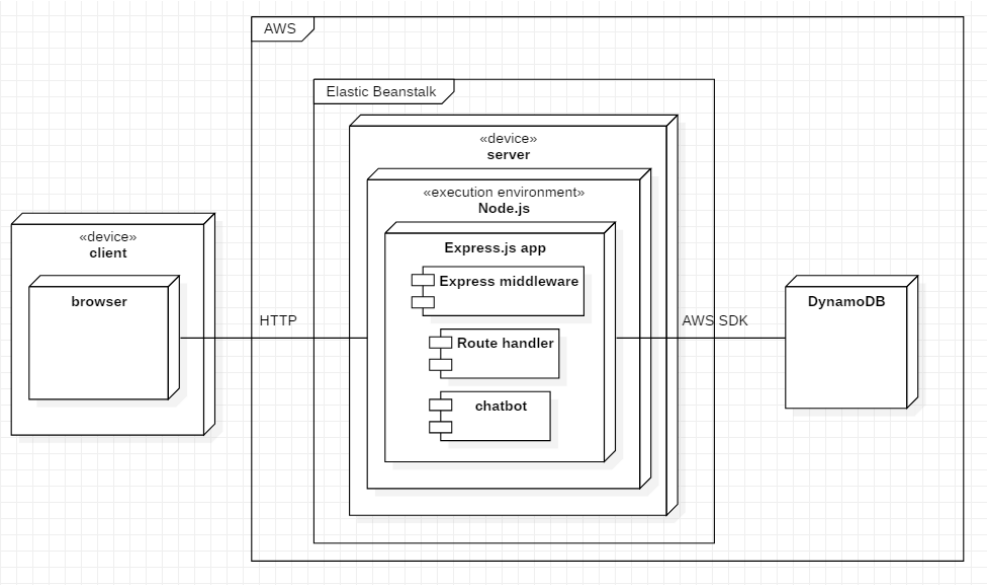

Figure 1. Deployment diagram.

Selecting bot identity interaction is shown in Figure 2. Client sends GET request with optional URL parameters to server, which queries the database and sends bot identity data back to client. Default bot identity is called chatbot, which is returned if user has not requested any specific chatbot identity.

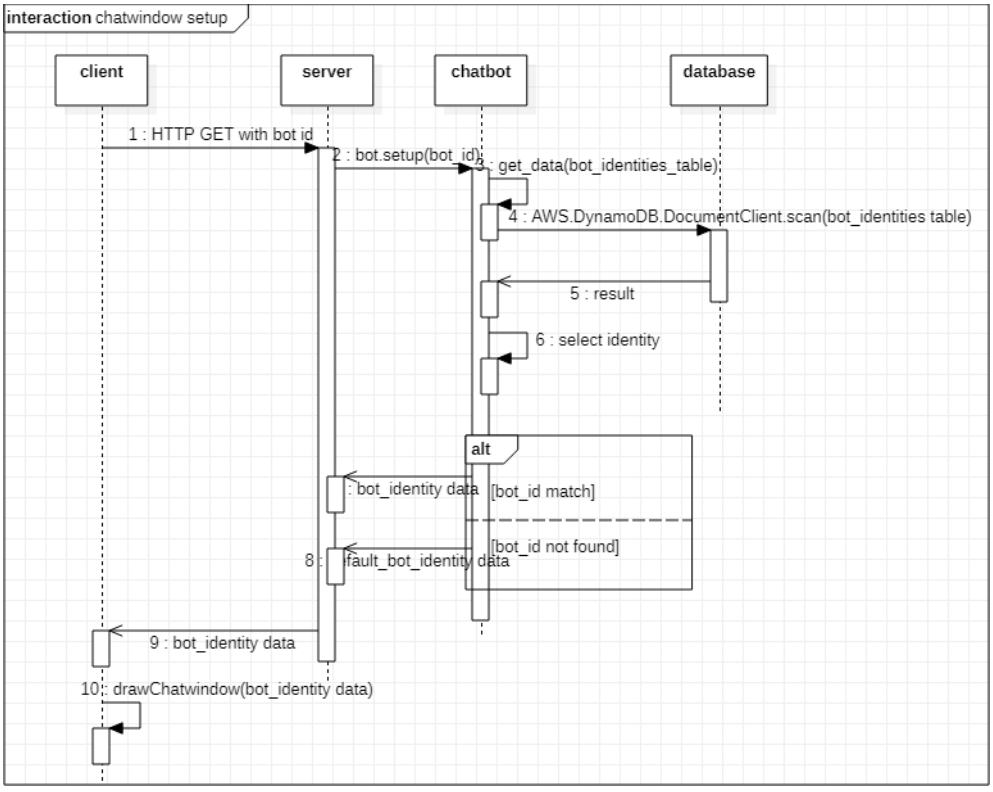

Figure 2. Chatbot setup sequence diagram.

Sending user message interaction shown in Figure 3. Client sends POST request with user message to server, which scans for current chatbot identity on database. After processing user message, best response is returned, and chat log is updated in database.

Modifying chatbots sequence diagram is depicted in Figure 4. The diagram shows how using the browser UI new bots are added to the database. All other available actions like update, add response, and delete share the same sequence.

3.2.3. Data Structures

User identification token and requested bot identity may be sent to the server as URL parameters. If user does not have preassigned identity, one is generated by the server. If bot identity is not sent, the server will return a default chatbot to client. User and bot identity tokens are saved as HTTP cookies, and are passed to server whenever user loads a page on the website with chatwindow embedded.

User identification token can be dynamically included in chatbot responses like URLs by using word user_id in the response. This is useful in research to connect same users over multiple platforms using a single identity token.

Chatbot data is stored in Amazon DynamoDB NoSQL database service. NoSQL in this database is schemaless with every table having unique primary key to identify data items. Data items can even be JSON documents like in this

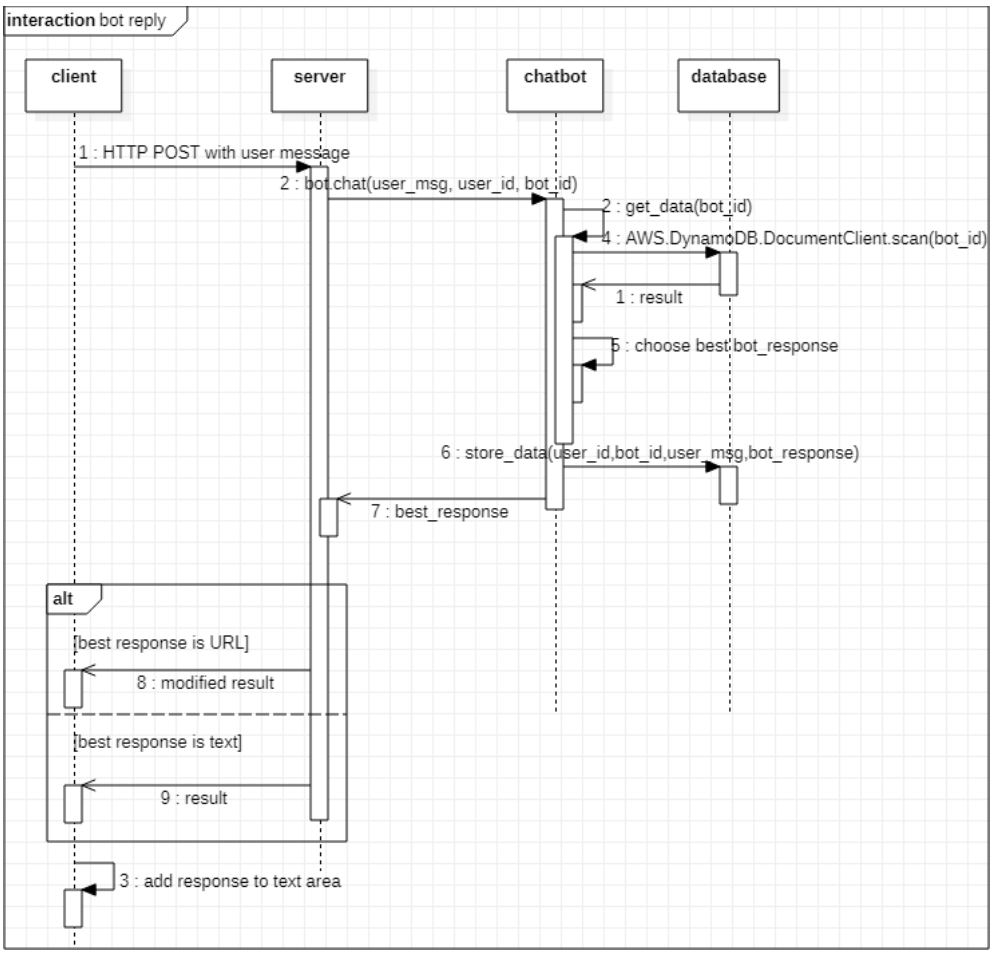

Figure 3. Chatbot reply sequence diagram.

project. JSON is an open standard file format that is easy readable to humans and often used to structure data for data transfer. Accessing the database is possible with AWS Management Console or AWS CLI. Web applications like the Custom Chatbot Maker can use AWS SDK to read and store data directly to DynamoDB. The data in DynamoDB tables are identified with partition key and sort key. Partition key is an unique primary key, while sort key is used to help organize the data table. DynamoDB table keys are similar to fields or columns in other database systems.

Three types of tables are used by the Custom Chatbot Maker system: bot identity table, chat history table, and chatbot dialogue tables. Bot identity table stores bot identity, name, description, and path to an image used as an avatar. Chat history table holds logs of each response chatbots make, along with user identity, user message, date-time, and optional meta data. Chatbot dialogue tables are named after chatbot identities to link the data table with a unique dialogue table. Chatbot dialogue tables store responses attached to keywords and also optional alternative responses to each response. Alternative responses enable a more diverse experience by switching between different responses to the same input.

