THE MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF KAZAKHSTAN

Kazakh National Research Technical University named after K. I. Satbayev

Institute of Information and Telecommunication Technologies

Department Of Cybersecurity, Data Storage and Processing

Kussembay Aruzhan Maratkyzy

Development of Intellectual ChatBot System

DIPLOMA WORK

Major 5B070300 – «Information systems»

Almaty 2019

THE MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF KAZAKHSTAN

Kazakh National Research Technical University named after K. I. Satbayev

Institute of Information and Telecommunication Technologies

Department Of Cybersecurity, Data Storage and Processing

ADMITTED TO DEFENSE
Head of Department Of Cybersecurity,
Data Storage and Processing,
PhD, associate professor
[illegible] N.A.Seilova
«13» 05 2019.

DIPLOMA WORK

Theme: «Development of Intellectual ChatBot System»

Major 5B070300 - «Information systems»

Performed:	Kussembay A.M
Reviewer, PhD, professor of Almaty Management University [illegible] T.I. Bakibayev «13» May 2019.	Scientific advisor, Tutor [illegible] A. T. Azhenov «13» 05 2019.

School of Engineering Management

Almaty 2019

THEMINISTRYOFEDUCATIONANDSCIENCE OF THE REPUBLIC OF KAZAKHSTAN

Kazakh national research technical University named after K. I. Satbayev

Institute of Information and Telecommunication Technologies

Department of Cybersecurity, Data Storage and Processing

5B070300 – «Information systems»

AFFIRM
Head of Department of Cybersecurity,
Data Storage and Processing,
PhD, associate professor
[illegible] N.A.Seilova
«13» 05 2019.

TASK
to perform the Diploma work

Student Kussembay A.M
Theme: Development of Intellectual Chatbot System (Projecting and developing a database of an expert system of a chatbot)
Approved by the order of the University Rector №1162-б from «16» 10 2018
Deadline for completion of work «13» 05 2019
Source data to work: Chatbot web applications, social media channels and instant messengers web integration projects, Web API documentation, results of a research about Ecommerce projects.
Summary of the diploma work:
a) the development of a database for a chatbot;
b) the development of chatbot's algorithms;
c) the research of the integration of systems.
The list of graphic material: presented 15 slides presentation work
Recommended main literature: 20 sources

SCHEDULE
of preparation of the Diploma work

The name of the sections, a list of issues	Deadline for submission to supervisor and consultants	Notice
Overview and analysis of existing IS solutions on the market	10.01.2019 - 08.03.2019.	
Writing the functional structure of IS	05.02.2019-10.03.2019.	
Writing program part	11.03.2019-28.04.2019.	

Signature
Consultants and normocontrol to complete a Diploma work indicating related sections of work

Section titles	Consultants, Full name (academic degree, rank)	Data of signing	Signatur
Development of program software	M.B Bauyrzhan, Master of Technical Studies, tutor	8.05.19	[illegible]
Normocontrol	O.V Kisseleva, PhD, senior-lector	8.05.19	[illegible]

Scientific adviser [illegible] Azhenov A. T.

Task was accepted for execution by the student [illegible] Kussembay A. M.

Date «13» 05 2019 year

MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF KAZAKHSTAN
SATBAYEV UNIVERSITY

REVIEW

of Diploma work
Kussembay A.M

Major 5B070300 – Information Systems
Theme: Development of Intellectual Chatbot System

Done:
а) Program part on 26-32 pages
б) Explanatory note on 7-25 pages

NOTES TO WORK

The work of Kussembay A.M is devoted to the creation of a web application that integrates with third-party services of social network and instant messenger and has a training mode. The work analyzes the subject of the study, describes the technologies used, the process of building a web application, interaction with the database, connection and integration with the third-party services. In the process of developing a system programming tools like Python, Django, SQLite were used. The goal of the work was fully achieved and the tasks determined in the project were implemented. The material of the diploma work is logically structured, written by the scientific style of presentation.
The diploma work has the following comment:
1. needs an improvement of the web application interface design;
2. functional improvement.

Review of work

Overall, the diploma work is performed at a high level, is completely in accordance with the requirements and tasks, is recommended for protection, and deserves 75% points.

Reviewer,
PhD, professor of Almaty
Management University
[illegible] T.I. Bakibayev
«13» May 2019.

School of Engineering Management

MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF KAZAKHSTAN
SATBAYEV UNIVERSITY

REVIEW

OF SCIENTIFIC ADVISER

of Diploma work
Kussembay A.M
Major 5B070300 – Information Systems

Theme: Development of Intellectual Chatbot System

The presented diploma work is very informative and fully corresponds to the issued task and theme of the project. The student has clearly formulated the objectives of scientific research, as well as the tasks to achieve it. They fully meet the project's theme and were implemented. The ultimate goal of the project has been completed and a system for processing customer requests from the instant messaging service Telegram and social media channel VKontakte with training mode has been developed that allows customers and users to use it for maximum business benefit. In the diploma project the relevance of the thesis and the degree of elaboration of the problem in modern science was described.

