Virtual Sleep Coach
Bachelor Thesis

Irene van der Blij
Kasper Grabarz
Mayke Kloppenburg
Magdalena Simidzioski

Faculty EEMCS, Computer Science

Virtual Sleep Coach
Bachelor Thesis

by

Irene van der Blij
Kasper Grabarz
Mayke Kloppenburg
Magdalena Simidzioski

in partial fulfillment of the requirements for the degree of

Bachelor of Science
in Computer Science

June 26th, 2017

Supervisor:	Dr. ir. W. Brinkman	TU Delft
Bachelor project coordinators:	Dr. O. W. Visser,	TU Delft
	Dr. H. Wang,	TU Delft
Client advisor:	Dhr. dr. J. Lancee	UvA

Delft University of Technology

Preface

This report was written as part of the bachelor thesis at Delft, University of Technology. The assignment came from Willem-Paul Brinkman, who works at the Interactive Intelligence group. The client was Jaap Lancee, an assistant professor at the Department of Clinical Psychology of the University of Amsterdam. The purpose of this project was to explore the options around making a virtual therapist for people with sleep disorders. This therapist was supposed to be able to have an intelligent conversation with the patient using the Socratic method of questioning. The task was to explore the possibilities around helping an insomnia patient with a virtual therapist and if it was feasible to make a virtual therapist that emulates a real therapist.

The target audience for this report are all people who are interested in virtual therapy. Both the computer science aspect and the psychology side of the project will be explained. This report will describe the challenges of the project, the design and implementation of the system and the development process. It will also give background information, information about the testing of the system and a small chapter on the ethical issues of this project. The report also contains elaboration on the design and implementation choices that were made.

The group was helped along by Willem-Paul Brinkman and Jaap Lancee. They helped by specifying the requirements of the project and commenting on our progress and process. Franziska Burger helped us improve our teamwork and process and acted as a co-client. We want to thank all of them for their time and input.

Irene van der Blij
Kasper Grabarz
Mayke Kloppenburg
Magdalena Simidzioski
Delft, June 2017

Contents

Summary **iii**

1 Introduction **1**

2 Problem description and analysis **2**
2.1 Problem description . . . 2
2.2 Requirements analysis . . . 3

3 Research report **4**
3.1 Cognitive Behavioral Therapy . . . 4
3.2 Software Technologies . . . 5

4 Design and implementation **10**
4.1 Design choices . . . 10
4.2 Implementation . . . 14
4.3 Code structure . . . 15

5 Testing **20**
5.1 Unit and end-to-end testing . . . 20
5.2 Use case testing . . . 20
5.3 User testing . . . 20

6 Process **24**
6.1 Development process . . . 24
6.2 Development tools . . . 24
6.3 Software Improvement Group . . . 25

7 Ethical issues **26**

8 Discussion and limitations **27**
8.1 Discussion . . . 27
8.2 Limitations . . . 28

9 Conclusion and recommendations **30**
9.1 Conclusion . . . 30
9.2 Recommendations for future development . . . 31

A Original project description **32**

B Remainder chapters research report **34**

C Cooperation contract **42**

D Software Improvement Group feedback **44**

E Infosheet **45**

Bibliography **47**

Summary

In this project we were asked to answer the question whether it is possible to create a virtual therapist that can help people who suffer from a sleeping disorder. In order to answer this question the goal of the project was to make such a therapist ourselves and explore the possibilities and limitations.

The first two weeks of the project were spent doing research. We looked into cognitive behavioural therapy and existing eHealth solutions for people with sleeping disorders. We also conducted research into existing chat bots, platforms and languages. At the end of the research phase we concluded that machine learning was the best way to make the agent smart, and for that reason we chose to write the code in Python and use rasa NLU as our platform for natural language processing.

During the design phase we stumbled upon the fact that we could not get access to a corpus of conversations between therapists and patients discussing sleep problems. This meant that we could not use machine learning on this corpus, which greatly influenced how smart the agent was going to be. Due to the lack of the corpus we decided to use api.ai instead of rasa NLU. This personalized artificial intelligence was incorporated in our web application, which communicates with the code via our server.

The executable and correct functioning of the software was tested with unit tests, where all modules and functions are tested individually after every new addition. The user interface was tested in a use case evaluation. Use cases are written descriptions of how people should interact with the system and describe how the system should respond to the user's actions. Each use case can be validated by simply performing the action as described in the use case and see if the response of the system is as described in the use case. User tests on the final product were also performed. This gave us insight in how people with no knowledge about the system would interact with the system.

The main purpose of this project was to explore the possibilities and limitations for creating a virtual therapy that treats insomnia. Even though the solution we created cannot yet be used as a standalone therapy, it shows a promising glimpse of the many possibilities to conduct virtual therapy. A project group with more time, money and help from a psychologist could greatly expand upon the project by buying or generating a corpus which can then be used for machine learning and by letting the psychologist model the conversation. The future of virtual therapy has yet to be discovered, but this project already shows an interesting course.

1 Introduction

Sleep problems are nowadays very common [1], and many people suffer from the consequences. They have a hard time participating ordinarily in society, because they cannot get enough sleep or cannot sleep at the right times. In order to cope with sleep problems, people can consult a therapist to find solutions. However, conventional therapy is costly in resources as well as time. Technological advancements might provide a new and better approach to deal with sleep problems.

The Interactive Intelligence group at the University of Technology in Delft is doing research into the effects of using virtual cognitive behavioural therapy (CBT). In this report virtual therapy means that there is an autonomous virtual agent that conducts the therapy. No help from a real therapist is used. The goal of this specific problem was to explore the technological possibilities of challenging patients' beliefs around sleep using the Socratic method. The project group will tackle this by making a virtual coach that will adopt the Socratic method of questioning to treat patients with sleep problems.

Exploring the option of using a virtual therapist to help patients with sleeping problems is worthwhile, since this has some compelling advantages over conventional therapy. The use of a virtual therapist will make therapy very easily accessible. Patients do not have to overcome the obstacle of searching for a therapist or leaving the house to get to one. People with busy lives will be able to do the therapy when they have the time, instead of being bound to an appointment with a therapist. It can also help people who are embarrassed to talk about their (sleep) problems, since the use of a virtual person can eliminate these feelings. Lastly, the virtual coach can greatly reduce the cost for receiving sleep therapy.

The aim of this report is to thoroughly describe the project and give a detailed view of what the project entailed. In the next chapter the problem will be described and analyzed by explaining the project and working out the requirements. In chapter 3 the important chapters of the research report can be found that were made in the first two weeks of the project. It discusses the research done into CBT, existing platforms, chat bots and languages. Chapter 4 elaborates on the design and implementation of the final product. It explains the design choices that were made, what functionalities the final product has and how the code is structured. The next chapter will give information about how the code and product was tested. In chapter 6 the development process will be discussed. This includes information about what methods and tools we used and how our cooperation went, as well as the feedback from Software Improvement Group. Chapter 7 will discuss the ethical issues that concern this project. In chapter 8 the research questions will be answered through a discussion and limitations of the project will be explained. Chapter 9 will contain the conclusion and recommendations for future development.

2 Problem description and analysis

This chapter will firstly explain the problem at hand, introducing multiple sub-problems. In the second section a requirements analysis will be given, which will show the possible functionalities of the final product in order of importance.

2.1. Problem description

The main question of this project was: 'What are the technical possibilities and limitations of implementing a specific CBT technique for insomnia treatment using a virtual therapist?' In order to answer this question, we had to try to make such a therapist ourselves. When doing this, multiple problems arise. Firstly, how do you make the virtual therapist appear intelligent? This is essential for the therapy, since the code behind the virtual therapist should decide what to say next, depending on what the patient just said. This goes hand in hand with generating a conversation that has a nice flow. The patient should feel that the conversation flow is natural and that he or she is understood by the therapist. There is also a whole different side to this project, namely the psychological aspects. In order to make a nicely flowing and intelligent therapist, a lot of knowledge about psychology, and specifically about cognitive behavioural therapy, is needed. It is important to know how a conventional therapist would tackle such a therapy session in order to create a virtual one yourself. The last question that the project team faces concerns the implementation of all the elements named above. What is the best code structure to tackle this problem? Are there existing chat bots, platforms or languages that will be useful during this problem? How do you internally code the conversation? During the project all these questions were answered, and the answers were used to create a virtual therapist that can help people with sleep disorders. The original project description can be found in appendix A.

Overview of questions

Main question: What are the technical possibilities and limitations of implementing a specific CBT technique for insomnia treatment using a virtual therapist?

Sub(-sub)questions:

- How is CBT used to treat insomnia, and is the same possible for a virtual therapist?
- How can you make the virtual therapist appear intelligent?
  - Are there existing eHealth solutions for sleep problems, and how do they do this?
  - What chat bots, platforms and languages exist for creating chat bots, and which of them is most useful for this project?
- How do you implement such a virtual agent?
  - Which programming language fits this project best?
  - How do you internally code the conversation?

2.2. Requirements analysis

In this section the requirements will be described using the MoSCoW (Must have, Should have, Could have, Won't have) method. The Must haves will be in the end result. Without the should haves, the product will still work, but the requirements are desired. Could haves will only be implemented if time allows it. Won't haves will not be implemented, however they may be interesting for future research.

Must haves

- The virtual agent makes use of Cognitive Behavioral Therapy techniques for sleep problems.
- The virtual agent conducts a conversation with the patient adopting a Socratic method of questioning and tries to find out the underlying reason for the sleep problems.
- The virtual agent always responds to the user, even when the user input was incomprehensible.

Should haves

- The patient is able to form a bond or attachment to the virtual agent.
- The user interface is intuitive so that it is clear for the patient how he should interact with it.
- The patient can continue with his conversation when returning to it at another point in time.

Could haves

- The conversation between the virtual coach and the patient is as natural as possible.
- The patient can answer a relapse prevention questionnaire to check the current situation.
- The patient can maintain a sleep diary.
- The patient can set reminders for the bed time or wake time.
- The application gives the possibility of choosing or trying out different therapies (for example SCT or relaxation therapy).
- The virtual agent has appropriate facial expressions according to the user input or own output.
- The virtual agent moves to the background or turns around when the patient has to fulfill a task on his own.
- The patient designs the virtual avatar according to his/her preferences.
- The patient can personalize settings such as choosing different speaking voices, changing the background picture and changing the font size.
- The virtual agent is able to stimulate the patient to continue with the therapy.

Won't haves

- The virtual agent uses speech recognition.
- The virtual agent is able to sense nonverbal behaviour.
- The virtual agent recognizes facial expressions and emotions of the patient.

3 Research report

This chapter contains the report made after the first two weeks of the project. The chapters that are not crucial for the rest of the report/project are removed and can be found in appendix B.

3.1. Cognitive Behavioral Therapy

This chapter gives the necessary background information in relation to cognitive behavioral therapy in general and cognitive behavioral therapy for sleep problems. The chapter will explain the fundamentals of cognitive behavioral therapy.

Cognitive and Behavior Therapy

Cognitive Therapy is a therapy that treats people with psychological problems. The focus of this therapy lies in treating the patient's thoughts. Beck [2] and Ellis [3] are seen as the founders of this new form of psychotherapy. The focus of the therapy lies specifically on the patient's thoughts. Beck [4] defines cognitive therapy '... as a set of operations focused on a patient's cognition's (verbal or pictorial) and on the premises, assumptions, and attitudes underlying these cognition's.

Behavior therapy is a therapy which is also used for treating psychological problems. However, this therapy focuses more on treating the behavior of the patient. Behavior treatment emerged in the beginning of the 20th century with the work of Watson [5] and Jones [6]. This therapy was effective for some neurotic disorders, but it had not much effect on disorders such as depression [7].

Cognitive Behavioral Therapy

Cognitive Behavioral Therapy (CBT) is a form of cognitive therapy merged with behavior therapy. With this therapy not only the patient's thoughts, but also the behavior of the patient is taken into account. [8]. Behavioural strategies are used to confront the patient with 'wrong' thoughts which are in most cases the cause of the dysfunctional behaviour [9–11]. CBT consists of a couple sessions where the patient and the therapist set up an agenda, construct homework assignments, challenge thoughts, adjust homework assignments and summarize the session [12].

Socratic method

One of the basic principles of CBT is the Socratic method [13–16]. This method is used to stimulate critical thinking of an individual by having a dialogue which consists of asking and answering questions [17]. A couple of researchers have tried to come up with a list of questions or guidelines that will help the therapist design the questions. Five basic principles for a successful Socratic dialogue are given by Wells [18]. Guidelines for improving the Socratic questioning method are provided by Padesky [16] who argues that the purpose of this questioning process should be more about discovering instead of only trying to change the patient's thoughts. However it remains very difficult to determine what a Socratic dialogue should look like, but in most cases a systematic way of asking questions is adopted [14].

Medical use of CBT

CBT has been used to treat a number of mental disorders. Among these are anxiety [19, 20], psychosis [21], personality disorders [22], eating disorders [23], depression [24–26], sleeping disorders [27] and many more. In many cases this therapy is proven to be very effective. The focus of this research will lie on the treatment of sleeping disorders with CBT.

3.2. Software Technologies

In this chapter we will present the different software technologies that are available for implementing our virtual agent. We start with a short description of the visual component and then focus the rest of the chapter on conversational software.

Visual component

For the visual component of the virtual agent we have been provided an avatar called TalkingCoach. The avatar has been developed in Unity, a cross-platform game engine, and is made to work through WebGL. WebGL is a JavaScript API used to render 3D and 2D graphics in any compatible web browser. It is possible to interact with the avatar using JavaScript and therefore we can easily interface it with sotware responsible for the conversational component of the virtual agent.

Theory

Even though Natural Language Processing (NLP) and conversational software has been researched since the 1950s, it is in the past two years that these fields have seen a drastic change. Partially due to advances in machine learning technologies, a type of conversational software popularly called 'bots' are taking over from apps. Or as Microsoft CEO Satya Nadella stated: "Bots are the new apps" [28]. In this section we will briefly go over some of the theory that drives conversational software.

Terminology

First we introduce some terminology that we will be using throughout the rest of this paper. This is the same terminology that has previously been used by Radziwill and Benton [29]. We first define conversational agents as software that mimics interaction with people. These are distinguished from other systems such as Interactive Voice Response (IVR) systems. IVR systems are systems which ask customers to press a number on their phone to choose between options when for example calling some big company. IVR systems and conversational agents are examples of dialog systems. We then further distinguish between chatbots and embodied conversational agents (ECA) which are both a type of conversational agent. An overview of this classification can be seen in Figure 2.

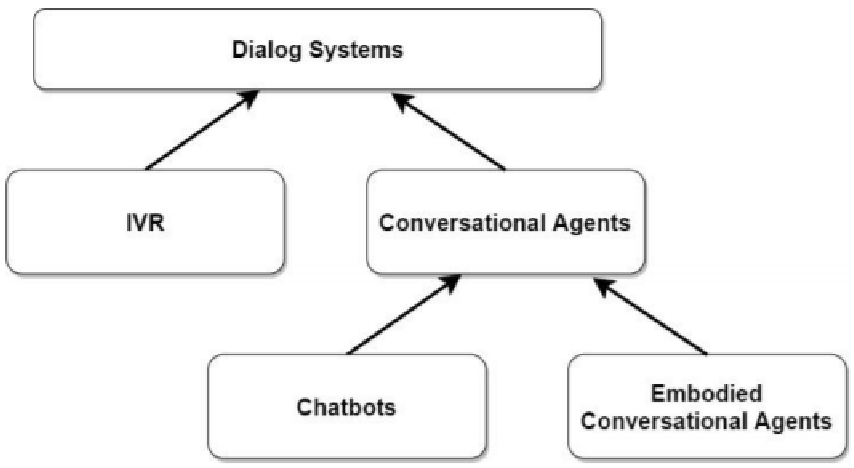


Figure 2. Relationships between classes of software-based dialog systems.

This diagram is presented by Radziwill and Benton [29].

Natural Language Processing

NLP is a part of Computer Science that is concerned with programming computers to process natural language corpora. This computer science field started to develop in the early 1950's with the work of Alan Turing [30] who created the well-known Turing Test used to test the intelligence of a computer. Additionaly, a 'universal grammar' was developed by Noam Chomsky [31] with which he aimed to generate all possible sentences of a given language. Since the introduction of machine learning algorithms for NLP many different programs have been developed.