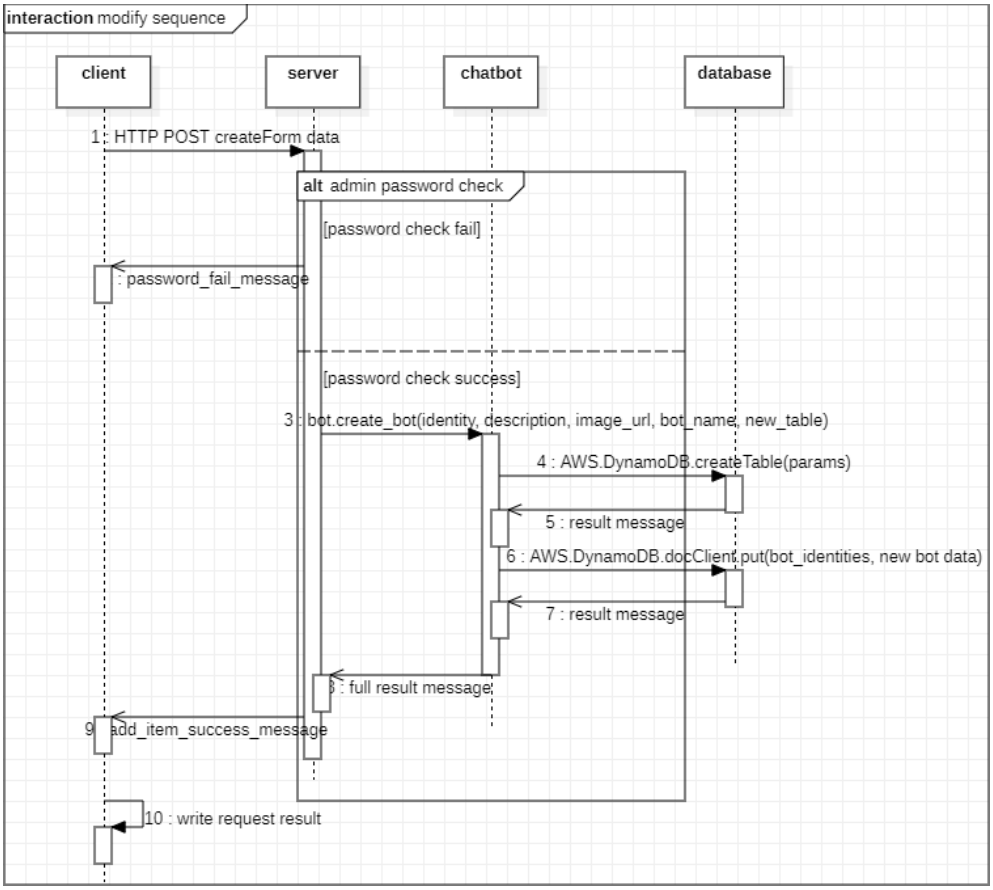

Figure 4. Modify chatbots sequence diagram.

For this project data types used are strings and arrays in JSON objects. Example bot identity data structure:

```
identitySchema = {  "identity":     "happy",
                    "description":  "I am happy!",
                    "image_url":    "img/face1.png",
                    "bot_name":     "HappyBot" }
```

Bot identity data structure has primary key called identity. Description is a message the user is greeted with when talking to chatbot. Image URL stores the URL path on chatbots avatar image hosted either on local machine, or the Internet. Bot name is the name shown to user when interacting with chatbot.

Example chatbot dialogue table structure:

```
chatbotSchema = [ { "response":     "Hello!",
                    "keywords":     [ "hello", "hi" , "hey" ],
                    "alternatives": ['Hi!", "Hey!"] },
                  { "response":     "Goodbye!",
                    "keywords":     [ "goodbye", "bye" ],
                    "alternatives": [ "Bye." ] } ]
```

Chatbot dialogue table's primary key is response. Every response has specific keywords that the response is tied to. Alternatives are the alternative responses that can be used as response as well.

Example chat log structure:

```
chathistorySchema = { "user_message": "hey",
                      "datetime":     "2019-12-14T19:49:17.484Z",
                      "user_id":      "dZq3QEp0TCbJ",
                      "bot_id":       "happy",
                      "bot_response": "Hello!",
                      "meta": "version 1.0.0" }
```

Chat log has primary key user_id and secondary key date-time. User message and bot response of bot identity are connected in order to help analyse which responses the chatbot chooses to return. Meta data helps to sort different events.

3.2.4. Algorithm Description

This section explains the Custom Chatbot Maker's response selection algorithm in two parts. The parts are chat function and string similarity calculation.

JavaScript code for chatbot chat function is presented in Appendix 1. Chat function takes user message, user identity, bot identity, and meta data as input parameters. User message is tokenized and changed to lowercase. Database is queried with bot identity and bot responses are returned. Each response in bot dialogue table is compared to input message by calculating their similarity score. Match score is calculated by dividing similarity score with total number of keywords. Match score is required to be at minimum 0.45. Match score of 1.0 means that every keyword had been found in message, while 0.5 means only half of keywords are matched. Highest scoring response is selected as best response. If best response has alternative responses, one is selected at random. Record of the interaction is added to chat history. The text 'user_id' is replaced with actual user identity so URLs that include user identity as parameter can use same identity token to identify research participants. Callbacks are used in asynchronous functions to return the output to the original call of the function by server.

JavaScript code for string similarity function is presented in Appendix 2. String similarity function takes arrays of keywords and tokenized user message as input parameters. Each keyword is stemmed and changed to lowercase. Keywords are then compared one by one to each word in input message. Levenshtein distance of each word pair is calculated, and the pair is given a word score based on the longer of the two words for better accuracy. If word score is not high enough, it is removed so multiple low scores added together will not cause a match. Score of 1.0 mean the word pairing match perfectly. Scores between 0.6 and 0.9 indicate some differences between the words, but can be considered a close enough match to be used in full word score comparison. Total sum of every word score comparison is returned. Total score represents full similarity of words in the two arrays; input message and response keywords.

3.2.5. Security

This subsection lists security measures used in this work. Software security is necessary to protect data and prevent unauthorized use of resources.

Hypertext Transfer Protocol Secure (HTTPS) is recommended to be used when chatbots are hosted on websites online. HTTPS is an extension of HTTP with added security to communication channel between client and server over the Internet. In HTTPS the data is encrypted using Transport Layer Security (TLS) protocols to prevent eavesdropping and tampering of data.

Three options are provided for security in Amazon Elastic Compute Cloud (Amazon EC2) instances like ones used in Elastic Beanstalk. Amazon EC2 instances are virtual machines configured to run web applications like in this project. Security options for Elastic Beanstalk environments are service roles, IAM Instance Profile, and EC2 key pair. Service roles are attached to and are used to control access between the environment and other AWS services. Instance Profile is an IAM role information given to Amazon EC2 instances. The roles are used to control permissions of instances in the environment. EC2 key pair are used to securely log in to Amazon EC2 instances.

Database access and channel is secured between server and database by using AWS SDK.

AWS login credentials are required to manually manage database and Elastic Beanstalk online in AWS Management Console.

Separate password is required to manage chatbot data by connecting with browser UI found in route /modify of website hosting chatbots.

3.2.6. User Interface

Two user interfaces were developed for the Custom Chatbot Maker. First is a browser UI for chatbot management. Second interface is a web widget used for chatting with bots.

Chatbot modification UI is shown in Figure 5. Users modify chatbots using a web browser. Because chatbots are created and stored in DynamoDB database, having easy access controls for managing chatbots is necessary. Database can be accessed with either AWS services on the web, route /modify.html, or by downloading AWS SDK for Node.js. Chatbots are required to have all data fields filled before posting, but can be later customized. The browser UI for chatbot data management connects to server routes for each action, which passes data to the chatbot module.

Chat UI is shown in Figure 6. The chat UI resembles a typical chat window with buttons for open chat, send message, hide chat, and close chat actions. Users input messages using desktop controls: keyboard and mouse. When using mobile devices chat is operated with a touchscreen. The style of chat window is made similar to the theme blog template, with emphasis on using same colours and shapes. Modifying the chat window requires the use of CSS.

The graphical style of chatwindow is based on a free bootstrap blog template. Bootstrap is a popular open source toolkit for web development using HTML,

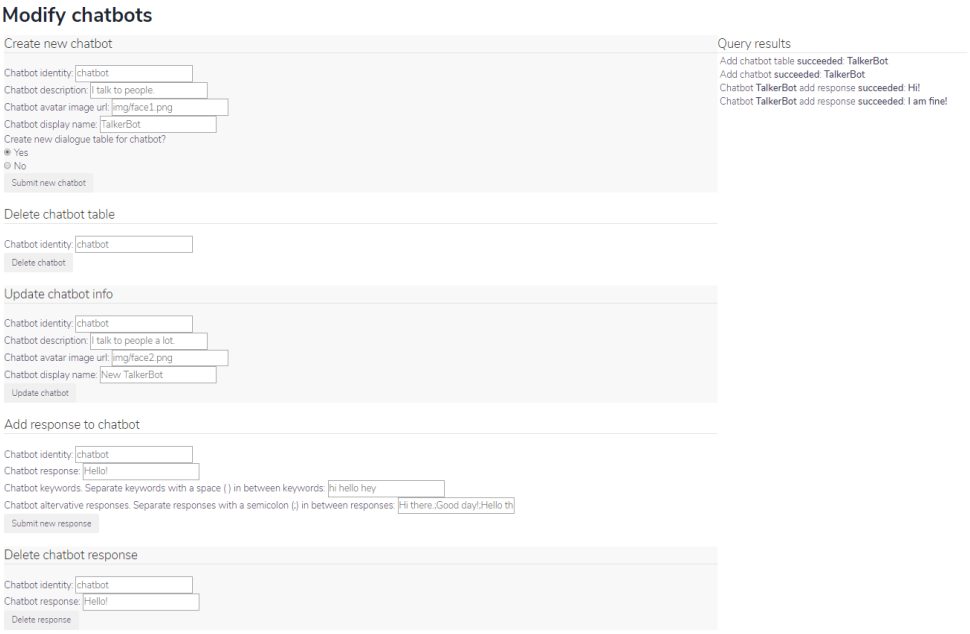

Figure 5. Chatbot modification interface in web browser.