A full review of the subject area, the technologies used, the advantages and disadvantages of the system were made with full dedication and without any mistakes. Methods of systems integration, interaction of web services, Web API of third-party systems were investigated. From the project, it is clear that the student has deeply studied and developed the problem of the intellectual chatbot system.

All parts of the diploma project are written and decorated in accordance with the State Standards, accurate and competent, relevant. Information, graphs and figures in the application are made quite qualitatively and correctly.

The diploma work developed by Kussembay A.M was performed completely in accordance with the requirements and tasks, is recommended for protection.

Scientific advisor,
Tutor
[illegible] A. T. Azhenov
«13» 05 2019.

Протокол анализа Отчета подобия

заведующего кафедрой / начальника структурного подразделения

Заведующий кафедрой / начальник структурного подразделения заявляет, что ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Кусембай А.М

Название: Development of Intellectual Chatbot System

Координатор: Алмат Аженов

Коэффициент подобия 1:0,4

Коэффициент подобия 2:0

Тревога:0

После анализа отчета подобия заведующий кафедрой / начальник структурного подразделения констатирует следующее:

[✓ ☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, работа признается самостоятельной и допускается к защите;]
☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;
☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, работа не допускается к защите.

Обоснование:

………
………
………
………
………

……………………… ……………………………………………………

Дата 13.05.19г.

Подпись заведующего кафедрой / [illegible]

начальника структурного подразделения [illegible]

Окончательное решение в отношении допуска к защите, включая обоснование:

Допускается к защите

Дата 13.05.19г

Подпись заведующего кафедрой / [illegible]

начальника структурного подразделения [illegible]

Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Кусембай А.М

Название: Development of Intellectual Chatbot System

Координатор:Алмат Аженов

Коэффициент подобия 1:0,4

Коэффициент подобия 2:0

Тревога:0

После анализа Отчета подобия констатирую следующее:

[✓ ☐] обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;

☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;

☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:

……
……
……
……
……
…………………………………………………………………

13.05.2018 [illegible]

Дата *Подпись Научного руководителя*

Краткий отчет

Университет:	Satbayev University
Название:	Development of Intellectual Chatbot System
Автор:	Кусембай А.М
Координатор:	Алмат Аженов
Дата отчета:	2019-05-06 05:42:37
Коэффициент подобия № 1:	**0,4%**
Коэффициент подобия № 2:	**0,0%**
Длина фразы для коэффициента подобия № 2:	**25**
Количество слов:	7 062
Число знаков:	43 460
Адреса пропущенные при проверке:	
Количество завершенных проверок:	44

>> **Самые длинные фрагменты, определеные, как подобные**

№	Название, имя автора или адрес гиперссылки *(Название базы данных)*	Автор	Количество одинаковых слов
1	ANALYSIS OF CRM IMPACT ON CUSTOMER LOYALTY IN HOSPITALITY INDUSTRY. CASE OF HILTON HOTELS CORPORATION *Uniwersytet Ekonomiczny w Krakowie (Wydział Ekonomii i Stosunków Międzynarodowych)*	Magdalena Rup	10
2	URL_ https://en.wikipedia.org/wiki/Service_of_process		5
3	URL_ https://en.wikipedia.org/wiki/Message_authentication_code		5
4	URL_ https://en.wikipedia.org/wiki/Message_authentication_code		5

>> **Документы, в которых найдено подобные фрагменты: из RefBooks**

Не обнаружено каких-либо заимствований

>> **Документы,содержащие подобные фрагменты: Из домашней базы данных**

Не обнаружено каких-либо заимствований

>> **Документы,содержащие подобные фрагменты: Из внешних баз данных**

Документы, выделенные жирным шрифтом, содержат фрагменты потенциального плагиата, то есть превышающие лимит в длине коэффициента подобия № 2

№	**Название** ***(Название базы данных)***	**Автор**	**Количество одинаковых слов (количество фрагментов)**
1	ANALYSIS OF CRM IMPACT ON CUSTOMER LOYALTY IN HOSPITALITY INDUSTRY. CASE OF HILTON HOTELS CORPORATION *Uniwersytet Ekonomiczny w Krakowie (Wydział Ekonomii i Stosunków Międzynarodowych)*	Magdalena Rup	10 (1)

>> **Документы,содержащие подобные фрагменты: Из интернета**

Документы, выделенные жирным шрифтом, содержат фрагменты потенциального плагиата, то есть превышающие лимит в длине коэффициента подобия № 2

