FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR’S THESIS

Title:	Investment voice conversational bot
Student:	Jan Drábek
Supervisor:	Ing. Jan Šedivý, CSc.
Study Programme:	Informatics
Study Branch:	Information Systems and Management
Department:	Department of Software Engineering
Validity:	Until the end of winter semester 2019/20

Instructions

Design and implement a voice interface based conversational bot for checking stocks portfolio with selected companies. The bot will inform the user about the current investment portfolio status, biggest gainers and losers and give detailed information about individual stocks. Use the Amazon Alexa platform for implementing the bot.

1) Review the Alexa Skills Kit capabilities for conversational bot.
2) Analyze the financial information web sites and their APIs
3) Select a financial information server and implement data scraping.
4) Implement on the Alexa platform a bot informing the user about current investment portfolio status, biggest gainers and losers, give detailed information about individual stocks.
5) Test and document the services.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 18, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Investment voice conversational bot

Jan Drábek

Department of Software Engineering
Supervisor: Ing. Jan Šedivý, CSc.

May 14, 2018

Acknowledgements

Firstly, I want to thank my thesis supervisor Ing. Jan Šedivý, CSc. for continuous support during the the creation of this thesis.

Besides my supervisor, I would like to thank my family and friends for providing supportive environment during my whole studies.

Declaration

In Prague on May 14, 2018 . . . .

Citation of this thesis

Drábek, Jan. *Investment voice conversational bot.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

Abstract

This project explores the analysis, design, and development of a voice-enabled intelligent conversation bot for the Amazon Alexa platform which allows users to get information about the stock trading market. Thanks to the intuitive voice interface, the goal of the application it to lower the barrier of getting into the stock trading domain for the general public.

The created chatbot focuses on educating the user about stock markets, giving a fast and easy way to find out about the news regarding the trading companies and generally keeping the user up to date on the most important trading information.

The application targets firstly people interested in stock market investments and aspiring investors, however, using this application could provide great value even to more experienced investors.

Keywords chatbot, voice-enabled, investments, stock markets, education, Amazon Alexa, Alexa Skill, python

Abstrakt

Tato práce rozebírá analýzu, návrh a vývoj hlasového inteligentního konverzačního bota na platformě Amazon Alexa, který umožňuje uživatelům získávat informace ohledně akciového trhu. Cílem aplikace je snížit vstupní bariéru do světa akciových trhů pro širokou veřejnost, využitím intuitivního hlasového rozhraní.

Vytvořený konverzační bot se zaměřuje na vzdělávání uživatele ohledně akciových trhů, umožnění rychlého a jednoduchého získání novinek ohledně obchodovaných společností a celkovému udržení informovanosti uživatele.

Aplikace cílí v první řadě na lidi se zájmem o investování na akciovém trhu a aspirující investory, přesto, může být velmi přínosná i pro již zkušenější investory.

Klíčová slova chatbot, ovládání hlasem, investice, akciový trh, vzdělávání, Amazon Alexa, Alexa Skill, python

Contents

List of Figures

Chapter 1

Introduction

The economy has been on a rise for many years now, and so has the stock market. In fact, as described in chapter 2.1.1, in the last five years, the value of the US market increased by 170%. This appreciation of value is a tremendous investment.

Unfortunately, not many people capitalize on this opportunity, even though they have the money. Especially the millennials have more money in savings than previous generations, as reported by the Bank of America[7]. But they are vastly uneducated about the stock market[8].

As described in chapter 2.1.1, most people do not own any stocks, and if they do, it is just very few. The reason for that is that for the general public, entering the stock market appears to be very scary[8].

The problem is that learning about trading can be difficult and time-consuming. With many specialized terms, data sources, unintuitive graphs and tens of factors to consider, the entry barrier is very high.

The opportunity is seen in voice-enabled chatbots. With these applications, the user can interact in natural language. This approach removes the friction of complicated user interfaces. Thanks to this technology, the user could interact with the trading application as if it was a personal broker. It would present the trading information in a user-friendly way and motivate the user to educate himself on this topic and ultimately even show how to capitalize on the growing economy trend.

This thesis discusses the general practices of designing a user-friendly chatbot build on the Amazon Alexa platform and addresses the process of analyzing and building a voice assistant application in the domain of stock market investing.

1.1 Goal

The goal of this thesis is to create a voice-enabled conversation chatbot in the stock market investments domain. The chatbot will be built on the Amazon Alexa platform.

The objective of the analysis phase of this project is to examine the existing solution on the Alexa platform and compare their futures with the anticipated application, research the financial information data sources and explore the Alexa Skills Kit platform. Additionally, introduce a business model for the created application.

Furthermore, explore the concepts of designing for voice user interface, design the Amazon Alexa chatbot based on the concepts and implement the application.

Chapter 2

State-of-the-art

2.1 Introduction to stock market trading

The OttoBot application targets firstly people interested in stock market investments and aspiring investors, however, using this application could provide great value even to more experienced investors.

The goal is to encourage the general public to learn about the stock market and improve financial literacy.

This section provides an overview of the stock markets and its characteristics and discusses the fundamentals of stock market investing.

Although the thesis focuses on the US market, most of the principles apply to other markets as well.

2.1.1 Opportunities in the stock market

Almost everyone has heard about the stock market, but very few people actively trade or even understand the fundamental principles how the financial markets work.

As stated by William J. O'Neil [9] we have had 12 bull markets and 11 bear markets in the past 50 years. A bull market is a state of the trading market where investments are rising in value and encourage buying, on the other hand, a bear market is a state where prices are falling and the value decreases [10][11]. These bull runs lasted on average 3 years and 9 months, and the market value increased on average by 100%. When looking at the bearish moves during the 50 years period, the average duration was 9 months and translated into a drop in value by about 30%.

This trend can be seen even on the more recent scale, in the last 5 years, the market increased in value by about 170%. This was determined by the S&P 500 index, which is calculated from the 500 largest public companies listed on American stock exchanges and roughly reflects the performance of the market. The lifetime chart of S&P 500 index is shown in figure 2.1.

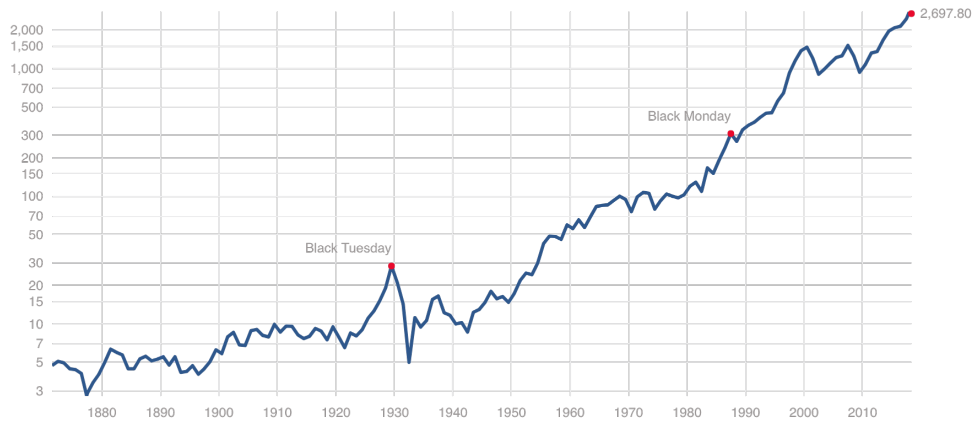


Figure 2.1: S&P 500 index lifetime chart [1]

What causes this long-term growth? It is the people creating real value with their businesses. Each year, there are hundreds of new products, services, and inventions that improve people's lives in some ways. Free people in free countries with desires work hard to improve their living conditions further.

The stock markets do not increase in value because of greed. It is because of ambitious businesses bringing real value.

The performance of the stock market is impressive and investing in stocks seem like a tremendous deal. But not many people own stocks.

Surprisingly, only about 54% of Americans own any stock, and if they do, they own small amount [12]. According to New York University economist Edward Wolff [13], the top 10 percent of households own 84% of the stock market wealth.

In Europe, the ownership of stocks is much lower. Even though, the German stock market being of the top performing, only about 12% of Germans own any stock or fonds [14]. In France, it is less than 8% [15].

According to William J. O'Neil [9], the reason why people are not taking advantage of the stock market is that the general public does not understand the market. For most people, it is complicated to get into stock market investing and make well-informed decisions. They are unsure, or even afraid.

2.1.2 What are stocks?

Most of the people have at least an idea what a company stock is. It usually goes like this:

"A stock represents a share of company ownership, it is a claim on the company's assets and profits."

That is not entirely true. Law treats companies as a legal person, this entitles companies to buy properties, borrow money, be sued, just as a regular natural person. This translates into assets ownership as well. If this 'person' buys assets, it owns them, not the shareholders.

On the other hand, if the company should go bankrupt, the shareholders are not at risk. Jurisdiction says the corporate property is legally separated from the shareholder's property.

Being a stock owner gives you some other benefits. A shareholder with large enough stake has the privilege to control the company indirectly, by appointing its board of directors, through voting on the shareholders' meetings.

Owning stock entitles you to have an adequate share of companies profits on a regular basis, known as dividends. Although most companies decided not to pay out dividends, instead reinvest this profit back into the company.

The purpose of stocks is to allow companies to raise capital from public investors. When a company is founded, there might be multiple co-founders with various amounts of equity (shares). As the company grows, there might be a need for an investor to step in. The investor gets some equity as a compensation for the capital he provides and the founders end-up with a lower percentage. At that point, the company is still privately held. There is typically s relatively small number of shareholders and it's fairly complicated to exchange the stocks. As the company matures, some of the investors might want to cash out their profits and the company might need more outside funding to expand. The company decides to file for an initial public offering (IPO), to enter the public market, where the stock can be easily publicly traded.

During the IPO the investors buy the stock directly from the company, this is called the primary market. After the company went public, the stocks start to trade on the secondary market, where the shareholders no longer buy from the company itself, instead, they trade with other shareholders owning the stock.

Other than selling stocks, there are more means for companies to raise capital. Issuing bonds is one of them. Bonds represent debt obligations. The business borrows money with an interest rate and commits to paying it back including an interest rate [16].

There are multiple types of stocks:

- common stock
- preferred stock
- stock classes

Common stock The most frequent type are common stocks. They represent voting rights and a claim on the companies profits (dividends). The dividend frequency can vary, although most companies pay them quarterly [17]. Dividends are not guaranteed, the value may vary on the business's performance in the given timeframe.

If a company goes bankrupt, the common stock owners are the last in line to get repaid.

Preferred stock In some ways similar to bonds are preferred stocks. The usually do not come with voting rights but investors are often guaranteed a fixed dividend. In the case of company liquidation, the preferred stockholders have the priority over common stockholders when it comes to compensation, nonetheless, they are behind bond owners.

Stock classes Different stock classes can be used, as a way to organize the issued stocks into collections with different conditions and privileges. Sometimes, corporations want to keep concentrated voting power in a certain group, to do that, they issue a class of stock with, for example, 10 votes per share. Additionally, they issue another class of stocks to raise capital, which comes with 1 vote per share. When multiple classes of stock, they are typically named as class A, class B, etc.

This chapter was inspired by a Investopedia article Stocks Basics [18].

2.1.3 The stock market

Stock markets are places where buyers and seller meet to make stock transactions. This can be both physical locations or virtual exchanges. Only public companies can be traded on the stock market.

The listed companies usually do not participate in the transactions. Investors usually buy the stock from other shareholders, who are willing to sell. This is called the secondary market.

There are numerous exchanges where trading is conducted. Some of the biggest exchanges are the New York Stock Exchange (NYSE), Nasdaq Stock Market (NASDAQ) or the London Stock Exchange (LSE). For a company, to be listed on an exchange, it needs to meet specific criteria and file for listing. After that, the business can be traded on that exchange.

The stock value is determined in a number of ways, usually is the traded price set through the auction process.

A shareholder, who wants to sell the stock, places an *offer*, also referred as *ask*, for how much money he is willing to sell the stock. A buyer places a *bid*, how much he is willing to pay for the stock. When the bid and ask price meet, the transaction is conducted. In a liquid market, there are many offers and bids waiting to be filled. The gap between the bid and ask prices is called spread, the smaller the spread is, the more liquidity the market has. This can be expressed with a simple equation:

$$\text{small spread} + \text{many bids} + \text{many offers} = \text{liquid market}$$

The stock prices change constantly, they are affected by the fluctuations in supply and demand. If there is increasing demand, the price goes up, if shareholders are selling off their stock, the supply increases and price falls.

What does stock represent? The value of a single stock multiplied by the number of issued stocks reflects how much the market values the company, so-called market capitalization. This company value can be translated into two main components:

$$\text{total value of company} = \text{assets} + \text{future cash flows}$$

First, the business has assets, this relatively easy to translate into a dollar value. The second component, which determines the companies potential value, is the future cash flows. This is the expectations for the company performance in the near future and is influenced by both internal and external factors.

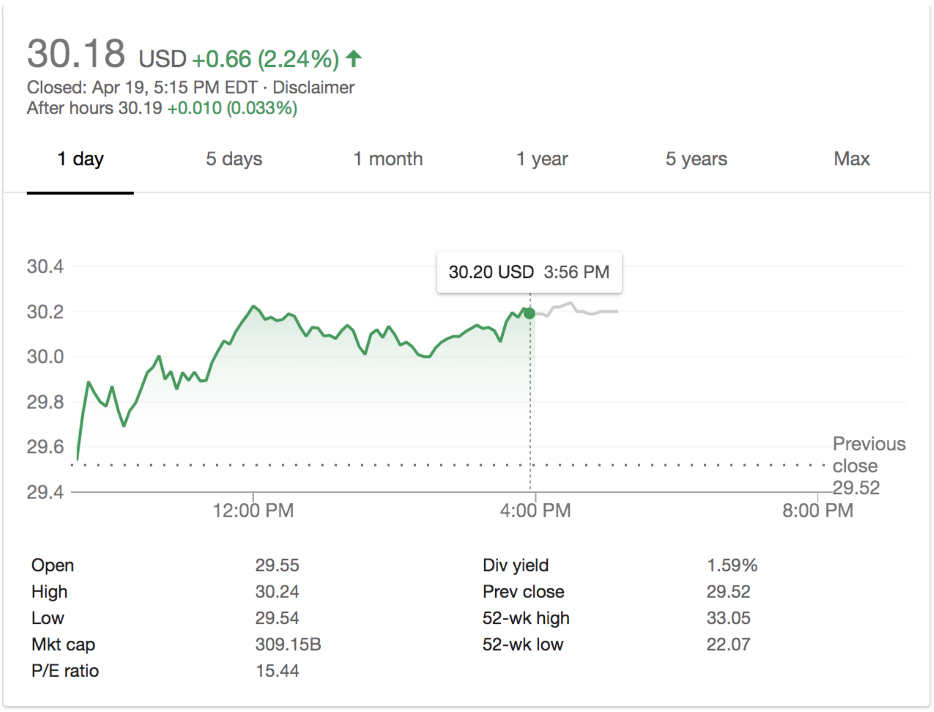
The hypothetical predictions of the future performance are mostly what drives the stock price.

Read Stock Quote Every publicly traded stock has its ticker symbol, this is an arrangement of characters representing the company in the stock market.

Some of the most basic information about the stock performance can be found on the stock quote summary 2.2. This overview can be easily found on the internet through the stocks ticker symbol.

A brief description of some of the information found on the stock quote in figure 2.2:

Summary - shows the current stock price, which is the last traded price, the 24-hour change in dollar value and in percentage relative to the stock price
Open - the value of the stock at the beginning of the trading day
High - highest value the stock reached during the trading day
Low - lowest value of stock during the trading day
Mkt cap - market capitalization of the company calculated as number of issued stocks multiplied by price of a single stock
P/E ratio - price/earnings ratio reflects how much are investors willing to pay for a stock, relative to its earnings, calculated as stock price divided by earnings per share [19]
Div Yield - the value of dividend paid each year, relative to the stock price, calculated as annual dividend per share divided by stock price [20]
Prev close - the closing trading price of the share on the previous trading day



30.18 USD +0.66 (2.24%)
Closed: Apr 19, 5:15 PM EDT · Disclaimer
After hours 30.19 +0.010 (0.033%)

Open	29.55	Div yield	1.59%
High	30.24	Prev close	29.52
Low	29.54	52-wk high	33.05
Mkt cap	309.15B	52-wk low	22.07
P/E ratio	15.44		

Figure 2.2: Stock Quote Bank of America [2]

Stock Market Indexes It is useful to capture the overall trend of a market or a market segment. This is the purpose of stock market indexes. These indexes represent aggregated prices of a number of different stocks, defining the segment.