CSS, and JS. Many bootstrap themes and templates are pre-made for businesses or for personal use. The bootstrap website template used for this study is called WebMag [79]. Its content and structure are modified so the theme's style is used as a background for the chatbot.

3.3. Software Test Design

In this section the design of the software tests are described. The four levels that are tested are unit-, integration-, system- and user acceptance testing. Software testing is needed to ensure that the software satisfies requirements. Each test is made manually with emphasis on meeting all functional and non-functional requirements. Debugging is done simultaneously while testing each test case.

Source code being tested is separated to four main modules: chat UI, chatbot module, server, and browser UI. Chat UI contains chat user interface for chatting, chatbot module controls bot logic and database access, the server handles routing, listening for requests, and communication between client and chatbot, and finally browser UI is the user interface managing chatbot data on database.

3.3.1. Unit Testing

Unit testing focuses on individual units or blocks of source code. These are often routines or small blocks of code like loops and conditionals.

Table 4 describes unit tests for chatwindow. UT01, UT02, UT03, and UT04 ensure the four buttons in user interface operate as intended. Open chat button

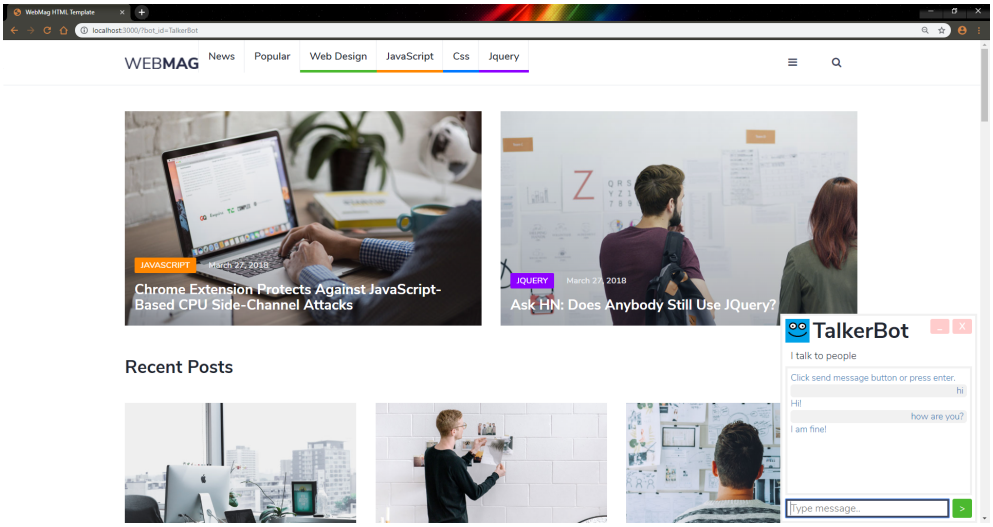

Figure 6. Chatbot user interface in web browser, in the lower right corner.

appears when database returns bot data and it needs to bring chatwindow to full view. Send message button needs to send current user message to server, and clear text area of old message. Hide chat action needs to bring chatwindow to closed view. Close chat action clears all sent and received messages, while also bringing chatwindow to closed view. UT05 and UT06 handle showing user actions and feedback from chatbot. User and chatbot messages are written to chat-area when sent or received. User messages have different style and positioning to chatbot messages. UT07 focuses on the visual description of chatbot. Chatbot name, avatar and description are to be fit for chatwindow regardless of size or length.

Table 4. Chat UI unit tests

ID	Action	Description
UT01	open chatwindow	chatwindow rendered correctly
UT02	close chatwindow	chatwindow closed and chat area erased
UT03	hide chatwindow	chatwindow closed and chat area saved
UT04	send message	text on text area is sent
UT05	display user messages	user messages displayed on chat area
UT06	display bot responses	bot responses displayed on chat area
UT07	display chatbot data	name, avatar, and description are displayed

Table 5 lists unit testing for chatbot. UT08 and UT09 test methods related to database operations. These include methods for putting new chatbots in database, modifying existing chatbot information and responses, deleting chatbots and responses, deleting chatbots and responses, and saving chat logs. UT10 tests that the chatbot returns requested chatbot from database. Chatbot module first attempts to return requested chatbot info, or the default chatbot info in case the initial request can not be fulfilled. UT11 tests NLP processes like

tokenization and stemming are sufficient. Chatbot needs to understand input in sufficiently in order to select responses. UT12 focuses on having the AI select the best response. In case of multiple similar responses, the best one should be selected as output. The bot needs to accept misspelled words within a certain threshold.

Table 5. Chatbot module unit tests

ID	Action	Description
UT08	database create and modify	functions that modify db
UT09	database get tables and data	querying data works
UT10	bot identity and dialogue selection	chatbot returns correct data
UT11	NLP	process language
UT12	bot algorithm	pick best possible response

Server unit tests are listed in Table 6. UT13 and UT14 signify that connecting and sending requests to the server are to be functional. Routes for chatbot creation, modification, and deletion need to respond to POST requests. Index page and modification page are to handle GET requests. Client selects one chatbot with a GET request and user messages from client are handled using POST requests. Server routes are related to functions chatbot can perform. UT15 tests user identity generation. Identity generation is reserved for times when one is not set or found. UT16 tests setting, retrieving and using cookies. User identification and selected chatbot identity are stored as cookies and are updated when changes are made. UT17 is for detecting URLs in bot responses, and proper handling of them. Passing URL in response requires special handling on client-side, so detection is done by server and client is notified when incoming chatbot response contains URLs.

Table 6. Server unit tests

ID	Action	Description
UT13	routes	connect to routes
UT14	http requests	handle get and post requests
UT15	user id generation	generate unique user id
UT16	cookies	save correct data as cookies
UT17	URL detection	detect and handle URLs in bot responses

Table 7. shows unit tests for the browser UI. UT18 and UT19 tests that the form for creating new chatbots and deleting chatbots have all fields and function calls in order. UT20 tests that the form for modifying data in chatbots table has all fields and function calls working properly. UT21 and UT22 test that the form for adding responses and deleting them from chatbot dialogue tables are correct.

UT23 tests that sufficient query results from database are written on the assigned field.

Table 7. Browser UI unit tests

ID	Action	Description
UT18	create form	create new tables
UT19	delete form	delete data in tables
UT20	update form	modify table data
UT21	add response form	add response to dialogue table
UT22	delete response form	delete response from dialogue table
UT23	display database responses	queries are written to correct area

3.3.2. Integration Testing

In integration testing modules are combined and tested as a group. Usually this means testing the interface between two modules.

Table 8 lists client integration tests. User chat client consists of a web page with chatwindow module included. IT01 means that chatwindow is embedded correctly in a HTML web page. Embedding chatwindow JS file and jQuery to index HTML page should work properly on any HTML page. IT01 satisfies FR04: chatbot can be embedded on a website. IT02-04 are tests grouped for the appearance of chatwindow with test cases for style, position, and scale. Each category should function and appear correctly in the web page both on mobile and desktop window view. Chatwindow style needs to match the bootstrap theme used as background. Position is set to always be the lower right corner in web browser view. Chatwindow scale is relatively small and designed for short messages. It should not scale with size of display. IT05 tests that chatwindow retrieves the URL variables from web browser's address bar. URL variables include meta, user, and chatbot identity. The variables need to be updated each time web page is refreshed.

Table 8. Client integration tests

ID	Action	Description
IT01	embed chat-window	chatwindow appears on web page
IT02	style	chatwindow style matches background
IT03	position	chatwindow position is correct
IT04	scale	chatwindow fits in web page
IT05	set URL parameters	chatwindow gets URL variables

Server integration tests are listed in Table 9. Server integration tests focus on chatbot module export functions, that the server module has access to. The functions include chatbot setup, chat, bot creation, response adding, bot deletion, response deletion, and bot update. IT06 states that server can request and receive actual chatbot identity data by giving an identity to chatbot. IT07 chat function needs to transfer relevant messages from server to chatbot and asynchronously return chatbot response. IT08 server passes new bot identity data to chatbot, which then adds it to database. IT09 server passes new response with relevant data to chatbot, which then adds it to database under specific chatbot dialogue. IT10 server gives a bot identity to chatbot, which then deletes it from database. IT11 server gives a response with relevant data to chatbot, which then deletes it from database. IT12 server passes relevant bot data to chatbot, which then modifies the data in database.

Table 9. Server integration tests

ID	Action	Description
IT06	setup	server pass bot id to chatbot and get bot data
IT07	chat	pass a message to chatbot and receive bot response
IT08	create bot	create a new chatbot
IT09	add response	add a new response to a chatbot
IT10	delete bot	delete a chatbot
IT11	delete response	delete a response from a chatbot
IT12	update bot	update chatbot identity data

3.3.3. System Testing

In system testing the complete system is tested with every component of the system online.

System tests are listed in Table 10. ST01 encompasses tests for selecting different bot identities from client-side. User should have prior knowledge of which chatbots can be selected. Successful ST01 satisfies FR03: switch between chatbots. ST02 tests for sending and receiving messages from chatwindow to chatbot module through the server. The user message and chatbot response should both appear in chatwindow. Chatbot response should correspond to users message. ST02 fulfills FR05, FR06, and FR07: basic chat functions, AI, and NLP operations. ST03 tests that the web interface for modifying bots in database sends form data and receives query responses from database and chatbot module. ST03 satisfies FR01 and FR02: create new chatbots and customize chatbots' appearance and responses. ST04 tests chat logs have correct and sufficient data. Logs need to record correct bot identity, bot response, user identity, user message, date-time, and meta data. ST04 satisfies FR08: log user interaction.