Dialogue Structure

In order to understand conversational agent algorithms, it is useful to model the structure of a dialogue. In Spoken Language Understanding (SLU) (a field closely related to NLP) a dialogue is modeled with the use of four concepts: intent, context, session and parameters (also known as slots) [32][33]. These concepts are often used by the various platforms for creating conversational agents that we will look at later and so it is useful to briefly go over them.

A session is a sequence of interactions between the user and conversational agent that belong to each other. For example the whole process of booking a flight. Each sentence within a session carries with itself an intent. The intent is the goal that the user wants to achieve. For example: "I want to order a pizza", "I wanna pizza", "May I please have 1 pizza" are all sentences with the same intent. Intent classification or prediction is the process of mapping a sentence to an intent. The parameters of a sentence are that part of the information content of the sentence which can be used to further specify an intent. For example in the sentence: "I would like to book a flight from Paris to Rome", the intent is to book a flight, and the parameters are Paris and Rome. Lastly the context is relevant information saved from previous intra-session sentences. The context is important because it can help the agent to understand poorly written or otherwise ambiguous sentences.

Conversational agent models

There are various ways to classify conversational agent models. One such way is by its response generation method, conversation length and domain [34]. Retrieval based models choose a response from a preexisting repository while generative models generate one from scratch based on the input and some generation algorithm. In retrieval based models responses can be chosen using complex algorithms such as those used in the field of machine learning or simpler ones like pattern matching. With pattern matching responses are triggered by matching pre-declared input patterns. This approach is very popular due to its relative simplicity and can be implemented using popular languages like ChatScript or AIML. A downside of a model based purely on pattern matching is that patterns have to be written manually. This quickly becomes unwieldy if we want patterns to cover many possible sentence variations including grammar mistakes. Models which make use of machine learning (ML) techniques tend to be more flexible and require less hard-coding of rules and responses. However, one requires a firm understanding of ML theory before devising such models. Additionally, systems based on these models need to be trained and this requires a large amount of test data. Because of the time constraints we chose to explore existing platforms for creating conversational agents including those which make use of ML and NLP algorithms. Some of them, like rasa NLU, are open-source while others, like api.ai (bought in 2016 by Google) or wit.ai (bought in 2015 by Facebook), are closed source.

Existing Chat bots

Chat bots are computer programs that are able to conduct a conversation with another chat bot or human. The conversation can be conducted either textually or auditory. Chat bots are mostly used in dialogue systems such as virtual assistants. Nowadays they are extensively used by messaging platforms, apps, websites and e-commerce.

ELIZA

ELIZA is one of the first chat bot programs created by Joseph Weizenbaum [35]. ELIZA was created in order to simulate a Rogerian psychotherapist [36] and was one of the first programs able to pass the Turing Test. However, ELIZA relies on pattern matching and recognizing cue words in a sentence which means that this program has no strong Artificial Intelligence (AI).

PARRY

PARRY is also an early example of a chat bot created by Kenneth Colby [37] that simulates a person with paranoid schizophrenia. PARRY is considered as a more advanced program than ELIZA because it adopts a conversational strategy.

A.L.I.C.E.

A.L.I.C.E. is a chat bot inspired by ELIZA that uses pattern matching to engage in a conversation with a human. A.L.I.C.E. is created by Richard Wallace [38] and is currently one of the best NLP programs. This program has has won the Loebner Prize [39] three times. Unfortunately A.L.I.C.E. is still not able to pass the Turing Test. Because A.L.I.C.E. relies on the same techniques as ELIZA, this program also does not have a strong AI.

Existing platforms

In this section an overview of existing platforms for creating chat bots will be given. Each platform will be described, and the advantages and disadvantages will be discussed. An overview of the advantages and disadvantages of the platforms can be found in table 3.1.

wit.ai

wit.ai is a closed source online platform for developers of chat bots. It was acquired by Facebook in 2015. It is completely free and allows you to create both open and private projects without any limitations. One of the benefits of wit.ai is that it has support for input in many different languages including Dutch and English. In April 2016 Facebook changed from an intent-orientated approach to a story-orientated approach of specifying the bots conversational logic [40]. Stories are example conversations between the bot and the user that also contain the bot's actions. Stories are used to train the ML model after which it is used to predict the bots action each time it receives input. An application sends the input along with optional context by making an API request to a wit.ai server. After that the server responds with the result of the intent prediction performed by the ML model. It is also possible to send voice messages and make use of the wit.ai speech recognition algorithms. It has several tools to aid development and debugging. It has a chat UI which you can use to test your bots logic and an inbox which contain previous sessions. You can then turn these sessions into wit.ai stories. The more stories have been defined, the better the ML model will be trained and the more accurate the bot becomes.

api.ai

api.ai has very similar features to wit.ai but has both a paid and a free version. It was purchased by Google in 2016 [41]. Just like wit.ai, it supports multiple languages inlcuding English and Dutch. Just like with wit.ai a client application sends it a query or an event name after which it uses its machine learning to perform intent prediction [42]. It allows developers to define intents and associate them with certain contexts, parameters, events or example conversations. Besides its machine learning model it uses these associations to chose the most likely intent for a given query. Furthermore it allows developers to control the dialogue flow by specifying intent priorities and contexts. Unlike wit.ai it does not allow free use of speech recognition algorithms. It does allow for easy integration with chat services and other platforms such as Facebook Messenger, Slack or Twitter [43].

rasa NLU

rasa NLU advertizes itself as an open-source tool for intent classification and entity extraction [44]. It has been developed by LASTMILE, a company that makes products for developers and corporations to help them make conversational software [45]. It claims to be a drop-in replacement for platforms like wit.ai, api.ai or Microsoft's LUIS. It is written in Python, but like the other platforms it also has an HTTP API. One of the goals of this platform is preventing chatbot developers from having to re-build their own NLP tools and making the bot community more open and collaborative [46]. It is possible to run it locally or on a private server, making it faster and eliminating privacy concerns. Because it is open-source it can be customized easily and allows for tuning of the ML model parameters. It currently only supports English or German input but it tries to make it easy for developers to add their own language [47].

	Pros	**Cons**
wit.ai	- Completely free - Voice recognition - Good language support	- Closed source - Runs on third party server
api.ai	- Uses pretrained ML models trained on large amount of data - Good language support	- Closed source - Runs on third party server - No voice recognition
rasa NLU	- Open source - Easily tweakable - Can run locally or on a private server - Uses strong open source NLP and ML libraries	- Only supports English and German (languages can be added) - Less popular than the tech giants and so probably harder to find support

Table 3.1: Pros and cons of existing platforms

Existing languages

In this section an overview of existing languages for creating chat bots will be given, as well as an description of each language. An overview of the advantages and disadvantages of each language can be found in table 3.2.

AIML

Artificial Intelligence Markup Language (AIML) [48] is an XML language designed for creating natural language programs. Since AIML is an XML based language, it is easy to learn and implement. AIML can be written in various text editors such as Notepad. Because this language uses pattern matching, the responses of the program are modeled as a decision tree. However, because of the weak pattern matching, this language requires a very large database of responses. As a consequence, the chat bot can return incorrect responses and it becomes very difficult to debug its knowledge base. The most important elements of AIML are tags, categories, patterns and templates. There exist AIML interpreters in a couple of languages among which Java, Python, C#, C^{++} and PHP. A popular chat bot created with this language is A.L.I.C.E. [38, 49, 50].

ChatScript

ChatScript [51] is a language designed for creating chat bots and interactive conversations by Bruce Wilcox. This language is considered powerful because it has a much stronger pattern matching than AIML. However, ChatScript is more complicated than AIML because it is both a scripting language and an engine. The most important elements of ChatScript are rules, patterns, topics and concept sets. The reason why ChatScript has a strong pattern matching is because it is organized in rules and topics. ChatScript first finds the topic and then executes a rule that falls under that topic. Additionally, the ontology of ChatScript consisting of verbs, nouns, adjectives and adverbs is very useful. MongoDB and PostgreSQL are also supported by this program. A popular chat bot that was made using this language is Suzette [49, 50, 52].

RiveScript

RiveScript [53] is a simple language for creating chat bots. This language is based on pattern matching and it is described as a simpler and more easy to use compared to AIML and Chatscript. RiveScript is only designed as a self-contained software library that can be used in already existing code. Additionally, it is considered a very flexible and powerful language which is easy to read and maintain. The most important characteristics of RiveScript are the flexible options to use wild cards, variable matching, arrays and optional words, the use of simplified expressions to match the user's input and dynamic replies. RiveScript is available for a couple of platforms among which JavaScript, C# and Python. A popular chat bot created with this language is Aires Bot [50, 54, 55].

	Pros	**Cons**
AIML	- Simple - Easy to learn and implement - Available for public use	- Weak pattern matching - Difficult to maintain - Needs a very large database for better results - Difficult to learn - No hosting services - Difficult to embed in a web page
Chatscript	- Strong pattern matching - Powerful and flexible - Open source	- Difficult to learn - No hosting services - Difficult to embed in a web page
RiveScript	- Easy to learn and implement - Easy to read and maintain - Open source - Flexible	- Less support available

Table 3.2: Pros and cons of existing languages

4
Design and implementation

In this chapter the design and implementation of the solution will be described. We will elaborate on how we handled our biggest challenge in this project. The design process and the implementation choices that we made will also be discussed and explained. Furthermore we will elaborate on why we chose to use some functionalities but did not make use of other possible functionalities.

4.1. Design choices

In this project we were provided with a basic version of the virtual avatar which had no conversation yet, this limited us in some of the choices. This section will explain what design choices were made and why, by discussing the functionalities present in our final project.

Log in

To allow the user to customize the application and have multiple sessions we implemented a login system. After registering, an account gets created on the server and the user can use his credentials to login. When the user logs in he can use the settings menu to tweak several settings in the Virtual Sleep Coach. The settings menu will be described in more detail in the next section. One of these settings is the storing of chat history. Because there is a login system a user can chose to store the chat history on the server and quit the application mid-session. The user can then come back at his own leisure, start the application and then continue his session. These and other user-dependent settings that might be implemented in the future are all made possible by using a login system.

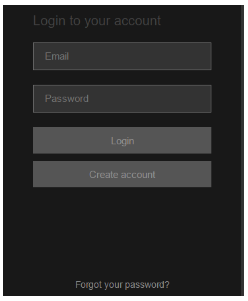


Figure 4.1: Login interface

Settings menu

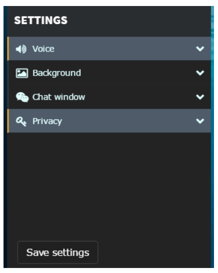


Figure 4.2: Settings menu interface

The settings menu consists of a panel on the left side with several expandable items where the user can choose between several different alternatives for each setting. The user can choose to turn off the voice or change the type of the voice. He can also change the background of the application. We chose to incorporate these features because it is generally known that the environment where the therapy takes place also has an influence on its success. Because we are dealing with digital therapy the user interface can and should be seen as the environment. Another item in the settings menu allows for customizing the chat window. The last item in the menu is an important one. Due to the private nature of the information that is disclosed in the messages the user sends to the server the default setting is not to store it. The last item in the menu allows the user to optionally turn on the storing of chat history so that he can continue his therapy at a later point in time.

Avatar

The avatar for the virtual therapist is made with unity. We made use of an already existing implementation of the avatar which we integrated in our own web application. The avatar moves its head and opens its mouth while talking. It also makes some hand gestures. However the avatar's gestures and its speech could be better synchronized. This will lead to a better and more realistic therapy. Since our primary focus was the flow of the conversation we did not spend much time on the looks and gestures of the avatar. We did look at options to make the voice of the avatar sound more natural. The patient can now choose between two different voices, whichever feels more natural to them. Another option that we considered is to enable the user to change the appearance of the avatar. This would also be a valuable addition to the application and probably improve the effectiveness of the therapy. Unfortunately because of time constraints and the priority level that it had in our project we did not implement it.

Figure 4.3: Virtual avatar

Dialog

The dialog between the patient and the avatar was our biggest challenge. Since the therapist was supposed to treat the patient's disorder, the flow of the conversation had to be very realistic and make the patient aware of his or her situation. CBT-I makes use of the Socratic method of questioning [16]. With this method the therapist should make the patient question his or her own beliefs and realize what is wrong. This way, the patient should stop having the 'wrong' thought which helps the sleeping disorder to only get worse. In order to conduct such a dialog, the virtual therapist had to understand the answers of the patient. To make the therapist smart and able to continue the therapy we made use of a NLP platform, namely api.ai in which we modeled the conversation based on a previous designed conversation tree.

Conversation tree

For creating and editing a conversation between the patient and the therapist we modeled the conversation as a tree structure. In this conversation tree we distinguish 16 different ways the conversation can proceed based on the 16 different cognitions that the patient can have. Each cognition represents a separate branch in the tree. However, because there are many questions that the therapist will ask independent of which specific cognition the conversation is about, the separate branches often come together. Therefore we have converted the conversation tree to a graph. A high-level structure of this graph can be found in figure 4.4. From this graph the therapist can ask about another cognition and get back to the top of the tree. Additionally, the therapist can change the conversation flow when the user input is not clear or not really helpful.

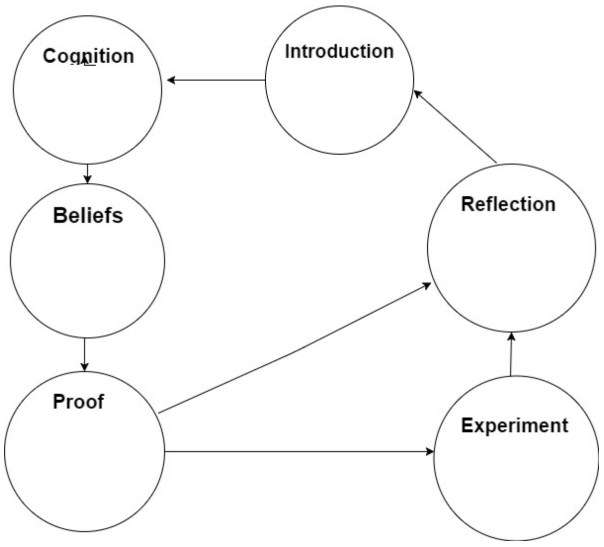


Figure 4.4: Conversation graph

However, the conversation graph is not complete. From all the possible paths that the conversation can go, we have worked out the cognitions up to the point where the therapist asks for an experiment, the results of the experiment are analyzed and the therapy ends. If the patient is not comfortable doing an experiment, then the therapist should be able to propose some other method or let the patient propose something else instead. If neither option works for the patient the therapist should end the session with a proposal about how to further continue dealing with the sleep disorder. For us this was a big challenge because we did not have any model to follow or suitable examples to look at.

Editor

For the purpose of training the api.ai chat bot we have created and editor. With this editor, therapists are able to add conversations to the training set and make the chat bot more sophisticated. The editor creates a custom json file which is accepted by api.ai and which can be uploaded to the intents section and used for training the chat bot. The designer of the conversation can simply add questions that a therapist would ask and possible responses from a patient. For the ease of use the name of the intent file is automatically being generated so that there is no overlap between a new and an existing intent. In the editor, a user can make a full conversation. The editor is written in Java and the for the GUI JavaFX was used. The code is structured according to the Model-View-Controller pattern and has been tested with JUnit. Figure 4.5 depicts a short UML of the program. The interface of the editor can be seen in figure 4.6.