Sometimes the S&P 500 index, is used to represent the performance of the US stock market. The S&P 500 is a market cap weighted index of 500 largest US companies, by looking at the performance of these companies a decent assumption can be made about the whole market. There are also indexes which inspect a category of stocks or some industry.

This chapter was inspired by a Investopedia article Stocks Basics [18].

2.2 Voice User Interfaces

The fundamental part of the OttoBot application is the utilization of voice user interface (VUI) instead of traditional graphical user interface (GUI).

Thanks to the VUI, the user interacts with the application using just his voice. In this chapter, I will cover the effects of using natural language as an interaction method, the current state of the technology and its role in the research project.

2.2.1 Introduction to VUIs

The first development of VUIs dates back to 1950s. In Bell Labs, a group of researchers developed a system capable of recognizing digits. This system was limited to single-speaker and wasn not much use outside of the lab.

Throughout the 1960s and 1970s, more sophisticated systems were developed, which started to support larger vocabulary and did not require the speaker to make a pause between each word.

In the 1990s, the first speaker independent system became reality and people started thinking about the commercial applications of this technology.

By 2000s, VUIs became mainstream. These systems were capable of understanding simple human requests over the telephone line and execute simple tasks. They were tremendously useful for people calling over an over again to get an updated on some simple information. Anyone with a traditional landline phone could use them by asking about the current weather, the traffic conditions, stock market information, ordering flights and more 24/7, all of this long before the smartphone era.

This resulted in significant cost savings because of automation of repetitive tasks. Suddenly, there was no need for specialized people to answer basic questions.

Ultimately these VUIs got a bad reputation. The reason was the overuse of these systems. The users often had to pass through the automatic voice system everytime they needed to contact the company, even with more complex requests the VUI systems were not capable of handling.

The systems caused so much frustration, that even dedicated services, like GetHuman (www.gethuman.com) started to appear, that provided contact information like phone numbers which should bypass the automatic VUI system.

2.2.1.1 The new generation of VUI systems

The latest iteration of this technology introduced the concept of virtual private assistants (VPA), sometimes called intelligent personal assistants (IPA). Opposed to the single purpose VUI systems, these assistants are creating a complex platform capable of handling request across multiple domains by delegating the requests to relevant service.

By opening this platform to third-party applications, developers around the world can build applications on top of the VUI personal assistant. This allows these assistants to become more powerful and grow much quicker, which would not be feasible to achieve by a single company.

One of the most challenging aspects of conversational interface is the ability to remember the context across multiple user requests as the dialog continues. The new generation of VUI systems performs much better in this regard compared to previous systems but there is still much progress to be made.

The essential characteristics of the IPA are:

- conversational interface
- personal context keeping
- service delegation

The biggest technology companies like Amazon, Google, Apple, Microsoft, and Samsung, recognize the disruptive potential of this technology and bet big on virtual assistants. They even integrate their proprietary solution straight into operating systems, internet of things (IoT) devices and recently even smart-speakers, which are devices dedicated solely to run the virtual assistant.

The first modern digital virtual assistant is considered to be Siri introduced by Apple in the iPhone 4s reveal in October of 2011, implemented right in the iPhone's operating system. Since then Siri is present in the whole Apple ecosystem including iPhone, iPad, Apple Watch, Macbooks and newly the smart speaker HomePod.

After the reveal of Siri in 2011, Google introduced it's competitor Google Now in 2012, which was a feature of Google Search and offered predictive cards with information and daily updates on information user might need.

Not even a year after that, Microsoft unveiled its Cortana as part of the future operating system. Surprisingly the e-commerce business Amazon followed by introducing its assistant Alexa with the release of the first smart-speaker called Amazon Echo in 2014.

In may of 2016, Google revealed the Google Assistant together with the smart-speaker Google Home as a successor to the Google Now service, tightly integrated with the Android operating system. The release timeline is visualized on image 2.3.

These VPAs strive to be exactly what they are called, they want to become your full-fledged personalized assistant. To do that, they have to be everywhere with you, on your phone, in your home in the form of smart-speaker, even car manufactures integrate them into their newest cars. This way, you can activate the assistant, simply just by saying its name, making it the ultimate hands-free experience.

VPAs just started to be really useful and provide real value to the users.

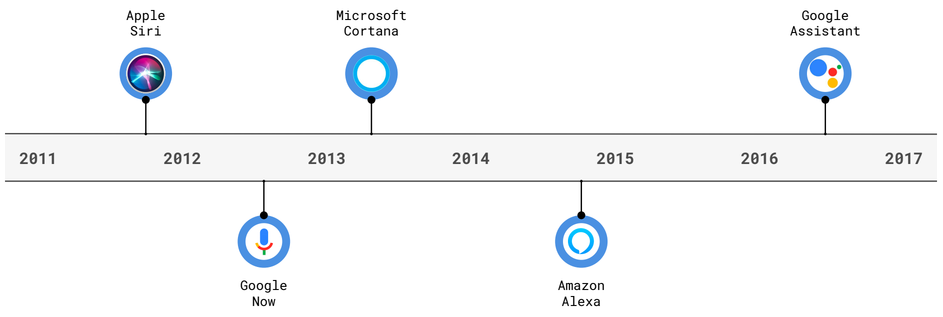


Figure 2.3: Release timeline of major VPAs

2.2.1.2 The technology behind VUIs

All of this is possible thanks to the most recent advancements in artificial intelligence (AI), natural language processing (NLP) and natural language understanding (NLU).

NLP is the technique of transforming the voice, as an audio input, into text. Achieving low error rate is crucial for the VUI systems. The error rate of professional transcribers is about 5.9% [21] for the Switchboard corpus, which is a data set of 2,400 two-sided telephone conversations among 543 speakers. The use of artificial intelligence recently made the word error rate to drop below 5%, surpassing the human accuracy 2.4.

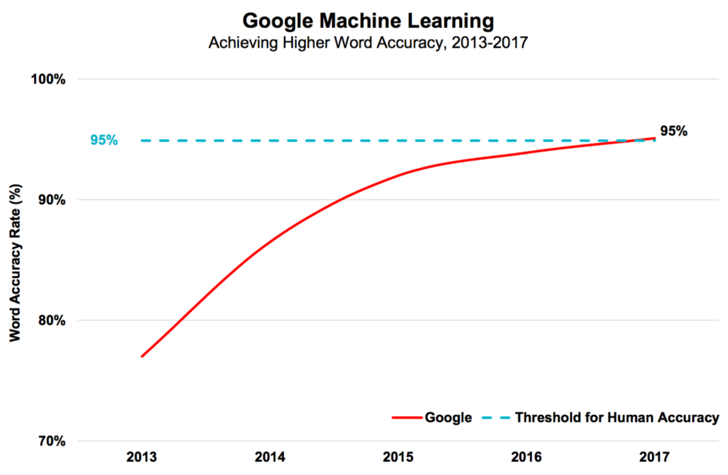


Figure 2.4: Advances in voice recognition accuracy [3]

The process of disassembling and parsing an unstructured input, in the case of VUIs, the natural language query, and producing a structured interpretation of the meaning behind the query is a big challenge. For example, there are dozens of ways how to ask about the current time, but the intention is always the same.

This is considered to be an AI-hard problem, meaning, solving this problem means making computers as smart as people [22].

2.2.1.3 Market status

Advances in voice recognition have fueled the growth of VPAs and their commercial applications. According to a survey by voicebot.ai [23] conducted in January 2018, 1 in 5 adults in the United States has access to a smart-speaker, which in most cases means there is a smart-speaker in their home.

The smart speaker is the fastest growing consumer technology we have seen in recent years, surpassing highly popular virtual and augmented reality or even wearable devices, according to a press release by Canalys [24]. There are expected about 56 million smart-speaker devices to be shipped in 2018, compared to a little over 30 million in 2017.

In the 2017 holiday season, the top 2 selling devices across all categories were Amazons Alexa devices, with tens of millions units sold, as stated in the Amazon report [25].

The clear leader of the smart-speaker market is in fact Amazon with it's Alexa platform with about 72% market share, followed by Google Home with a little over 18%.

The preference Americans have shown for Alexa is considered not to be founded in the products superior performance, but thanks to the 2-year headstart on Google Home. In fact, as Karen Hao is pointing out [4], a study conducted by digital marketing firm 360i, Amazons Alexa is significantly worse in executing commands and responding to questions across various topics, compared to Google Assistant as illustrated on figure 2.5.

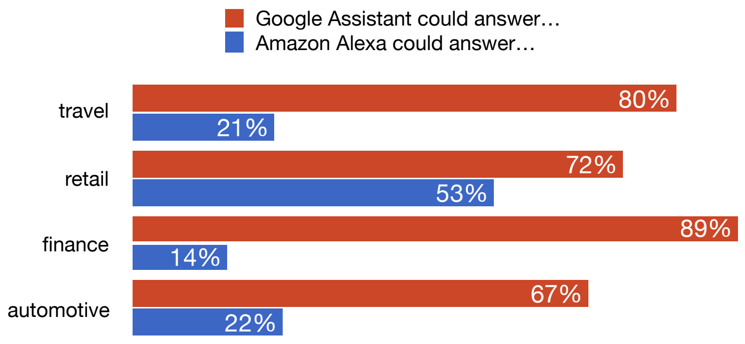


Figure 2.5: Assistant outperformed Alexa in every category of questions [4]

Amazons dominance is considered to be thanks to the 2-year headstart on Google Home. This shows that the virtue of being first is a tremendous advantage. For the current users, Alexa is simply good enough.

Amazon has the lead in the smart-speaker market, however, on the mobile platform, there is more competition. Ease of use is the key to success. The integration of assistants right into the operating system is a huge advantage for companies developing them. Apple on iOS announced there were 375 million monthly active Siri users globally, which is more than 3/4 of all iPhone users [26].

On Android, about 51 million users have tried the Google Assistant which represents about 43% devices supporting this technology [23], as visualized on figure 2.6. Even though Apple is here the clear leader, Siri has had a huge, almost, 5-year headstart and Google Assistant is catching up fast.

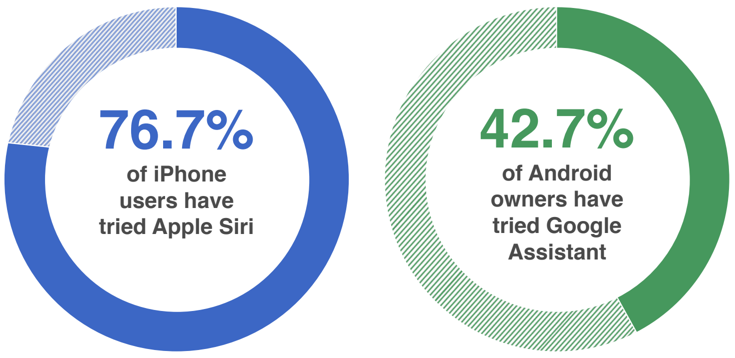


Figure 2.6: Voice Assistant Trial Rate on Smartphones

This shows that VPAs are much more accessible from smartphones, however, the comparison of usage between smartphones and smart-speakers points out that frequency of use of assistants through smart-speakers far exceeds use on smartphones, as displayed on figure 2.7.

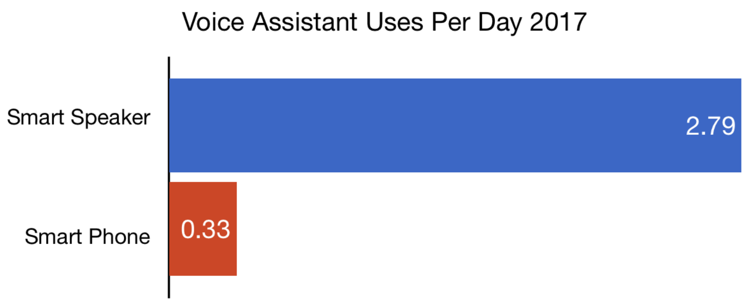


Figure 2.7: Smart speakers are used much more frequently [5]

According to a report from May 2017 conducted by Verto Analytics [27], personal assistance apps on smartphones are used on average 10 times a month, this translates to 0.33 times per day.

In comparison, smart-speakers are used about 2.8 times a day, or about eight times more frequently than assistance apps on smartphones, as stated by Voicebot.ai [5].

Based on all of this insight, I have decided to build the educational stock investments application OttoBot on the Alexa platform.

Amazon Alexa is the clear leader in the smart-speaker market, with about 50 million users just in the US [24]. The lack of presence on the mobile platforms is not a big issue for OttoBot since there is not as much need for this product, considering the vast options of mobile applications from the same domain.

2.2.2 The new world of human-computer interaction

For many years the human-computer interaction (HCI) was mediated through graphical user interfaces. In the past 2 decades, a huge effort has been put into improving the experience with better GUI design.

Nowadays for many users, natural language is already the default mode of interacting online, this is mostly happening in the form of human to human (H2H) interaction via hugely popular messaging, but we can see this trend in human to machine (H2M) interaction as well.

Many tech companies see chatbots and natural language user interfaces as the next big thing. Microsoft CEO Satya Nadella linked the transition to natural language interfaces to the revolution like the introduction of GUIs, Web and mobile internet.

This rapid shift to H2M VUIs means application developers need to drastically rethink the design of user interfaces and HCI to be suitable for voice.

2.2.2.1 Design for conversation

As noted by Følstad and Brandtzæg [28] VUI applications need to be designed for conversation from the ground up.

The designer's repertoire is greatly reduced, there is no swiping, scrolling, no buttons or images. The visual aids are very limited or non-existent. The features and content of the underlying service are mostly hidden, in fact, the interaction is much more dependant on the user's input. That's why the application design needs to move from the exploratory approach, where users explore what content and futures are available thanks to GUI.

The new HCI paradigm should be the goal-oriented approach, understand what the user needs and how he should be served best to bring most value as dialog evolves. The success depends on how well the application supports conversation process while providing useful output.

A big part of this is error recovery. Considering infinite ways how the user can formulate the request, failure is inevitable in some cases. The important part is a well-designed error recovery, to get back on track without frustrating the user.

2.2.2.2 Design as a multi-agent system

The current HCI approach with graphical applications often considers only one user.

One of the main interfaces for VUI applications are smart-speakers. Typically these devices are shared by multiple people, like people in a household or office. It is important to design voice applications with this aspect in mind. The design has to consider all the ethic and privacy issues that arise from multiple users using the same device [28].

2.2.2.3 Opportunities

There are several big opportunities to take advantage of when utilizing VUIs and VUI design.

Fight digital divides New technologies, unfortunately, tend to broaden the digital divides across age groups, genders, and social status. Only the most tech-savvy users are willing, or even are capable, to adopt the newest technologies.

Voice user interfaces, thanks to natural language interaction, could help to combat this undesirable tendency [29]. In fact, Verto Analytics data [27] shows, that women use personal assistants slightly more frequently than men. And interestingly, older people tend to use assistants more often. When compared to the adoption of other new technologies, this drastically different demographics.

Personalized Generate most relevant responses based on personal needs, preferences and digital literacy thanks to profiling of each user.

Data-driven improvements When interacting with graphical applications, user click on available buttons. Using VUI applications is not like that, user talks to the app what he thinks is the right way of query formulation. It's quite simple to collect feedback if he lands in an error state or does not get a satisfactory result. This enables massive volumes of data to drive the design improvements.

2.2.3 VUI and OttoBot

NLP and NLU just recently improved in a way that VUI are capable of providing real added value to the users of voice-enabled applications.

In the context of the OttoBot application, leveraging this new technology is a great way to get the general public interested in stock market investing and educate them on this topic. This is achieved by lowering the amount of effort necessary to get the important information, both for anyone who wants to learn about investing and for more experienced investors.

Lowering the entry-barrier for anyone to get into stock investing is critical to motivate the general public to educate themselves on this topic. Providing an easy to use application users can talk to, does just that. For someone to learn about investing, there is no need to show an overwhelming amount of data, serving just the right information is much more beneficial.

For investors, it is critical to stay up to date with the latest information and news, to be able to react appropriately. This can be very tiresome and time-consuming. The goal would be to have an assistant to provide the insights and report investments status. That is exactly what the OttoBot app strives to provide.