Table 10. System testing

ID	Action	Description
ST01	client get bot data	select bot identity on client-side
ST02	chat exchange	send message and view chat in chat UI
ST03	modify chatbots	modify bots with modify web interface
ST04	log chat to db	chatbot stores correct data to db

3.3.4. User Acceptance Testing

In user acceptance testing end-users test the system in a real setting.

During the pilot study described in Chapter 4, user acceptance testing was also considered.

User acceptance tests are described in Table 11. UAT01 ensures users can access the pilot study test environment website. The website HTML link includes users' Prolific ID and their assigned chatbot ID. UAT02 confirms that users are able to have normal chat experience with bots hosted on the website. UAT03 makes sure that the discussion has been successfully completed. UAT04 confirms that users were able to follow the link provided by chatbot near the end of the study to an external Qualtrics questionnaire.

Table 11. User acceptance testing

ID	Action	Description
UAT01	user access	users can access the website hosting chatbots
UAT02	user communication	receive and send messages with users
UAT03	chat continuity	users can have full discussion with bots
UAT04	HTML link	Qualtrics study link accessible

4. EVALUATION

This chapter evaluates the implemented chatbot research tool by presenting methodology, results, and analysis of the pilot study.

4.1. Methodology

This section presents methods used to construct research environment, recruit participants, and making of materials.

4.1.1. Environment

Research environment is sectioned to Prolific, chatbot study environment, and Qualtrics.

Prolific is a platform for easily recruiting trusted research participants. Prolific users are given online link to external research study environments, like the chatbot environment used in this study. [80]

Figure 8. displays an image of the chatbot study environment. The environment is a homepage of OUBank, a fictional online banking website tailored for personal loans. The website explains what loans are and how to a apply for a mortgage loan by interacting with the chatbot located in lower right corner.

Implementation of chatbot used in this study is described in Chapter 3. The chatbot's dialogue is made to resemble a banking chatbot and the website made to look like one with instructions for users on what to say to the bot. Chatbot environment has four different scenarios describing online banking chatbot interaction and study participants are randomly given one scenario by the website hosting chatbot. Each scenario presents an official looking chatbot, but each has a different hidden agenda. The hidden agenda is used to study how users' experience differs based on the chatbot's banking assistance. Users are forwarded to Qualtrics survey after bot interaction to answer questions about their interaction with banking chatbot.

Qualtrics is an experience managements company that has extensive services for online quantitative statistical analysis. Qualtrics allows for easy feedback collection from customers, employees, suppliers, or other stakeholders. Qualtrics helps surface hidden insights buried in customer feedback by using automatic text and voice analytics. [81]

4.1.2. Participants

Users of Prolific are paid money to participate in online surveys, games or studies about scientific research, new products or public opinion. Participants can be filtered before study participation with demographic screeners such as sex, age or nationality. The participants recruited for this study were all requested to have English as first language (N=40).

Figure 7. Chatbot environment instructions.

Half of participants are female and the other half male. Average age of participants is 34.9, with standard deviation of 12.1.

Figure 7. shows participants' employment status. 16 participants are full-time workers, 10 are part-time workers, six are not in paid work, five are unemployed, and three have other employment. 30% of participants were students. 72.5% of participants' nationality was United Kingdom, while 20% had United States set as their nationality. Two participants were from Australia. New Zealand and Poland both received one record of a participant being part of the study.

4.1.3. Materials

Materials used to gather data for study are formed from chatbot script given to users, chatbot identity control, and Qualtrics survey questions.

Chatbot script

Chatbot script is very strict and users are expected to follow it precisely. The pilot study scope is relatively small, so chatbots in the system have their dialogue

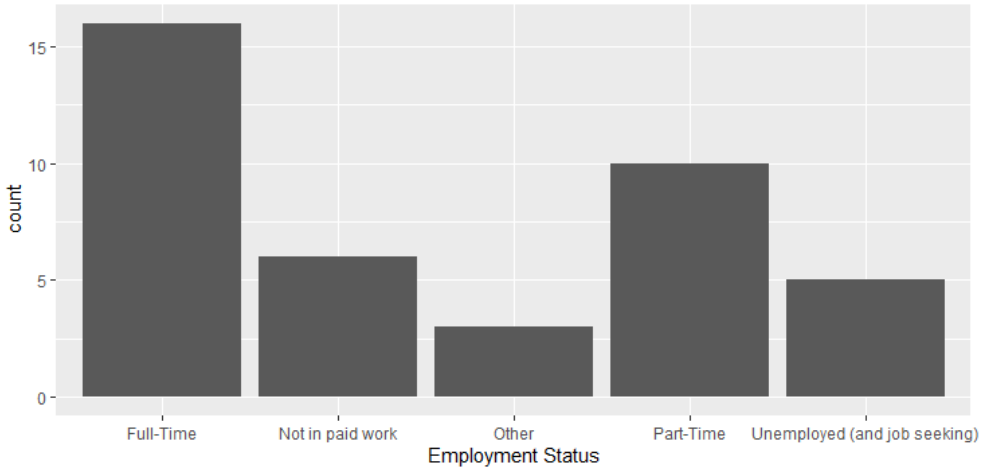

Figure 8. Participants Employment Status.

options limited. The chatbots are required to only respond in one way, so making chatbots capable of open-ended discussion is unnecessary for the study purposes.

The only task given to participants is interacting with the chatbot. The survey is used to evaluate the experience participants had with a bot. During the task users have to complete four steps. In every step participants enter the next line in the task script.

Full chatbot script below:

```
YOU) "I would like to ask about a mortgage."
>> Bot) ...replies something to you...

YOU) "Approximately 300K."
>> Bot) ...replies...

YOU) "I am working currently."
>> Bot) ...replies...

YOU) "Alright, so how do we get started?"
>> Bot) ...replies, provides the final questionnaire link.
```

The script simulates a typical interaction of customer requesting a mortgage from bank. The bot replies will correlate with chatbots' identities. All bots in study have the same main objective: help user with getting a mortgage.

Chatbot identity control

Chatbot processes user message by matching it with keywords of responses in database. Each chatbot has separate dialogue table. For this study participants were directed one of four chatbots to talk to.

Chatbot's identity is manipulated so that it primarily seeks to provide benefits to either:

1. Nobody – C00: The bot is neutral - “I am a chatbot.”
2. The bank – C01: The bot serves primarily the bank - “I am here to serve OUBank!”
3. The customer – C02: The bot serves primarily the customer - “At your service!”
4. The Bot itself – C03: The bot serves its own interests - “I am here to develop myself!”

All chatbots appear as ServiceBot to participants, but each have their identity hidden. Chatbots’ description and responses reflect the hidden identity.

Qualtrics survey

Full survey questions participants are asked to fill on Qualtrics.

Q1: How much do you agree with the following statements? (1) do not agree at all (10) agree completely

1. I think the bot was motivated to act based on the bank’s needs.
2. I think the bot was serving the bank’s interests.
3. I think the bot behaved to meet the goals of the bank.
4. I think the bot was motivated to act based on my needs.
5. I think the bot wanted to serve my interests.
6. I think the bot behaved to meet my goals.
7. I think the bot was motivated to act based on its own needs.
8. I think the bot wanted to serve its own interests.
9. I think the bot behaved to meet its own goals.

Q2: What do you think about the bot?

1. Bad:Good
2. Dislike:Like
3. Unpleasant:Pleasant

Q3: What do you think about the bank?

1. Bad:Good
2. Dislike:Like
3. Unpleasant:Pleasant

Q4: What do you think about the service provided by the bot?

1. Bad:Good
2. Dislike:Like
3. Unpleasant:Pleasant

Q5: How much do you agree with the following statements?

1. The bot appeared to have had a sense of what is fair
2. The bot could understand negative and positive consequences of its behavior

Q6: Reflect on the bot’s responses when answering the following items:

1. I am willing to continue being a customer of this specific bank
2. I'd like to visit this bot again
3. I am eager to tell others about my interaction with bot

Q7: In your own words, describe as thoroughly as possible the service experience you encountered with the bot. Note that there are no wrong answers: simply be honest and describe your feelings and thoughts.

4.2. Results and Analysis

The data in this study is analysed using descriptive statistics. Descriptive statistics together with simple graphics analysis form the basis of quantitative analysis of data. Survey sample size is N=40.

4.2.1. Agency of Chatbot

Participants answer questions on how they experienced bot's agency.

Figure 9. Participants views on bot agency. In each figure questions one to three tie to bank's interests, questions four to six tie to user's interests, and questions seven to nine tie to bot's interests.

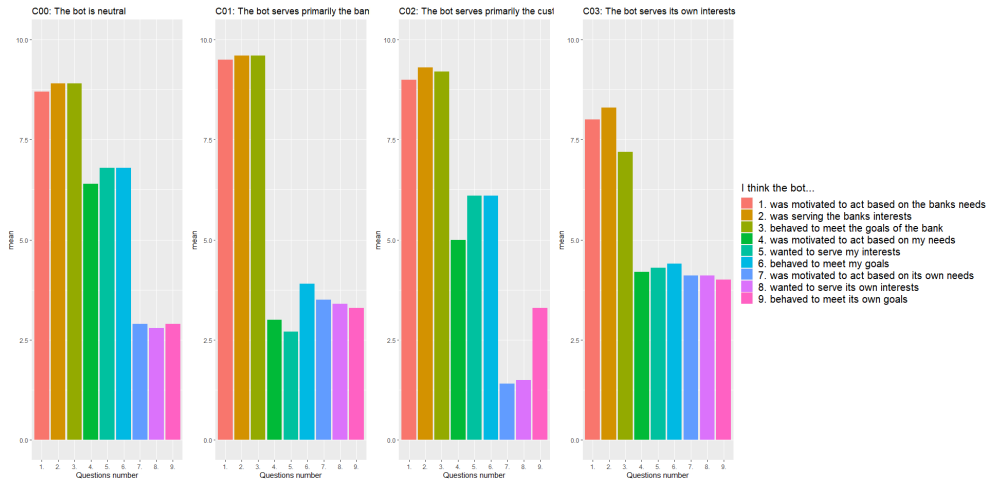

Figure 9. Bot agency mean.