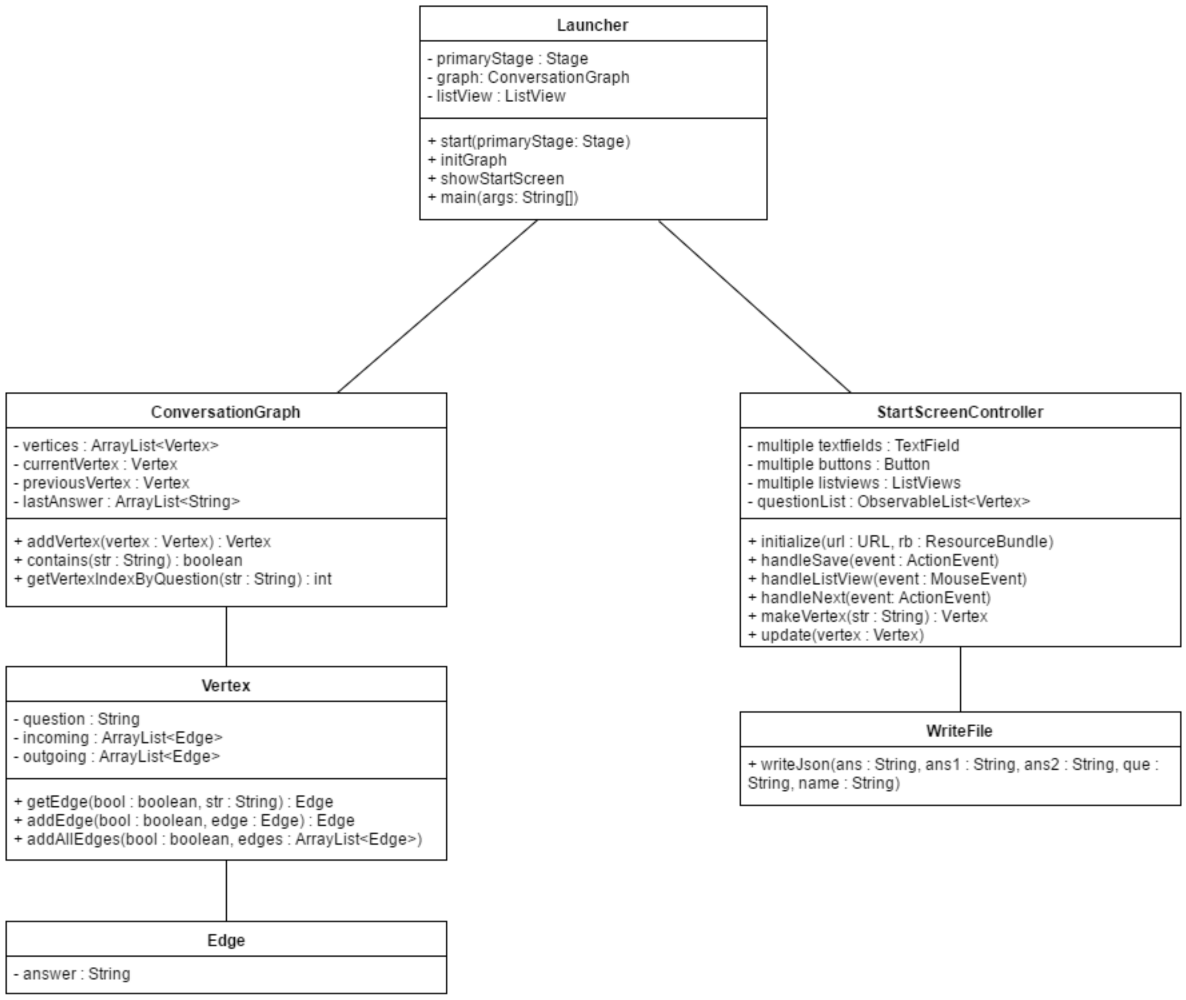
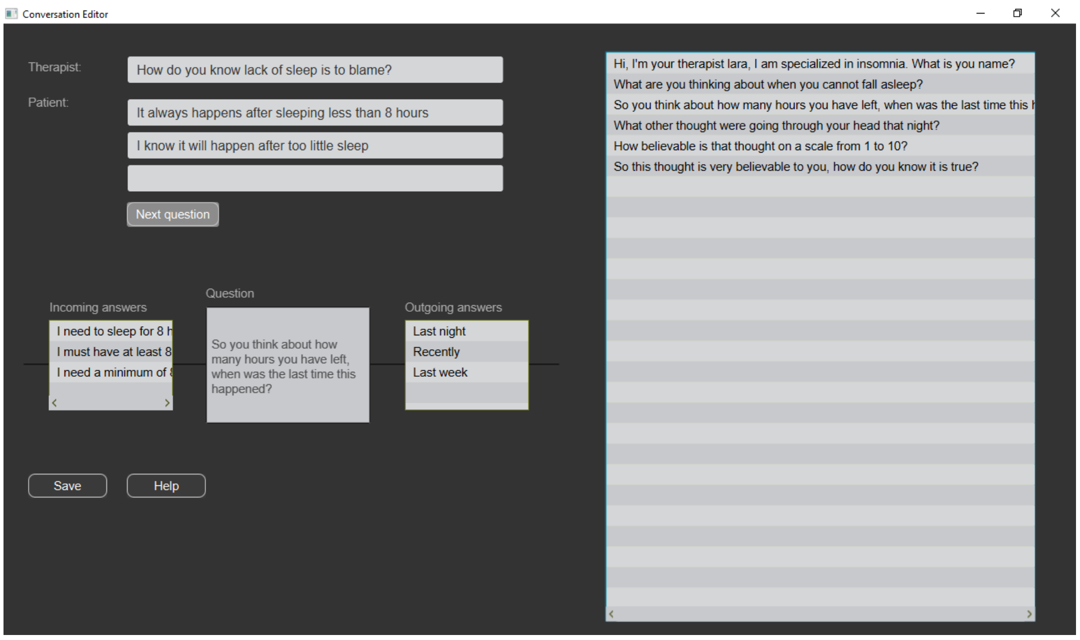

Launcher
- primaryStage : Stage
- graph: ConversationGraph
- listView : ListView
+ start(primaryStage: Stage)
+ initGraph
+ showStartScreen
+ main(args: String[])
ConversationGraph
- vertices : ArrayList<Vertex>
- currentVertex : Vertex
- previousVertex : Vertex
- lastAnswer : ArrayList<String>
+ addVertex(vertex : Vertex) : Vertex
+ contains(str : String) : boolean
+ getVertexIndexByQuestion(str : String) : int
StartScreenController
- multiple textfields : TextField
- multiple buttons : Button
- multiple listviews : ListViews
- questionList : ObservableList<Vertex>
+ initialize(url : URL, rb : ResourceBundle)
+ handleSave(event : ActionEvent)
+ handleListView(event : MouseEvent)
+ handleNext(event: ActionEvent)
+ makeVertex(str : String) : Vertex
+ update(vertex : Vertex)
Vertex
- question : String
- incoming : ArrayList<Edge>
- outgoing : ArrayList<Edge>
+ getEdge(bool : boolean, str : String) : Edge
+ addEdge(bool : boolean, edge : Edge) : Edge
+ addAllEdges(bool : boolean, edges : ArrayList<Edge>)
WriteFile
+ writeJson(ans : String, ans1 : String, ans2 : String, que : String, name : String)
Edge
- answer : String

Figure 4.5: UML of the editor

Socratic questioning

In order to design a virtual conversation we needed an example of a real conversation with a therapist and a patient with a sleep disorder. Unfortunately, getting such an example was more difficult than we imagined. We still needed some kind of instructions on how to design the conversation and what kind of questions to use in the therapy but this was very hard to find. With all the research that we had done


Conversation Editor

Therapist: How do you know lack of sleep is to blame?

Patient: It always happens after sleeping less than 8 hours

I know it will happen after too little sleep

Incoming answers

I need to sleep for 8 h

I must have at least 8

I need a minimum of 8

Question

So you think about how many hours you have left, when was the last time this happened?

Outgoing answers

Last night

Recently

Last week

Hi, I'm your therapist lara, I am specialized in insomnia. What is you name?

What are you thinking about when you cannot fall asleep?

So you think about how many hours you have left, when was the last time this h

What other thought were going through your head that night?

How believable is that thought on a scale from 1 to 10?

So this thought is very believable to you, how do you know it is true?

Figure 4.6: Conversation editor interface

on Insomnia, CBT-I and Socratic questioning and with some instructions of our client we tried to model a conversation ourselves. In this conversation we have come up with a specific order of questions that the therapist should asks the patient. This is represented in figure 4.7. In a conversation with a real therapist this order would probably vary depending on the situation. In our application, with every new session, the order of questions remains the same. This might pose an issue for how realistic an virtual therapist can be. Mainly because a conversation can hardly be the same each time. Because of the order of the questions, the therapist forces the patient to have a very similar conversation each time. This might make the patient think more about how he is talking to a computer and maybe make the patient stop with the therapy. Unfortunately it remains difficult to change the order of the questions during such a conversation as a real therapist could do. However sometimes there is some random order depending on what the patient says. For example when the patient does not know how to respond, the virtual coach may ask for another cognition already at the beginning of the therapy.

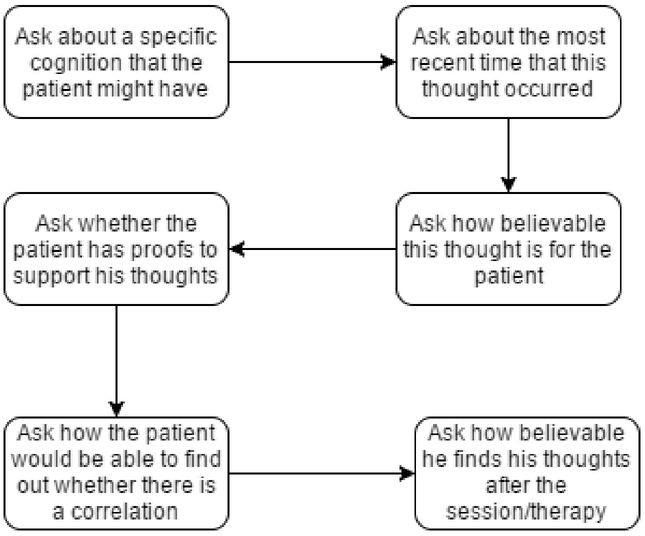

Ask about a specific cognition that the patient might have
Ask about the most recent time that this thought occurred
Ask how believable this thought is for the patient
Ask whether the patient has proofs to support his thoughts
Ask how the patient would be able to find out whether there is a correlation
Ask how believable he finds his thoughts after the session/therapy

Figure 4.7: Socratic model used in the application

4.2. Implementation

This section will discuss the choices that were made with regard to the implementation of the product.

Programming languages

For developing our server application we have chosen to use Python. One of the reasons we made this decision is because it is considered to be a strong and reliable language by a large portion of the web development community. It is used on servers run by big organizations such as Google, Facebook and NASA and has a large and experienced user base making it easy to find support. It is flexible and has virtually no compile times compared to languages such as Java. This will make it more pleasant to develop our application and give us extra time for our project. Compared to PHP, also a commonly used language for server-side development, Python is considered to be more secure due to its better design. Another reason for choosing Python is the existence of many popular frameworks used by researchers in fields such as AI, NLP, ML and general purpose server-side development.

For the editor we have chosen to use the programming language Java. Java is a language that was already well known to all of us, and it is a language that is broadly used to develop enterprise software. The advantage of using Java over most other programming languages is that Java has a lot of build in libraries, which already solve lots of problems common in applications. By using Java, we could also use JavaFX for creating the user interface of the application, which is easy and intuitive.

Use of existing platforms

After our research into existing platforms, we concluded that we wanted to use rasa NLU because this platform gave us the opportunity to tweak the natural language processing algorithm such that it would suit our purpose. We still believe that this is the best way to go, but unfortunately we had to change our course. In order to tweak the natural language processing algorithm we needed a large corpus of conversations between therapists and patients about sleep problems. It was not possible for us to get our hands on such a corpus, and therefore we chose to use api.ai instead of rasa NLU. However, for future development if it becomes possible to use rasa NLU, this could be easily integrated and used together with api.ai.

api.ai [42] is a platform for creating chat bots that understand natural language. The reason that we chose this platform to work with is the ease and functionality that it provides. It is user friendly and it can easily be integrated in a web application. To create a chat bot, api.ai provides intents, entities and contexts. Using these features simultaneously, api.ai enables the designer to create a smart chat bot which learns how to respond to user input by adding new examples. Of the available features we only used intents and contexts to design our therapeutic conversation.

Intents [56] are features which enable the designer to define user inputs and possible responses from the chat bot. For each user input, the chat bot has one or several specific responses. One intent can also hold several user inputs to which the bot has to respond. One intent groups together similar user inputs. This enables the chat bot to give meaningful responses to similar inputs. However, this solution was not ideal in our case because the chat bot does not know at which point of the conversation it is. Another issue were the 'yes', 'no' and 'I don't know' answers. To be able to reply accordingly to one of these inputs, the chat bot had to know what its previous response was. To solve this problem we used another feature that api.ai provides, namely contexts. The conversation can be improved by adding contexts for each intent.

A context enables to chat bot to handle vague inputs and yes/no answers. For each intent an input and output context can be specified. With these contexts the chat bot can keep track where it is in the conversation. The input context specifies which intent should precede the current one, and the output context specifies which intent should be next. Using this feature we were able to make the chat bot 'understand' the input of the user and reply accordingly to continue talking on the same topic.

A tricky part of modeling a conversation with a NLP machine were the 'I don't know' answers. Additionally to using contexts we solved this problem by generating several random responses. With these responses the patient is given time to think about the asked question. If the patient is not able to provide a good answer with which the conversation can proceed, the chat bot asks for another cognition and goes back to the top of the tree. This way the conversation is not stopped but rather continues with another branch in the tree.

Web application

Writing the the Virtual Sleep Coach as a web application has many benefits. Because the client module is written in HTML, CSS and Javascript, any user with a modern browser just has to go to the address of the server to be able to use it. This means the user has to do little effort to be able to use the product and also that it can be easily tested on users during development. Furthermore by making the interface responsive using appropriate CSS rules, users can navigate to the site on their phone making it even easier for them to use the product. Most cross platform issues that might have been there if the product was written in another language are reduced to browser compatibility issues. This can be further mitigated by only writing javascript and CSS code that is widely supported by most commonly used browser versions. Except for the server, deployment issues such as installation of dependencies and the time needed for this are all virtually non existent. Lastly, writing a web application comes with some benefits for the developers too. Creating a user interface in CSS, HTML and Javascript means there are no compile times and code can be easily tested using a modern browser like Chrome which come with built in developer tools. Because we chose python to write our server code, there were no compile times associated with that either. Python itself is a very forgiving language and allows for quick development. Compared to languages like Java or C++ much more can be achieved with fewer lines of code. Because many Artificial Intelligence libraries are available in Python it makes it easy to incorporate them into our product when needed.

Implemented requirements

Our final product is a web application which provides the possibility for the user to take a full therapy. The virtual coach can apply CBT-I with Socratic questioning. In the final version we have implemented all must-have requirements and some of the should and could haves.

From the must-have requirements our application includes them all. The virtual coach is able to conduct a conversation with the patient adopting a Socratic way of questioning. In other words, the virtual coach is able to ask the patient questions, understand the patient's answers and ask the next question accordingly. The virtual coach is also able to continue with the conversation when getting 'I don't know' or incomprehensible answers.

Additional features that we have added are the skip button which enables the user to skip a part where the coach gives an explanation or example. The interface with which the user interacts is intuitive and clear. Another feature that we implemented as a should have is the possibility to save the conversation and continue at a later time. From the could haves we implemented the possibility to change the avatar's voice. Now the user has an option to choose between the default voice and a more natural voice. The patient can also personalize the interface by changing the background and moving text boxes.

4.3. Code structure

For the interested reader, this section will go into the code structure of the server-side as well as the client side and provides insight into the architecture of the system.

Server-side

Because our server-side application was written as a Flask app, there is some code that follows its rules and conventions. This code also contains the entry point and the routes of our web applications and can be found in *server.py*. This file is where the Flask app object itself gets instantiated and where a secret key used for signing session cookies is set. Here is also where the Controller object and Database object get instantiated. The Database class is responsible for any interaction between our application and our database.

The main purpose of the $server.py$ module is to define routes which can be called by HTTP clients to request certain services. One of these is the *authenticate* route. By supplying email and password parameters the HTTP client can try and authenticate with the server using this route. When authenticated, a session gets started and the server checks the database if it should start storing chat history. If the supplied email and password are wrong it returns an HTTP 401 error. The user can register using the *register* route. This route takes an email, password and name parameter and after checking if the email does not already exists, creates a new account. If the required parameters are not present it returns an HTTP 400 error. The *saveUserData* route and *getUserData* are used to store or retrieve the user settings object. Both of these routes first check if the user is logged in and return a suitable message when this is not the case. An HTTP client can log out by accessing the *logout* route. This is done by simply clearing the session variables. The most important route is the *api* route. This is the route through which the main interaction between the avatar representation in the client code and the logic in the backend of the server code happens. Any chat message to and from the client passes through here and, if the user opted for it, this is also where the storing of chat history occurs. The chat messages sent by the client and the responses from the back-end are done by calling the getResponse method on the Controller object. Finally the */* route is used to serve the client application.

For interaction with the Sqlite database we wrote the Database class which can be found in *database.py*. It has a saveUserSettings and getUserSettings to easily save or retrieve the settings object from the database. It takes an email argument to find the correct row in the database. Similar functions are the storeValueForUser and the getValueForUser methods. These can be used if only one value in the database needs to be set or retrieved. To check if a user exists or to insert a new user there are the userExists and insertNewUser methods. To check if a user has a certain password the userHasPassword method with email and password arguments can be used.