Chapter 3

Analysis

This chapter examines the existing solution and their functionalities, compared to the OttoBot application, presents the requirements for the product, analyses the financial data sources and Alexa programming interface and finally introduces the vision and business model.

3.1 Existing solutions

The intention of the project is to build an Amazon Alexa voice-enabled application for the purpose of educating users on the stock market and providing useful information about the traded companies, as well as option to create a virtual portfolio for easy access of favorite companies.

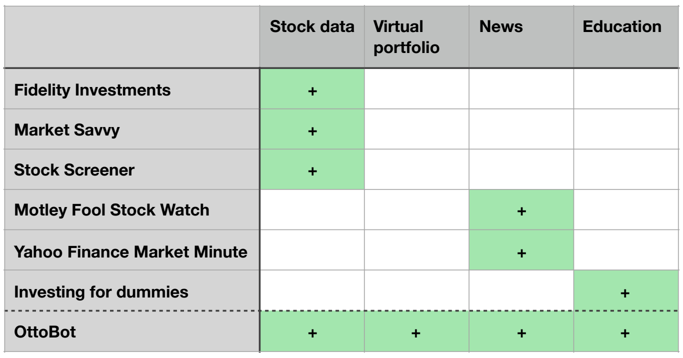
Because of these specifications, only applications for the Amazon Alexa platform, known as Alexa Skills where considered.

There are many Alexa Skills in the domain of finance and investing, however, the vast majority of these Skills are just products of developers experimenting with the Alexa platform. Thus most of the Skills do not provide much value if any at all.

All of the existing relevant applications could be divided into 3 categories, based on the type of information they serve:

1. Stock factoid data
2. News
3. Educational content

No Skills can be found on the Alexa platform, which would cover all of the functions of OttoBot. Below all the relevant applications are briefly described and a comparison with OttoBot is presented in table 3.1.



	Stock data	Virtual portfolio	News	Education
Fidelity Investments	+			
Market Savvy	+			
Stock Screener	+			
Motley Fool Stock Watch			+	
Yahoo Finance Market Minute			+	
Investing for dummies				+
OttoBot	+	+	+	+

Figure 3.1: Comparison of functions of competition and OttoBot

3.1.1 Stock factoid data

Existing Skills that might be considered as competition in this category provide hard information about the traded stock, like close price, market capitalization etc.

Fidelity Investments This might be the closest competitor to the anticipated OttoBot application. The Fidelity Skill can report last close price of a traded stock, give a report on the whole market state by reporting the DOWJ, NASDAQ, S&P 500 indexes, and report the markets biggest stock gainers and losers. There is no option to assemble a portfolio, get stock news or get explained the investing terms. This Skill is focused on more experienced investors. The biggest complaints about the app are: Skill is too verbose or even annoying, prices are not real-time, no possibility to assemble a virtual portfolio.

Market Savvy This Skill focuses solely on reading the data from the stock quote card, like close price, open price 52 week low etc. Unfortunately, this app does not provide any information which would give the user more insight into the company.

Stock Screener The purpose of this application simple, to serve real-time stock price information, there are no other features to this Skill.

3.1.2 News

Skills in this category can deliver news from the finance domain.

Motley Fool Stock Watch Serves users a report of the news in the stock market investing domain as an audio recording. This report is provided by a multimedia financial-services company Motley Fool, it is about stock investing in general and is updated on daily basis.

Yahoo Finance Market Minute This application is very similar to the Motley Fool Stock Watch Skill, the difference is in the source of the information, which is, in this case, Yahoo Finance. Other than that, this Skill has no other functions.

3.1.3 Educational content

This category contains Skills that provide educational content about stock market investing, like trading terms explanations.

Investing for dummies Upon triggering this Skill, the application tries to educate users by randomly explaining one of 100 investing terms.

3.2 Requirements

Requirements for the stock market investing helper app can be divided into two categories. The first category is functional requirements, stating the behavior of the application's functions. The second category is quality attributes of the application, judging the operation of the system, like scalability.

All of the requirements have requirement levels assigned as described in protocol RFC 2119 [30].

3.2.1 Functional requirements

The functional requirements were formulated based on the thesis instructions, the insights gained during the research of the stock market, conducted as part of the thesis, investigation of existing solution and agreement with the thesis supervisor. All of them are categorized and described below.

The requirements with the requirement level MUST are annotated by F#, where # represents a sequential number of the functional requirement.

3.2.1.1 General

Among the general functional requirements is the support for at least a dozen publicly traded companies on the US stock market. Furthermore, the application should support multiple users, but there is no need for support of multiple users on one device at the same time.

These are the desired functions:

- include at least a dozen publicly traded US companies; F1.1 [MUST]
- support multiple users on different devices; [SHOULD]

3.2.1.2 Traded company information

The application allows users to ask about the information regarding a publicly traded company. The most basic information the application provides is the stock price, this should be not older than the last market close price. Other information about the stock quote will be a welcome future.

The functions are:

- get stock price; F2.1 [MUST]
- get market capitalization; [FUTURE]
- get P/E ratio; [FUTURE]
- get dividend yield; [FUTURE]

3.2.1.3 Virtual portfolio

Allow users to create a watchlist of companies to resemble investing portfolio, to get quick access to the favorite companies. The user has to be able to add stocks to the portfolio and remove them.

An option for a quick report about the stocks in the portfolio has to be provided, to find out about the biggest gainers and loser in the watchlist.

A synchronization of the portfolio across multiple Alexa devices would be a nice future.

These are the futures:

- report portfolio performance; F3.1 [MUST]
- add a stock to the portfolio; F3.2 [MUST]
- remove a stock from the portfolio; F3.3 [MUST]
- a synchronization of the portfolio across multiple Alexa devices; [MAY]
- store portfolio on a server in case the user loses access to the device; [MAY]

3.2.1.4 News

Give the users the option to get the latest news about a publicly traded company, to inform them what is being published about the business in order to help them make better investing decisions.

The features regarding news are the following:

- present recently released article title mentioning the company; F4.1 [MUST]
- send a condensed version of the article to a device with a screen; [MAY]

3.2.1.5 Education

The application should be able to explain some of the investing terms to educate the user about investing. In the future, it could provide more educational content from the stock market like investing strategies.

The futures:

- explain investing term; [MAY]
- present and explain investing strategies; [FUTURE]

3.2.1.6 Authentication

Authenticate the user via service like Facebook in order to store user-specific data on the server and be able to map them back to the user.

These are the futures:

- implement Facebook login; [MAY]
- implement Google login; [MAY]

3.2.1.7 Investing strategy portfolio

In the future, the application could suggest an investment portfolio based on the market strategy selected by the user and historical data.

Futures:

- portfolio based on investment strategy and historical data; [FUTURE]

3.2.2 Quality attributes of the application

Here are listed the non-functional requirements objecting the quality of the application. These were formulated based on the thesis instructions and consultation with the thesis supervisor.

- Q1 - Alexa platform implementation
- Q2 - intuitive user interface
- Q3 - possibility for implementing new futures
- Q4 - scalability
- Q5 - low latency to complement a pleasant user experience

3.2.2.1 Analysis of requirement Q1

As stated by the thesis instructions the application has to be implemented on the Alexa platform using the Alexa Skills Kit (ASK) application programming interface (API).

3.2.2.2 Analysis of requirement Q2

To achieve pleasant user experience, the voice user interface design principles have to be taken into consideration when building the product.

3.2.2.3 Analysis of requirement Q3

It should be fairly easy to extend the capabilities of the app by implementing new futures.

3.2.2.4 Analysis of requirement Q4

The application has to be built on easily scalable services in order to ensure easy scalability in case of increased popularity.

3.2.2.5 Analysis of requirement Q5

For the purpose of convenient interaction with the application, the latency on the service responses has to be reasonably low.

3.2.3 Processes, use-case and scenarios

Modeling of the processes, use-cases and scenarios is based on the second[1] and third[2] lecture of the course Software Engineering 1 at FIT CTU.

3.2.3.1 Processes

Processes closer illustrate the interaction of the user with the application and better describe the requirements of the system. All of the processes assume the OttoBot Skill is activated on the device, this is done by saying "Alexa open OttoBot".

P1 - Get stock price Modeled in the activity diagram in figure 3.2.

For finding out the price of a traded stock, the user has to formulate a question or command from which it is obvious that he wants to know about the current value of the stock. This query has to include the stock ticker.

For a subset of tickers, even saying the company name instead of the ticker is supported.

If the query is correctly recognized and the ticker symbol is in the stock database, the application responds with the current price of the stock, otherwise, the application will inform the user that it can not get data for this stock.

[1] https://edux.fit.cvut.cz/courses/BI-SI1/_media/lectures/02/02.prednaska.pdf
[2] https://edux.fit.cvut.cz/courses/BI-SI1/_media/lectures/03/03.prednaska.pdf

P2 - Add stock to the portfolio Modeled in the activity diagram in figure 3.3.

In order to add new stock to the portfolio, the user needs to express this intention in a query including the stock ticker.

The user has to be authorized, otherwise the app will inform the user about this constraint and cancel the action.

If the ticker is not understood correctly or is not among the supported stocks, the application will inform the user. After the app recognized the ticker, the ticker will be repeated back to the user together with a request for confirmation to finish the operation. The user can confirm the action by a positive response, this will add the stock to the portfolio. In the case of negative response, the stock will not be added to the portfolio.

P3 - Remove stock from the portfolio Modeled in the activity diagram in figure 3.4.

To remove stock from portfolio user asks the application with proper query clearly showing his intentions, including the ticker symbol of stock to be removed from the portfolio.

If the stock is currently included in the portfolio, the user will be asked for a confirmation of the removing action. Otherwise, the app will inform the user that the removing operation can not be completed. If the user confirms the action with a positive response, the stock will be removed from portfolio, if not, the operation will be canceled.

P4 - Report the portfolio status Modeled in the activity diagram in figure 3.5.

The user has to tell the app a query which indicates clearly the intention of getting a portfolio report.

If the portfolio is empty, the user gets a response informing him about this matter. In case there are stocks in the portfolio, a response informing about the performance of the stocks in portfolio will be generated and presented to the user.

P5 - Get news about a company Modeled in the activity diagram in figure 3.6.

To get the news about some company, the user has to formulate a query indicating his intention, including the ticker symbol of the company he wants to get the news about. If the ticker is not supported, the app will inform the user by presenting an error message.

If the ticker is recognized, the app will generate a response mentioning multiple titles of articles mentioning the company and ask if the user wishes to get more information on one of the articles. If the user's answer is affirmative, the app asks which of the articles he wants to know about. The user responses.

The expected response is an ordinal value. If the answer is recognized as correct, the article is sent to the Alexa application, otherwise, the user is informed that article was not found.

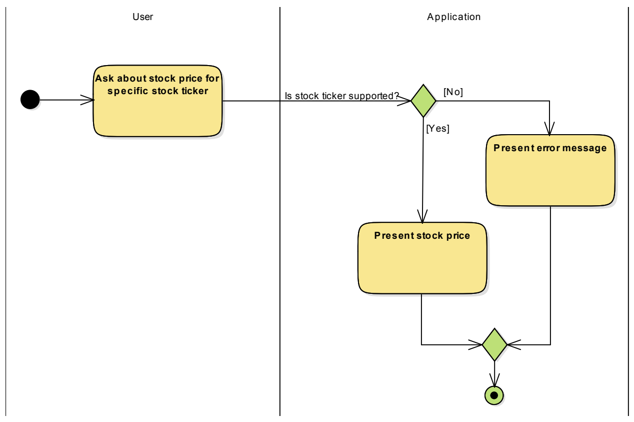

User
Application
Ask about stock price for specific stock ticker
Is stock ticker supported?
[No]
[Yes]
Present error message
Present stock price

Figure 3.2: Process P1 - Get stock price

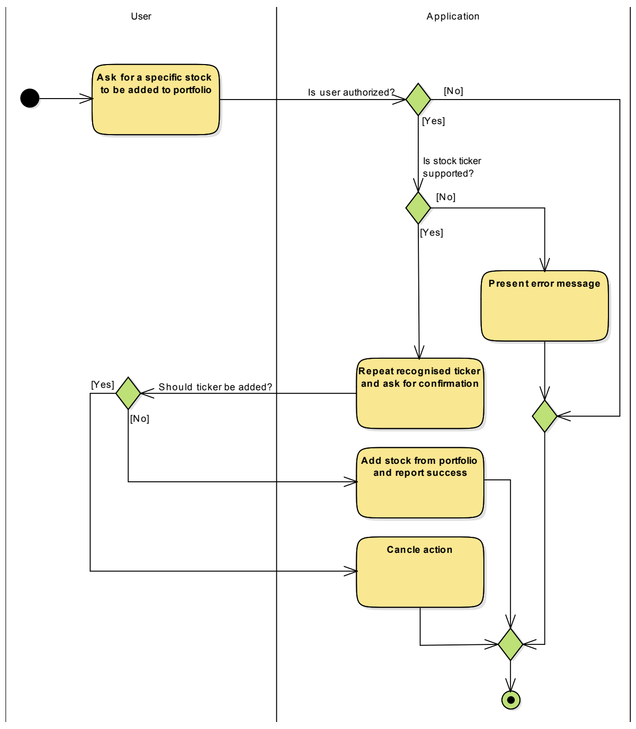

User
Application
Ask for a specific stock to be added to portfolio
Is user authorized?
[No]
[Yes]
Is stock ticker supported?
[No]
[Yes]
Present error message
Repeat recognised ticker and ask for confirmation
[Yes]
Should ticker be added?
[No]
Add stock from portfolio and report success
Cancle action

Figure 3.3: Process P2 - Add stock to the portfolio

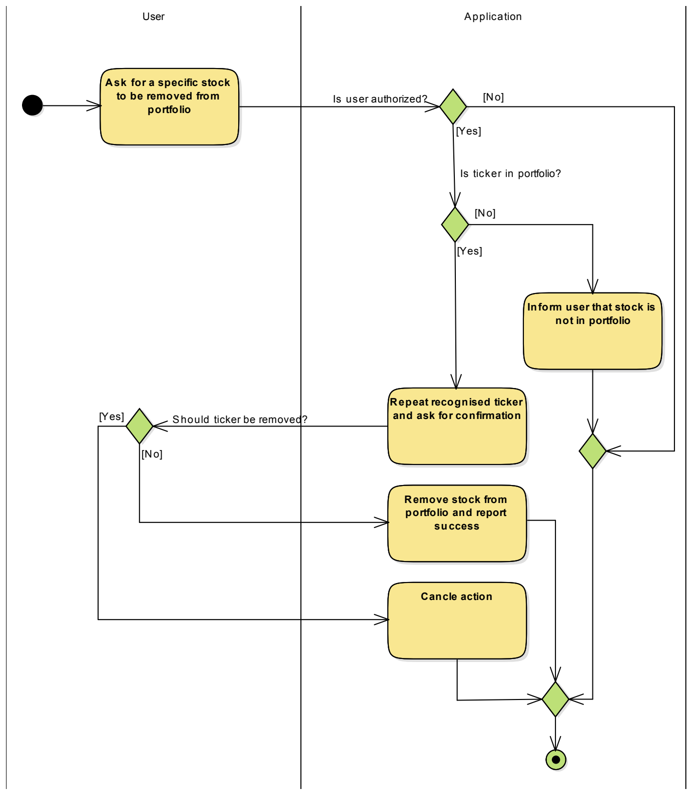

User
Application
Ask for a specific stock to be removed from portfolio
Is user authorized?
[No]
[Yes]
Is ticker in portfolio?
[No]
[Yes]
Inform user that stock is not in portfolio
Repeat recognised ticker and ask for confirmation
[Yes]
Should ticker be removed?
[No]
Remove stock from portfolio and report success
Cancle action