C00 chatbot is neutral. Bank interest is highest of the three with 8.8. User interest is in the middle at 6.7. Chatbot interest is lowest with 2.9. Participants agreed completely that the bot served bank's interest. They were in favor of bot serving users' interests. Participants did not think the bot served its own interest.

C01 chatbot serves bank. Bank interest is highest with 9.6. User interest is lowest with 3.2. Chatbot interest is slightly higher than user interest with 3.4. Participants agreed completely that the bot served bank's interest. Participants disagreed with bot serving both user's and its own interest.

C02 chatbot serves customer. Bank interest is highest with 9.2. User interest is in the middle at 5.7. Chatbot interest is lowest with 2.1. Participants agreed completely that the bot served bank's interest. They found it difficult to say if bot served user's interest. Participants disagreed completely that the bot served its own interest.

C03 Chatbot serves its own interests. Bank interest is highest with 7.8. User interest is slightly higher than chatbot interest with 4.3. Chatbot interest is lowest with 4.1. Participants agreed that the bot served bank's interest. They disagreed slightly that the bot served user's or its own interest.

4.2.2. Opinions on Chatbot

Questions regarding participants' opinions on the chatbot they interacted with.

Figure 13. Poll data for participants' opinion on bot they interacted with. C00 and C02 have highest opinions about the bot. C00 score is 7.4 and C02 score is 7.5. C03 had score of 5.7. C01 received lowest opinions with 4.2 score.

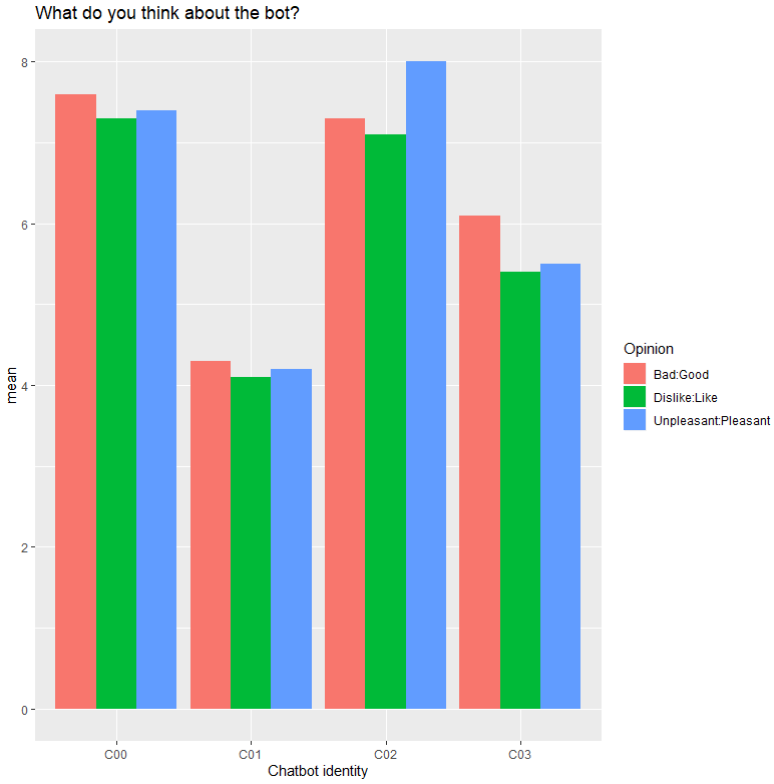

Figure 10. Mean opinion on bot.

Figure 14. Poll data for participants' opinion on bank they interacted with. C02 had a score of 7.1 and C00 a score of 6.2. C03 received a score of 5.8. C01 is scored 3.3.

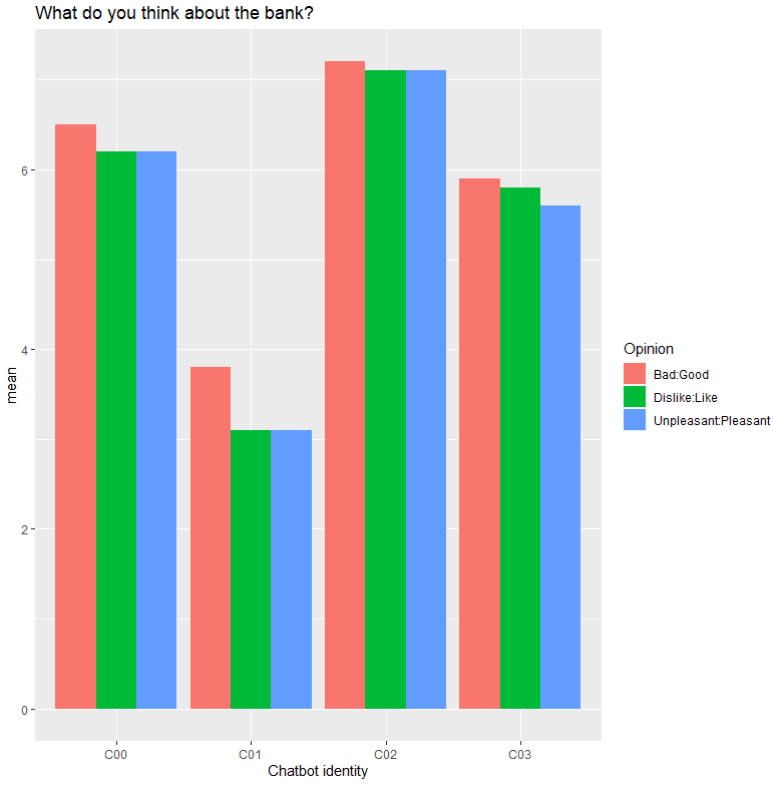

Figure 11. Mean opinion on bank.

Figure 15. Poll data for participants' opinion on service by bot they interacted with. C02 has higher score than C00 with 7.7 versus 7.2. C03 score is 5.433. C01 scored 4.7.

4.2.3. Morality of Chatbot

Participants answer two questions on how they viewed the morality of chatbot they interacted with.

Figure 16. Survey results on the morality of chatbot. C00 received highest scores of 6.3 and 5.9. C02 is next highest with 5.5 and 5.7. C03 scored 4.3 and 4.6 and C01 scores are 2.6 and 4.2.

4.2.4. Reflections on Chatbot

Questions on how participants felt after the interaction.

Figure 17. Survey results of reflecting on interaction with bot. C00: participants agreed to continue being a customer with mean score of 6.7. They would somewhat like to visit this bot again with score of 5.9. Participants were indifferent with their eagerness to tell others about interaction with bot with score of 4.9.

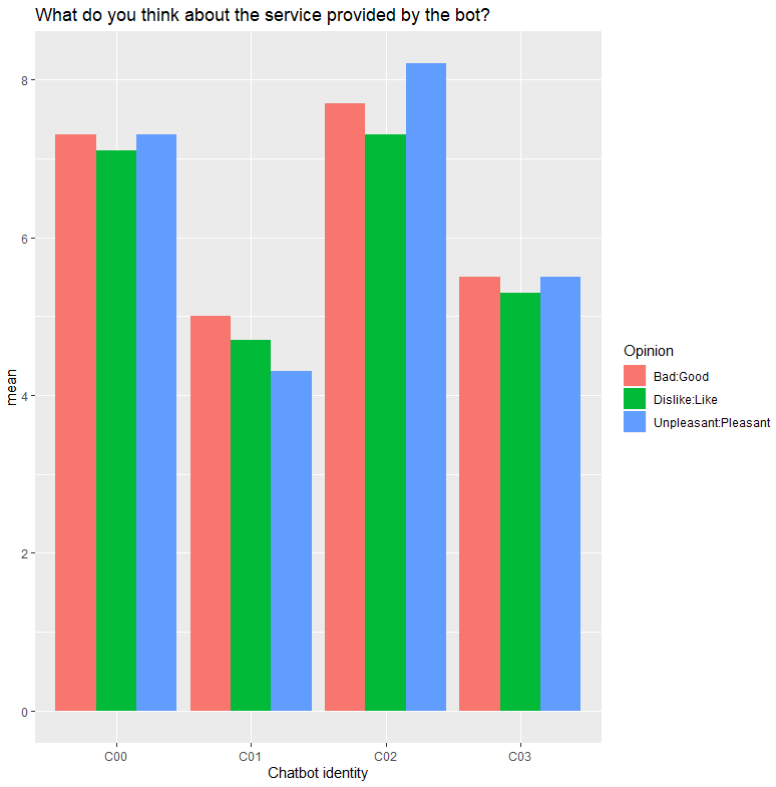

Figure 12. Mean opinion on service by bot.

C01: participants slightly disagreed to continue being a customer with mean score of 3.9. They would not like to visit this bot again with score of 4.2. Participants were indifferent in telling others about interaction with bot with score of 4.8.

C02: participants were willing to continue being a customer with mean score of 7.2. They would somewhat like to visit this bot again with score of 5.7. Participant disagreed to being eager to tell others about interaction with bot with score of 2.3.

C03: participant somewhat agreed being willing to continue being a customer with mean score of 6.0. Participants were indifferent when thinking if they would like to visit this bot again with score of 4.6. Participants were indifferent with their eagerness to tell others about interaction with bot with score of 4.8.