№	**Источник гиперссылки**	**Количество одинаковых слов (количество фрагментов)**
1	URL_ https://en.wikipedia.org/wiki/Message_authentication_code	10 (2)
2	URL_ https://en.wikipedia.org/wiki/Service_of_process	5 (1)

АҢДАТПА

Сұхбатбот зияткерлік жүйесін дамыту

Дипломдық жұмыста әлеуметтік желі және жедел мессенджердің бөгде сервистерімен біріктірілген веб-қосымшаны құру сипатталған. Жұмыста пәндік салаға талдау жүргізілді, пайдаланылған технологиялар, веб-қосымшаны құру процесі, деректер базасымен өзара іс-қимыл, бөгде сервистермен қосылу және интеграциялау әдістері сипатталған. Бұл жоба электрондық коммерция саласына, атап айтқанда интернет-дүкендерге бағытталған, олардың алдында қолдау қызметін жақсарту және автоматтандыру міндеті тұр.

АННОТАЦИЯ

Разработка интеллектуальной системы чатбот

В дипломной работе описывается создание веб-приложения, которое интегрировано со сторонними сервисами социальной сети и мгновенного мессенджера. В работе проведен анализ предметной области, описаны использованные технологии, процесс построения веб-приложения, взаимодействие с базой данных, методы подключения и интеграции со сторонними сервисами. Данный проект нацелен на сферу электронной коммерции, в частности интернет-магазинов перед которыми стоит задача улучшить и автоматизировать службу поддержки.

ANNOTATION

Development of Intellectual Chatbot system

The thesis describes the creation of a web application that integrates with third-party services of social network and instant messenger. The work analyzes the subject of study, describes the technologies used, the process of building a web application, interaction with the database, connection and integration with the third-party services. This project focuses on e-commerce area, in particular online stores, which are faced with the task of improving and automating customer support.

CONTENTS

INTRODUCTION

In modern society, information technologies have become firmly established in the life of every individual. In Kazakhstan and around the world there is a rapid development of tools and technologies related to the exchange of information. In our country today, the state program "Digital Kazakhstan" designed for 2018-2022 is being actively implemented. It includes a developed information and communication environment, e-government services, e-commerce, digital libraries, block chain technologies, cybersecurity, e-logistics, smart cities. This means that these areas are actively growing and are in high demand.

The number of users of social media channels and instant messengers is growing very fast. Today, they are in great demand, this is due to a change in the field of mobile Internet: high speeds, low prices and the wide distribution of smartphones. Already, 2 billion people use messaging applications, and according to predictions, by 2021 the number of users will increase to 2.48 billion [1].

The relevance of the chosen topic can be explained by the fact that virtual communication is growing in importance and becoming one of the main types of communication between people in modern world. People chat with each other on social media channels and instant messengers on a daily basic because it is much easier and faster than trying to communicate through calling them. Instant messengers are also have a very easy to understand interface that make users like it. This project will help small and medium businesses to improve their customer service and communication between costumers and the company.

The aim of the thesis is to develop an intelligent chatbot system with training mode that will have integration with social media channel like VKontakte and instant messenger Telegram. The main objectives for the introduction of an automated intellectual chat bot system:

- research systems integration methods;
- consider the methods of interaction between web services;
- study instant messenger’s Web API.

Final goal of the thesis is to create a CRM system for small and medium. businesses in order to interact with sales channels through one interface and automate customer support through chatbot. How a developed end system can affect a business:

- improve customer service experience;
- permanent availability with quick answers;
- increase the worker’s productivity;
- maximize customer engagement and loyalty;
- increase resource profit.

1 Subject of study

The object of study of this diploma work is a chatbot. The scope for it is electronic commerce also called e-commerce projects in particular online stores, which became a main key element of the sales industry. An online store is an interactive website that advertises a product or service, accepts purchase orders, offers the user a choice of calculation options, a way to receive an order, and invoice for payment. From a technical point of view, an online store can be defined as a multifunctional software module, embedded in a website and designed to ensure the sale of products of the company via the Internet. Online stores can be classified by type of sales, payment methods and the number of product categories presented on the site. In addition, online stores differ in how they receive income from the site owner.

Selling your own goods and services through the Internet is the main way of commercial use of sites for most manufacturing and commercial enterprises. The company creates a site and places on it information about their products and services, prices and guarantees for buyers. Then they try to attract visitors and show them this information.

The Internet has become the main source of information for a huge number of potential buyers. More and more people, before making any major purchase, are looking for information on the web about manufacturers and sellers of the goods they need. After examining the alternative offers of sellers, consumers make an informed choice, which ends with a deal.

Nowadays, it is much easier and more convenient to shop online. You do not have to get to the actual store to buy a product; you can comfortably sit at home and find things to buy that will be delivered straight to your house. Most of the online stores even offer fittings and if the item did not fit you or you just did not like it, then you can easily return it back. The amount of people that prefer this method of shopping is growing each day.