Figure 3.4: Process P3 - Remove stock from the portfolio

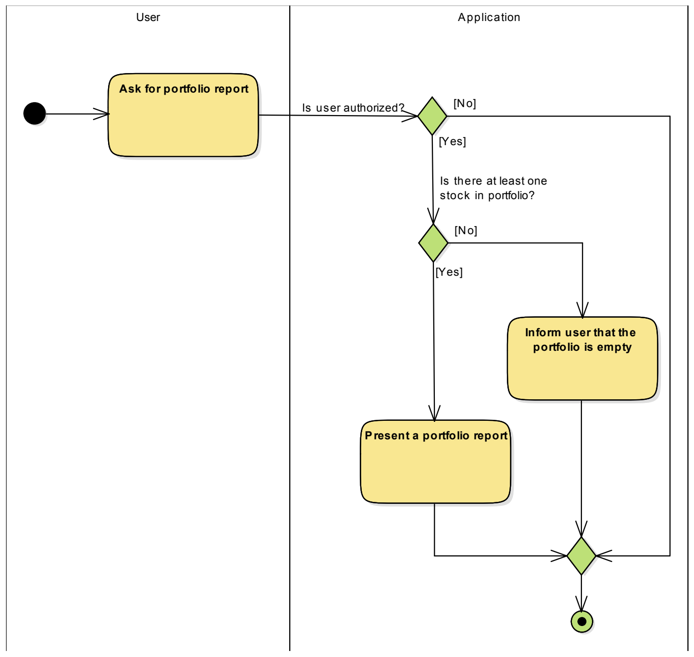

User
Application
Ask for portfolio report
Is user authorized?
[No]
[Yes]
Is there at least one stock in portfolio?
[No]
[Yes]
Inform user that the portfolio is empty
Present a portfolio report

Figure 3.5: Process P4 - Report the portfolio status

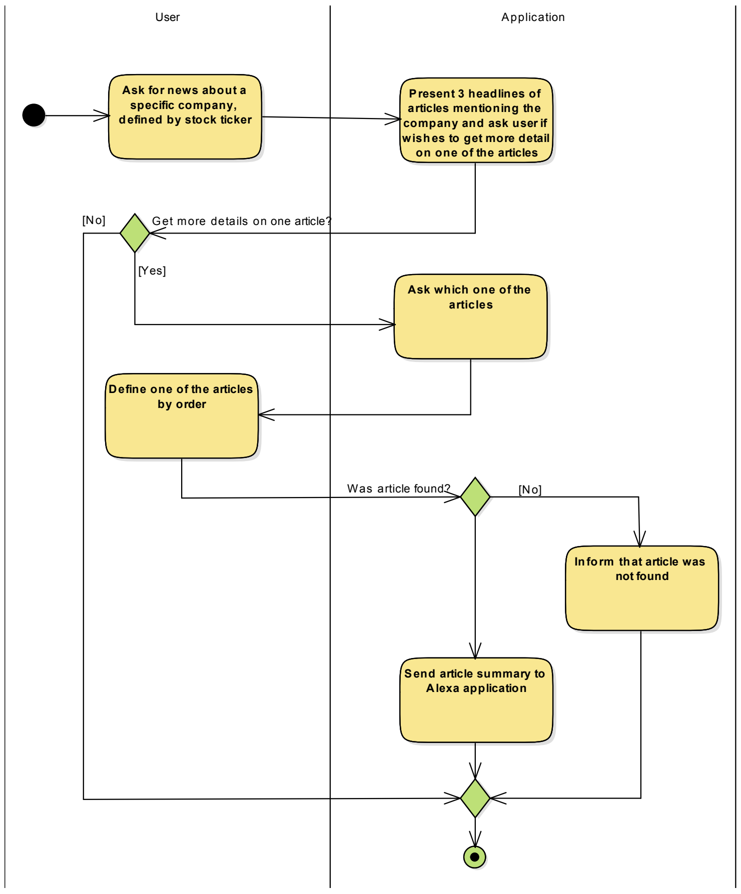

User
Application
Ask for news about a specific company, defined by stock ticker
Present 3 headlines of articles mentioning the company and ask user if wishes to get more detail on one of the articles
[No]
Get more details on one article?
[Yes]
Ask which one of the articles
Define one of the articles by order
Was article found?
[No]
Inform that article was not found
Send article summary to Alexa application

Figure 3.6: Process P5 - Get news about a company

3.2.3.2 Actors

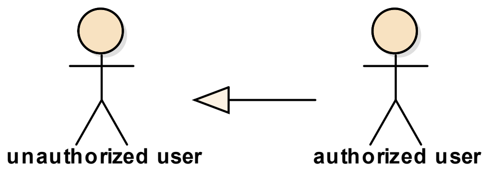


Figure 3.7: Actors diagram

There are 2 actors distinguished in the use cases, one being authorized user, the other unauthorized user, as shown in figure 3.7. An authorized user has all the privileges an unauthorized has, additionally he can perform actions the unauthorized user is not capable of. An unauthorized user can become authorized by linking the account in the Amazon Alexa Skills settings.

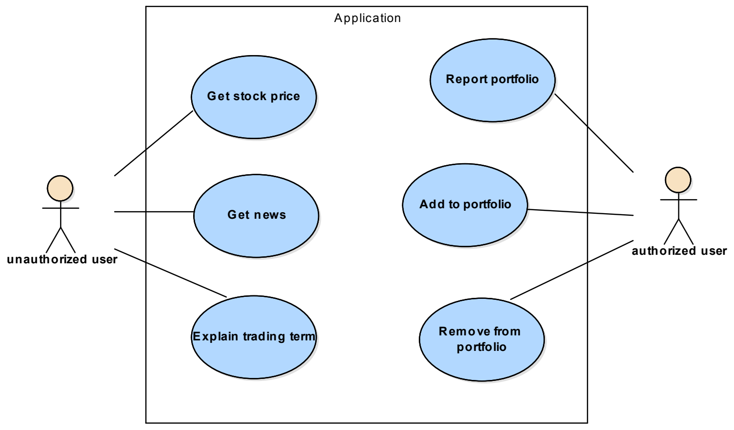


Figure 3.8: Use-case diagram

3.2.3.3 Use-case diagram

As the use-case diagram on figure 3.8 shows, an unauthorized user can ask for the stock price, get an investing term explained or request the latest information for a traded company.

An authorized user is additionally allowed to interact with a virtual portfolio. The user can add stocks to the portfolio, remove them and get a brief report on the portfolio's stock performance.

Below is a description of the examined use-cases, annotated by UC#, where # represents a sequential number of the use-case.

All of the use cases and particularly use-case scenarios assume that the OttoBot application is already open on the device.

3.2.3.4 UC1 - Stock price

The user can query the stock price of any supported stock by including the stock ticker in the command. For a subset of the stocks, even saying the company name, instead of the ticker symbol is allowed.

Preconditions

- no preconditions

Scenarios The only scenario for this use-case is querying the stock price by saying a command from which is the intention of getting the stock price clear.

Main scenario

1. The use-case scenario begins with user ordering the app to provide stock price information. The command has to be structured in a way so that

it is obvious that user is asking about stock price and must contain the stock ticker.

2.
   - stock ticker supported: The application responds with the stock price.
   - otherwise: An error is presented to the user.

Process mapping

- P1

3.2.3.5 UC2 - News

Allows the user to ask about the latest news on a supported publicly traded company, by formulating a query with clear intention for the news information.

Preconditions

- no preconditions

Scenarios The only scenario for this use-case is ordering the app to provide the latest news about a publicly traded company. The desired action should be apparent, and the command should include the stock name of the company, as identification. The user is then presented with three headlines of articles mentioning the company and asked if he wants more information on one of the articles. If he confirms, he is asked which of the articles, otherwise the action ends. If the answer is recognized successfully, the article is sent to the Alexa app.

Main scenario

1. The use-case scenario begins with user ordering the app to provide news about a specific company. The command must contain the stock ticker.
2. The user is presented with 3 headlines of recently released articles about that company and asked if he wants more information on one of the articles.
3.
   - positive response: User is asked to specify one of the articles.
   - negative response: The action ends.
4.
   - success: The article is sent to the Alexa app.
   - failure: User is informed, that the article was not found.

Process mapping

- P5

3.2.3.6 UC3 - Report portfolio

The user can ask about the performance of his virtual stock portfolio, by formulating a command in natural language.

Preconditions

- no preconditions

Scenarios The main scenario for this use-case is asking the app to report the portfolio status, to do that, the user has to be authenticated. If there are no stocks in the portfolio, the user is informed about this situation.

Main scenario

1. The use-case begin with the user ordering the app to report the stock portfolio performance. This command has to be clear and on point.
2.
   - success: One by one, the stocks in the portfolio are presented to the user including their recent performance.
   - user not authenticated: User is informed, that he needs to authenticate in order to use this function.
   - portfolio empty: User is informed, that his portfolio is empty.

Process mapping

- P4

3.2.3.7 UC4 - Add stock to portfolio

The user can add a stock to his virtual portfolio by formulating a command in natural language, which indicates this action. The command has to include the stock ticker.

Preconditions

- user is authenticated (otherwise he is informed that he needs to setup account linking)

Scenarios The main scenario for this use-case is ordering the app to add a stock to the portfolio, in order to do that, user has to be authenticated. If the stock is already in the portfolio, user is informed about that.

Main scenario

1. The use-case begin with the user ordering the app to add a specific stock to portfolio. The stock name has to be included in the command.
2.
   - success: The stock ticker is repeated to the user and confirmation is requested.
   - stock already in portfolio: User is informed that his portfolio already contains this stock.
3.
   - confirmed: Stock is added to the portfolio and a success message is presented to the user.
   - cancel: Stock is not added to the portfolio.

Process mapping

- P2

3.2.3.8 UC5 - Remove stock from portfolio

The user can remove a stock from his virtual portfolio by formulating a command in natural language, which indicates this action. The command has to include the stock ticker.

Preconditions

- user is authenticated (otherwise he is informed that he needs to setup account linking)

Scenarios The main scenario for this use-case is asking the app to remove a stock from the portfolio, in order to do that, user has to be authenticated. If the stock is not in the portfolio, user is informed about that.

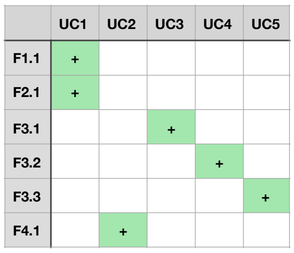
Main scenario

1. The use-case begin with the user ordering the app to remove a specific stock from portfolio. The stock name has to be included in the command.
2.
   - success: The stock ticker is repeated to the user and confirmation is requested.
   - stock not in portfolio: User is informed that his portfolio does not contain this stock.
3.
   - confirmed: Stock is removed from the portfolio and a success message is presented to the user.
   - cancel: Stock is not removed from the portfolio.

Process mapping

- P3

3.2.3.9 Functional requirements coverage check



	UC1	UC2	UC3	UC4	UC5
F1.1	+				
F2.1	+				
F3.1			+		
F3.2				+	
F3.3					+
F4.1		+			

Figure 3.9: Check of the coverage of functional requirements by use-cases

The table in figure 3.9 shows the use-case coverage of all functional requirements with the requirement level MUST.

3.3 Financial information APIs analysis

In this chapter some of the most popular financial application programming interfaces (API) will be examined, to find out which is best suited for the OttoBot project. These data sources are Alpha Vantage, Google Finance API, and Quandl.

There are several expectations about the stock market data sources:

- **Free to use** - for the purpose of providing the OttoBot application free of charge;
- **Provides information about stocks** - like stock price;
- **Targets stocks traded on the US exchanges** - because the application targets the US market American companies should be supported;
- **Stock price updated at least daily** - in order to serve relevant information to the users. A real-time stock price would be ideal, services usually charge for this functionality;
- **Provides historical stock price data** - for the purpose of reporting price changes to the user;
- **Fast** - to deliver seamless conversational experience without significant delays;

Alpha Vantage This is one of the most popular financial data sources, mainly because of great documentation and easy usage, all without any charges.

The biggest advantage of this Alpha Vantage is support for real-time stock price querying. The API provides many query modification options, this enables easily querying daily, weekly or monthly stock price information.

The drawback with the API are the limitations in terms of batch requests, for getting data for multiple stocks at once. For batch requests, it is only possible to get data for the current day. This is insufficient for the use case of reporting price changes in a portfolio, instead, a request for each stock in the portfolio has to be sent to achieve the desired functionality.

Google Finance API Unfortunately this free API became deprecated during the development of the OttoBot application, currently, Google Finance API has not been stable since late 2017. Requests seem to fail at random or take much longer to be handled.

On the other hand, this data source supports batch requests even for historical data.

Quandl Quandl is a platform providing data across multiple domains, including finance and investing. For the purpose of the projects, the most suitable data set is WIKI Prices, which is free of charge.

The Wiki Prices dataset is updated daily at 9:15 afternoon Eastern Standard Time and covers more than 3,000 US tickers, which is good enough for the purpose of this project.

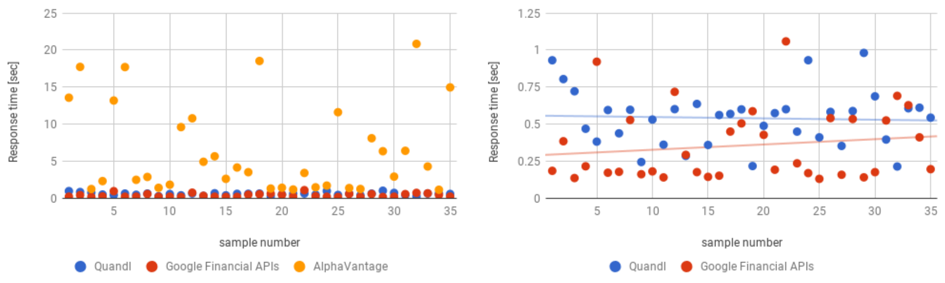
Even batch requests are supported. Unfortunately, the API does not seem to be very reliable, when implemented in the Alexa Skill, the requests got timed-out in some cases.

3.3.1 Comparison

In order to carefully test the performance of the APIs, a testing Python script was created. Three request types were examined:

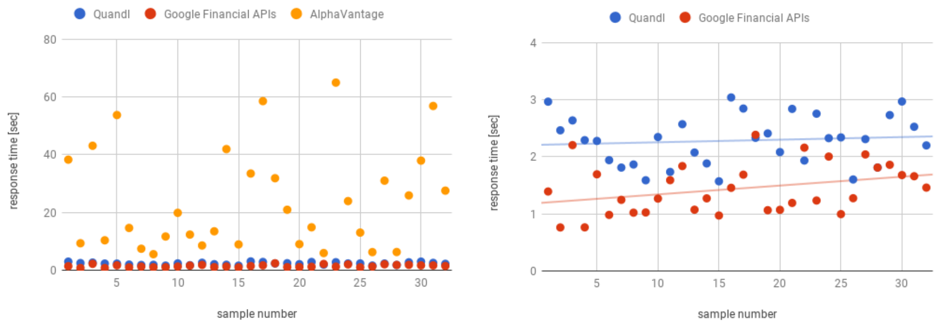
- **single quote** - query stock price for given ticker symbol
- **multiple single requests** - query stock price of 5 stocks by sending a single request for each stock in sequence
- **batch request** - query 5 stocks at once with one request

Over 30 requests of each type were sent to each of the APIs and the response time was recorded.



(a) (b)

Figure 3.10: Datasource performance - single query test



(a) (b)

Figure 3.11: Datasource performance - sequence query test

Single quote As shown on the figure 3.10a, the Alpha Vantage API performance is very unstable, some of the requests take up to 20 seconds, which is greatly insufficient for use in the project.

On the plot 3.10b, only Quandl and Google Finance are displayed in order to get better scale. Here, the Quandl data source is on average a little over 0.5 second which would be reasonable, but in some cases, the response time is close to 1 second. Regarding the request for a current stock price is considered to be very frequent in the application, response time around 1 second is inadequate.

On the other hand, the Google Finance API performs on average a little bit better.

Sequential requests Regarding sending multiple single requests in sequence, the Alpha Vantage API is incredibly slow as shown on plot 3.11a.

The comparison of Quandl and Google Finance is shown on plot 3.11b. Both of the data sources are too slow, using this approach would not provide good conversational experience.