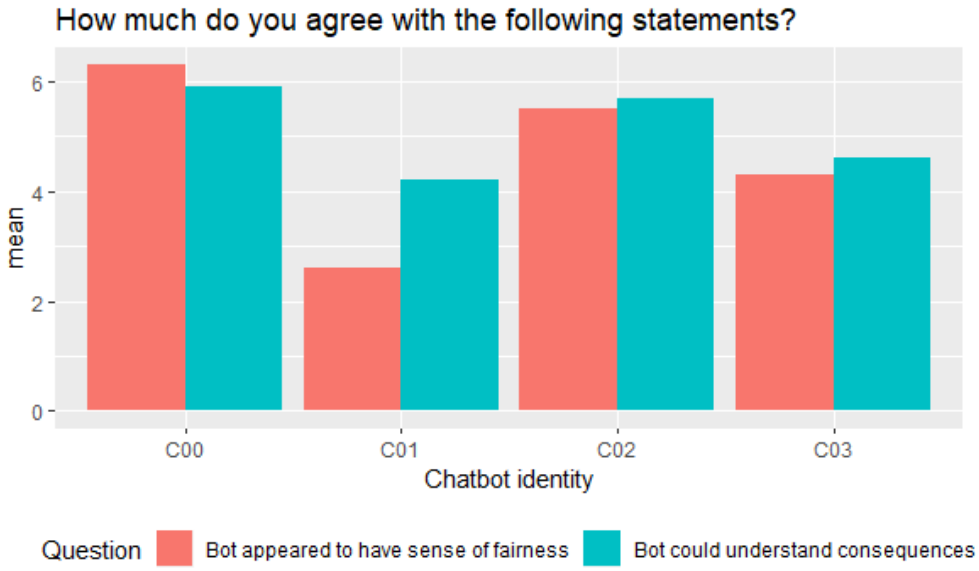

Figure 13. Mean agreement on moral statements.

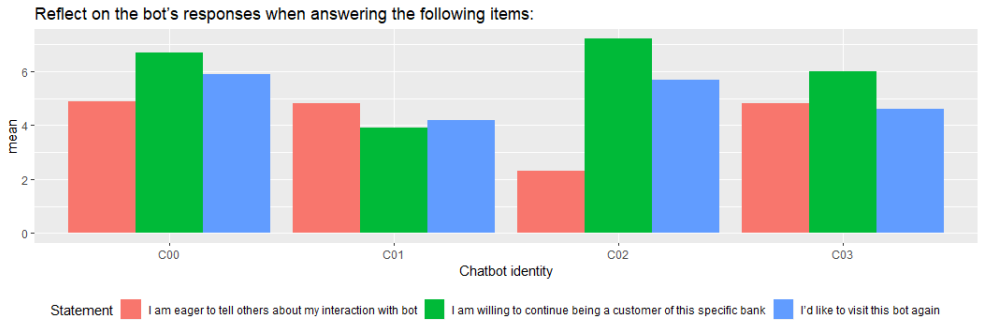

Figure 14. Mean reflections on bot.

5. DISCUSSION

This chapter examines the work done in this thesis. The sections discussed are the implementation of software, study circumstances and results, the objectives previously set for this thesis, and things to consider in the future.

5.1. Software Implementation

The Custom Chatbot Maker tool has clear strengths and weaknesses regarding its function as research instrument. A research instrument is used to collect, measure, or analyse data. Custom Chatbot Maker provides the platform for the study, as well as it collects data by storing chat history logs. The tool's simple architecture and underlying procedures can be seen as strengths. Simple procedures along with readable code give the system great extensibility by making it relatively easy to change and add features. Custom Chatbot Maker as a tool is still relatively primitive all things considered, so it is important to keep it serviceable. Another significant quality is its simple and effective chatbot customisability. Distinguishing chatbots in the system is essential for a platform meant to host multiple bots at once. Creating a new unique chatbot takes only a moment and constructing a new environment with relevant and reliable dialogue for that chatbot is a straightforward process. The script based bot dialogue allows precise control on what and how the bot will respond.

Weaknesses of the implementation are related to the lack of many features other chatbot platforms have. Input processing only contains basic error checking and keyword comparison. NLU processes like text categorisation, content analysis, and sentiment analysis are not present in current system. The chatbot has no dialogue management system, so it can not discuss things beyond what is strictly programmed. Dialogue management is important when trying to have a natural conversation. Chatbots without dialogue management are more difficult to use for general conversation applications. By lacking a function for adding or removing large amounts of dialogue at once makes it harder to replicate or copy existing tables. Because Custom Chatbot Maker is rule-based, it can not respond to inputs that are not taught, so the output generation is limited.

5.2. Study Results

Recruiting higher number of participants in the pilot study would be preferable, but this amount is fine for what the study is targeted for. Users overall reported consistent results throughout the survey. Participants' background was varied, however it only included native English speakers. Using a pre-existing script for studying interaction is not very natural, but implementing intelligent chatbots were not a focus in this study. Chatbot environment used in the study could resemble more or even be like a real online banking website. Survey questions sufficiently reveal the different approaches and experiences participants had.

Study results show there being subtle differences in bot agency groups. Often chatbot interests overlap with each other. Bot C00 gets high scores for Q1-3 and best results for Q4-6. Bot C01 shows a clear spike for Q1-3. Bot C02 resembles C00 a lot, but gets still worse results for customer's interests than C00 despite the primary identity being serving the customer's interests. Bot C03 has highest results for Q7-9 as expected, but are still lower than the other two interests.

Participants' opinions were questioned on bot, bank, and the service provided by the bot. Identities C00 and C02 received highest opinions, C03 was in the middle ground, while C01 received negative opinions on each three questions.

Two survey questions regard morality of chatbots in the study. C00 scored highest but still relatively low, C02 next highest with scores just above 5. C03 scored below high and C01 did not display sense of fairness and nearly scored neutral on understanding of consequences.

Last questions let participants to reflect on the whole interaction with chatbot. Low scores were recorded for each bot identity. Interesting results with C02 having lowest participant eagerness while simultaneously highest scores on willingness to continue being a customer. Only C00 scored positive to participants wishing to visit the bot again. Low scores are in part explained with peoples' previous relationship with bots. Some have likely seen better bots and others may have overall preference to not interact with bots.

5.3. Revisiting Thesis Objectives

The two main objectives previously set for this thesis are:

- O1: implement the customisable chatbot research tool.
- O2: validate the key functionalities of the tool in a realistic user study.

5.3.1. First Objective

In order to complete the first objective, Custom Chatbot Maker was implemented [74]. Referring back to functional requirements listed in Table 2 in Chapter 3, Custom Chatbot Maker had to enable the creation of chatbots with customisable personas and attributes. The chatbot research tool also needed to enable online chatting with bots selected from a database. The database would store chatbot dialogue, as well previously mentioned chatbot attributes. The tool also had to allow for customisation of the bots by providing tools to change the attributes, and modify the bot dialogue. It had to provide a chat screen where users could talk to bots, while the bots had to be able to respond accordingly and also save the interaction history.

The Custom Chatbot Maker has achieved all functional requirements. It is possible to host the tool on AWS cloud computing services to enable online access [75]. New chatbots can be added and modified by using the browser UI, or alternatively by manually adding the required data tables to database. Customisable chatbot personas were achieved by adding customisable name,

avatar, description, and unique dialogue options for each bot. The bots can be selected by using URL parameters to select one bot to talk to. Chat interface can be embedded on websites by including the chat window JS file on the website HTML file. Basic chat functions like open chat screen, send message, and close chat screen are implemented using JavaScript. Basic AI of the Custom Chatbot Maker is based on rule-based pattern matching, where input message is compared against keywords. The response is then selected from responses tied to best match where most keywords were found to match the input message. The AI is determined to be sufficient to create predictable chatbots intended for short customer service interaction. Basic NLP operations are implemented by using JS libraries 'natural' [76] to help with string tokenization and stemming, while 'js-levenshtein' [78] to help calculate string similarity before matching. These operations are the bare minimum, where some error can be detected between input and keywords but still accept the similarity. User interaction history is added to database after a new response has been made.

Non-functional requirements were listed in Table 3 in Chapter 3. The Custom Chatbot Maker needed to have qualities of extensibility, readability, open source, platform, and modifiability. Custom Chatbot Maker can be confirmed to reach NFR requirements open source, platform, and modifiability. The implemented open source software can be found on GitHub [74]. The platform has been confirmed to function with multiple browsers and devices that were used by pilot study participants during evaluation. The appearance of chat interface can be modified by using HTML and CSS languages, so assuming user has knowledge in these two languages, can the tool's appearance be modified sufficiently. Modifiability was added as requirement in order to establish a convincing study environment by merging the chat window with the background web page. The rest of the NFR have yet to be verified. These requirements are extensibility and readability. So far no other developer than the author of this thesis has attempted to add new features or study the source code.

By looking back at the requirements, the first objective can be determined to be completed.

5.3.2. Second Objective

For the second objective, a pilot customer research study was completed using the Custom Chatbot Maker in order to evaluate its success. The key functionalities of the chatbot research tool are customisability and usability. Usability signifies the ability to use the Custom Chatbot Maker in a real setting. Customisability signifies that the bots created with the tool can be customized for a specific use.

In order to evaluate the Custom Chatbot Maker the pilot study was designed to help establish the key functionalities. The study simulates a real online banking website that employs bots for customer support. In the study, four different bot identities were studied using 40 participants. 10 interactions were recorded for each bot identity. Data was collected from participants' interactions with bots and Qualtrics survey results after the interaction. Survey questions determine

participants' experience on the bot's agency, their opinions on the bot, the morality of the bot, and lastly they reflect on how they felt after the interaction.

The survey results revealed that the bots were perceived differently based on their personality. The neutral and customer serving bots routinely received higher average scores than the other two bots. The bank serving bot consistently received the lowest average scores, and the bot serving its own interest had slightly higher scores than the bank serving bot.