In *controller.py* the Controller class is defined. The purpose of the Controller class is to connect the main modules together and where any logic needed for connecting them is implemented. This way even if at a later point more modules are added to the application it remains clear how they interact just by looking at that file. At the same time it becomes easier to allow the classes to interact if most of them live together as members of the same object. The Controller class has a getResponse method which returns the response generated from the input passed as an argument. The method delegates this call to an instance of the Agent class.

The Agent class which is found in *agent.py* is the class responsible for generating the chat responses based on user input. It contains instances of the State class and the ApiAiClient class. The ApiAiClient class is where the interaction with the agent on the api.ai server occurs. It has a getApiResponse method where an ApiAI object gets instantiated. The constructor of this object takes a client access token. This token is used by the api.ai server to choose the agent to send the user input to. The ai object then gets used to construct a request object. This object is then used to send the user input to the *api.ai* agent. The response then gets used to form an object which gets returned by the Agent. This object subsequently gets sent to the client application for processing and presentation.

For delivering multiple choice output there is the State class defined in the $state.py$ file. This class takes a Graph object for which the class is defined in $graph.py$. This class represents a conversation graph which consists of nodes defined by the Node class. The nodes of the graph get created after the graph reads the $convo.txt$ file where the nodes are defined in their textual representation. The Node class can be found in $node.py$. A node represents the state of the conversation and the paths from those nodes represent the different type of directions the conversation can take. A node consists of an id used to identify the node, the text field which holds a textual name that $api.ai$ sends with each response, and paths which depending on the user input lead to the next state of the conversation. When the agent calls the $setState()$ method on the State object, it uses the text field of the node to retrieve the node that represents the current state of the conversation. There is also a response_type field which can be used by the client to visually format the response for presentation to the user. The State class has a response method and a current_node field which holds the Node object representing state of the conversation. When the Agent class detects that the $api.ai$ agent did not understand the user input three times in a row, it calls the getFallBackResponse to generate a response that contains the possible answers the user can give. These answers are based on the last question associated with the conversational state and are read by the Graph from *convo.txt* upon startup.

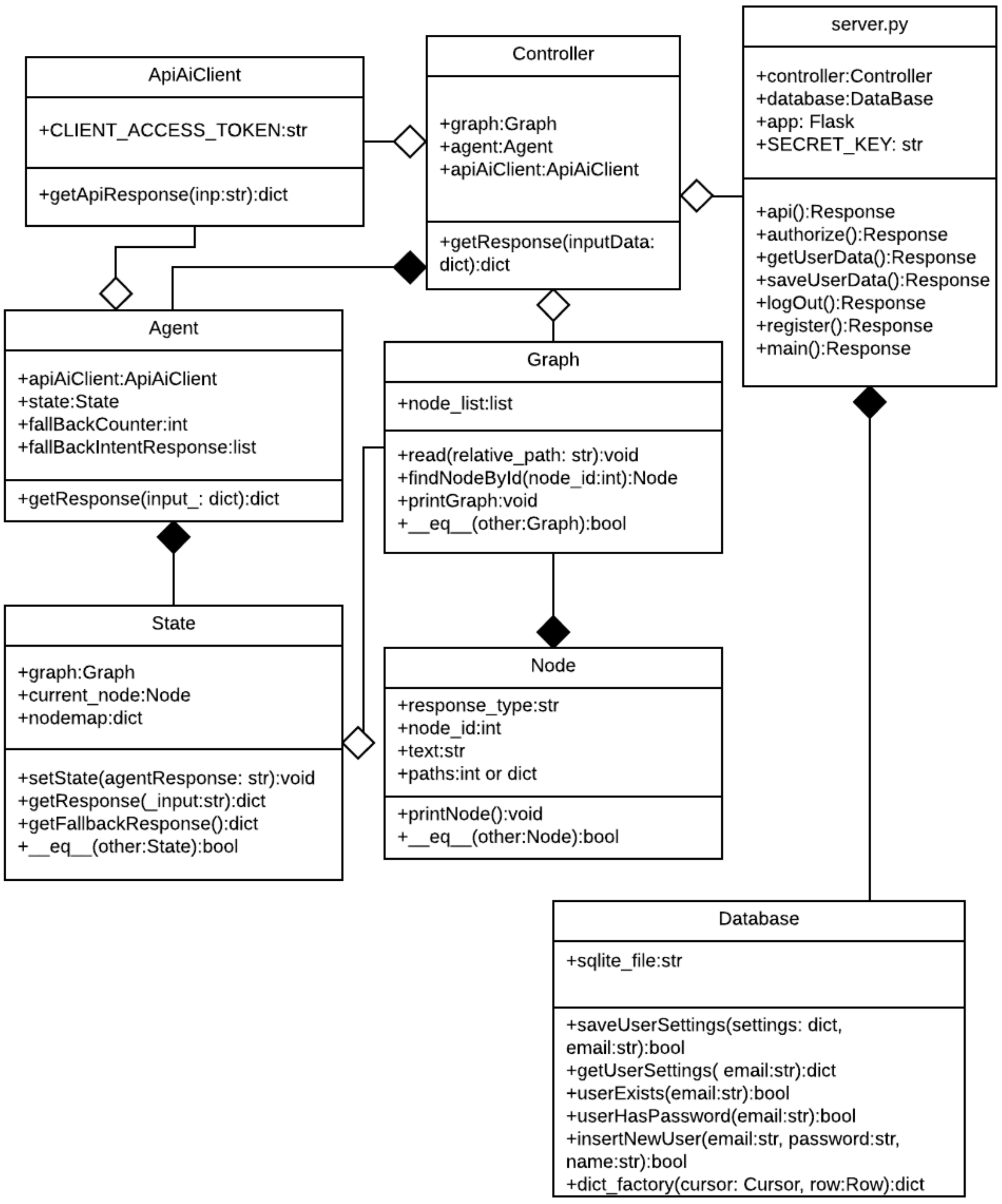


Figure 4.8: Server application structure

Client-side

The client application gets started when the browser opens the *index.html* file. This file contains some basic html markup for the header and footer and markup for the settings menu. Besides that there is the *login.html* file where the user gets directed to and presented with the login form. Most of the client application is written in object-orientated Javascript. Elements in the UI are defined as classes and each class has its own file. The entry point of the javascript code can be found in *main.js*. This file has a main function which gets called when the DOM of the client application has been loaded. In Docking.js there is code that handles the drag and docking logic of the ChatPanel. The styling of the UI is done using css and can be found in the style.css file for the main page and login.css file for the login page.

In the main function the ChatPanel and the StartPanel classes get instantiated. These hold the logic for the corresponding panels visible to the user in the UI. The TextToSpeach object which is responsible for the text-to-speech functionality also gets created here. Furthermore some initial configuration is performed and a call gets made to the server to retrieve the user settings in case the user has previously logged in.

The ChatPanel object has a method called show which gets called when the user presses on the start button of the StartPanel. This object also has a send button which gets the getResponse method as an onClick callback. When the user presses this button the getResponse method gets called with the user input. This method stops the avatar from speaking and if the input is not empty it passes that as an argument to the makeAjaxCall method together with some additional user dependent configuration data and the displayMessage as an onSuccess callback. Upon a successful response the displayMessage method gets called with the response as input. The displayMessage method contains a switch method which, depending on the response type, calls the ChatPanel object displayMessage method with the appropriate arguments. When the response type is of type *Error*, it logs the response to the console. If the response type is unknown it also logs an appropriate message to the console. Lastly, there is a saveUserSettings method which gets called when the user presses the save button in the settings menu. This method makes an AJAX call with the settings object after which the server stores this object in its database if the user has logged in.

The ChatPanel class has fields for its jQuery elements which get set when its $render()$ function gets called. This function also returns the root jQuery element which can then be appended to the DOM by whoever calls this function. The$attachEvents()$ function gives the send button and the skip button their click handlers and also makes sure that when the user presses enter the send click handlers gets triggered. Furthermore there are the $show()$ and $hide()$ functions which show or hide the ChatPanel. Lastly there is a displayMessage function which uses a switch statement to instantiate a chat message with or without choice answers. This function then delegates the actual appending of the message to the DOM to a a method called _displayMessage. The ChatPanel uses the ChatMessage class for simple chat messages and the ChoiceMessage class for messaged with multiple choice answers. The ChoiceMessage uses the ChoiceList class that contains an array of choices and logic used by the ChoiceMessage class when a user clicks on an answer.

For our text to speech logic we have designed a TextToSpeech class. This class internally makes use of either the responsiveVoice or speechSynthesis objects. These are external objects which have a slightly different API and a different voice type. Depending on which voice type the user chooses in the settings menu one of these get used. The class is written in such way to abstract away the differences in their API.

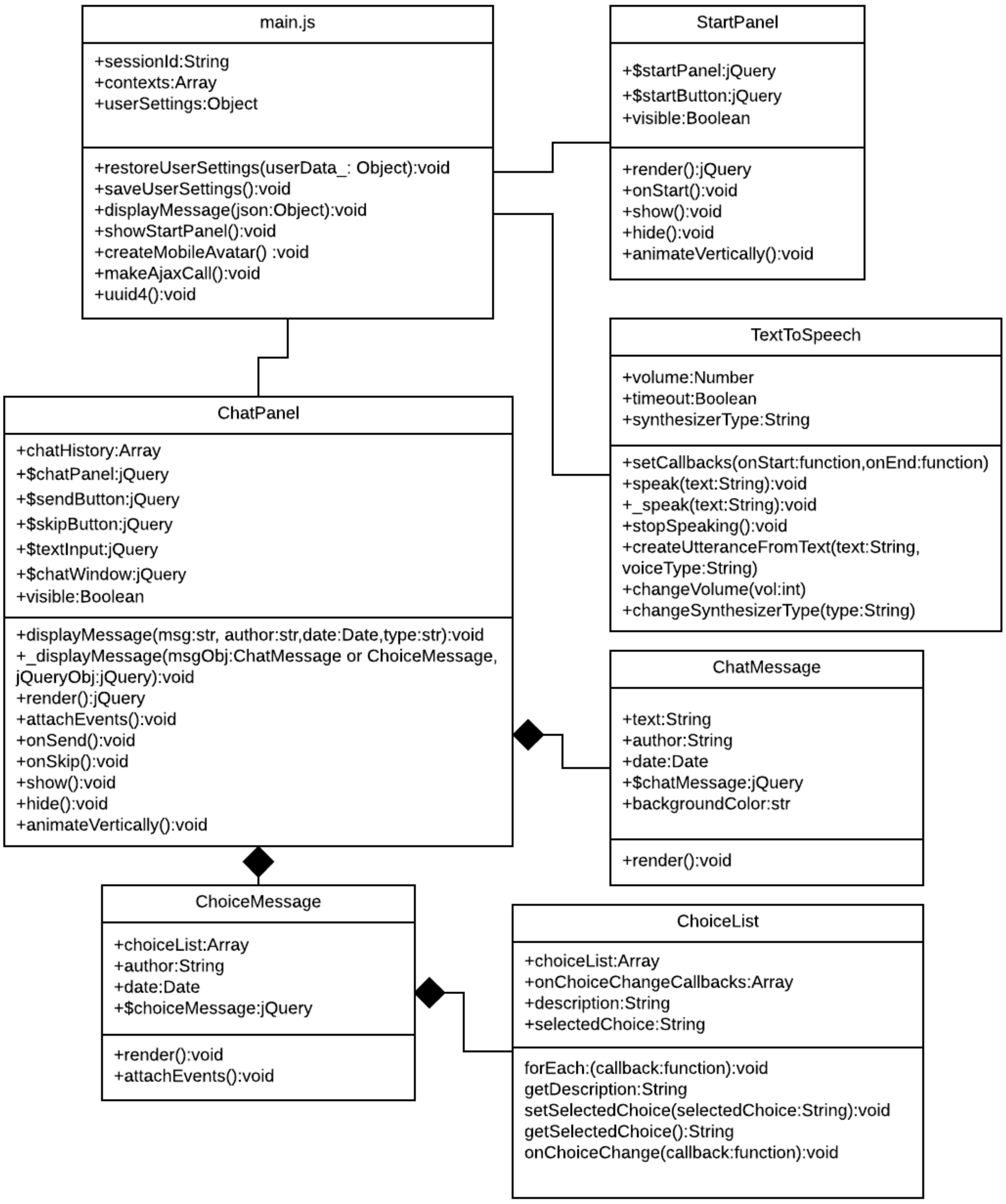


Figure 4.9: Client application structure

5 Testing

In this chapter we will describe how we tested our code and product.

5.1. Unit and end-to-end testing

The editor is tested with the JUnit framework and the virtual coach with the PyUnit framework. The tests are mainly unit tests. Unit tests are classes that test one class with a minimum of referring to other classes. This can be achieved with mocking. The Mockito framework instantiates mock objects that can be used instead of an actual object. This ensures that only one class is tested. The editor also has end-to-end test. An end-to-end test simulates starting the program and performing multiple actions to see if the program responds correctly.

5.2. Use case testing

The user interface was tested by setting up use cases. Use cases are written descriptions of how people should interact with the system and describe how the system should respond to the user's actions. Each use case can be validated by simply performing the action as described in the use case and see if the response of the system is as described in the use case.

Figure 5.1 shows a high level use case diagram. Many use cases can already be derived from this diagram, such as: when the patient is in the state 'Sign in page' and enters correct credentials, the patient should be sent to the 'Welcome page logged in'. The interface change of this use case is shown in figure 5.2 The use case diagram is a high level use case diagram since it does not show all the use cases. For example from the arrow 'Edit settings' in the state 'Session' an entire new use case diagram could be made, showing all the possible settings and what should happen when each of them is clicked.

5.3. User testing

For the purpose of evaluating our final product we organized a testing day where we asked a couple of students to chat with our virtual therapist. Of course, not all of them had a sleep disorder, so in order to start the conversation and give them an idea about how to talk to the therapist we first explained what the purpose of our application was. We also gave them a couple of examples and possible 'wrong' cognitions that a real patient would have and asked them to take up the role of someone who suffers from insomnia.

The examples given in figure 5.3 will give a better picture of how the therapist works during a user test. The first picture shows how the therapist starts the conversation with the patient. The second picture shows what our test person answered and what the therapist's response was for the given answer.