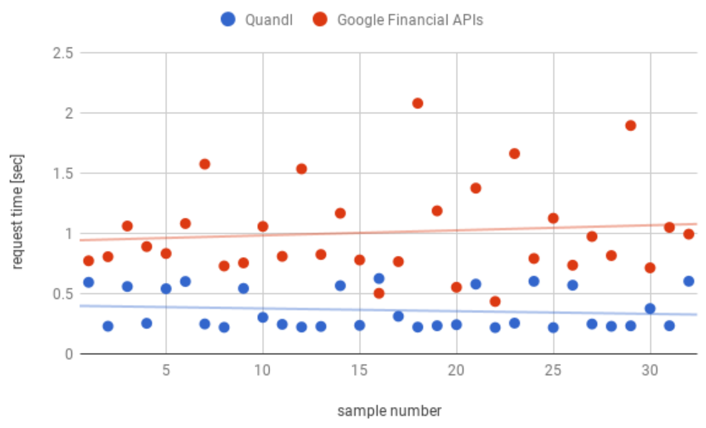


Figure 3.12: Datasource performance - batch query test

Batch request Batch requests are only supported by Google Finance and Quandl, the performance of these APIs is shown on plot 3.12.

Google Finance shows inconsistent performance with many samples over 1.5 seconds response time. Quandal performs notably better with response time somewhere around 0.5 seconds.

Conclusion The Quandl API seems to be the best performing overall but is still unreliable and can be in some cases slow, this is an issue especially for the single stock requests.

The biggest flaw with the Google Finance API is the fact it is no longer supported and all the reliability issues may even increase and cause more problems, additionally, Google might very likely shut the API down entirely.

Alpha Vantage as a data source is lagging in all categories and is entirely insufficient for the use in the OttoBot project.

Mostly because the latency issues, creating own database of traded stock prices appears as the best solution. The Quandl platform provides API to download the Wiki Prices dataset, this can be used to populate own database with historical data. Additionally, a data scraper needs to be created which will update the database with the newest stock prices every day, once the data is available. This approach will greatly reduce the data retrieval latency.

3.4 Alexa Skills Kit

The role, responsibilities, capabilites and concepts of the Alexa Skills Kit APIs (ASK) will be discussed below.

The ASK is a set of tools, APIs, and documentation that empowers developers to create voice-enabled applications for the Alexa platform. These applications are called Skills.

The Alexa platform is based on the AWS public cloud and the Skills are distributed through the Alexa Skills Store.

The Alexa Skill consists of 3 basic components and the ASK APIs facilitate the interaction between these components. The communication process consists of 6 steps. To introduce this process, below is a walkthrough for the query "What is the current price of Apple stock?", as presented in figure 3.13.

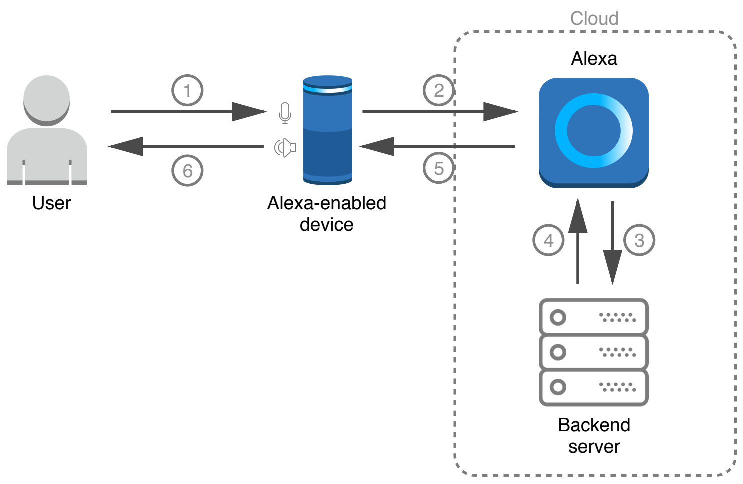


Figure 3.13: Communication with the ASK components, inspired by [6]

Communication flow description for figure 3.13:

1. The user says: "What is the current price of Apple stock?".
2. The Alexa-enabled device forwards the voice command to the Alexa service in the cloud.
3. Alexa service parses the command and triggers the backend service.
4. A response is generated on the backend server and passed back to Alexa in the cloud.
5. Alexa prepares the reply which is returned to the physical Alexa-enabled device.
6. Alexa-enabled device answers the query with a response like "Apple is currently trading for $162.32."

3.4.1 Capabilities

The key capabilities of ASK are fast and easy Skill development and distribution. The developer does not necessarily need to be experienced programmer to develop a simple Alexa Skill. The ASK API handles the natural language processing (NLP) and the developer just needs to define the interaction dialog and build program responsible for generating the responses.

Where ASK comes short are the means of monetizing the application. Currently, Amazon is trying to motivate developers by rewarding Skills with the highest engagement with rewards like AWS credit [31]. For now, it is not possible to charge the users for using the Skill. Fortunately, Amazon recently announced that in-app purchases[3] will be available in 2018.

One of the other limitations of ASK is the inconsistent performance of the voice recognition system. With growing feature list of the app, the possibility of an improper understanding increases. This can occur in similarly sounding words or commands. For example, when asking for stock price, as more companies get supported, similarly sounding company names can get mixed up. This is something Amazon is always working on and trying to improve, so better performance can be expected in the near future.

3.4.2 Development concepts

The Skill development process consists of four stages. These are: build, test launch, measure. All of these stages are inspected and configured in the ASK web interface. During this project, the focus was mainly on the build stage. The build stage itself is separated into 3 phases:

1. Skill setup
2. Interaction model creation
3. Development of the backend server

Skill setup When creating new Alexa Skill, first the developer needs to define the name. After that, the endpoint for connecting the Skill to the responses will be generated, this can be a Lambda function or HTTPS server.

Interaction model creation In the ASK web interface, the developer defines the interaction model. Here the invocation name is set. That is the trigger word or phrase the user will use to reference the Skill. Additionally, so-called Intents need to be modeled. Each Intent represents users intention for specific action for the application to perform. This could be getting information about stock price or reporting portfolio performance. Each of these interactions can contain variables called Slots, to capture important

[3]Possibility to purchase extra content or services inside the already installed application.

data pieces from the user's commands. When asking for the stock price, the slot could be the company name. When defining these Intents, it is necessary to provide sample commands how the users will invoke this intent. If there is a need for slots in the commands, they have to be defined in slot types. In the context of getting the stock price, a slot type might be a list of supported companies. This list contains all the companies which can be expected in the command. Based on the interaction model, the ASK service generates an NLP model capable of understanding these commands and representing them in a structured form for the backend.

Development of the backend server The backend is the programmatic logic for generating responses. The backend receives the structured command, including the intent name and slot name with its value and produces the response.

3.5 Vision & Business model

3.5.1 Vision

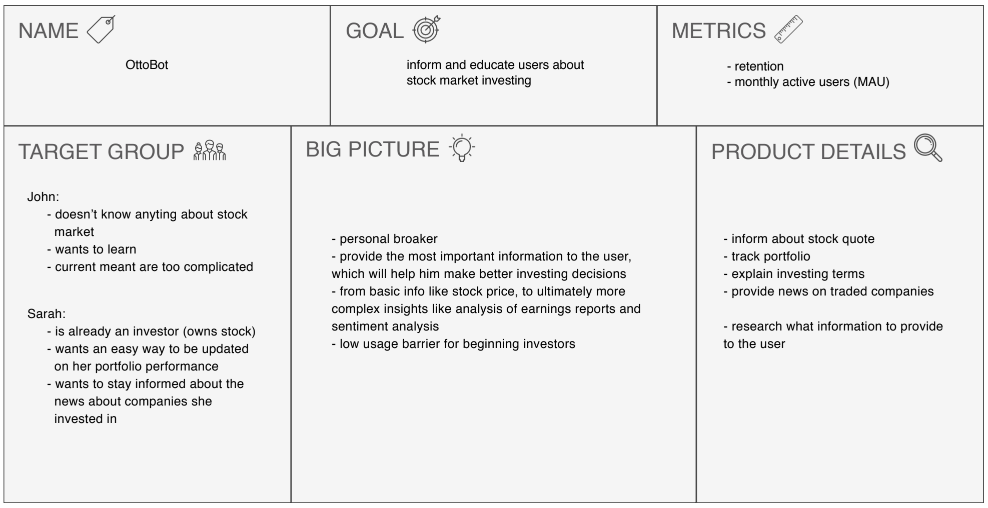
The vision for the OttoBot product is to build a voice-enabled application that provides the most significant trading information. To build a personal broker.

The app should present the information in a condensed an easily understandable way, to create a low entrance barrier for inexperienced users. Additionally, it should be able to educate the user and introduce him to the world of stock trading.

The product should fulfill the needs of 2 specific groups of people. The first potential user group is someone who does not know, much about trading and wants to learn about it. For this person, trading analyses are too technical and hard to understand. He wants hands-on experience. The second user group can be represented as someone who already owns some stock. He wants to track the performance of his portfolio and be kept up to date on the news about the traded companies to operate efficiently and promptly make informed trading decisions.

It is unfeasible to implement all of the functionality introduced in the vision already in the first version. It is essential to bring a product to the market and validates the vision. Section 3.2 outlines the futures intended to be supported in the first version.

Two primary metrics will measure the performance of the product. These key performance indicators (KPIs) are retention and the number of monthly active users (MAU). It is important to continually track how many users are using the app and if the users keep interacting with the app in the long term.


NAME

OttoBot

GOAL

inform and educate users about stock market investing

METRICS

- retention
- monthly active users (MAU)

TARGET GROUP

John:
- doesn't know anyting about stock market
- wants to learn
- current meant are too complicated

Sarah:
- is already an investor (owns stock)
- wants an easy way to be updated on her portfolio performance
- wants to stay informed about the news about companies she invested in

BIG PICTURE

- personal broaker
- provide the most important information to the user, which will help him make better investing decisions
- from basic info like stock price, to ultimately more complex insights like analysis of earnings reports and sentiment analysis
- low usage barrier for beginning investors

PRODUCT DETAILS

- inform about stock quote
- track portfolio
- explain investing terms
- provide news on traded companies

- research what information to provide to the user

Figure 3.14: OttoBot Product Canvas

The Product Canvas template was utilized, to shape the vision, set the target personas and pick the right features, inspired by Roman Pichler [32]. Figure 3.14 corresponds to the filled canvas.

3.5.2 Business model

The business model of the OttoBot application is divided into two phases. In the first phase, the application will not generate any revenue. That is possible thanks to the low operating cost of the product. For a limited number of users, the infrastructure for running the app is free of charge. The traffic up to 1000 concurrent users, which is a bottleneck set by the AWS Lambda cloud function (more on that in section 4.1), provides enough growth space to build and test the product. The application will be provided free of charge to build a strong userbase and validate the concept.

Once a substantial user base has been created, and the vision validated, an affiliate marketing monetization model will be implemented. Third party services will be marketed in a non-intrusive way through the application in exchange for a commission. For example, the app does not offer to buy or sell stock, which is something the users might want to do at some point. Suggesting a trading platform to do that, is providing the user with useful information and creating a valuable revenue stream at the same time. Some of the many potential partners are Plus500, eToro, Fidelity, and RobinHood.

Waiting with the affiliate marketing until the product has a strong userbase will secure a better negotiation position.

The userbase milestone is unclear at this point and will be set as soon as the product has been launched, and first metrics have been collected.

Chapter 4

System design

This section describes in detail the infrastructure and architecture of the system. The components and the communication between them will be illustrated extensively. The deployment of the software will be explained as well.

4.1 System infrastructure

From the beginning of the project, it was clear that the system will utilize cloud computing[4] in the form of Infrastructure as a Service[5] (IaaS), concretely the Amazon Web Services (AWS). This decision was made after consultation with the thesis supervisor.

Some of the many reasons are flexibility, scalability, security, integration with Alexa platform and expenses. Infrastructure build on AWS is very easily scalable based on the current needs, thanks to services like load balancers and autoscaling across many geographical regions. With the support of virtual private clouds (VPC), the system can be secured with private/public subnets and firewalls. The integration with the Alexa platform and Alexa Skills Kit APIs (ASK) is seamless and operating the system on a reasonably small scale is free of charge.

On figure 4.1, an overview of the system infrastructure is displayed.

The OttoBot application backend runs on the virtual private cloud OttoBotVPC on AWS, this is an infrastructure of services and platforms dedicated to the OttoBot application. For security reasons, the VPC contains 2 subnets, one private and one public, in 2 different availability zones[6].

The public subnet serves as an entry point into the application backend. Only the public subnet has an internet gateway all of the communication with the outer world is routed through it.

[4]On-demand access to shared pools of configurable computing resources [33].

[5]Type of cloud computing service providing computing infrastructure like computers, data storage space and networking [34].

[6]Isolated locations in a geographical region running the Amazon Web Services [35].

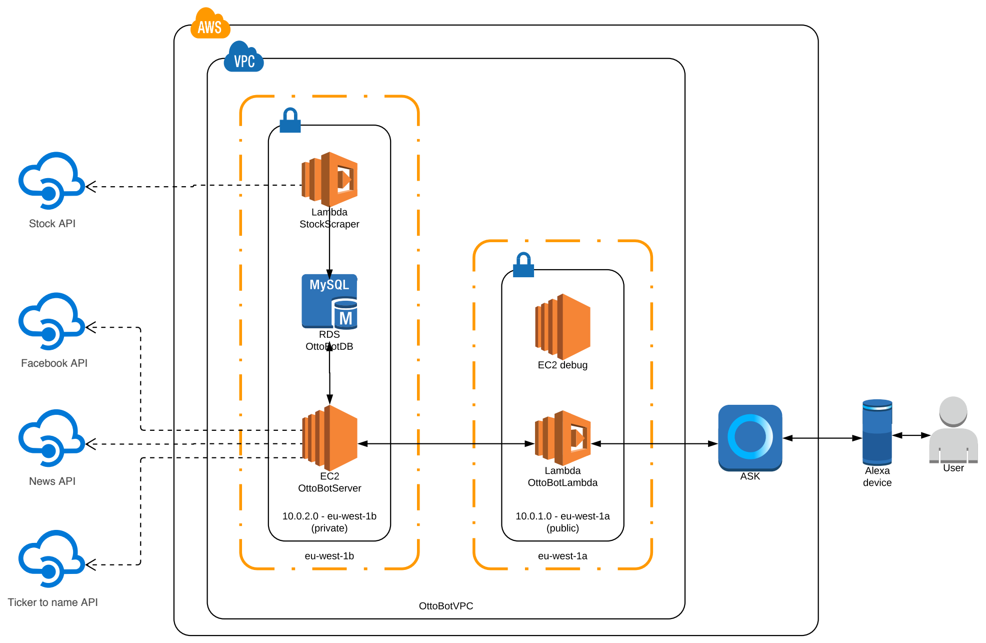


Figure 4.1: System infrastructure

Most of the OttoBot application backend runs on the private subnet, this is where the data scraping and response generation takes place. In cases where internet connection is needed, the requests are routed through the public subnet. For simplicity, connections to APIs on the left side of figure 4.1, are drawn directly out of the network, even though the communication is mediated by the public subnet.

The user interacts with the application via an Amazon Alexa enabled device. The device sends the requests to the ASK APIs to perform NLP and NLU on the command.

The ASK APIs extract the intents and variables, as described in chapter 3.4, and pass the request in form of a JSON format to the to the OttoBot-Lambda function inside the public subnet in the VPC. The AWS Lambda is an event-driven, serverless computing platform[7]. The only purpose of the OttoBotLambda lambda function is to forward the JSON in an HTTPS POST request to the OttoBotServer.

OttoBotServer is an EC2 instance[8], which is an easily scalable virtual server instance for computational purposes.

This EC2 instance is where an appropriate response to the user's request gets generated. Once all necessary data is gathered from all the data sources, a JSON response is generated and returned as an answer to the POST request.

[7]more information on AWS Lambda `https://aws.amazon.com/lambda`

[8]more information on AWS EC2 `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide`

The response goes all the way back to Alexa device where the generated message is presented to the user.

As mentioned in the Financial information APIs analysis in section 3.3.1, to remove latency bottlenecks, stock prices need to be scraped into a database. This is a relational database Amazon RDS named OttoBotDB. The database serves as a data source for stock prices, additionally, it stores users and their portfolios. The OttoBotDB is further described in section 4.3.

To scrape new stock prices into the database every day, a Lambda function named LambdaStockScraper is used, which pulls the latest stock prices from the Quandl WIKI Prices API. The LambdaStockScraper is triggered every day at 16:00 GMT to download the stock prices for the last day and save them to the OttoBotDB.

The lambda execution is invoked by a CloudWatch[9] Rule. The rule is a trigger for the LambdaStockScraper every day at 16:00 GMT. This keeps the OttoBotDB database updated on the stock prices.