Chatbots are used in various business areas to improve customer service and provide them with technical support:

- in insurance, a bot is able to help conclude an agreement and apply for payment;
- in the communal sphere, he informs about tariff changes, accepts meter readings and accident notifications;
- in medicine, a chat bot can record a patient to a doctor and conduct an initial survey;
- in the restaurant, bots can reserve a table and take an order;
- in online stores they help arrange the purchase, delivery and payment;
- in call centers of large companies, robots provide communication with clients and provide them with services at any time of the day or night.

My project allows automating the feedback for the client through online chatbot that will provide the instant response for the costumers available 24/7, eventually improve the customer service, user loyalty, and essentially increase the sales and straight profit. Through constant interaction with people, the chatbot will learn to imitate real conversations and respond to verbal or written inquiries, helping them find answers. Thus, after each dialogue, it will become smarter and provide more help for the business. Chatbots respond to consumer needs for responsiveness and personalization, optimizing online experience and improving it. This way of interaction helps increase consumer activity, and allows brands to interact with their audience in a more natural way like friends. Chat bot is always in touch 24/7 and 365 days a year.

1.1 Chatbot: A definition

A chat bot is a computer program, actually a virtual interlocutor, that works on the basis of established rules and algorithms. It is able to imitate human behavior while communicating with a potential client. It can work on any known platform, for example, Messenger, Telegram, Vkontakte, etc.

The term chatbot consists of two other terms - chat and bot. The meaning can be better understood by examining the two components separately. The Oxford Dictionary defines chat as “an informal conversation” and more specifically as “the online exchange of messages in real time with one or more simultaneous users of a computer network” [2]. I Bot is defined as being “(chiefly in science fiction) a robot” with the specific characteristics of representing “an autonomous program on a network (especially the Internet) which can interact with systems or users, especially one designed to behave like a player in some video games” [2].

There are two types of bots [3]:

- simple, that perform a limited number of actions, inform on standard queries and work according to the established rules and only;
- smart, that use elements of artificial intelligence, they study themselves, conduct a full-fledged dialogue with the client and are able to learn.

My chatbot will be smart and have the training mode which will provide the ability to learn from answers of online store worker’s, based on those answers it will reply to the costumers later on its own.

I can classify chatbots based on features they have into following categories:

One feature chatbots. They only one feature is provided by a large proportion of chatbots. These chatbots are functionally limited but user-friendly. One example is a Facebook chatbot called Instant Translator [4]; the person selects a language to be translated into at the beginning. From there, Instant Translator will simply translate all of the text it receives into the chosen language selected.

Proactive Chatbots. This category characterizes chatbots that push user information rather than answering conversational questions. The person does not need to communicate with the chatbot hereby, but only uses it as a service on certain times to receive information. An example would be a service that sends a daily weather forecast to the user.

Group Chatbots. There are a variety of functionality chatbots that can provide when interacting with a group of people rather than just a single user. These chatbots are restricted to platforms that provide the necessary features for group conversations to use chatbots. Roll [5] for a messenger service called Kik is a simple example for a group chatbot ; when trying to send a question to Roll, the chatbot responds with a random name selected from the group members.

Simplification Chatbots. In some cases, chatbots can be used to provide users with an easier interface for complex and difficult existing tasks, traditionally involving many formal and bureaucratic steps. One instance is the DoNotPay service. It is marketed as "the first robot lawyer in the world" [6] and the system helps the customer with basic legal issues, like fighting tickets for parking.

Personal Assistants. This category includes chatbots that integrate many different characteristics and it can be viewed like their own systems. This category includes Apple's Siri and Amazon's Alexa.

Optimization Chatbots. By developing a chatbot for consumers to connect to the product, this category attempts to make current products more available. Companies like to reduce the friction for users when using their products. The most evident aspect of chatbot systems at the moment is the ease with which users can access products. Companies want to maximize the use of products through making them accessible through chatbots ' conversational interfaces. As apparent from the commitment of so many well-established companies, companies are keen to be present on instant messenger platforms.

The chatbot system for my project can be classified as optimization chatbot.

1.2 A brief history of chatbots

A while back every chief had a personal assistant who was in charge of schedules, calls, routine work and similar things but in 1966 it had all changed. The professor Joseph Weizenbaum presented the first virtual assistant interlocutor “ELIZA” [7] to the world. The discovery of the new technology inspired people to start thinking and creating so called robots that would do different monotonous tasks.

By 1990, the industry and the technology tools had evolved to quite an extent, which as per the "Turing Test," virtual companions had started testing for humanity. The participants contacted the human first, then the computer. At the earliest stages, it was easy to distinguish in which the robot is, but 30 percent of the respondents have been unable to determine whether a young person or a computer was talking to them on one of the tests in 2014.