The results of the user tests varied. However these results are certainly biased because we have not had the chance to test our application with real patients. Some students were better in acting like they had insomnia and some had more difficulty with this. The main obstacle was giving the therapist an input which she would expect and therefore understand. This is of course because we have a very limited data set and each user gave an almost totally different response. At this point we had to give

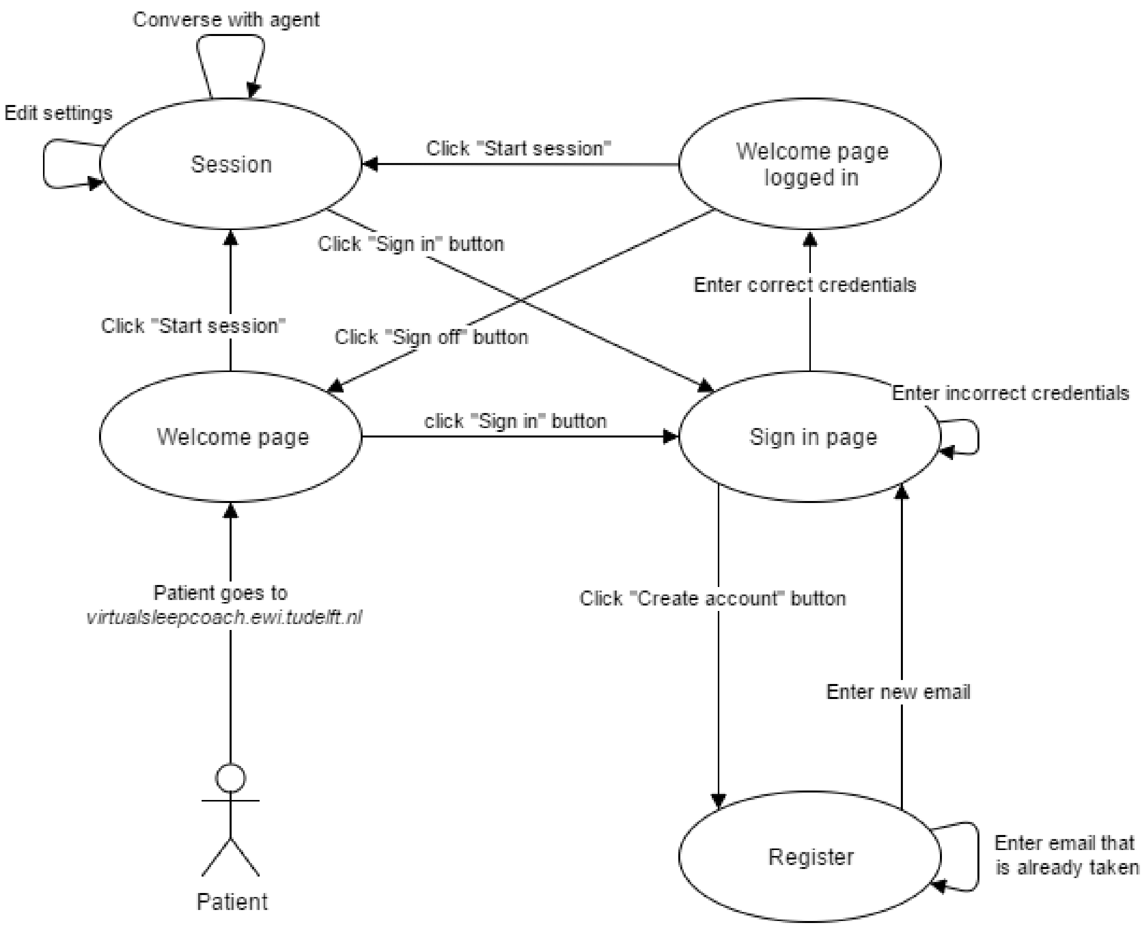


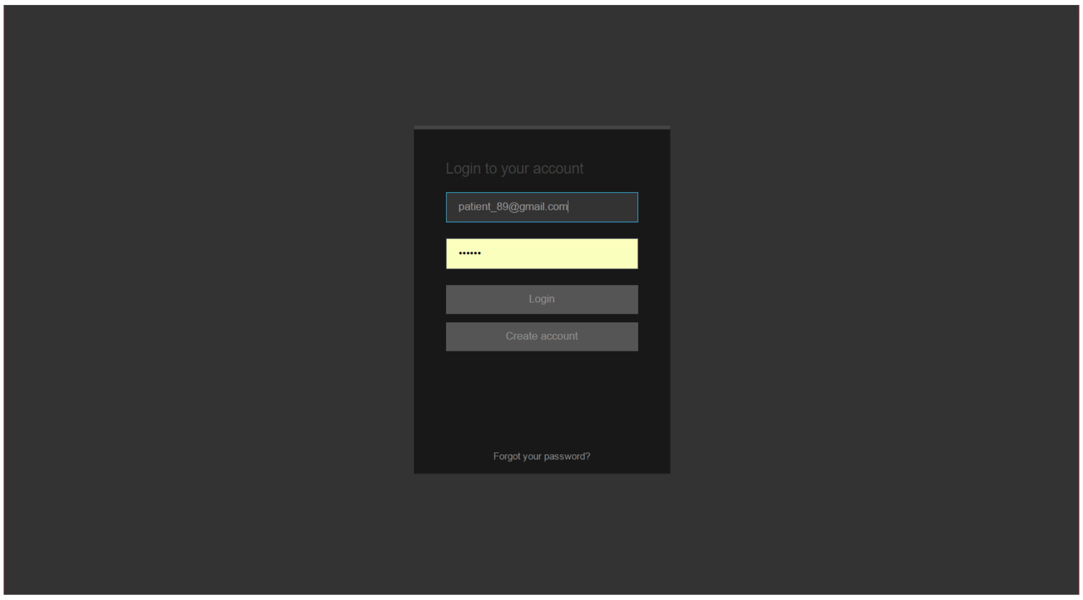
Figure 5.1: Use case diagram

the users more instructions in order to be able to continue with the test. Another problem that many stumbled upon were the 'I don't know' answers which the users replied with a lot during the session. Many times this caused the conversation to go in a loop where the therapist would constantly asks for a new cognition.

The overall grade that was given to our virtual therapist was a 7. The users ratings were as follows:

- Appearance - 8
- Natural feeling - 6
- Conversation flow - 5
- Settings menu - 7
- User Interface - 8.5

The users liked the appearance of our female avatar and also the gestures that she makes while talking. However, they did not have the feeling that they were talking to a real therapist, mainly because she did not understand all of their broad inputs. This is also a reason for the grade of the conversation flow. They did like the user interface very much, especially the possibility to move the text box. Also the settings menu, which enabled them to change a background, voice or save the conversation, was liked. The users also said that they would have liked to be able to change the appearance of the avatar.



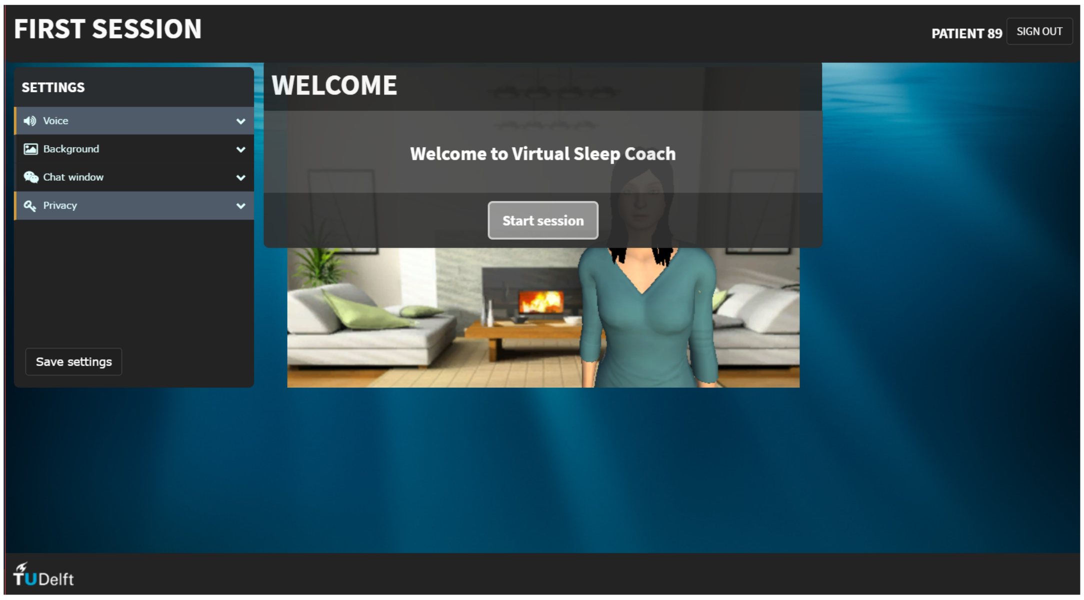
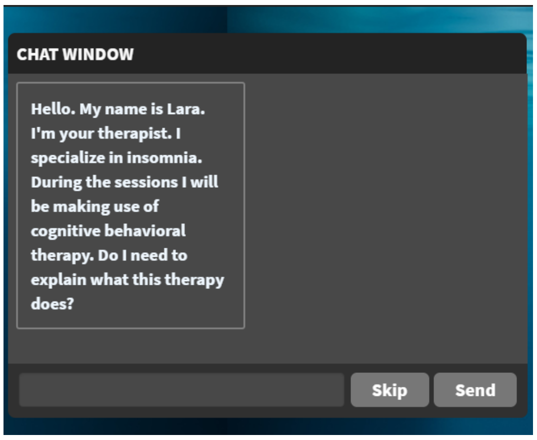
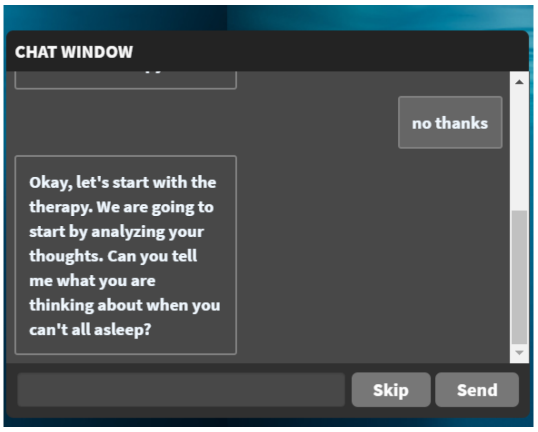


Figure 5.2: The top screen shows the sign in page. When correct credentials are entered, the user is directed to the bottom screen: the welcome page when logged in.


CHAT WINDOW

Hello. My name is Lara. I'm your therapist. I specialize in insomnia. During the sessions I will be making use of cognitive behavioral therapy. Do I need to explain what this therapy does?

(a) user test part 1


CHAT WINDOW

no thanks

Okay, let's start with the therapy. We are going to start by analyzing your thoughts. Can you tell me what you are thinking about when you can't all asleep?

(b) user test part 2

Figure 5.3: User test example input and response

6

Process

In this chapter we will give an overview of our process and the tools that we used in our development process.

6.1. Development process

The process of the project is a very important part. Therefore it is of great importance to choose the right tools to work with. Based on previous experiences and reviews we have chosen the following models and tools that have helped us to finish this project successfully.

Scrum

During the courses and projects in the Computer Science Bachelor we have learned that working with an Agile methodology is very efficient and useful. Some aspects from the Agile methodology that we focused on are having frequent meetings with the coach as well as the client, and having frequent showcase and delivery of a working product. As a team we decided to make use of the Agile framework SCRUM, with a sprint length of one week. With making use of this method we were able to have an organized project with an iterative process. This was very useful, because each week we had a working version of our product and we could make better plans for upgrading with the weekly feedback that we got.

Team process

Teamwork, cooperation and communication are key to making a project successful. In order to make this team process as easy as possible from the get go, we decided to set up a cooperation contract and sign it. The cooperation contract contains conditions about time and organization, and it explains the sanctions when these rules are not followed. The cooperation contract can be found in appendix C. Despite our effort to make the teamwork go as smoothly as possible, we did have some difficulties in the group. Tension arose due to miscommunication between team members. This was talked about in a meeting with a supervisor present to lead the conversation. Conclusion of the meeting was that all team members should communicate more clearly and make sure that it is understood correctly by the other members of the team. After this meeting there were no issues anymore concerning the communication.

6.2. Development tools

In this section an overview of the development tools that we used will be given. It also includes a further elaboration on each tool and the way in which we used it.

Bitbucket

Bitbucket is a hosting service which is used for development of projects that make use of Git. It is a source code management tool, which is very important in a project where multiple people work on the same code. A very useful feature of this tool is that you have the ability to make pull requests.

This allows the fellow developers to first review the code changes made by someone else before they merge into the main branch. Our team made extensive use of pull requests, so that everybody was up-to-date regarding our code.

Taiga.io

Taiga.io is a project management platform for Agile development. We chose this platform because of its simplicity and functionality. We were able to plan and track our process very easily with the tools and boards that were provided. During the weekly sprint planning meeting a task division was made, which was then entered in Taiga.io. Whenever the status of a task changed (either to do, in progress or done), this was altered in this tool. This gave a very clear overview of what still had to be done for the sprint, and what everybody was working on at a certain moment in time.

6.3. Software Improvement Group

As part of the final bachelor project, the code of each group had to be analyzed twice by the Software Improvement Group (SIG). This is a company that performs a static code review to determine whether your code is maintainable. Since the results from the second SIG evaluation will only come in after the report deadline, only the results from the first SIG evaluation will be discussed below.

Feedback

On the SIG maintainability scale our project scored 4 out of 5 stars, which is above average. We did not manage to get the perfect score of 5 out of 5 stars due to a lower score for duplication in our code. For determining the duplication score, SIG looks at the percentage of redundant code, or in other words, code that is present multiple times in the system. Code should only be present once in the project, and if it is used more often, the same code should be called instead of writing it again in another place. In our system duplication can be found in EndPanel, ExplanationPanel and StartPanel. SIG also noticed the presence of test-code in our project and they hope that this will also grow as we add more functionalities to the system. The original feedback received from SIG can be found in appendix D.

Improvements

To increase the maintainability of our system, we looked at the classes EndPanel, ExplanationPanel and StartPanel as SIG suggested. We analyzed and compared the code between the classes and like SIG had noticed, there was some duplication and redundant code present. This code was removed, and we hope that in the next SIG evaluation we will get the perfect score of 5 out of 5 stars.

7
Ethical issues

As is the case with most projects, this assignment raises some ethical questions. This chapter will elaborate on that. It is important to pay attention to the ethical side of things, especially since this project deals with patient data. The patient data and the metadata will be discussed. There are other issues, but the aforementioned two will be focused on.

Patient data

The most important ethical question is in relation to patient data. Important questions are: Who should be able to see the data? What can be done with this data?

In essence, this project aims to imitate the conversation between a patient and a psychologist. This means the same confidentiality should be upheld. Patient data should be very secure and not easily accessible. That answers the first question. The people who are able to see the data should only be the people who would normally have access to it. However, there is an issue with this. Psychologists and other medical professionals who would normally have access are usually not able to solve any technical problems with the program. The confidentiality needs to be taken into account when dealing with technical staff.

The second question seems straightforward if the data is considered comparable to other patient/psychologist data. The data can be used to help the patient in the next therapy session, for example. But this in itself could be a challenge. The programmer will probably need to write the program so it can work with the data. Like in the real world, the patient should be able to give permission to put the information in their patient files. The data should not be sold in any circumstance.

Metadata

Meta data is data that gives information about other data. In this case meta data could be how much time a patient spends on the site and who the patient is. Again the questions ”Who should be able to see the data” and ”What can be done with this data” are relevant. The answer may not be as straightforward as one may think. On the one hand it can be argued that metadata is just as confidential as the actual data and should be treated the same. Importance of metadata should not be underestimated; the knowledge that some person has been in therapy and the duration can also be powerful.

On the other hand, metadata can be viewed as less confidential ad patient data. If it is compared to the real world: someone could keep track of your visits to the medical professional and use this information. Of course, this does require more physical effort and time.

To determine whether metadata is less confidential, it could be useful to ask what the consequences would be if it is treated as not confidential. As mentioned before, metadata can give powerful knowledge. For example, the knowledge of someone suffering from insomnia could help with targeted advertising. This is not desirable and may be even detrimental to the patient. In conclusion, the metadata should also be treated as very confidential to prevent these things from happening.

8 Discussion and limitations

This chapter will first answer the research questions asked at the beginning of the project. Afterwards this chapter will elaborate on the limitations of the project.

8.1. Discussion

This section will discuss our findings during the research and project phase. These findings will be discussed by answering the sub(-sub)questions that were given in chapter 2.

Q1: How is CBT used to treat insomnia, and is the same possible for a virtual therapist?

In CBT behavioural strategies are used to confront the patient with his/her 'wrong' thoughts, which are in most cases the cause of the dysfunctional behaviour. One way of conducting CBT is with the Socratic method. This method is used to stimulate critical thinking of the patient by having a dialogue which consists of asking and answering questions. Not only is this method used to change thoughts, a big part of the questioning process is also about discovering the wrong thoughts.

With natural language processing this Socratic method is also a good option to conduct CBT for a virtual agent. By asking specific questions the virtual agent can find out what the wrong cognition is of the patient, and then challenge that cognition to change it.

Q2: How can you make the virtual therapist appear intelligent?

Intelligence of the virtual agent is an important for the effectiveness of the therapy. A patient should feel that the virtual agent is intelligent enough to understand their problems. This section discusses how intelligence can be achieved.

Are there existing eHealth solutions for sleep problems, and how do they do this?

During our research phase we found quite a lot of applications that were made to help people with sleep problems. The applications that used CBT were very similar in their manner of bringing the therapy and the therapy itself. They ask the patients to fill in questionnaires and sleep diaries and the application gives the patients education about sleep. Unfortunately, none of the applications involved a sophisticated AI therapist.

Since this was the case, we also looked into existing conversational agents. We found some conversational agents targeting various mental disorders, used for teaching and practicing with social situations. However, studies that were conducted on these agents had a very small sample size and were therefore negligible. Unfortunately, we did not learn much about which AI would be best from the existing applications.

What chat bots, platforms and languages exist for creating chat bots, and which of them is most useful for this project?

For existing chat bots we looked into ELIZA, A.L.I.C.E and PARRY. The first two rely on pattern matching and recognizing cue words, which does not produce a very strong AI. PARRY is somewhat more advanced since it adopts a conversational strategy. For existing platforms we looked into wit.ai, api.ai and rasa NLU. The disadvantage of the first two is that they are closed source. rasa NLU is open source, which would mean that we could tweak the machine learning algorithm. For existing languages we considered AIML, ChatScript and RiveScript. AIML had a lot of disadvantages as opposed to the other ones, such as weak pattern matching and the difficulty to maintain. The advantage of RiveScript over ChatScript is that RiveScript is easier to learn.