Because of the fact, that the OttoBotServer EC2 instance is running inside a private subnet without a direct internet connection, there is a need for a middleman to access this instance for actions like remote login into (SSH).

For the administrative, debugging and monitoring purposes, there is an EC2 instance in the public subnet of the VPC. The administrator has to remotely login into the EC2 in the public subnet, in this instance he can login into the OttoBotServer in the private subnet.

4.2 System architecture

The system backend handles HTTPS requests, just like a web application backend. In that sense, the architecture is similar to a web app.

In order to create a maintainable and easily extendable application, it is structured into a 3 layered architecture. A 3 layered architecture separates the code of the application based on their responsibility into presentation layer, business logic layer and data layer. This concept supports scalability and helps with maintanance [36]. The presentation layer is represented by the Alexa enabled device and the ASK API. It takes the input from the user and interprets the JSON response generated by the business layer.

On the application backend side, the code is structured into a data and a business logic layer.

The data layer abstracts the access to external data sources from the application logic. Access to the database and APIs is provided by a set of implementation independent methods, which can be used by the business logic layer. In the case of the relational database, an ORM framework is used in order to handle object mapping between the relational database and object-oriented code.

[9]more information on AWS CloudWatch `https://aws.amazon.com/cloudwatch`

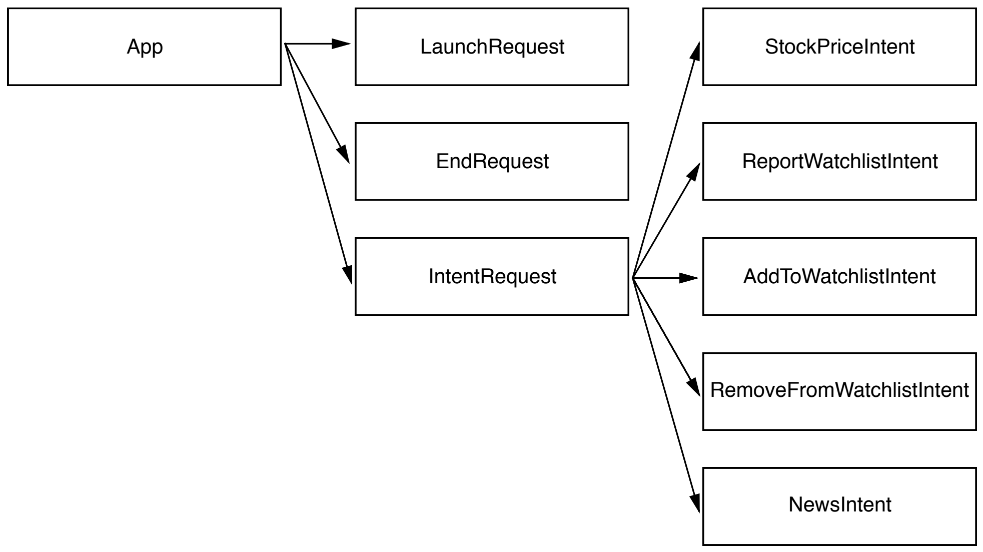


Figure 4.2: Request delegation on the backend

The business logic layer is where the structured user's request arrives and a relevant response is generated.

The code for handling a particular intention, like a query for the stock price, is structured into separate modules.

First, the app package delegates the handling of a request based on its type to appropriate package, as shown in figure 4.2. The request types are launch, intent, and end. In the case of a simple launch or end request, either a response welcoming the user to the app or a goodbye response is generated. If the incoming request is of type intent, the response generation is delegated to the appropriate package, as shown in figure 4.2. The single purpose package collects the required data from the data layer and builds a response.

Execution environment The backend code on the OttoBotServer EC2 instance is running in a virtual environment. This virtual environment is a Docker container. Docker is a tool to virtualize the execution environment for the contained application to make it easier to develop, test and deploy the product. Virtualization will cause the execution environment to be exactly the same no matter where the application is running.

The backend server is then a Flask application running inside the Docker container 4.3.

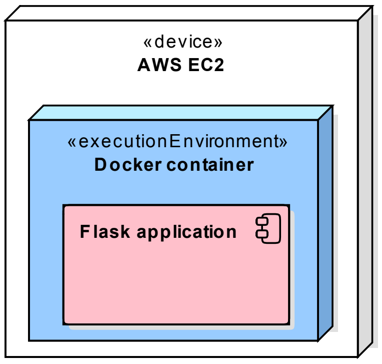


Figure 4.3: Dockerized application on EC2

4.3 Database model

The application uses an SQLite database. The database contains 3 simple tables: Stock, Watchlists, Users.

The Stock table stores the historical prices of stocks. This table is updated every day at 16:00 GMT with the latest prices by the StockScraper lambda function.

The app uses Facebook API to authenticate the users. Upon linking the Facebook account a Facebook id can be retrieved to uniquely identify the user. This id is saved in the table users together with the Facebook name. At this point General Data Protection Regulation (GDPR) complience has not been implemented, the regulation has to be met, before publishing the application.

The watchlists table is used, to persistently save the portfolio for each user. The collection of ticker symbols for specific user makes his portfolio.

The relational database model is shown in figure 4.4.

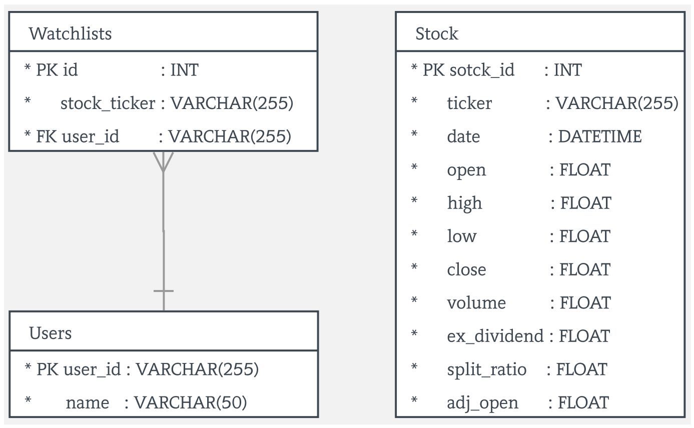


Figure 4.4: Relational model for ottobotdb

Chapter 5

Realization

In this chapter, the implementation of the application will be discussed. First, the used tools and technologies will be introduced. In the sections below some of the implementational details as well as testing will be covered.

5.1 Technologies & Tools

This section describes the technologies and tools selected for the realization of the project.

5.1.1 Technologies

The application backend is implemented in Python 3.6 together with the Flask framework. The reasons for using Python are the author's experience with Python and the AWS's support for this language.

Because the application backend behaves very similarly to a web server, the Flask web framework is a great fit. Flask is very popular for Alexa Skill development because it is lightweight and very easy to use. Additionally, the SQLAlchemy toolkit is utilized to simplify database interaction. This toolkit takes care of object-relation mapping and handles database migrations.

The Flask application runs inside a Docker container on the OttoBotServer EC2 instance.

The microservices running as Lambda functions, namely the OttoBot-Lambda and StockScraper, are implemented in the Python programming language.

The thesis is written in the LaTeX typesetting macro language.

5.1.2 Tools

Here will be listed the software tools used in the process of realization of this thesis.

Atom 1.26 This bachelor thesis project is written in the Latex language. The Atom editor with the package latex was used to compile the latex document into a PDF format.

Enterprise Architect (version 11) The visual modeling tool Enterprise Architect was used to model processes, use-case diagram, and actors diagram. All of these diagrams are modeled in the UML language.

PyCharam Professional 2017.3 PyCharm is an integrated development environment (IDE) designed specifically for the Python programming language. Among other tools, it provides graphical debugger, integration with version control systems (VCS), support for web frameworks like Flask and integrated unit tester.

Postman 6.0 Postman is an API development tool. It provides an easy way to emulate and test HTTP requests.

Version control system During the process of developing the OttoBot system, there was a need for versioning the code, for this purpose the Git version control system was used.

5.2 Implementation

OttoBot is an application build on the Alexa platform. As shown in figure 3.13, the Alexa Skill utilizes 3 basic components: the physical Alexa-enabled device, Alexa cloud service, and a backend generating the responses.

The Alexa-enabled device represents the user interface. The developer needs to configure the Skill in the Alexa cloud, by setting up the Skill and defining interaction model, and implement the backend server.

5.2.1 Interaction model

The role of the ASK interaction model is described in section 3.4.

First, it is necessary to define the invocation name, in case of this project, the name is Otto investments.

Second, the developer needs to define the intents. In the OttoBot Skill, for every use-case, there is an intent modeled. In addition to these custom defined intents, the model contains some basic self-explanatory built-in intents, like HelpIntent and StopIntent. Below, the WhatsTheStockPriceIntent intent will be described in detail, the implementation of the other intents is analogous.

The intent WhatsTheStockPriceIntent represents the action of querying the stock price. When a user wants to find out the stock price, a possible command would be something like “What’s the value of AAPL stock?”. Here the interaction model needs to understand 2 elements. First what type of intent the command is. That is determined based on the whole sentence, Second, there is a need to capture the stock ticker as a variable in the sentence, a so-called slot. Here the variable is the stock ticker AAPL.

There could be hundreds of possible ways of asking about the stock price. It is impossible to exactly define all of them. For this reason, some sample utterances need to be provided for the interaction model. Based on these samples an artificial intelligence NLP model will be generated to recognize all potential commands for this intent.

As shown in the developer portal screenshot in figure 5.1 the sample utterances contain a slot type, which represents the stock ticker symbol.

When defining slot types, it is necessary to provide all the expected values the slot might represent. In the case of the LIST_OF_TICKERS slot type, the list contains more than 3000 supported tickers.

For better user experience, it is possible to define synonyms for the tickers, which are words which will be resolved to the value of the ticker. Regarding the ticker list, some of the tickers have defined company name as a synonym. This allows the users to refer to the stock by the company name, instead of the ticker. Even when the user uses the company name, the name will be resolved as the ticker symbol, this greatly simplifies the code. The slot type LIST_OF_TICKERS modeled in the ASK developer web interface is shown in figure 5.2, at this point, only a subset of tickers has a defined synonym.

By modeling this intent, the ASK API will be able to resolve the commands for stock price and forward it to the backend in a structured form, that is a JSON format. The JSON output for the command "What the value of Apple stock?" is shown in figure 5.3, this JSON is sent to the backend as a POST request. Some of the most interesting segments like the recognized intent and the resolved ticker are in the figure 5.3 circled blue.

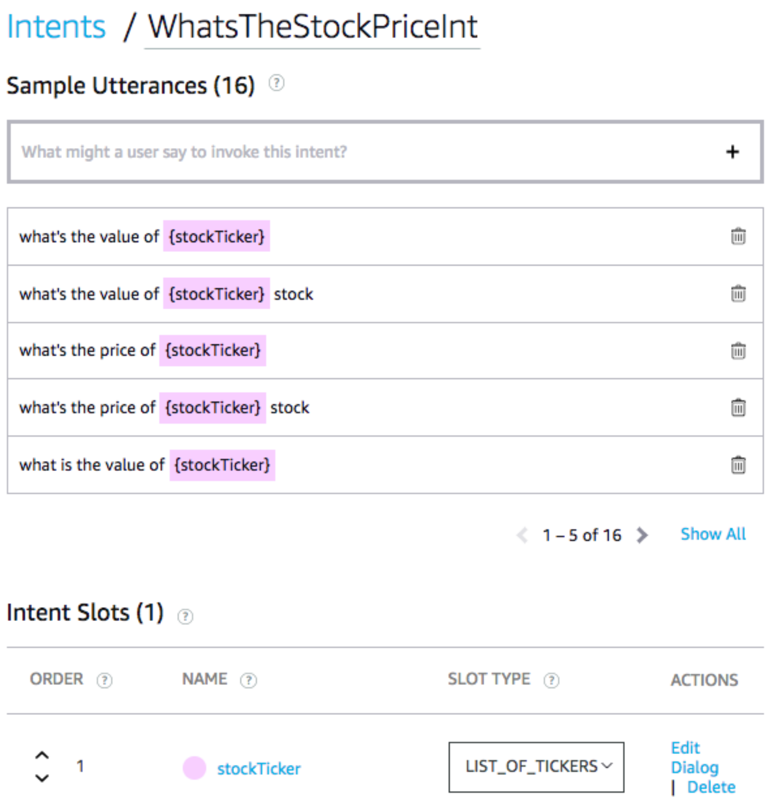
5.2.1.1 Account linking

The implementation consists of 2 steps, creating a Facebook app and configuration of account linking in the Alexa Skill.

To support Facebook login, a Facebook app with the authentication service, with redirect URIs, needs to be set up. This redirect links point to the OttoBot Alexa Skill.

Account linking needs to be configurated in the Alexa Skill as well, here it is necessary to fill the Facebook app ID, app secret, and the redirect links to connect the apps. The Alexa Skill account linking configuration is illustrated in screenshot 5.4.

In the OttoBotServer Flask application, a package called FacebookAPI implements a method for getting the unique user id and name from Facebook, code snippet 5.5.


Intents / WhatsTheStockPriceInt

Sample Utterances (16)

What might a user say to invoke this intent?

what's the value of {stockTicker}
what's the value of {stockTicker} stock
what's the price of {stockTicker}
what's the price of {stockTicker} stock
what is the value of {stockTicker}

1 – 5 of 16 Show All

Intent Slots (1)

ORDER	NAME	SLOT TYPE	ACTIONS
1	stockTicker	LIST_OF_TICKERS	Edit Dialog | Delete

Figure 5.1: ASK dev portal - Interaction model WhatsTheStockPriceIntent

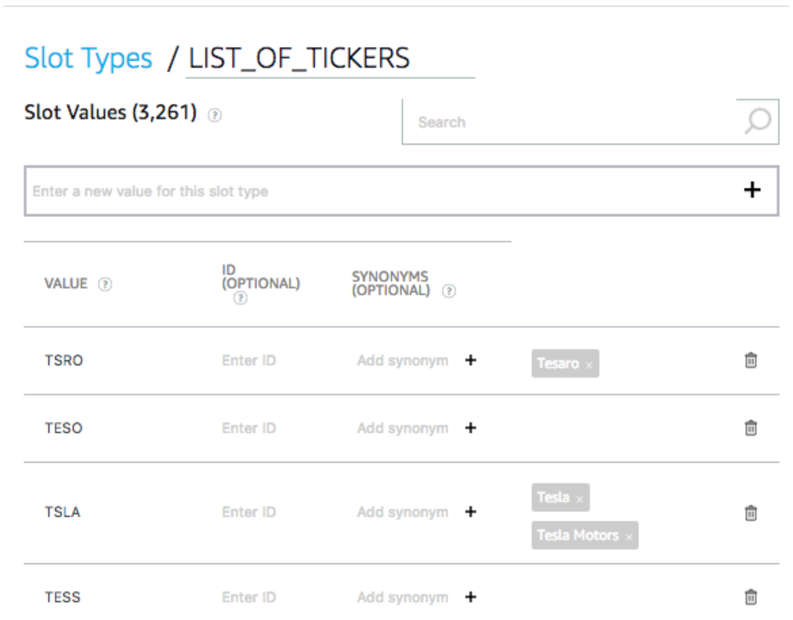
As soon as a user wants to use futures which require authentication, he will be asked to allow Facebook account linking in the Alexa mobile app or web app.

5.2.2 Backend server

From the ASK API, the request is routed to the endpoint, defined in the Skill's configuration. This endpoint is the OttoBotLambda microservice.

The implementation of the lambda function is straightforward, code snippet 5.6. The lambda service runs the lambda_handler function, this function passes the JSON in as HTTPS POST request further to the EC2 OttoBot-Server instance, running on 10.0.2.152/api/. Additionally, the function produces several logs, including the record of execution time on the OttoBot-Server.