These programs have already been called artificial intelligence (AI) cautiously by this time. In addition, if it could then be called a questionable statement, then AI is a rather real phenomenon with the introduction of trained neural networks.

2017 was a year of the performance of chatbots with an integrated neural network. People have come to a virtual robots implementation that was more practical. They modified them, gave them specific different tasks and gave them the form of instant messengers.

1.3 Components

Generally speaking, chatbots can be considered as a combination of three parts, namely the interface between the chatbot and human users through a messaging platform, the intelligence that enables the chatbot to understand and solve customer requests and gain knowledge from each interaction, and integration, which includes mainly connection with other channels and systems.

Chatbots are also made up of inputs and outputs. The best way to describe input is to record and enter data into a system and provide the system with instructions. The user's input into the chat must match one of the predefined inputs in order for the computer system to be able to understand it correctly in order for a chatbot to work. The three key elements therefore include: chatbot must be able to adhere to its user's characteristics, be suitable for the tasks it is meant to perform, and the function and environment for which it is intended. After implementation, ensure that suitable feedback is obtained so that the system itself can be evaluated and adapted.

Output can always be described as converting information from a computer system into a form that can be recognized by a human being. As with inputs, it is important to understand what and who the system is designed for in order to set up a suitable input-output structure.

The chatbot environment consists of a few different parts. One of them is the user messages: they are a dynamic input that the agent can receive at any given time. They consist in a string representing the actual text sent by the user, and a metadata structure containing additional information like a pointer that links the message to the structure representing the particular conversation it belongs to, and possibly the time the message was sent, on which platform the message was sent, etc.

Another significant part of the environment the agent has access to is the company’s backend which contains additional information about the user and the database’s state. The agent can both inspect and influence certain aspects of the backend. The chatbot also has the ability to send replies to the users in order to obtain new information, or simply to tell them that their request has been accepted and treated accordingly.

The user sends a message to the agent, which is our chatbot system that interacts with backend client server and saves and extracts data from there. Then he sends a reply to a user back. Visual representation of the chatbot environment is presented in the Figure 1 below:

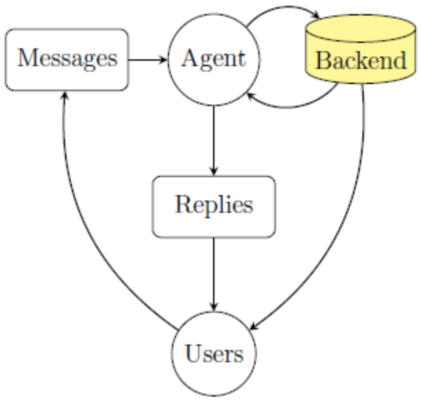


Figure 1 – Chatbot environment

1.4 Opportunities and challenges

The time of interacting through telephone calls or face-to-face contact with customers being the only possible option for communicating is over. With chatbots becoming extremely popular, they are also showing their advantages and benefits improving the business.

Optimization of the solution of standard repetitive tasks that can be formalized in the form of business logic. This applies to both business communications with the outside world, for example, customers or suppliers and internal communications.

The obvious gain for the company in these cases is to reduce costs and save staff time. However, there is also an unobvious one - increasing staff motivation. Standard operations are present in the life of even the most qualified employees and do not give them joy at all. Freed from them, people will be able to do creative tasks, for which, in fact, have chosen their profession.

Their permanent 24/7 availability is another advantage of chatbots [8]. All the time, chatbots are online, processing queries from the user. In addition, the problem of waiting until one gets through to an employee is also solved as most chatbots provide the function of processing unlimited queries simultaneously without causing issues.

Chatbots also have the ability to boost cross-selling and upselling and help customers when shopping online to find the perfect item. This enables chatbots to give recommendations based on the past personal data, preferences and history of order. It will also enable the subsequent one-click ordering, increasing the customer's convenience.

Personalization - chat bots save all customer information. Contact begins with the consent of the client to the processing of personal data. Client personalization comes after each interaction with the robot. The program remembers and analyzes which product a person bought, which accessory he came up with and which one was not needed, and so on, ad infinitum. Thanks to this, we can confidently increase the number of cross-sell sales.

Work with clients - the more often they communicate with the robot, the more it collects information about users. The platform will remove from you the task of collecting real and relevant information. Thus, the chatbot is used for both sales and marketing. It supports loyal customers interested in the brand with push notifications.

Unlike applications or websites, messengers communicate through dialogue, and people do not need to learn a new interface. And for the Millennials, messengers are also a favorite tool. Messengers are becoming new sales and marketing channel, and chat bots are the tool for this channel.