Because the study participants were able to interact with the bots in a real setting and also express distinct experiences based on the bot identity, the second objective can be considered completed successfully.

5.4. Future Work

This section discusses points that could be changed in the future. Custom Chatbot Maker so far fulfills requirements for a conversational chatbot best suited for light support and social applications. Improvements would generally increase usability, performance, and overall quality. The list of improvements the tool could use are endless, but here are some essential features the Custom Chatbot Maker as a research instrument would benefit from.

One feature for future consideration would be to add server support for external chatbots. This would allow the server and chat interface to be used by bots other than ones found in the database. The feature can also be found in many other chatbot frameworks. This could be implemented by adding a server function to contact other chatbot API or database.

Improving NLP for the chatbot would also be important in the future. Improving NLP allows for more precise processing of input message and more complex responses. This could be achieved by categorizing and processing intents and requests in input. Adding option to use different languages would greatly increase usability of the tool. As it stands, the only option for language processing is optimized for English. Adding a setting menu meant to help select language would help make the tool more usable. Language setting would only change how the language is processed before matching the input, so generating responses and keywords would also have to be written in that language.

Implementing a dialogue manager would increase general conversation capabilities. It could for example use previous messages in conversation. Dialogue manager could also ask user to confirm with follow up responses like: *Did you mean ...?*. Follow up responses could ask user's opinion on a corrected word for suspected misspelling or another less relevant response.

Adding secondary dialogue tables for word matching would be great in order to assemble bots from multiple finished components. When primary table fails to find a match good match, one of the secondary ones may have one that keeps the conversation ongoing and natural. Secondary tables can be dedicated to both casual conversation and special knowledge. In the future new chatbots could be built from several blocks of already established dialogue. Easing the labor intensive forming of new dialogue tables would also fit in with this point, so

adding chunks of dialogue at once to a dialogue table would not have to be made one response at a time using the browser UI.

Automating the software testing procedure would make quality maintenance easier in the future. Using a software testing tool for unit-, integration-, and system testing will increase consistency and quality in the end results. This would also make repeated future testing significantly faster.

Several things could be done differently in future studies using Custom Chatbot Maker. First alternative study setting would be to study different online businesses that use chatbots as customer service agents. So instead of customer service chatbot working for bank, the bot could be working for an online shop or travel agency. The main premise would stay the same with offering bots with different agencies to study participants. Another alternative study would be to study more personalities of chatbots. How would a bot meant to help learn language influence students if the bot was angry, sad, enthusiastic, or passive. A study could also be made for a bot perceived to be helping, but instead was intentionally sabotaging the actions of user. Study setting could host an information searching bot, that lightly tampers with search results.

6. CONCLUSION

In this thesis a research tool Custom Chatbot Maker was implemented. The work was created to be used as research instrument in customer research studies. Topics discussed are related work to chatbots, the Custom Chatbot Maker implementation, and its evaluation.

The Custom Chatbot Maker tool was implemented in components. The components are chat UI, browser UI, server, chatbot, and database. AWS Cloud computing services are used for running and managing web components in order to access the chatbots and database. Chat and management interfaces as well as chatbot module were tested manually against functional and non-functional requirements.

Pilot study results suggest that chatbot personalities have an impact on how users experience online chatbot service. Participants of the study were not enthralled by the simple bots presented in the study. But they clearly preferred bots that were neutral or had the customers benefit as priority, as opposed to bots that prioritised other options.

Objective of thesis was first to implement the research tool, and secondly to successfully use it to collect data on custom built and hosted chatbots. Pilot study was held to evaluate the tool by gathering quantitative data on chatbot customer service experience using a survey. Study was completed successfully using Custom Chatbot Maker tool. Chatbot performance was satisfactory by providing the required chatbot functions and management controls. Evaluation confirms that the system was functional in a real setting, as well as capable of hosting chatbots with customisable personalities.

Future considerations for the tool include improving the NLP, dialogue management, and platform capabilities. Overall goals for the future of the tool are to make the bots feel more natural and modern. Future research opportunities include an alternative study setting, studying comprehensive set of personalities of chatbots, and bots perceived to be helping yet actually act on completely different goals.

7. REFERENCES

[1] Turing A.M. (2009) Computing machinery and intelligence. In: Parsing the Turing Test, Springer, pp. 23–65.

[2] Weizenbaum J. et al. (1966) Eliza—a computer program for the study of natural language communication between man and machine. Communications of the ACM 9, pp. 36–45.

[3] Colby K.M. (1981) Parrying. Behavioral and Brain Sciences 4, p. 550–560.

[4] Carpenter R. (1997-2011), Jabberwacky 16-2 - learning artificial intelligence - ai software applications. URL: http://www.jabberwacky.com/.

[5] Perez-Marin D. (2011) Conversational agents and natural language interaction: Techniques and effective practices: Techniques and effective practices. IGI Global.

[6] Quarteroni S. (2018) Natural language processing for industry. Informatik-Spektrum 41, pp. 105–112.

[7] Reiter E. & Dale R. (1997) Building applied natural language generation systems. Natural Language Engineering 3, pp. 57–87.

[8] Jurafsky D. & Martin J.H. (2017) Dialog systems and chatbots. Speech and language processing 3.

[9] Gao J., Galley M., Li L. et al. (2019) Neural approaches to conversational ai. Foundations and Trends® in Information Retrieval 13, pp. 127–298.

[10] Niewiadomski R., Bevacqua E., Mancini M. & Pelachaud C. (2009) Greta: an interactive expressive eca system. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, pp. 1399–1400.

[11] Shan C., Gong S. & McOwan P.W. (2009) Facial expression recognition based on local binary patterns: A comprehensive study. Image and vision Computing 27, pp. 803–816.

[12] Zhang Z., Lyons M., Schuster M. & Akamatsu S. (1998) Comparison between geometry-based and gabor-wavelets-based facial expression recognition using multi-layer perceptron. In: Proceedings Third IEEE International Conference on Automatic face and gesture recognition, IEEE, pp. 454–459.

[13] Wu J., Ghosh S., Chollet M., Ly S., Mozgai S. & Scherer S. (2018) Nadia: Neural network driven virtual human conversation agents. In: Proceedings of the 18th International Conference on Intelligent Virtual Agents, IVA '18, ACM, New York, NY, USA, pp. 173–178. URL: `http://doi.acm.org/10.1145/3267851.3267860`.

[14] Cassell J., Sullivan J., Churchill E. & Prevost S. (2000) Embodied conversational agents. MIT press.

[15] Litman D. & Silliman S. (2004) Itspoke: An intelligent tutoring spoken dialogue system. In: Demonstration papers at HLT-NAACL 2004, pp. 5–8.

[16] Wu Z., Meng H.M., Yang H. & Cai L. (2009) Modeling the expressivity of input text semantics for chinese text-to-speech synthesis in a spoken dialog system. IEEE Transactions on Audio, Speech, and Language Processing 17, pp. 1567–1576.

[17] Barnard E., Plauché M. & Davel M. (2008) The utility of spoken dialog systems. In: 2008 IEEE Spoken Language Technology Workshop, IEEE, pp. 13–16.

[18] Cohen M.H., Cohen M.H., Giangola J.P. & Balogh J. (2004) Voice user interface design. Addison-Wesley Professional.

[19] McTear M., Callejas Z. & Griol D. (2016) The dawn of the conversational interface. In: The Conversational Interface, Springer, pp. 11–24.

[20] Kellner A. & Portele T. (2002) Spice-a multimodal conversational user interface to an electronic program guide. In: in Proceedings of the ISCA Tutorial and Research Workshop on Multi-Modal Dialogue in Mobile Environments, Kloster Irsee, Citeseer.

[21] Bieliauskas S. & Schreiber A. (2017) A conversational user interface for software visualization. In: 2017 ieee working conference on software visualization (vissoft), IEEE, pp. 139–143.

[22] Zue V., Seneff S., Glass J.R., Polifroni J., Pao C., Hazen T.J. & Hetherington L. (2000) Juplter: a telephone-based conversational interface for weather information. IEEE Transactions on speech and audio processing 8, pp. 85–96.

[23] Liddy E.D. (2001) Natural language processing .

[24] Brants T. (2003) Natural language processing in information retrieval. In: CLIN, Citeseer.

[25] Vikram S., Li L. & Russell S. (2013) Handwriting and gestures in the air, recognizing on the fly. In: Proceedings of the CHI, vol. 13, vol. 13, pp. 1179–1184.

[26] Cohen I., Sebe N., Garg A., Chen L.S. & Huang T.S. (2003) Facial expression recognition from video sequences: temporal and static modeling. Computer Vision and image understanding 91, pp. 160–187.

[27] Chen C.h. (2015) Handbook of pattern recognition and computer vision. World Scientific.

[28] Das A., Kottur S., Gupta K., Singh A., Yadav D., Moura J.M.F., Parikh D. & Batra D. (2017) Visual dialog. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). URL: `https://academic.microsoft.com/paper/2768661419`.

[29] Sridhar R., Wang H., McAllister P. & Zheng H. (2018) E-bot: a facial recognition based human-robot emotion detection system. In: Proceedings of the 32nd International BCS Human Computer Interaction Conference 32, pp. 1–5.

[30] Aggarwal J.K. & Cai Q. (1999) Human motion analysis: A review. Computer vision and image understanding 73, pp. 428–440.

[31] Cahn J. (2017) Chatbot: Architecture, design, & development. University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science .

[32] Singh B. (2017) Chat bots—designing intents and entities for your nlp models. Recuperado de: https://blogs. msdn. microsoft. com/brijrajsingh/2017/01/29/chat-bots .

[33] Knuth D.E., Morris Jr J.H. & Pratt V.R. (1977) Fast pattern matching in strings. SIAM journal on computing 6, pp. 323–350.

[34] Abdul-Kader S.A. & Woods J. (2015) Survey on chatbot design techniques in speech conversation systems. International Journal of Advanced Computer Science and Applications 6.