We decided not to use the scripting languages, because of their limitation to create an intelligent agent. This is because of the simplistic pattern matching algorithms involved in generating responses. In order to make the agent as intelligent as possible, the agent should be able to learn from previous experiences. The constant innovations in the field of machine learning have made creating such an agent a real possibility. This is the reason we have chosen to focus on bot platforms such as api.ai, wit.ai and rasa NLU all of which make use of machine learning algorithms. First we started focusing on rasa NLU, since this allowed us to tweak the machine learning algorithm. But because we did not have access to training data we switched to api.ai.

Q3: How do you implement such a virtual agent?

When you have an idea of what you want to make, you first have to decide how you are going to make it. This section will discuss the questions around the implementation of the virtual agent.

Which programming language fits this project best?

For implementing the logic behind the virtual coach we chose the programming language Python. For Python there exist many popular frameworks that are used for server-side development, NLP and ML. Also this language is used on servers of many large corporations which implicates that there is a lot support available. It is flexible and has virtually no compile times compared to languages such as Java. This will make it more pleasant to develop our application and give us extra time for our project. Compared to PHP, also a commonly used language for server-side development, Python is considered to be more secure due to its better design.

For the editor we have chosen to use the programming language Java. Java is a language that was already well known to all of us, and it is a language that is broadly used to develop enterprise software.The advantage of using Java over most other programming languages is that Java has a lot of build in libraries, which already solve lots of problems common in applications. By using Java, we could also use JavaFX for creating the user interface of the application, which is easy and intuitive.

How do you internally code the conversation?

For coding the conversation we chose to use the chat bot platform api.ai. This platform provided us with enough features to design a conversation between the virtual coach and a possible patient. Before using api.ai we started modelling the conversation in the from of a graph which we read as a text file. This made it possible for us to begin and experiment a bit with the different components of the conversation.

8.2. Limitations

Our task was to implement a virtual therapist that is able to conduct a successful therapy. This included asking questions to a patient and understanding the input that the patient provides. In order to implement such an intelligent therapist we had to make use of Natural Language Processing. A big limitation for us was that we did not have access to a large corpus of conversations between therapists and patients regarding sleep problems. Because of the absence of this corpus, the possibility of using machine learning to process the inputted natural language was eliminated. We believe that using machine learning to process the patients' input would have yielded the best results concerning the correctness of the answer that is given back by the agent. Since we could not use this method to do natural language processing, the correctness of the answers given by the agent is limited.

Since all the team members are computer science students, there was not a lot of knowledge about psychology present. Even though during the research phase of the project we learned a lot about psychology and specifically cognitive behavioural therapy, we can still not say that we are therapists. This turned out to be quite a limitation. Since we could not use machine learning, as described in the paragraph above, we had to make up conversations about sleeping problems between a patient and therapist ourselves, and think of the many answers a patient could give to certain questions. Not being therapists ourselves, and not having (full time) help from a therapist, it was nearly impossible for us to create a psychologically correct dialogue. In the product as it is now the dialogue looking at the psychological correctness could be improved a lot.

Our client mainly wanted us to find out whether it would be possible to create a virtual coach that could help people with sleep problems by using cognitive behavioural therapy. We only had 10 weeks to answer this question. Due to time restriction we decided not to implement dialogue for all possible cognitions for sleep problems. Instead, we focused on only one cognition: 'I need 8 hours of sleep to be able to function properly during the day'. The idea behind choosing only one cognition was that we would still be able to show whether virtual therapy was possible or not, but not have to implement dialogue for every different cognition (which we did not have the time for). Our client agreed that if the therapy would work for one cognition, it would also work for the others. The dialogue for the other cognitions could be implemented in the same way by people who might want to pick up this project after we are finished. The current product is therefore limited to supporting dialogue only concerning one cognition.

9 Conclusion and recommendations

This chapter will provide the conclusion of this report and project. It will also propose ideas for future development to facilitate further improvement of the product.

9.1. Conclusion

During this project we were asked to answer the question whether it is possible to create a virtual therapist that can help people who suffer from a sleeping disorder. In order to answer this question we did a lot of research, and tried to make such a virtual therapist ourselves.

In the research phase of the project we delved into cognitive behavioural therapy; something that we knew nothing about. We also looked at already existing eHealth solutions for sleep problems to see if we could learn or use something from those applications. Besides researching the psychological side of the project, we also did research into existing chat bots, platforms and languages. Based on this research we made design choices for the project, such as which languages and platforms we were going to use.

During the implementation phase we came across some problems which caused us to rethink some of our design choices. The biggest problem that we came across was that we did not have access to a large corpus of conversations between therapists and patients about sleeping problems. Our initial idea was to design a machine learning algorithm that would use this corpus as training data. Since this was no possibility for us due to financial reasons, we changed our platform choice from rasa NLU to api.ai and we had to alter our ideas and expectations that we had about the final product.

Another problem we ran into was due to the psychological nature of the project. As was mentioned earlier, none of us had any prior knowledge about psychology. Even though we learned a lot about psychology and specifically cognitive behavioural therapy, none of us is now able to play the role of a psychologist (that would require years of extra studying). For that reason we decided that it was not feasible for us to model the conversation between the therapist and patient in a way that it would be completely psychologically correct. Also, due to this reason and time constraints, the conversation is quite limited, meaning that the conversation tree is far from complete. Unfortunately, this also made the application harder to test.

The two problems as discussed above have caused that our final product will not solve sleeping problems for most people. The virtual therapist cannot be used as standalone therapy to treat insomnia. The product could, however, be useful as an addition to conventional therapy.

The main purpose of this project was to explore the possibilities and limitations for creating a virtual therapy that treats insomnia. Even though the solution we created cannot yet be used as a standalone therapy, it shows a promising glimpse of the many possibilities to conduct virtual therapy. A project group with more time, money and help from a psychologist could greatly improve the project by buying or generating a corpus which can then be used for machine learning and by letting the psychologist model the conversation. The future of virtual therapy has yet to be discovered, but this project already shows an interesting course.

9.2. Recommendations for future development

As was already discussed in the limitations section, we did not have access to a large corpus of conversations between therapists and patients about sleep problems. This eliminated the possibility to use machine learning for processing the natural language. If a project group was to pick up this project with the intentions of improving it, we would suggest them to either find such a corpus or generate one themselves. We were not able to do that ourselves because of financial reasons. We believe that having such a corpus and using this as training material for machine learning could greatly improve the quality of the answers given by the agent.

If the product were to be kept as it is now, we would recommend to use it as an additional option to conventional therapy with a therapist. As mentioned above, the artificial intelligence of the agent still has potential to be improved, and is right now not able to understand everything the patient might say. Even with improved artificial intelligence it will still be very hard for the agent to understand the patient fully. By keeping a real therapist in play, the patient is still able to discuss issues the agent may not understand or does not give satisfying answers to. A possibility can be that the agent keeps track of patients' input that he did not understand, which is then sent to the real therapist, who can pick up on this in a therapy session.

There are some more relevant improvements that could be made to the application, one of them being the relation that the patient has with the therapist. As mentioned before, this relationship plays an important role in the success of the therapy. Especially when a patient wants to stop with the therapy, this feeling of obligation towards the virtual coach can be crucial. Therefore, this 'relationship' between the therapist and the patient is a valuable feature.

Another improvement that would bring more success to the application is the reading and understanding of facial expressions. With this feature the conversation could be modified to take into account the emotions of the patient and respond accordingly. This would make the virtual therapy a lot more realistic and lift the application to a different level. Also, this feature could help the chat bot establish some sort of relationship with the patient more easily.

A
Original project description

Project description

In this project, we would like to study the possibilities and limitations of equipping a virtual sleep/depression coach with a method (for example, an ontology, a BDI model of the patient, or value-based argumentation) to reason with patients about their false beliefs around sleep problems or depression. The students' tasks would consequently involve identifying a suitable reasoning method, implementing the method, designing the flow of the dialog between the patient and a virtual coach, and interfacing with the virtual coach so that it is able to execute the dialog. If time permits, a small user study could be carried out. We provide a virtual coach implemented in Unity.

Background

A frequent problem in mental illnesses in general and insomnia in particular is that people hold false beliefs about the effects of their behavior, which in turn could even enhance the illness. For example, people with sleep problems may worry that they cannot concentrate on anything if they do not get at least 8 hours of sleep and may, as a result of this worrying, not be able to sleep. A key component of Cognitive Therapy is therefore to challenge these beliefs by asking the patient for evidence (Why do you think you need 8 hours of sleep? Why do you feel you cannot concentrate? Does your work performance actually show this? Has your boss mentioned to you that you are not performing at the expected level? Could you ask your colleagues whether they can confirm this? Etc.)

With the developments in information and communication technology over the last decades, the rise in prevalence of mental illnesses, and the limited capacities to provide face-to-face therapy to all those in need, mental health experts are increasingly turning towards ICT solutions for therapy. Research conducted with kidney patients in our group has found the inclusion of a virtual agent in a weight-loss program to positively affect the outcome.

While many completely automated systems exist for changing people's behavior, few have targeted cognitions in an interactive manner.

Company description

At the Interactive Intelligence group, we are interested in researching and enabling the cooperation between humans and artificial agents. To this end we combine knowledge from psychology and cognitive science on the one hand, and artificial intelligence and computer science on the other hand to generate insight and methods in the fields of human-computer interaction, affective computing, and cognitive engineering. Willem-Paul Brinkman is especially focused on the possibilities of creating digital therapists and coaches for health and wellbeing and studying the various ways in which technology (apps, virtual reality, avatars) can be used to support humans in changing their behavior.

For this project, we draw on expertise gained in the SleepCare project (http://www.ikgalekkerslapen.nl/) and combine it with current research into supporting junior researchers in managing work-related stress (http://ii.tudelft.nl/distressless/).

Auxiliary information

The students can word on-site. We do not pay a stipend. English or Dutch is fine. The group size is not limited. Four students can be accommodated but it is possible/likely that for some tasks the students will then be split into two subgroups of two. For example, two students create an ontology for the domain and two students focus on modeling the user.

B

Remainder chapters research report

Cognitive Behavioural Therapy

Here an overview of existing, commonly used cognitive behavioral therapies for sleep problems will be given. The chapter will conclude by discussing the effectiveness of these therapies.

Cognitive therapy for sleep problems

In this section different cognitive therapies for sleep problems will be discussed. The focus will lie on therapy for insomnia, since this is the most common sleep problem reported to therapists [57]. The therapies discussed are the ones that are used the most by therapists.

Stimulus Control Therapy

Stimulus Control Therapy (SCT) is a therapy which helps the patient to acquire a consistent sleep rhythm. It focuses on associations with the bed and bedroom; the bed and bedroom should be a cue for sleep, and not for other activities such as watching television and worrying. Therefore, the patient should only go to bed when he or she feels tired. The patient should get up at the same time every day, even when he or she did not get a lot of sleep that night. Napping during the daytime should also be avoided. Lastly, SCT requires the patient to get out of bed and go to another room when he or she is unable to fall asleep. The patient should only go back to bed when feeling sleepy again. [58]

Relaxation Therapy

Relaxation Therapy (RT) is constructed to ease arousal, both cognitive and somatic. Attention-focusing procedures can be used to alleviate cognitive arousal. Examples of this are imagery training, where the imagination of the patient is used to envision a time, place or person that makes the patient feel calm and pleasant, and thought stopping, which aims to eliminate troublesome, reoccurring thought patterns. To ease somatic arousal, patients can be taught progressive muscle relaxation. Another option is autogenetic training, which is a relaxation method for body and mind that teaches self control skills. Biofeedback is another option, which tells the user about his body signals such as muscle tension and heartbeat. [59]

Sleep Restriction Therapy

Sleep Restriction Therapy (SRT) is a therapy which is based on restricting the time spent in bed by the patient. The therapy starts out by restricting this time to the average amount of time the patient actually sleeps during the night. For patients with severe sleeping problems this is usually only several hours. Each week, the time spent in bed can be increased or decreased by fifteen to twenty minutes or stay the same, depending on the sleep efficiency. The sleep efficiency is a percentage that is determined by dividing the actual hours of gotten sleep by the time spent in bed, multiplied by 100. When the sleep efficiency is lower than 80% the time spent in bed is decreased and when the sleep efficiency is higher than 90% the time spent in bed is increased. When the sleep efficiency is between 80% and 90%, the time spent in bed stays the same. The patient continues to adjust the time spent in bed each week until a stable amount of time is reached, which should be the optimal sleep duration. [59]

Sleep Hygiene Education

Sleep Hygiene Education (SHE) is often used in combination with other sleep therapies. Sleep hygiene refers to a list of behaviours, physical circumstances and other sleep-related aspects, which discourage the patient to engage in behaviour that interferes with sleep and encourage them to engage in behaviour that improves sleep. There is no absolute consensus about which behaviours and aspects should be included in sleep hygiene, but most lists have many overlapping items. Examples are not to consume alcohol and caffeine, to consume a light bedtime snack since hunger can disturb sleep, and to wake up at a fixed time. [60]

Paradoxical Intention Therapy

Paradoxical Intention Therapy (PIT) is a therapy in which the patients are told that they should try to stay awake as long as possible. They should not engage in activities that disturb sleep, such as watching television or reading a book, and the physical conditions should also encourage sleeping, i.e. the lights should be off. The patients should just lie in bed and keep their eyes open for as long as possible. [61]

Effectiveness of the therapies

One study [59] did a meta-analysis of treatment efficiency which included all of the therapies discussed above. The study found that SCT, RT and SRT produced percentages of improvement that were statistically significant. SHE was not effective at all, when used as a stand-alone therapy. SCT and SRT are the most effective therapies according to this study. Another study [62] did a controlled comparison of SCT, RT and PIT. Their findings were that all three therapies reduced problems relating to sleep significantly. No distinctions were observed between the three therapies. Another paper [63] reviews 48 clinical trials and two meta-analysis'. The paper concludes that SCT, RT and SRT all have significantly positive effects on sleep problems. PIT was also found to have benefits, but it produces smaller treatment gains than the other therapies. As in the other paper, SHE as stand-alone therapy did not produce significant improvements.

All these results should be treated with some caution, since most of the studies use subjective measurements, such as sleep diaries and questionnaires. Besides this point, we conclude that SCT and SRT are the most effective therapies for treating sleeping problems.

eHealth Solutions for Sleep Problems

This chapter will describe some existing eHealth solutions for sleep problems. It will also discuss existing conversational agents.

Existing eHealth solutions for sleep problems

Many studies on digital sleep aids have been conducted. Many using CBT were very alike in their manner of bringing the therapy and the therapy itself. There were almost no applications involving a sophisticated AI therapist. However, the studies on these applications are still useful, as they have overlap in problems this project will face as well, e.g. adherence. In this section some interesting interventions were selected.

Sleepio App

Sleepio [64] is an online program. It uses CBT and has six sessions tailored to a person. It starts with the user filling in a questionnaire. The answers are used to tailor a personalized program. The user has a weekly meeting with an animated figure, "The Prof", and has to keep up a sleep dairy. The Prof gives education about negative thoughts, the sleep schedule, life style and the bedroom. This app includes online tools, such as reminders of work the user has to do. There is also an option to chat with other users. The program has been clinically proven and is used in many trials for research papers.

Sleep Healthy Using the Internet (SHUTi)

SHUTi [65] is a web-based program using CBT. The program can be tailored to an individual‘s need by completing a short online sleep dairy. The course lasts six weeks and the user has to complete a 40-minute session every week. The sessions include interactions, quizzes, puzzles, personal stories, homework review and other information. In between the sessions, the user has to practice techniques and strategies the user has learned. If the user completes at least 5 sleep dairies a week, the program will create a personal sleep recommendation. SHUTi has been tested with multiple randomized controlled trials, it had positive results.

RETURN2SLEEP

Return2Sleep [66] is an online program lasting six weeks. It includes education, sleep hygiene, stimulus control, relaxation training, behavioural therapy and sleep restriction control. In this intervention [67], information is conveyed with audiovisual clips, graphs and mp3 files. The user gets homework each week after completing a module.

Internet intervention van Straten

This online intervention [68] is based on self-help materials, textbooks and research literature and has been discussed with sleep experts. The intervention has six weekly lessons, which include elements of CBT for insomnia. *"Every lesson contained information, examples of other people carrying out the treatment, and homework. After finishing the homework, the coach received a notification. Within 3 working days the coach provided online feedback on the homework"* ([68], p. 1542).