When the request arrives to the OttoBotServer, it is routed to the appropriate package based on the request type, and in the case of an intent, further based on the intent type, as illustrated on figure 4.2. The intent routing to the appropriate package is handled by the handle_intent method, code snippet



Slot Types / LIST_OF_TICKERS

Slot Values (3,261)

Search

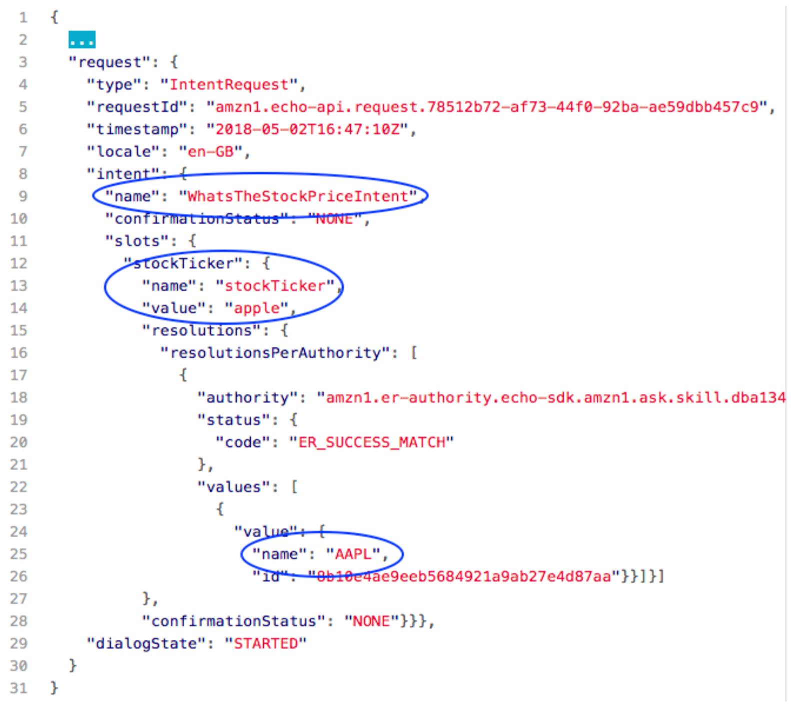
Enter a new value for this slot type

VALUE	ID (OPTIONAL)	SYNONYMS (OPTIONAL)	
TSRO	Enter ID	Add synonym	Tesaro
TESO	Enter ID	Add synonym	
TSLA	Enter ID	Add synonym	Tesla Tesla Motors
TESS	Enter ID	Add synonym	

Figure 5.2: ASK dev portal - Slot type LIST_OF_TICKERS

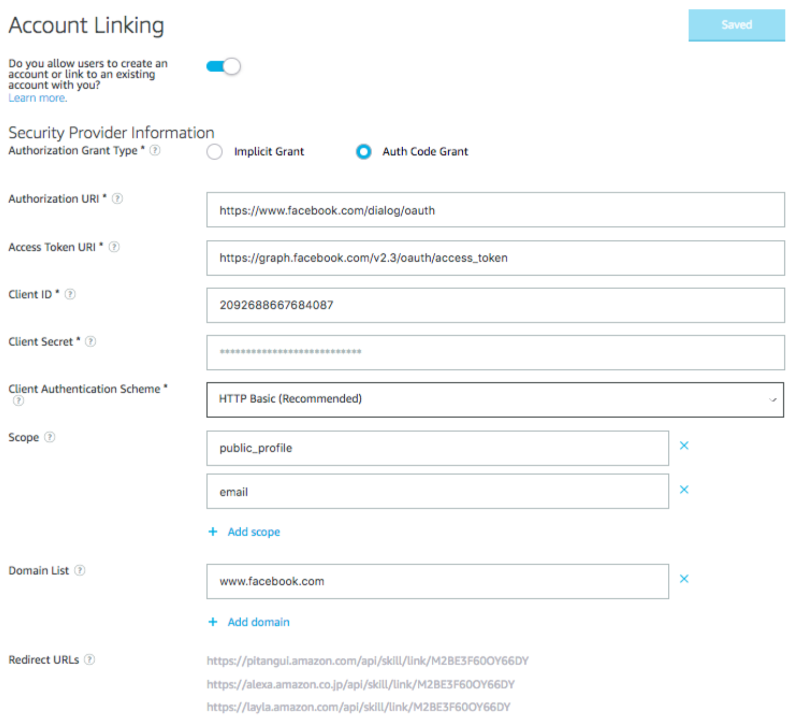
5.7.

In the case of the WhatsTheStockPriceIntent, the request is routed all the way to the intent_stock_price package, code snippet 5.8. Here the method handle_get_stock_price_intent collects the necessary data. That is the stock price for the specific ticker. Afterward, a response is built and returned.



```
{
  ...
  "request": {
    "type": "IntentRequest",
    "requestId": "amzn1.echo-api.request.78512b72-af73-44f0-92ba-ae59dbb457c9",
    "timestamp": "2018-05-02T16:47:10Z",
    "locale": "en-GB",
    "intent": {
      "name": "WhatsTheStockPriceIntent",
      "confirmationStatus": "NONE",
      "slots": {
        "stockTicker": {
          "name": "stockTicker",
          "value": "apple",
          "resolutions": {
            "resolutionsPerAuthority": [
              {
                "authority": "amzn1.er-authority.echo-sdk.amzn1.ask.skill.dba134
                "status": {
                  "code": "ER_SUCCESS_MATCH"
                },
                "values": [
                  {
                    "value": {
                      "name": "AAPL",
                      "id": "8b10e4ae9eeb5684921a9ab27e4d87aa"}}]}]
          },
          "confirmationStatus": "NONE"}}},
    "dialogState": "STARTED"
  }
}
```

Figure 5.3: Resolved WhatsTheStockPriceIntent JSON request



Account Linking

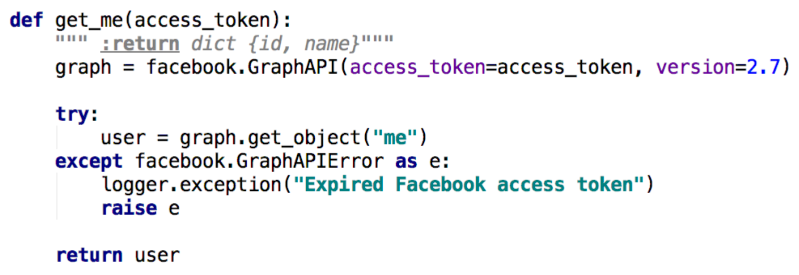
Saved

Do you allow users to create an account or link to an existing account with you? Learn more.

Security Provider Information

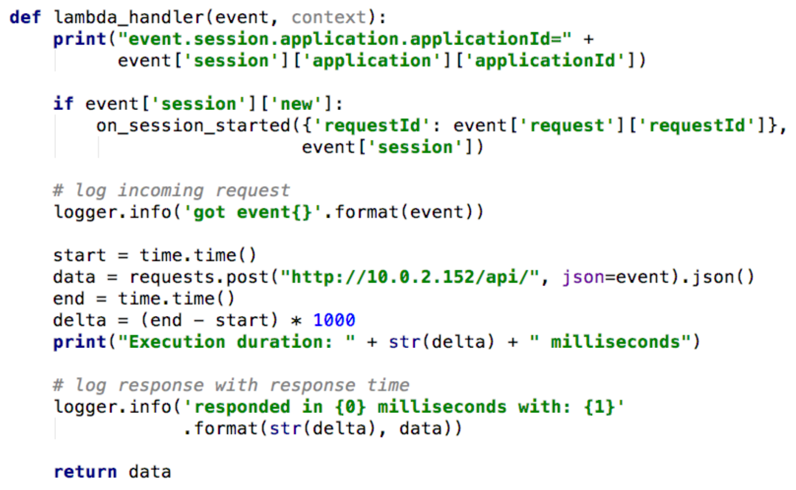
Authorization Grant Type *	Implicit Grant [✓ Auth Code Grant]
Authorization URI *	https://www.facebook.com/dialog/oauth
Access Token URI *	https://graph.facebook.com/v2.3/oauth/access_token
Client ID *	2092688667684087
Client Secret *	**************************
Client Authentication Scheme *	HTTP Basic (Recommended)
Scope	public_profile
	email
	+ Add scope
Domain List	www.facebook.com
	+ Add domain
Redirect URLs	https://pitangui.amazon.com/api/skill/link/M2BE3F60OY66DY
	https://alexa.amazon.co.jp/api/skill/link/M2BE3F60OY66DY
	https://layla.amazon.com/api/skill/link/M2BE3F60OY66DY

Figure 5.4: ASK dev portal - Facebook account linking configuration



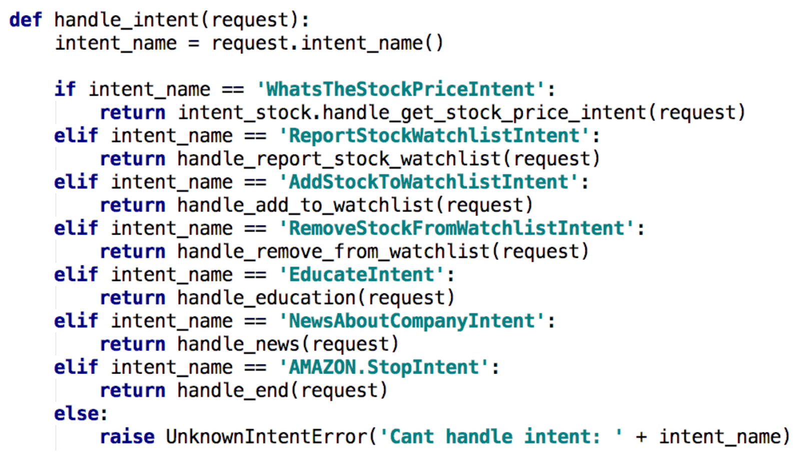
```
def get_me(access_token):
    """ :return dict {id, name}"""
    graph = facebook.GraphAPI(access_token=access_token, version=2.7)

    try:
        user = graph.get_object("me")
    except facebook.GraphAPIError as e:
        logger.exception("Expired Facebook access token")
        raise e

    return user
```

Figure 5.5: Implementation of get_me method in FacebookAPI package



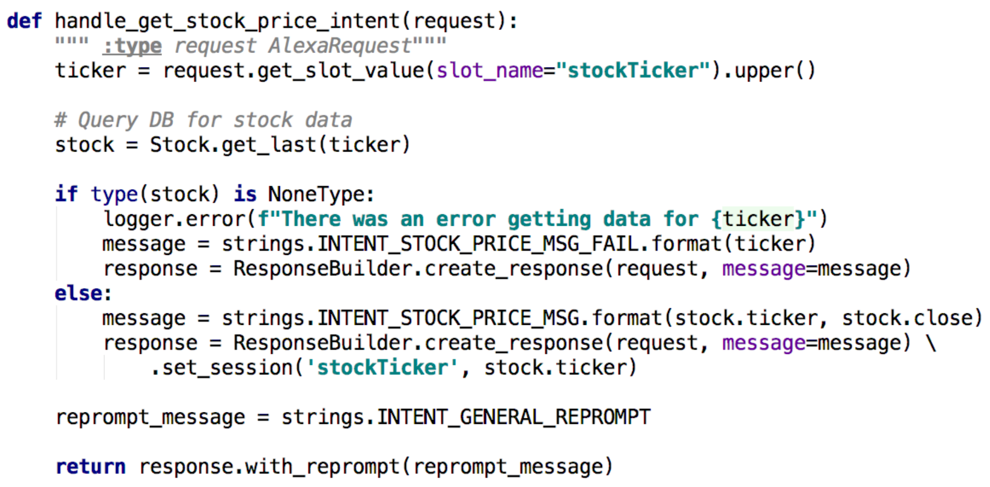
```
def lambda_handler(event, context):
    print("event.session.application.applicationId=" +
          event['session']['application']['applicationId'])

    if event['session']['new']:
        on_session_started({'requestId': event['request']['requestId']},
                           event['session'])

    # log incoming request
    logger.info('got event{}'.format(event))

    start = time.time()
    data = requests.post("http://10.0.2.152/api/", json=event).json()
    end = time.time()
    delta = (end - start) * 1000
    print("Execution duration: " + str(delta) + " milliseconds")

    # log response with response time
    logger.info('responded in {0} milliseconds with: {1}'
                .format(str(delta), data))

    return data
```

Figure 5.6: Implementation of routing requests to OttoBotServer in OttoBot-Lambda



```
def handle_intent(request):
    intent_name = request.intent_name()

    if intent_name == 'WhatsTheStockPriceIntent':
        return intent_stock.handle_get_stock_price_intent(request)
    elif intent_name == 'ReportStockWatchlistIntent':
        return handle_report_stock_watchlist(request)
    elif intent_name == 'AddStockToWatchlistIntent':
        return handle_add_to_watchlist(request)
    elif intent_name == 'RemoveStockFromWatchlistIntent':
        return handle_remove_from_watchlist(request)
    elif intent_name == 'EducateIntent':
        return handle_education(request)
    elif intent_name == 'NewsAboutCompanyIntent':
        return handle_news(request)
    elif intent_name == 'AMAZON.StopIntent':
        return handle_end(request)
    else:
        raise UnknownIntentError('Cant handle intent: ' + intent_name)
```

Figure 5.7: Implementation of delegating the intent handling to appropriate package



```python
def handle_get_stock_price_intent(request):
    """ :type request AlexaRequest"""
    ticker = request.get_slot_value(slot_name="stockTicker").upper()

    # Query DB for stock data
    stock = Stock.get_last(ticker)

    if type(stock) is NoneType:
        logger.error(f"There was an error getting data for {ticker}")
        message = strings.INTENT_STOCK_PRICE_MSG_FAIL.format(ticker)
        response = ResponseBuilder.create_response(request, message=message)
    else:
        message = strings.INTENT_STOCK_PRICE_MSG.format(stock.ticker, stock.close)
        response = ResponseBuilder.create_response(request, message=message) \
            .set_session('stockTicker', stock.ticker)

    reprompt_message = strings.INTENT_GENERAL_REPROMPT

    return response.with_reprompt(reprompt_message)
```

Figure 5.8: Implementation of response generation for requested stock price

5.3 Deployment

As mentioned in chapter 5.2, the application is running on several servers and has multiple components.

The primary backend of the application is responsible for generating the responses for incoming requests. This component is the Flask application inside a Docker container. For simple deployment, the docker image for this container is hosted in the cloud and can be quickly updated to newer version thanks to 2 custom scripts, both developed as part of this project.

First of the scripts is the docker_publish.sh, it creates an updated image for the application and uploads it to the cloud repository.

The script refresh_docker.sh, which is executed from the hosting server, stops the running container, downloads the new version, and runs it. In these 2 simple scripts, the entire deployment of the OttoBotServer backend is taken care of.

The last created script is the archive_script.sh for the routing Lambda function. This is the component which delegates request from the ASK API to the OttoBotServer. All this script does, is create a zip file with the new source code. This archive file can then be easily uploaded to the OttoBotLambda function.

All of the scripts are included with this project on the enclosed medium.

5.4 Testing

This chapter describes the testing process of the OttoBot application. After consultation with the thesis supervisor, only functional testing was conducted.

5.4.1 Functional tests

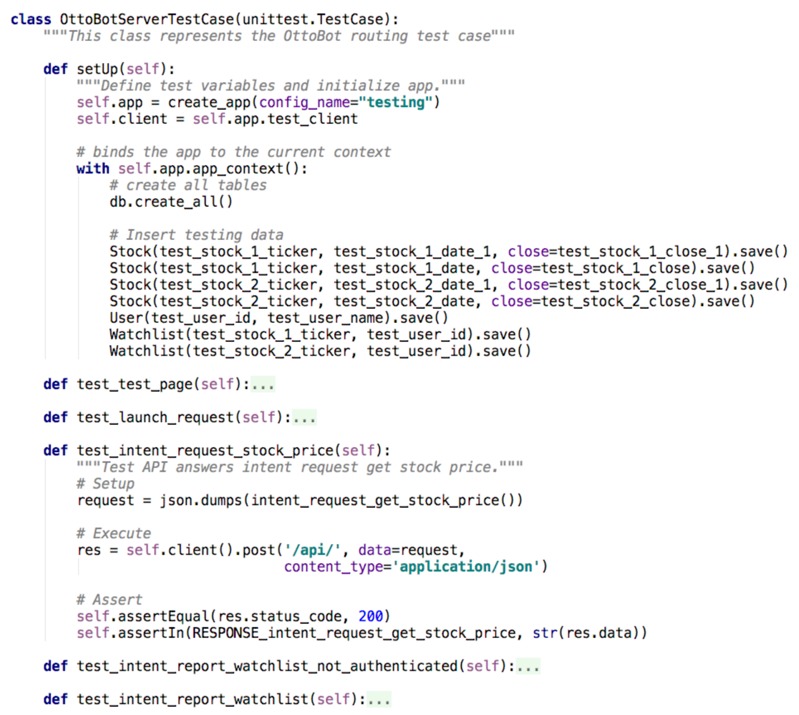
During the development of the application backend, namely the Flask server, the test-driven development methodology (TDD)[10] was employed. As an outcome of this approach, for all of the futures and test-cases, there exists a functional test. These tests validate if a proper response is generated for a given request.