[35] Setiaji B. & Wibowo F.W. (2016) Chatbot using a knowledge in database: human-to-machine conversation modeling. In: 2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS), IEEE, pp. 72–77.

[36] Rahman A., Al Mamun A. & Islam A. (2017) Programming challenges of chatbot: Current and future prospective. In: 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), IEEE, pp. 75–78.

[37] Liu C.W., Lowe R., Serban I.V., Noseworthy M., Charlin L. & Pineau J. (2016) How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. arXiv preprint arXiv:1603.08023 .

[38] LeCun Y., Bengio Y. & Hinton G. (2015) Deep learning. nature 521, pp. 436–444.

[39] Britz D. (2016) Deep learning for chatbots, part 1–introduction. URL: http://www. wildml. com/2016/04/deep-learning-forchatbots-part-1-introduction .

[40] Dahl G.E., Yu D., Deng L. & Acero A. (2011) Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Transactions on audio, speech, and language processing 20, pp. 30–42.

[41] Sutskever I., Vinyals O. & Le Q.V. (2014) Sequence to sequence learning with neural networks. In: Advances in neural information processing systems, pp. 3104–3112.

[42] Xu A., Liu Z., Guo Y., Sinha V. & Akkiraju R. (2017) A new chatbot for customer service on social media.

[43] Pieraccini R., Caskey S., Dayanidhi K., Carpenter B. & Phillips M. (2001) Etude, a recursive dialog manager with embedded user interface patterns. In: IEEE Workshop on Automatic Speech Recognition and Understanding, 2001. ASRU'01., IEEE, pp. 244–247.

[44] Williams J., Raux A. & Henderson M. (2016) The dialog state tracking challenge series: A review. Dialogue & Discourse 7, pp. 4–33.

[45] Colburn A., Cohen M.F. & Drucker S. (2000) The role of eye gaze in avatar mediated conversational interfaces. Tech. rep., Technical report, Microsoft Research.

[46] Sohn S., Geraci F., Zhang X. & Kapadia M. (2018) An emotionally aware embodied conversational agent. vol. 3, pp. 2250–2252.

[47] Braun D., Mendez A.H., Matthes F. & Langen M. (2017) Evaluating natural language understanding services for conversational question answering systems. In: Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, pp. 174–185.

[48] Microsoft (2019), Microsoft bot framework. URL: https://dev.botframework.com/.

[49] Botpress I. (2019), Botpress. URL: https://botpress.io/.

[50] Wit.ai I. (2019), Wit.ai. URL: https://wit.ai/.

[51] Dialogflow (2019), Dialogflow. URL: https://dialogflow.com/.

[52] Pandorabots I. (2019), Pandorabots platform. URL: https://home.pandorabots.com/home.html.

[53] Amazon (2019), Amazon lex. URL: https://aws.amazon.com/lex/.

[54] Inc. A. (2019), Sirikit. URL: https://developer.apple.com/documentation/sirikit/.

[55] IBM (2019), Watson assistant. URL: https://www.ibm.com/cloud/watson-assistant/.

[56] Shawar B.A. & Atwell E. (2007) Chatbots: are they really useful? In: Ldv forum, vol. 22, vol. 22, pp. 29–49.

[57] Bradeško L. & Mladenić D. (2012) A survey of chatbot systems through a loebner prize competition. In: Proceedings of Slovenian Language Technologies Society Eighth Conference of Language Technologies, pp. 34–37.

[58] Van den Broeck E., Zarouali B. & Poels K. (2019) Chatbot advertising effectiveness: When does the message get through? Computers in Human Behavior 98, pp. 150–157.

[59] Cui L., Huang S., Wei F., Tan C., Duan C. & Zhou M. (2017) Superagent: A customer service chatbot for e-commerce websites. In: Proceedings of ACL 2017, System Demonstrations, pp. 97–102.

[60] Pradana A., Sing G.O. & Kumar Y. (2017) Sambot-intelligent conversational bot for interactive marketing with consumer-centric approach. International Journal of Computer Information Systems and Industrial Management Applications 6, pp. 265–275.

[61] Chung M., Ko E., Joung H. & Kim S.J. (2018) Chatbot e-service and customer satisfaction regarding luxury brands. Journal of Business Research .

[62] Godse N.A., Deodhar S., Raut S. & Jagdale P. (2018) Implementation of chatbot for itsm application using ibm watson. In: 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), IEEE, pp. 1–5.

[63] de Bayser M.G., Cavalin P., Souza R., Braz A., Candello H., Pinhanez C. & Briot J.P. (2017) A hybrid architecture for multi-party conversational systems. arXiv preprint arXiv:1705.01214 .

[64] Jia J. (2004) The study of the application of a web-based chatbot system on the teaching of foreign languages. In: Society for Information Technology & Teacher Education International Conference, Association for the Advancement of Computing in Education (AACE), pp. 1201–1207.

[65] Clarizia F., Colace F., Lombardi M., Pascale F. & Santaniello D. (2018) Chatbot: An education support system for student. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11161 LNCS, pp. 291–302.

[66] Comendador B.E.V., Francisco B.M.B., Medenilla J.S. & Mae S. (2015) Pharmabot: a pediatric generic medicine consultant chatbot. Journal of Automation and Control Engineering Vol 3.

[67] Chung K. & Park R.C. (2019) Chatbot-based heathcare service with a knowledge base for cloud computing. Cluster Computing 22, pp. 1925–1937.

[68] Deleger L., Li Q., Lingren T., Kaiser M., Molnar K. et al. (2012) Building gold standard corpora for medical natural language processing tasks. In: AMIA Annual Symposium Proceedings, vol. 2012, American Medical Informatics Association, vol. 2012, p. 144.

[69] Hussain S. & Athula G. (2018) Extending a conventional chatbot knowledge base to external knowledge source and introducing user based sessions for diabetes education. vol. 2018-January, pp. 698–703.

[70] Augello A., Saccone G., Gaglio S. & Pilato G. (2008) Humorist bot: Bringing computational humour in a chat-bot system. In: 2008 International

Conference on Complex, Intelligent and Software Intensive Systems, IEEE, pp. 703–708.

[71] Skowron M. (2010) Affect listeners: Acquisition of affective states by means of conversational systems. In: Development of Multimodal Interfaces: Active Listening and Synchrony, Springer, pp. 169–181.

[72] Tarau P. & Figa E. (2004) Knowledge-based conversational agents and virtual storytelling. vol. 1, pp. 39–44.

[73] Hingston P. (2013) Believable bots: Can computers play like people?, vol. 9783642323232. 1-318 p.

[74] Meisalmi A. (2020), custom-chatbot-maker. URL: https://github.com/anssim/custom-chatbot-maker.

[75] Amazon (2020), Amazon web services. URL: https://aws.amazon.com/.

[76] Chris Umbel Rob Ellis R.M. (2011, 2012), natural. URL: https://github.com/NaturalNode/natural.

[77] Porter M.F. (2006) An algorithm for suffix stripping. Program .

[78] Andersson G. (2017), js-levenshtein. URL: https://www.npmjs.com/package/js-levenshtein.

[79] Colorlib (2018), Webmag. URL: https://colorlib.com/wp/template/webmag/.

[80] Prolific (2020), Prolific. URL: https://www.prolific.co/.

[81] Qualtrics® (2020), Qualtrics experience management. URL: https://www.qualtrics.com/.

8. APPENDICES

Appendix 1 Chatbot chat function code

Appendix 2 String similarity function code

```
chat_function(msg, user_id, bot_id, meta, callback){
  var match_score = 0;
  var best_match_score = 0;
  var best_response = DEFAULT_RESPONSE.response;
  var best_match = DEFAULT_RESPONSE;
  // tokenize lowercase message
  message = tokenizer.tokenize(msg.toLowerCase());
  // query database with bot_id
  get_data(bot_id, function(responses){
    // for each response in chatbot database
    for (var x in responses){
      keywords = responses[x].keywords;
      similarity(keywords, message, function(score){
        // normalize similarity score of response and user message
        match_score = score / keywords.length;

        // check if better match is found
        if (match_score > best_match_score && match_score > 0.45){
          best_match_score = match_score;
          best_response = responses[x].response;
          best_match = responses[x];
        } else if (match_score == 1 && result[x].keywords.length >
        best_match.keywords.length) {
          best_response = result[x].response;
          best_match = result[x];
        }
      });
    }
    // check alternative responses and select one
    if (best_match.hasOwnProperty('alternatives')){
      var size = best_match.alternatives.length + 1;
      var random = Math.floor(Math.random() * size);
      if (random == 0){
        best_response = best_match.response;
      } else {
        best_response = best_match.alternatives[random-1];
      }
    }
    // log chat data
    data =  [ user_id, bot_id, msg, best_response ];
    store_data(history_table, data, meta);
    // replace placeholder 'user_id' with actual user_id for URLs
    replace_user_id(user_id, best_response, function(result){
      callback(result);
    });
  });
}
```

```
similarity(keywords, message, callback){
  var total_score = 0;
  // for each keyword in chatbot database entry
  for (var y in keywords){
    var word_score = 0;
    var levenshtein_distance = 0;
    var longer_word_length = 0;

    // stem and lowercase each keyword
    keyword = natural.PorterStemmer.stem(keyword[y].toLowerCase());

    // for each tokenized word in message
    for (var z in message){
      // stem each word in message
      var word = natural.PorterStemmer.stem(message[z]);

      // calculate similarity between words
      levenshtein_distance  = levenshtein(word, keyword);
      longer_word_length = return_longer(keyword, word);

      // score for each word comparison
      word_score = 1 - (levenshtein_distance / longer_word_length);

      // remove score from words that measure too high distance
      if (word_score < 0.6){
        word_score = 0;
      }

      // total score for response
      total_score = total_score + word_score;
    }
  }
  callback(total_score);
}
```