CBT-I coach

CBT-I (Cognitive Behavioral Therapy-Insomnia) is a mobile app which helps with the delivery of and adherence to CBT-I. [69] This application is for people who are undergoing CBT-I with a professional, it does not replace the therapy. The app has 4 sections: a sleep dairy, tools, learning and reminders. Entries in the sleep diary can be done by filling out some questions, e.g. "Did you nap", "What time were you in bed", "What time did you go to sleep?". It also has graphical information on the data the user provides in the sleep dairy. The app also includes education on sleep, such as stages of sleep and sleep regulators. It has sleep hygiene recommendations and stimulus control guidelines to help the user create better sleep habits. CBT-I coach implements sleep restriction therapy and a number of relaxation exercises. In the tools section, CBT-I coach addresses cognitive therapy-related skills.

SleepCare

Sleepcare [70] is a mobile application. It has a sleep diary and various exercises involving sleep restriction, sleep hygiene and relaxation. This application has a conversation screen where the user can interact with the application. These conversations are inspired by face-to-face therapy. This app also tried to address the user's adherence by deploying persuasive strategies such as sending reminders.

Existing conversational agents

The most important aim of this project is to make an agent that can talk to a patient and ask the right questions. There are existing conversational agents out there targeting various mental disorders [71]. Some notable examples are a virtual psychiatric nurse, SimSensei and KASPAR. The agents were mostly used for teaching and practicing with social situations. However, most studies that conducted a trial had a small sample size. Although the evidence is sparse, the agents are promising.

Use of existing solutions

Existing solutions can be viewed in two different ways: existing applications using CBT to treat sleep problems and virtual agents that can have a somewhat natural conversation with a patient. No applications were found that combined both. A few CBT applications did have an animation or chat, but these did not have a real conversation or the ability adopt the Socratic method.

Applications using CBT for treating insomnia are available and tested by many scientific papers. They have a very similar setup; e.g. they all have a sleep dairy and many use the data to tailor sleeping advice to the patient. As read in review papers ([67] [72] [73]), the applications had mild to moderate positive results. However, many researchers reported problems with adherence to the therapy. Therefore, these standard CBT-I formats may be incorporated in the project, but extra attention needs to be paid to adherence.

Secondly, this project aims to incorporate an intelligent coach in the therapy. This coach needs to be able to respond to a patient and adopt the Socratic method. The virtual agent needs to know when to ask which questions. An example is SimSensei [74], a system that uses both verbal and nonverbal behaviours of the user to let a virtual human conduct an interview. However, this project will not make use of nonverbal behaviours as this is not feasible in the time given. Instead of verbal input, written input will be used. Another type of system that is available are chat bots. More on this subject is in the chapter "Software Technologies".

The Virtual Agent

The virtual agent will play a major role in the final therapeutic application. The virtual agent is the 'person' that the patient will interact with during the therapy, and it is therefore important how the agent is perceived by the patient. The coming sections will discuss how appearance and communication can be used to enhance the credibility of the agent. The chapter will conclude with discussing how a patient can bond with a virtual agent.

Appearance

The physical appearance of the virtual agent can influence how the patient perceives the information given by the agent. It is therefore important that the physical appearance of the virtual agent is determined in a way that makes patient think that the dialogue is trustworthy and credible. A youthful male agent with the same ethnicity as the patient is the best way to accomplish this [75]. The gender choice is explained by experiments which show that people are more inclined to follow the advice coming from men than from women. Also, men as well as women rated a man-voiced computer more positively than the same woman-voiced computer. When looking at age, teenagers and patients up to their late twenties prefer an agent that is about the same age as them. Older people prefer a younger character, since they associate being younger with being at ease with technology. The ethnicity of the agent should be the same as the patient's ethnicity, because this will eliminate ethnic prejudices and stereotypes.

Non-verbal communication

The face is probably the most important factor in communicating emotional states. According to Mehrabian, 55% of the meaning of a message is determined by facial expressions [76]. For having a dialogue that is perceived as credible, smiling is the most important facial expression. Smiling appropriately and at the right times improves the credibility. Credibility and approachability are also enhanced by brow-raising and smiling. The virtual agent should make use of every moment where this is appropriate. Diverse movements such as eye blinking, eyebrow movement and small movements with the head are also important components during a dialogue or conversation.

Another important form of non-verbal communication is eye contact. The eye contact during a conversation can have a big influence on whether the information given is perceived as trustworthy. Usually a direct stare when giving the information conveys the most trust. Throughout a dialogue, almost half of the time the focus of attention is on the eyes of the other person. Besides establishing trust, eye contact has many other functions during communication, including establishing the level of interest, persuasion and transmission of emotion.

Verbal and textual communication

There are different options when looking at communication with an agent. The two most straightforward ways of communication are verbal and textual communication. Since we use verbal communication as primary communication means in everyday life, this is perhaps the most natural form of communication. In order to incorporate this means of communication into an agent, a speech recognition system is needed. A major disadvantage of such a system is that it is prone to errors. Also, in noisy environments the system is unreliable. Therefore, many systems use the alternative: textual communication which can be inputted via for example a smartphone or tablet. This type of communication is more robust and reliable. Despite the disadvantages of speech recognition and the robust alternative, speech recognition will make the patient feel like the agent is more of a friend and more socially present than with textual communication [77].

The paragraph above looks at how an agent system can process input from the user, but it is also important how the system outputs its communication. Even when speech recognition is not used, the system can still use speech as an output means. The way a voice sounds can influence how the information is perceived by the other party. The sound of a voice already provides information about the gender, age and ethnicity of the person, and it can even provide information about the emotional state and personal characteristics of the person [78]. A voice that is described as pleasant or attractive by most people should be articulate, refrain from sounding monotonic, low in pitch, high in pitch range and speak at a suitably loud volume. These voice characteristics lead to a person being perceived as confident and relaxed. These voice characteristics are also often associated with characteristics such as competence and honesty, which will make the person be perceived as more credible. [75]

Bonding with a virtual therapist

One of the issues raised is bonding with the virtual agent. This will help patients be more open towards the therapist. It may also help solve the adherence problem. Therefore, the question that needs to be answered is "How does a person bond with a virtual person?". To answer this question, one must first answer a few other questions: "How does a person start a friendship in general?", "Can a person have a bond with a computer?" and "Does nonverbal communication influence bonding?".

So how and why do people initiate a friendship? There are many reasons for a person to start and maintain a relationship, reinforcement theory assumes a person A is attracted to person B if A is rewarded in the presence B; these rewards do not have to be material. *"An example is Byrne's 'Law of Attraction': The attraction towards a person A is a positive linear function of the proportion of positive reinforcements (positive reinforcement divided by total reinforcements) received from A."* ([79], p. 3) There are also two other, somewhat similar theories. Social exchange theory assumes the costs of the exchange have to be outweighed by the rewards. Equity Theory assumes that the input and outcome ratios of both people involved are equal. [79]

A first stage of friendship can be seen as the initiation phase. In this stage, the patients will assess the virtual agent on attractiveness and their disregard criteria. The virtual agents needs to be attractive for the patient, however it differs from person to person what they deem attractive and which aspects will make them reject the virtual agent. An effect of friendship on an interaction is more self-disclosure, e.g. showing emotions. This is desirable for the interaction the patient will have with the virtual agent. An aspect that can reinforce a relationship is having a virtual agent that is like the patient. Another aspect is whether a person feels that the other person likes him or her. This is important in the first stage of forming a bond. [79] In conclusion, important elements are: " *similarity of attitudes, the notion of rewards, joint activities or relational values (trust, confidence, intimacy), reciprocity, certain aspects like social class and common interests that define exclusion or disregard criteria. ""* ([79], p. 3)

To answer the second question, a person can indeed form a bond with even a simple computer. A study showed that users applied several regular social rules to their interaction with computers. For example, they applied politeness norms and gender stereotypes to the computer. the agents used were not sophisticated agents with faces or a personality. The social responses were not difficult to create, according to this study, the human-computer interaction is social. [80]

Another aspect that can influence bonding is the nonverbal communication of the therapist. An important aspect is the facial expression of the virtual agent. A study let people tell a story to a virtual character. This character gave either appropriate or inappropriate facial expressions while they were telling the story. The research showed that the participant faced with the appropriate expressions took less time to tell their story and reported feeling more understood. The people who were faced with the virtual agent with inappropriate facial expressions reported feeling no connection, whereas about one-third of the other group did. [81]

Design Choices

This chapter will give an overview of the made design choices for the to be made sleep coach application. These design choices are based on the findings of the research as presented in the previous chapters. This chapter will provide a short motivation for each design choice.

Chosen Software Technologies

We decided that we will not build the conversational component of our virtual agent using only scripting languages. The main reason for this choice is their limitation to create an intelligent agent. This is because of the simplistic pattern matching algorithms involved in generating responses. Our goal is to make the conversation with the virtual agent as natural as possible. In order to do this the virtual agent should be able to learn from previous experiences. This way the bot will get more experienced and intelligent as it conducts more conversations with various patients. In time the virtual agent should simulate a very skilled therapist that will suit the needs of many patients. The constant innovations in the field of ML have made creating such an agent a real possibility. This is the reason we have chosen to focus on bot platforms such as api.ai, wit.ai and rasa NLU all of which make use of ML algorithms. One of our goals is to build a system through which it will be possible to leverage these and possibly other similar platforms. This system should be able to combine their strengths or easily switch from one platform to another. We hope to achieve this by using a loosely coupled architecture consisting of a server application, the bot platforms and our client application. Most popular bot platforms expose a similar REST API which will make it easier to design a well defined and easy to use interface through which our server application can interact with them. The browser component of the virtual agent will communicate with our server application using our own REST API.

For developing our server application we have chosen to use Python. One of the reasons we made this decision is because it is considered to be a strong and reliable language by a large portion of the web development community. It is used on servers run by big organizations such as Google, Facebook and NASA and has a large and experienced userbase making it easy to find support. It is flexible and has virtually no compile times compared to languages such as Java. This will make it more pleasant to develop our application and give us extra time for our project. Compared to PHP, also a commonly used language for server-side development, Python is considered to be more secure due to its better design. Another reason for choosing Python is the existence of many popular frameworks used by researchers in fields such as AI, NLP, ML and general purpose server-side development. Lastly, we chose Python because of our desire to fine tune and possibly extend the rasa NLU platform. The fact that rasa NLU is written in Python means we will not have to switch between development environments to do this. An additional benefit of this is the increased amount of programming done by us in the same language. This will improve our ability to learn from and help each other effectively.

Besides making use of existing bot platforms we also want to include a fallback component in our server application. After the user's input is processed and no reasonable response has been generated this component will make sure that the therapy can still continue. It will present the user with a few options that he can choose to write to the virtual agent. This way the conversation between the user and our virtual agent can be guided into a reasonable direction. To retain a sense of freedom the user will always be able to optionally switch back to chat mode. For implementing our fallback component we are leaning towards using Chatscript over the other scripting languages due to its superior pattern matching. However we might ultimately opt for our fallback component to be written purely in Python due to the strength of this language for AI related tasks.

Conclusion

This chapter will summarize the results of the research that has been conducted. For the purpose of our project we looked into Cognitive Behavioral Therapy for sleep problems. The results show that insomnia is currently the most common sleep disorder. Therefore, we have chosen to focus on CBT-I and design a virtual coach that will treat this disorder. The appearance of the virtual agent is an important aspect, therefore we want to make several agents with different appearances from which the patient can choose. Another important aspect, as has been shown in the results, is the bonding of the patient with the virtual coach. It seems that nonverbal communication with the patient plays a very important role in the effects of the therapy. For this purpose, we want to implement several components of nonverbal communication such as eye contact. What concerns the structure of the final product, we decided that we will not use a script language to create the virtual agent. Instead, we will make use of a platform named rasa NLU. This platform is chosen because it gives the opportunity to tweak the NLP algorithm such that is suits our purposes. Because this platform relies on the programming language Python, we have decided to implement our final product in Python. Another reason for choosing Python is the fact that there exist many popular frameworks in Python that are used for server-side development, NLP and ML. Also this language is used on servers of many large corporations which implicates that there is a lot support available. Implementing all the features that we have chosen, the result should be a web-based virtual coach who, adopting the appropriate techniques, will treat patients with sleep problems successfully.

C

Cooperation contract

Door ondertekening van dit contract, geven de ondergetekenden aan zich te houden aan de voorwaarden zoals gedefinieerd in dit contract gedurende de looptijd van dit project.

Projectdefinitie

Projectnaam: Virtuele slaapcoach
Duur van het project: 24-04-17 t/m 05-07-17

Voorwaarden

Tijd

- Iedereen is aanwezig op de afgesproken tijd, tenzij er een geldige reden is, bijvoorbeeld OV vertragingen.
- Bij te laat komen moet een reden voor het te laat komen worden gegeven, anders is er sowieso sprake van een geldige reden. De groep bepaalt door meerderheidsstemming of de reden geldig is. Geen geldige reden kan leiden tot sancties (zie 'sancties').
- Groepsleden dienen bij meetings (bijv. met de groep of begeleiders) aanwezig te zijn. Mocht dit niet mogelijk zijn, moet het afwezige groepslid dit tijdig laten weten. Het afwezige groepslid draagt zelf de verantwoordelijkheid dat zijn/haar input alsnog in de meeting besproken wordt.
- Groepsleden laten tijdig weten of zij te laat zijn of niet op komen dagen, zij laten het minimaal een uur van tevoren weten.
- Alle teamleden verklaren bij ondertekening dat ieder minimaal het aantal verplichte studie-uren aan het project besteed.

Organisatie

- Taken zullen eerlijk worden verdeeld (in elke vrijdag meeting). Als een teamlid zich benadeeld voelt, zal hij/zij dit aangeven in de meeting. De groep zal dit bespreken en eventueel corrigeren.
- Er zal een officiële meeting plaatsvinden elke aan het eind van de week; de begintijd van deze meeting is tevens de deadline van de sprint. In het midden van de week zal een onofficiële meeting plaatsvinden waarin de voortgang van de sprint besproken wordt.
- Besluiten worden genomen door eerst een argumenten ronde te doen, met hierna een eventuele discussie. Hierna wordt gestemd en de meerderheid beslist.
- Er dienen geen (grote) beslissingen genomen te worden zonder overleg met de rest van het team.
- Code dient niet zonder overleg met de groep omgegooid te worden.
- Als een teamlid structureel zijn/haar deadlines niet haalt, dan kunnen er sancties volgen.

Sancties

- Wanneer de voorwaarden niet nagekomen worden of de samenwerking niet goed verloopt, zal er een meeting gepland worden waarin de situatie en eventuele maatregelen besproken worden.
- Ontslaan van een groepslid kan voorkomen in de volgende situaties:
  - Structureel niet halen van deadlines
  - 3 keer te laat komen zonder geldige reden
  - Als er 2 keer afspraken (bijv. meetings) zonder geldige reden niet zijn nagekomen.
  - Als er meerdere voorwaarden (of dezelfde meerdere keren) van het contract niet nageleefd worden.

  Voordat deze stap genomen wordt, zal er een meeting zijn waarin de situatie besproken en gestemd wordt over het ontslaan.

Ondertekening

Datum:

Naam	**Ondertekening**
Irene van der Blij	....
Kasper Grabarz	....
Mayke Kloppenburg	....
Magdalena Simidzioski	....

Dit document was ondertekend door alle bovenstaanden op 3 mei 2017.

D

Software Improvement Group feedback

De code van het systeem scoort 4 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door een lagere score voor Duplication.

Voor Duplication wordt er gekeken naar het percentage van de code welke redundant is, oftewel de code die meerdere keren in het systeem voorkomt en in principe verwijderd zou kunnen worden. Vanuit het oogpunt van onderhoudbaarheid is het wenselijk om een laag percentage redundantie te hebben omdat aanpassingen aan deze stukken code doorgaans op meerdere plaatsen moet gebeuren.

In dit systeem is er duplicatie te vinden tussen EndPanel en ExplanationPanel en StartPanel. Het is in dit soort gevallen beter om het gedeelde stuk in een aparte methode onder te brengen en deze vervolgens aan te roepen.

De aanwezigheid van test-code is in ieder geval veelbelovend, hopelijk zal het volume van de test-code ook groeien op het moment dat er nieuwe functionaliteit toegevoegd wordt.

Over het algemeen scoort de code bovengemiddeld, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkelfase.