Running all the test-suites[11] will automatically test all the core futures.

The code snippet 5.9 shows a small part of one of the test suits. In the test cases, a sample JSON POST request is sent to the Flask application. The request is similar to the one, the server might receive from Alexa. After a response is received, the test validates if the message holds the expected content.

[10]Before coding a new future, an automatic test is implemented to test the anticipated future.

[11]Collestion of tests that check a part of the application



```python
class OttoBotServerTestCase(unittest.TestCase):
    """This class represents the OttoBot routing test case"""

    def setUp(self):
        """Define test variables and initialize app."""
        self.app = create_app(config_name="testing")
        self.client = self.app.test_client

        # binds the app to the current context
        with self.app.app_context():
            # create all tables
            db.create_all()

            # Insert testing data
            Stock(test_stock_1_ticker, test_stock_1_date_1, close=test_stock_1_close_1).save()
            Stock(test_stock_1_ticker, test_stock_1_date, close=test_stock_1_close).save()
            Stock(test_stock_2_ticker, test_stock_2_date_1, close=test_stock_2_close_1).save()
            Stock(test_stock_2_ticker, test_stock_2_date, close=test_stock_2_close).save()
            User(test_user_id, test_user_name).save()
            Watchlist(test_stock_1_ticker, test_user_id).save()
            Watchlist(test_stock_2_ticker, test_user_id).save()

    def test_test_page(self):...

    def test_launch_request(self):...

    def test_intent_request_stock_price(self):
        """Test API answers intent request get stock price."""
        # Setup
        request = json.dumps(intent_request_get_stock_price())

        # Execute
        res = self.client().post('/api/', data=request,
                                 content_type='application/json')

        # Assert
        self.assertEqual(res.status_code, 200)
        self.assertIn(RESPONSE_intent_request_get_stock_price, str(res.data))

    def test_intent_report_watchlist_not_authenticated(self):...

    def test_intent_report_watchlist(self):...
```

Figure 5.9: Automatic functional test for stock price use-case

5.5 Documentation

Alexa Skills are distributed through the Alexa Skills Store. Below is a product description including a user guide and examples, optimized for publishing in the Alexa Skills Store.

5.5.1 Alexa Skills Store documentation

OttoBot is a stock market trading helper. The Skill provides information about current stock market status, news, offers virtual portfolios, and is even capable of explaining some of the advanced trading terminologies.

All of this will help you learn about the trading market and provide information to make better investing decisions.

To Enable the Skill: Search for “OttoBot” in the Alexa mobile app or at Amazon.com. Once you find the skill, open the detail and click the enable button. The skill can be also enabled via voice throu a Alexa-enabled device by saying “Alexa, enable Otto investments”.

To Get Started: Say ”Alexa open Otto investments”.

5.5.1.1 Supported futures:

Information about stock price You can ask what is the trading price of a particular stock. For a subset of the companies, it is possible to define the stock with the company name, instead of the stock ticker symbol. The presented value is the stock price. Try saying “How much is IBM stock?”.

Information about company’s market cap Ask about the current market capitalization of a company. Say “What is the market cap of Apple?”.

Virtual portfolio Simply have quick access your favorite stock. You can add stocks to a watchlist, so-called virtual portfolio. By adding these stock to the portfolio, you can request a report of the portfolio at any time to get a quick overview of all of the companies. If you have more then 2 stock in the portfolio, first the report will tell you what are your best and worst performing stocks. Of course, you can also remove the stocks from the portfolio.

Trading terminology explained If you are curious what does a particular trading term mean, ask OttoBot. The app is capable of explaining more than a dozen of investing terms. Simply say something like “What are some news about Amazon?”.

Trading strategies explained You can ask about some trading strategies. The application will explain a random strategie. Just say something like “What are some trading strategies?”.

Latest news Get the most recent news about any of the over 3000 supported companies. Say “What are some news about Amazon?” and you will be presented with the titles of 3 articles mentioning Amazon.

At this point, OttoBot supports only interaction in the English language. The Skill currently supports over 3000 publicly traded US companies.

Interaction Examples

- “What is the value of Apple stock?”
- “Add Microsoft to my portfolio.”
- “How is my portfolio doing?”
- “Give me the latest news on Tesla.”
- “What is P/E ratio?”

Invocation name Otto investments

Chapter 6

Conclusion

The goal of this thesis was to create a voice-enabled conversation chatbot in the stock market investments domain, build on the Amazon Alexa platform. The objectives were to examine the existing applications, review the Alexa Skills Kit capabilities, and analyze the financial information websites as a data source. Furthermore, design and implement the application to report portfolio status and provide information about stocks. Finally, to test and document the service.

During the existing solution research, very few quality applications were found, and none of them would support all of the proposed functionalities. That served as a validation that there is a market opportunity for this type of application.

After reviewing the capabilities of Alexa Skills Kit, some limitations were found, mostly regarding the lack of monetization options on the Alexa platform. This fact was taken into consideration, and an appropriate business model was introduced. The business model is relying on partnerships and user engagement.

The research of financial information data source identified several limitations. First of them was the necessity to pay to the data providers in order to have access to real-time stock prices. Based on the business model, the operation cost of the application has to be as low as possible, because of that it is unfeasible to pay for the stock data source. It was concluded that the close price from the last trading day if sufficient for the application. Another identified problem was the high latency of the stock information data source. That was discovered after conducting performance testing of several data sources. This issue was solved by implementing stock data scraping from one of the data sources into a private database to minimize response time.

The Alexa Skill was built on the Amazon Web Services infrastructure and requested futures implemented, including automatic functional tests for each of the use cases.

Finally, the application was documented with a guide to the user how to enable the Skill and interact with it.

Overall the goal of the project was met, an application was created, implementing all of the requested functionalities and even some more. During the project, the most prominent challenge was learning the ASK APIs. Because of the rapid development of this technology, even during the application development, the ASK APIs was changing significantly.

The developed Alexa Skill enables the user to quickly and efficiently get information about the stock market. The potential is much larger than that, as the ultimate goal would be to create a virtual private broker who presents complex information in an easily understandable form. Features like social media sentiment analysis on the traded companies, generating insights regarding earnings reports and suggesting portfolio based on investing strategies would make the application even more powerful.

Bibliography

1. *SP 500 Historical Prices*. 2018. Available also from: http://www.multpl.com/s-p-500-historical-prices.
2. *Stock Quote Bank of America Corp*. Google, 2018. Available also from: https://www.google.com/search?q=NYSE:BAC&source=lnms&tbm=fin&sa=X#scso=uid_LajwWpW-BMOosAHYk4c4_5:0.
3. MEEKER, Mary. *Internet Trends*. 2017. Available also from: http://www.kpcb.com/file/2017-internet-trends-report.
4. HAO, Karen. *Amazon Echo's dominance in the smart-speaker market is a lesson on the virtue of being first*. Quartz, 2018. Available also from: https://qz.com/1157619/amazon-echos-dominance-in-the-smart-speaker-market-is-a-lesson-on-the-virtue-of-being-first/. [Cited 2018-04-11].
5. KINSELLA, Bret. *Smart Speaker Owners Use Voice Assistants Nearly 3 Times Per Day*. 2018. Available also from: https://www.voicebot.ai/2018/04/02/smart-speaker-owners-use-voice-assistants-nearly-3-times-per-day/. [Cited 2018-05-04].
6. STOJANOVI, Slobodan. *Alexa, tell me how to build a skill - Hacker Noon*. Hacker Noon, 2018. Available also from: https://hackernoon.com/alexa-tell-me-how-to-build-a-skill-1aefdabc279.
7. AMERICA, Bank of. *2018 Better Money Habits Millennial Report*. 2018. Available also from: https://bettermoneyhabits.bankofamerica.com/content/dam/bmh/pdf/ar6vnln9-boa-bmh-millennial-report-winter-2018-final2.pdf. Bank of America. [Cited 2018-04-02].
8. MARTIN, Emmie. *Here's why millennials would rather save than invest*. CNBC, 2017. Available also from: https://www.cnbc.com/2017/12/29/why-millennials-would-rather-save-than-invest.html. [Cited 2018-04-02].

9. O'NEIL, William J. *How to Make Money in Stocks: A Winning System in Good Times Or Bad*. McGraw-Hill, 1995. ISBN 0070480176.
10. *Bull market Definition in the Cambridge English Dictionary*. Cambridge English Dictionary. Available also from: https://dictionary.cambridge.org/us/dictionary/english/bull-market. [Cited 2018-04-02].
11. *Bear market Definition in the Cambridge English Dictionary*. Cambridge English Dictionary. Available also from: https://dictionary.cambridge.org/us/dictionary/english/bear-market. [Cited 2018-04-02].
12. JONES, Jeffrey M. *U.S. Stock Ownership Down Among All but Older, Higher-Income*. Gallup, 2017. Available also from: http://news.gallup.com/poll/211052/stock-ownership-down-among-older-higher-income.aspx. [Cited 2018-04-02].
13. WOLFF, Edward N. *Household Wealth Trends in the United States, 1962 to 2016: Has Middle Class Wealth Recovered?* 2017. Available from DOI: 10.3386/w24085. Working Paper. National Bureau of Economic Research.
14. REZMER, Anke. *Zehn Millionen Deutsche haben Aktien âĂŞ so viele wie vor der Finanzkrise*. Handelsblatt GmbH, 2018. Available also from: http://www.handelsblatt.com/finanzen/maerkte/aktien/aktionaerszahlen-zehn-millionen-deutsche-haben-aktien-so-viele-wie-vor-der-finanzkrise/20977306.html. [Cited 2018-04-03].
15. FAY, Pierrick. *Bourse : les FranÃğais reprennent goÃżt aux actions*. Les Echos, 2017. Available also from: https://www.lesechos.fr/18/07/2017/lesechos.fr/030452304204_bourse---les-francais-reprennent-gout-aux-actions.htm. [Cited 2018-04-03].
16. HAYES, CFA Adam. *Bond*. Investopedia, 2018. Available also from: https://www.investopedia.com/terms/b/bond.asp. [Cited 2018-05-01].
17. *Dividend Frequency*. Investopedia, 2018. Available also from: https://www.investopedia.com/terms/d/dividend_frequency.asp. [Cited 2018-05-01].
18. HAYES, CFA Adam. *Stocks Basics: What Are Stocks?* Investopedia, 2018. Available also from: https://www.investopedia.com/university/stocks/stocks1.asp. [Cited 2018-05-01].
19. *Price-Earnings Ratio - P/E Ratio*. Investopedia, 2018. Available also from: https://www.investopedia.com/terms/p/price-earningsratio.asp. [Cited 2018-05-04].
20. *Dividend Yield*. Investopedia, 2018. Available also from: https://www.investopedia.com/terms/d/dividendyield.asp. [Cited 2018-05-04].

21. XIONG, Wayne; DROPPO, Jasha; HUANG, Xuedong; SEIDE, Frank; SELTZER, Mike; STOLCKE, Andreas; YU, Dong; ZWEIG, Geoffrey. Achieving Human Parity in Conversational Speech Recognition. *CoRR*. 2016, vol. abs/1610.05256. Available from arXiv: `1610.05256`.

22. YAMPOLSKIY, Roman V. Turing Test as a Defining Feature of AI-Completeness. In: *Artificial Intelligence, Evolutionary Computing and Metaheuristics: In the Footsteps of Alan Turing*. Ed. by YANG, Xin-She. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 3–17. ISBN 978-3-642-29694-9. Available from DOI: `10.1007/978-3-642-29694-9_1`.

23. KINSELLA, Bret; MUTCHLER, Ava. *Voicebot Smart Speaker Consumer Adoption Report*. 2018. Available also from: `https://www.voicebot.ai/wp-content/uploads/2018/03/smart_speaker_consumer_adoption_report_2018.pdf`. Technical report. [Cited 2018-05-04].

24. Smart speakers are the fastest-growing consumer tech; shipments to surpass 50 million in 2018. *Canalys*. 2018. Available also from: `https://www.canalys.com/static/press_release/2018/press-release-040118-smart-speakers-are-fastest-growing-consumer-tech-shipments-surpass-50-million-2.pdf`. [Cited 2018-05-04].

25. *Amazon Celebrates Biggest Holiday; More Than Four Million People Trialed Prime In One Week Alone This Season*. Business Wire, 2017. Available also from: `https://www.businesswire.com/news/home/20171226005146/en/Amazon-Celebrates-Biggest-Holiday-Million_People-Trialed-Prime`. [Cited 2018-04-11].

26. FEDERIGHI, Craig. *WWDC 2017 Keynote*. Available also from: `https://developer.apple.com/videos/play/wwdc2017/101/?time=3344`. [Cited 2018-04-13].

27. HWONG, Connie. *Rise of the Machines: How AI-Driven Personal Assistant Apps Are Shaping Digital Consumer Habits*. 2017. Available also from: `http://research.vertoanalytics.com/hubfs/Files/VertoAnalytics-Personal-Assistant-Report.pdf`. Technical report. Verto Analytics. [Cited 2018-05-04].

28. FØLSTAD, Asbjørn; BRANDTZÆG, Petter Bae. Chatbots and the New World of HCI. *interactions*. 2017, vol. 24, no. 4, pp. 38–42. ISSN 1072-5520. Available from DOI: `10.1145/3085558`.

29. BRANDTZÆG, Petter Bae; HEIM, Jan; KARAHASANOVIĆ, Amela. Understanding the New Digital divide-A Typology of Internet Users in Europe. *Int. J. Hum.-Comput. Stud.* 2011, vol. 69, no. 3, pp. 123–138. ISSN 1071-5819. Available from DOI: `10.1016/j.ijhcs.2010.11.004`.

30. BRADNER, Scott. *Key words for use in RFCs to Indicate Requirement Levels*. Internet Engineering Task Force, 1997. Available also from: `https://tools.ietf.org/html/rfc2119`. [Cited 2018-05-04].

31. *Earn Money for Developing Alexa Skills that Customers Love*. Amazon. Available also from: https://developer.amazon.com/alexa-skills-kit/rewards. [Cited 2018-04-22].

32. PICHLER, Roman. *A Product Canvas for Agile Product Management, Lean UX, Lean Startup*. Roman Pichler, 2017. Available also from: https://www.romanpichler.com/blog/the-product-canvas/.

33. *What is Cloud Computing? - Amazon Web Services*. Amazon. Available also from: https://aws.amazon.com/what-is-cloud-computing/. [Cited 2018-04-27].

34. *Types of Cloud Computing*. Amazon. Available also from: https://aws.amazon.com/types-of-cloud-computing/. [Cited 2018-04-27].

35. *Regions and Availability Zones*. Amazon. Available also from: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html. [Cited 2018-04-27].

36. *What is Three-Tier Architecture? - Definition from Techopedia*. Techopedia. Available also from: https://www.techopedia.com/definition/24649/three-tier-architecture. [Cited 2018-04-29].

Appendix **A**

Acronyms

IPO Initial public offering

NYSE New York Stock Exchange

NASDAQ Nasdaq Stock Market

LSE London Stock Exchange

VUI Voice user interface

GUI Graphical user interface

VPA Virtual private assistants

AI Artificial intelligence

NLP Natural language processing

NLU Natural language understanding

HCI Human-computer interaction

H2H Human to human

H2M Human to machine

ASK Alexa Skills Kit

API Application programming interface

KPIs Key performance indicators

MAU Monthly active users

IaaS Infrastructure as a Service

AWS Amazon Web Services

VPC	Virtual private clouds
IDE	Integrated development environment
VCS	Version control systems
GDPR	General Data Protection Regulation
TDD	Test-driven development

Appendix B

Contents of enclosed CD

```
readme.txt ........................ the file with CD contents description
documentation ..................... the directory of documentation files
src ....................................... the directory of source codes
  impl ........................................ implementation sources
  thesis .............. the directory of LaTeX source codes of the thesis
text ........................................... the thesis text directory
  BP_Drabek_Jan_2018.pdf .............. the thesis text in PDF format
```