So, with that being said, chatbots ' four main advantages include convenience, cost reduction, the ability to maximize customer experience and engagement, and minimize customer service employee work hours. In addition, customers will be operated quickly as the time will be spent waiting for an inquiry to be answered will be decreased and chatbots will provide immediate answers.

It is hard to imagine a technology that has no disadvantages and potential risks and chatbot technology is no exception [9]. Each client is an individual with its own identity and criteria, allowing service delivery to be deeply unique and customized to the particular individual. Therefore, developers have to keep that in mind when creating a chatbot system. It has to be highly personalized which is hard to do and needs a better understanding of chatbot algorithms.

In general, the issue of simultaneity or inseparability has to be addressed by almost all service systems. This motivates the fact that services are generated and consumed at the very same time most of the time, which means that development will not start ahead of the user's input provision. Of course, this will also affect any process of inquiry handling. Demand is hard to predict, so it may be that the various employee scheduled less time than he should have in order to answer correctly and promptly to all inquiries. The importance of consistent performance and the ability to manage different services at a similarly excellently perceived level are further evaluated. All these processes, however, are intangible, making measuring the effectiveness of service provision even more difficult.

Because programmers develop chatbots, pre-programmed knowledge exists and an input will only be recognized if it agrees with an expected path. As soon when something, which does not fit the expected direction, is put in to the chatbot, the output will also be greatly affected. The output will therefore be either monotonous or frustrating for the customer, failing to provide the correct response and leaving users unhappy. One major issue in positioning the provision of input forms sarcasm and irony, as chatbots seem to be unable to translate sarcasm the right way, considering it genuine. Even though chatbots can be very advanced with artificial intelligence, they will never be fully able to replace a human being and their critical thinking.

The convenience of shopping online and finding products for a customer through chat is an advantage of chatbots. However, this can also result to probable misunderstandings, changing the buying decisions of the customer. For example, the client can change their taste or preference that will lead chatbot to fail and lose its logic and algorithm. It could become a time-consuming mission to narrow down every filter characteristic, like size, pattern, type and fabric, reducing client convenience and removing the benefit of this aspect.

As proposed, chatbots could potentially lead to a decreasing labor supply due to the reduction in customer support man-hours. The main jobs to be affected are those with small level and monotonous tasks, becoming a standardized process substituted by a chatbot.

Also, another risk is the threat of social engineering attacks is very high since the global use of chatbots has only boomed over the last few years. Users should always be able to trust the chatbot as well as any data, but confidential information in particular needs to be treated securely.

1.5. Formulation of the problem

The main goal of this diploma work is to create a system for processing customer requests from the instant messaging service Telegram and social media channel

VKontakte with training mode. The main objectives for the introduction of an automated intellectual chat bot system:

- create a modern and flexible interface for visualization graph of dialogue and interaction with him;
- build an extensible web application architecture for later development;
- realization of interaction between client application and server via the WebSocket protocol for working with the application in real time;
- embed into storage database architecture information about the dialogues;
- connect the system with instant messenger Telegram and social media channel Vkontakte implementing Web API using machine-based interactions REST and SOAP.

This chatbot will allow clients to configure automatic answers to users' questions as well as automate routine operations using scripts.

The system is able to work with two types of questions: custom and standard. In the first case, the administrator independently registers the essence of the question and the keywords to which the bot will respond. For example, during business hours, the operator can answer customer questions, while the chatbot will contribute these questions and answers to the database. After that, not during business hours, the chatbot will be able to pull out answers to questions from the database on its own and respond to users outside of business hours.

2 Selection of programming tools

When looking for potential solutions to write a chatbot, the development of a system has been found, you have to know at least one of server programming languages. Deciding which programming language to understand and then use was required. It also is important to be able to integrate with the REST (Representational State Transfer) API (Application Programming Interface) technologies, which are provided by instant messengers namely Telegram Bot API and VKontakte API. For several reasons, Python has been selected as the server language.

2.1 Programming language and libraries

Python is a general-purpose programming language primarily intended to increase the programmer's own productivity instead of the code he wants to write [10]. Simply put, you can write actually nearly anything with no substantive problems on Python language (web and desktop applications, games, automation scripts, complicated calculation systems, life support systems, and far more). In addition, the entry threshold is minimal, and the code is descriptive and easy to understand in so many ways. Because of the code's simplicity, it becomes easier and more enjoyable to maintain programs written on Python especially in comparison to Java or C++. Moreover, from a business perspective, this involves a cost savings and an increase in the efficiency of employees.

One of the key reasons for choosing a programming tool called Python is that it offers a good web solution. It has a big constantly growing amount of libraries, frameworks, modules available. They are written in C and on Python itself, and all suitably qualified coders are able to develop them. In particular, developing this system Django web framework solution was used. It is extremely user-friendly, safe and quick. It is extremely stable and reliable programming tool that can be used in a number of applications. Most professional developers tend to choose Django, particularly due to shorter development time and ease of installation.