E

Infosheet

General information

Project title: Virtual Sleep Coach
Client organization: University of Amsterdam
Presentation date: June 3rd, 10:00

Project description

During this project we were asked to explore the possibilities and limitations of creating a virtual therapist that can help people who suffer from a sleeping disorder. In order to answer this question, the goal of the project was to make such a therapist ourselves.

The first two weeks of the project were spent doing research. We looked into cognitive behavioural therapy and existing eHealth solutions for people with sleeping disorders. We also conducted research into existing chat bots, platforms and languages. At the end of the research phase we concluded that machine learning was the best way to make the agent smart.

During the design phase we stumbled upon the fact that we could not get access to a corpus of conversations between therapists and patients discussing sleep problems. This meant that we could not use machine learning on this corpus, which greatly influenced how smart the agent was going to be. We had to rethink our design choices concerning the artificial intelligence and decided to use a different platform. The personalized artificial intelligence was incorporated in our web application, which communicates with the code via our server.

The testing of the product is based on unit tests, where all modules and functions are tested individually after every new addition. The user interface was tested by setting up use cases. User tests on the final product were also performed. This gave us insight in how people with no knowledge about the system would interact with the system.

The main purpose of this project was to explore the possibilities and limitations for creating a virtual therapy that treats insomnia. Even though the solution we created cannot yet be used as a standalone therapy, it shows a promising glimpse of the many possibilities to conduct virtual therapy. A project group with more time, money and help from a psychologist could greatly improve the project by buying or generating a corpus which can then be used for machine learning and by letting the psychologist model the conversation. The future of virtual therapy has yet to be discovered, but this project already provides a bright outlook.

Members of the project team

Name: Irene van der Blij (i.c.vanderblij@student.tudelft.nl)
Contribution: Front- and back-end developer

Name: Kasper Grabarz (k.p.grabarz@student.tudelft.nl)
Contribution: Environment setup, Front- and back-end developer

Name: Mayke Kloppenburg (m.l.kloppenburg@student.tudelft.nl)
Contribution: Software testing, Scrum master, Editor

Name: Magdalena Simidzioski (m.simidzioski-1@student.tudelft.nl)
Contribution: Conversation modeling, Editor, Front-end developer

All team members contributed to the research report, the design, the implementation, the testing, and preparing the final report and the final project presentation.

Client and coach

Client: Jaap Lancee, assistant professor Department of Clinical Psychology University of Amsterdam
Coach: Willem-Paul Brinkman, Interactive Intelligence group TU Delft

The final report for this project can be found at: http://repository.tudelft.nl.

Bibliography

[1] CBS, *Personen naar bij de huisarts bekende diagnose; leeftijd, geslacht,* (2016), `http://statline.cbs.nl/Statweb/publication/?DM=SLNL&PA=83110ned&D1=0&D2=0&D3=1-16,l&D4=96&D5=0&D6=a&HDR=T,G4,G1,G5,G2&STB=G3&VW=G`.

[2] A. T. Beck, *Cognitive therapy and the emotional disorders,* New York (1976).

[3] A. Ellis, *Reason and emotion in psychotherapy.* (1962).

[4] A. T. Beck, *Cognitive therapy: Nature and relation to behavior therapy,* Behavior therapy **1**, 184 (1970).

[5] J. Watson, *Behaviorism*, Phoenix books (University of Chicago Press, 1930).

[6] M. C. Jones, *The elimination of children's fears.* Journal of Experimental Psychology **7**, 382 (1924).

[7] G. L. Thorpe and S. L. Olson, *Behavior therapy: Concepts, procedures, and applications* (Allyn & Bacon, 1997).

[8] S. Rachman, *The evolution of cognitive behaviour therapy.* (1997).

[9] S. D. Hollon and A. T. Beck, *Cognitive and cognitive-behavioral therapies.* (1994).

[10] J. Bennett-Levy, *Mechanisms of change in cognitive therapy: The case of automatic thought records and behavioural experiments,* Behavioural and Cognitive Psychotherapy **31**, 261 (2003).

[11] J. E. Bennett-Levy, G. E. Butler, M. E. Fennell, A. E. Hackman, M. E. Mueller, and D. E. Westbrook, *Oxford guide to behavioural experiments in cognitive therapy.* (Oxford University Press, 2004).

[12] S. M. Bögels and P. Van Oppen, *Cognitieve therapie: theorie en praktijk* (Bohn Stafleu van Loghum, 2015).

[13] K. Seeskin, *Dialogue and discovery: A study in Socratic method* (SUNY Press, 1987).

[14] J. C. Overholser, *Elements of the socratic method: I. systematic questioning.* Psychotherapy: Theory, Research, Practice, Training **30**, 67 (1993).

[15] R. A. S. B. . E. G. Beck, A. T., *Cognitive therapy of depression,* New York: The Guilford Press. (1979).

[16] C. A. Padesky, *Socratic questioning: Changing minds or guiding discovery,* in *A keynote address delivered at the European Congress of Behavioural and Cognitive Therapies, London*, Vol. 24 (1993).

[17] A. T. Beck, G. Emery, and R. Greenberg, *Anxiety disorders and phobias: A cognitive approach,* Basic, New York , b58 (1985).

[18] A. Wells, *Cognitive therapy of anxiety disorders: A practice manual and conceptual guide.* (Wiley, 1997).

[19] C. Otte, *Cognitive behavioral therapy in anxiety disorders: current state of the evidence,* Dialogues Clin Neurosci **13**, 413 (2011).

[20] E. Robinson, N. Titov, G. Andrews, K. McIntyre, G. Schwencke, and K. Solley, *Internet treatment for generalized anxiety disorder: a randomized controlled trial comparing clinician vs. technician assistance,* PloS one **5**, e10942 (2010).

[21] A. T. Borrego and L. S. Bonome, *Cognitive-behavioral therapy for chronic psychosis,* Actas Esp Psiquiatr **37**, 106 (2009).

[22] A. K. Matusiewicz, C. J. Hopwood, A. N. Banducci, and C. Lejuez, *The effectiveness of cognitive behavioral therapy for personality disorders,* Psychiatric Clinics of North America **33**, 657 (2010).

[23] R. Murphy, S. Straebler, Z. Cooper, and C. G. Fairburn, *Cognitive behavioral therapy for eating disorders,* Psychiatric Clinics of North America **33**, 611 (2010).

[24] E. Driessen and S. D. Hollon, *Cognitive behavioral therapy for mood disorders: efficacy, moderators and mediators,* Psychiatric Clinics of North America **33**, 537 (2010).

[25] P. S. Foroushani, J. Schneider, and N. Assareh, *Meta-review of the effectiveness of computerised cbt in treating depression,* BMC psychiatry **11**, 131 (2011).

[26] A. Spirito, C. Esposito-Smythers, J. Wolff, and K. Uhl, *Cognitive-behavioral therapy for adolescent depression and suicidality,* Child and adolescent psychiatric clinics of North America **20**, 191 (2011).

[27] C. A. Espie, S. D. Kyle, C. Williams, J. C. Ong, N. J. Douglas, P. Hames, and J. Brown, *A randomized, placebo-controlled trial of online cognitive behavioral therapy for chronic insomnia disorder delivered via an automated media-rich web application,* Sleep **35**, 769 (2012).

[28] J. Temperton, *Apps are dying. long live the subservient bots ready to fulfil your every desire,* (2016).

[29] N. M. Radziwill and M. C. Benton, *Evaluating quality of chatbots and intelligent conversational agents,* arXiv preprint arXiv:1704.04579 (2017).

[30] A. M. Turing, *Computing machinery and intelligence,* Mind **59**, 433 (1950).

[31] N. Chomsky and S. Structures, *The hague,* Mouton & Co (1957).

[32] A. Bhargava, A. Çelikyilmaz, D. Hakkani-Tür, and R. Sarikaya, *Easy contextual intent prediction and slot detection,* in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013* (2013) pp. 8337–8341.

[33] P. Bashmakov, *Advanced natural language processing tools for bot makers – luis, wit.ai, api.ai and others (updated),* (2016), `https://stanfy.com/blog/advanced-natural-language-processing-tools-for-bot-makers`.

[34] D. Britz, *Deep learning for chatbots, part 1 – introduction,* (2016).

[35] J. Weizenbaum, *Computer power and human reason: From judgment to calculation,* (1976).

[36] C. R. Rogers *et al.*, *Client-centered therapy,* (1951).

[37] R. C. Parkinson, K. M. Colby, and W. S. Faught, *Conversational language comprehension using integrated pattern-matching and parsing,* Artificial Intelligence **9**, 111 (1977).

[38] R. Wallace, *Artificial linguistic internet computer entity (alice),* (2001).

[39] H. Loebner, *Loebner prize,* described at: http://www. loebner. net/Prizef/loebner-prize. html. Breazeal, C (2001).

[40] A. L. . T. Wit, *Bot engine,* (2016), `https://wit.ai/blog/2016/04/12/bot-engine`.

[41] A. Kharpal, *Google bought a chatbot start-up for a future where we'll all be talking to robots,* (2016), `http://www.cnbc.com/2016/09/20/google-bought-a-chatbot-start-up-for-a-future-where-well-all-be-talking-to-robots.html`.

[42] api.ai, *Introduction,* (2017), `https://docs.api.ai/docs/key-concepts`.

[43] api.ai, *Api.ai integrations,* (), https://docs.api.ai/docs/integrations.

[44] rasa NLU, *rasa nlu: Open-source bot tool for natural language understanding,* (), https://rasa.ai/.

[45] LASTMILE, *Lastmile: Next-generation conversational ai,* https://golastmile.com/.

[46] A. Nichol, *Comments on post "rasa nlu",* https://www.producthunt.com/posts/rasa-nlu-2/comments/405005.

[47] rasa NLU, *Language support,* (), http://rasa-nlu.readthedocs.io/en/stable/languages.html.

[48] R. Wallace, *Artificial intelligence markup language (aiml),* (2001), http://www.alicebot.org/aiml.html.

[49] M. L. McNeal and D. Newyear, *Chatbot creation options,* Library Technology Reports **49**, 11 (2013).

[50] A. Weeratunga, S. Jayawardana, P. Hasindu, W. Prashan, and S. Thelijjagoda, *Project nethra-an intelligent assistant for the visually disabled to interact with internet services,* in *Industrial and Information Systems (ICIIS), 2015 IEEE 10th International Conference on* (IEEE, 2015) pp. 55–59.

[51] B. Wilcox, *Chatscript,* (2009), http://chatscript.sourceforge.net/.

[52] B. Wilcox, S. Wilcox, B. Psych, and D. F. Arts, *Suzette, the most human computer,* (2010).

[53] N. Petherbridge, *Rivescript,* (2010), https://www.rivescript.com/.

[54] S. Gupta, D. Borkar, C. De Mello, and S. Patil, *An e-commerce website based chatbot,* .

[55] N. Petherbridge, *Aires bot,* (2013), https://www.rivescript.com/bots.

[56] api.ai, *Intents,* (), https://docs.api.ai/docs/intents.

[57] A. L. Chesson Jr, W. Anderson, M. Littner, D. Davila, K. Hartse, S. Johnson, M. Wise, and J. Rafecas, *Practice parameters for the nonpharmacologic treatment of chronic insomnia. an american academy of sleep medicine report. standards of practice committee of the american academy of sleep medicine.* Sleep **22**, 1128 (1999).

[58] R. R. Bootzin, D. Epstein, and J. M. Wood, *Stimulus control instructions,* in *Case studies in insomnia* (Springer, 1991) pp. 19–28.

[59] C. M. Morin, J. P. Culbert, and S. M. Schwartz, *Nonpharmacological interventions for insomnia,* Am J Psychiatry **151**, 1172 (1994).

[60] E. J. Stepanski and J. K. Wyatt, *Use of sleep hygiene in the treatment of insomnia,* Sleep medicine reviews **7**, 215 (2003).

[61] L. M. Ascher and R. M. Turner, *Paradoxical intention and insomnia: An experimental investigation,* Behaviour Research and Therapy **17**, 408 (1979).

[62] R. M. Turner and L. M. Ascher, *Controlled comparison of progressive relaxation, stimulus control, and paradoxical intention therapies for insomnia.* Journal of Consulting and Clinical Psychology **47**, 500 (1979).

[63] C. M. Morin, P. J. Hauri, C. A. Espie, A. J. Spielman, D. J. Buysse, and R. R. Bootzin, *Nonpharmacologic treatment of chronic insomnia. an american academy of sleep medicine review.* Sleep **22**, 1134 (1999).

[64] Big Health, *Sleepio app,* (2010), https://www.sleepio.com/.

[65] BeHealth Solutions, *Shuti,* (2007), http://www.myshuti.com/.

[66] University of Manitoba, Health Sciences Centre Winnipeg, *Return2sleep,* (2009), http://www.return2sleep.com/.

[67] Y.-y. Ye, N.-k. Chen, J. Chen, J. Liu, L. Lin, Y.-z. Liu, Y. Lang, X.-j. Li, X.-j. Yang, and X.-j. Jiang, *Internet-based cognitive–behavioural therapy for insomnia (icbt-i): a meta-analysis of randomised controlled trials,* BMJ open **6**, e010707 (2016).

[68] A. Van Straten, J. Emmelkamp, J. De Wit, J. Lancee, G. Andersson, E. van Someren, and P. Cuijpers, *Guided internet-delivered cognitive behavioural treatment for insomnia: a randomized trial,* Psychological medicine **44**, 1521 (2014).

[69] E. Kuhn, B. J. Weiss, K. L. Taylor, J. E. Hoffman, K. M. Ramsey, R. Manber, P. Gehrman, J. J. Crowley, J. I. Ruzek, and M. Trockel, *Cbt-i coach: a description and clinician perceptions of a mobile app for cognitive behavioral therapy for insomnia,* Journal of clinical sleep medicine: JCSM: official publication of the American Academy of Sleep Medicine **12**, 597 (2016).

[70] C. H. Horsch, J. Lancee, F. Griffioen-Both, S. Spruit, S. Fitrianie, M. A. Neerincx, R. J. Beun, and W.-P. Brinkman, *Mobile phone-delivered cognitive behavioral therapy for insomnia: A randomized waitlist controlled trial,* Journal of Medical Internet Research **19**, e70 (2017).

[71] S. Provoost, H. M. Lau, J. Ruwaard, and H. Riper, *Embodied conversational agents in clinical psychology: A scoping review,* Journal of medical Internet research **19** (2017).

[72] J. M. Trauer, M. Y. Qian, J. S. Doyle, S. M. Rajaratnam, and D. Cunnington, *Cognitive behavioral therapy for chronic insomnia: a systematic review and meta-analysis,* Annals of internal medicine **163**, 191 (2015).

[73] B. I. Voinescu, A. Szentagotai, and D. David, *Internet-administered cognitive-behavioral therapy for insomnia,* Journal of Cognitive and Behavioral Psychotherapies **13**, 225 (2013).

[74] D. DeVault, R. Artstein, G. Benn, T. Dey, E. Fast, A. Gainer, K. Georgila, J. Gratch, A. Hartholt, M. Lhommet, *et al.*, *Simsensei kiosk: A virtual human interviewer for healthcare decision support,* in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems* (International Foundation for Autonomous Agents and Multiagent Systems, 2014) pp. 1061–1068.

[75] A. J. Cowell and K. M. Stanney, *Embodiment and interaction guidelines for designing credible, trustworthy embodied conversational agents,* in *International Workshop on Intelligent Virtual Agents* (Springer, 2003) pp. 301–309.

[76] A. Mehbabian, *Orientation behaviors and nonverbal attitude communication1,* Journal of communication **17**, 324 (1967).

[77] E. C. Grigore, A. Pereira, I. Zhou, D. Wang, and B. Scassellati, *Talk to me: Verbal communication improves perceptions of friendship and social presence in human-robot interaction,* in *International Conference on Intelligent Virtual Agents* (Springer, 2016) pp. 51–63.

[78] K. R. Scherer, *Methods of research on vocal communication: Paradigms and parameters,* Handbook of methods in nonverbal behavior research , 136 (1982).

[79] B. Stronks, A. Nijholt, P. van der Vet, and D. Heylen, *Designing for friendship: Becoming friends with your eca,* (2002).

[80] C. Nass, J. Steuer, and E. R. Tauber, *Computers are social actors,* in *Proceedings of the SIGCHI conference on Human factors in computing systems* (ACM, 1994) pp. 72–78.

[81] J. W.-E. Wong and K. McGee, *Frown more, talk more: effects of facial expressions in establishing conversational rapport with virtual agents,* in *International Conference on Intelligent Virtual Agents* (Springer, 2012) pp. 419–425.