The unquestionable advantage is that on almost all platforms and systems, the Python interpreter can be implemented. The first of its kind programming language was C, but when writing a platform agnostic program, its data types on various machines could take up differing amounts of memory. It is not a drawback for Python.

One advantage is also the simplicity of Python syntax. It lacks headers and redundant code. Python is versatile, modern and easy to use. Because Python has a wide range of uses — like development, scripting, scientific use, and so on — a huge community has sprung up around it, with which developers get tremendous support. Python’s popularity is growing rapidly fast so there is also extensive documentation available for developers.

From the used modules can be identified:

- Numerical Python — complex mathematical functionality such as altering whole arrays and matrices ;
- Tkinter — constructing applications using a graphic user interface ;

- OpenGL — using a complex library of two- and three-dimensional Open Graphics Library graphical modeling .

The main libraries that I used were scrapy, tensorFlow, pattern, bokeh.

Scrapy is a freeware, open-sourced framework available for Python for web crawling [11]. It gives an opportunity to a programmer to send requests for a website and analyze the HTML script you are receiving back as a reply. It makes the whole experience much easier and more convenient for the developer. Scrapy provides a powerful framework for extracting the data, processing it and then save it. Scrapy uses spiders, which are self-contained crawlers that are given a set of instructions . In Scrapy it is easier to build and scale large crawling projects by allowing developers to reuse their code.

And there is also an option for establishing the HTTP server with this library, which can command it through HTTP server requests. The server generates the answer response that is essentially is data formatted in JSON that also contains the scratched data. That advantage of scrapy helps programmers to spend more time developing a code instead of formatting and scratching data. To install scrapy library into the Python project I used the pip install scrapy command. It is going to download Scrapy on my whole web application system. I deployed a Scrapy project by starting the command: scrapy startproject webchatbot once the setup was fully complete.

TensorFlow is an openly available Python's programming tool library that was designed by Google Inc. in order to do fast numerical computing using neuro networks. It is a basic library that also can be used either directly to develop Deep Learning systems or to use wrapper additional libraries to ease the processes of built on top of the TensorFlow [12]. Computation is an also defined in the structure of a guided graphs in terms of the data stream, the procedures and the functions.

With the help of this library, you can calculate the nodes that have zero or more of input and output. The data that is moving between those nodes is also known as the tensors that are the several-dimensional true value of array datatype.

There is data flowing, branch, loops and a state changes are defined on the chart diagram. Also there are special edges that could be used to synchronize with the actions within the chart, such as the waiting for the calculation to be completed on the amount of input data.

The operation is an abstract calculation data named that can take the input data characteristics and generate the output data characteristics. For instance, you can define an operations to add the numbers or divide them. It can run on single CPU systems, GPUs as well as mobile devices and large scale distributed systems of hundreds of machines.

Pattern is a web mining module for the Python programming language. It has tools for data mining (Google, Twitter and Wikipedia API, a web crawler, a HTML DOM parser), natural language processing (part-of-speech taggers, n-gram search, sentiment analysis, WordNet), machine learning (vector space model, clustering, SVM), network analysis and canvas visualization [13]. For data mining functionality method, it contains the special APIs used for the data mining technology from the web applications such as Facebook, Telegram, Vkontakte and etc.

For machine learning functionality method, it includes the computer-learning model and perceptron that can be used to classify, regress, and cluster the tasks.

For installing this library, I used Python’s command with syntax of pip installation pattern. I used the pattern.db module that contains wrappers for databases like SQLite, MySQL, Unicode CSV files and Python's datetime. It offers a convenient way to work with tabular data, for example retrieved with the pattern.web module.

Bokeh is the highly customizable visualization based library for using on Python programming tool that targets the basic representation website browsers. Bokeh has so called bindings in multiple programming languages tools. Those same bindings create the JSON file that functions as an input data for Script library, which in the switch portrays data to traditional website browsers [14].

Bokeh can also generate beautiful and interactive visual representation over an enormous amount of broadcasting datasets with strong-performance of the interactivity. It also can assist any developer that wants to build the customizable mathematical diagram, dashboard solutions and applications for the data quickly and effortlessly. Bokeh has several advantages:

- enables the developer to quickly build the difficult numerical plots and by basic commands;
- offers the programmer the function of the output in different media such as html, notebook and server;
- gives a visualization that can also be embedded on the web solution tool Django;
- allows you to develop the specific visualization compiled in other libraries such as matplotlib, seabornlib, ggplots;
- has the ability of versatility to use interaction, the templates, the layouts and the various visualization designing options.

Bokeh library provides the programmer powerful and flexible features that deliver straightforwardness and complex customization for a developer. As shown in the text below, it offers the user with a multiple option visualization design interfaces. They are charts, plots and models. Charts that are a first-level design layout interface that is used as rapidly and concisely to create the complex mathematical plots. Plotting that is also called secondary-level design interface focused on the visual glyph composition of data. Models that are a simple small-leveled design that gives the power to the application programmers the ability of maximum potential flexibility.

2.2 Web framework

A framework is a set of ready-made libraries and tools in development that help you build web applications. The solution was chosen for the project is a Django framework that is written on Python.

In the common real life situations, Django is equipped with most libraries and tools needed. Django also provides an important for a web application security.

Also because of the fact that Django is an open source programming tool and a highly popular framework, a responsive community of developers has formed around it. There is a lot of Django documentation that is available for any developer. If you ever get stuck-it is not hard to find a solution.

Django Framework uses the MVC (Model-View-Controller) concept, which helps you achieve high write code speed and application productivity and high-quality debugging [15]. The following architecture of MVC methods is shown on the Figure 2 below:

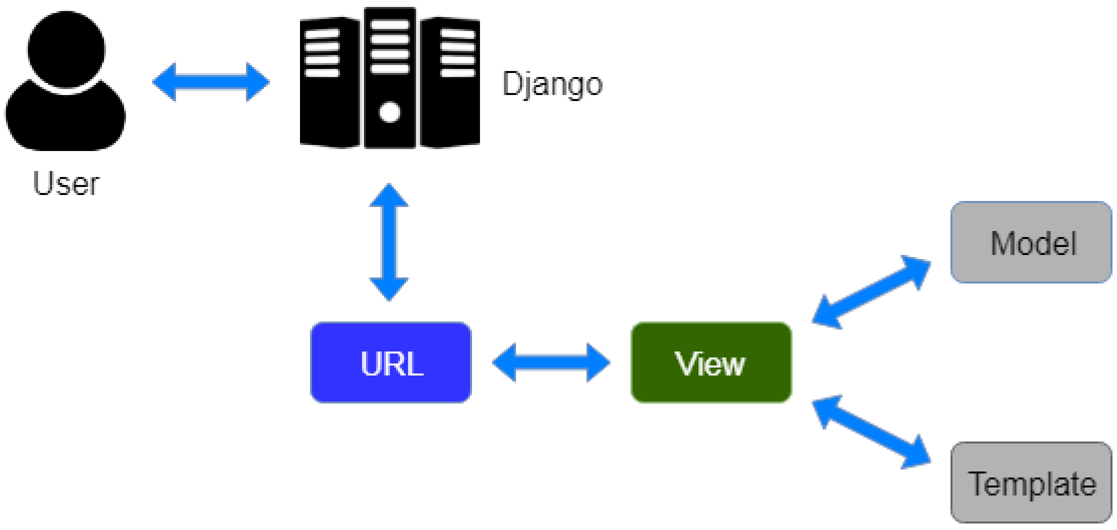


Figure 2 – Django MVC model architecture

Dividing the project into three parts, such as the definition of the database, presentation and work logic, makes Django Framework easy for everyone to understand, even a beginner level developer.

The Django Framework is fully Object Oriented. A Django application consists of at least 3 main files, namely:

- __init__.py is a very important file that is needed for Python to consider the directory where this file is located as an application;
- views.py - application logic, contains a set of functions that Django can call during an application call;
- models.py - work with the database.

The user's first stage of request enters the router also known as the URL dispatcher, which makes the decision which function must be called to handle the request. The choice is based on the results of a list of rules composed mainly of a regular expression and the function name: if such a URL is the function.

The router-invoked function is called a view. Inside there could be any business logic, in my case, this data is checked and saved to the database when a request came with data from the reply form of the client.

Data on the application is stored in a database. Relational database MySQL has been used in the thesis work. It is possible to create, read, modify and delete data in the database, which can also be called CRUD. A special language SQL (structured query language) is being used to query the data on the database.

Models are used in Django to operate with the database. They allow you to define tables and ask questions about a familiar developer of pythons, which is far more convenient.

The data collected from the database is prepared for forwarding. You can replace them with a template and send them as an HTML file. But this happens only once in a single-page application when an HTML page is created to which all JavaScript scripts are attached. The data is serialized and sent in JSON format in other cases. Despite having its own nomenclature, such as naming the callable objects generating the HTTP responses "views", the core Django framework can be seen as an MVC architecture.It consists of an object-relational mapper (ORM) that mediates between data models (defined as Python classes) and a relational database ("Model"), a system for processing HTTP requests with a web templating system ("View"), and a regular-expression-based URL dispatcher ("Controller"). The web application architecture is shown on the Figure 3 below:

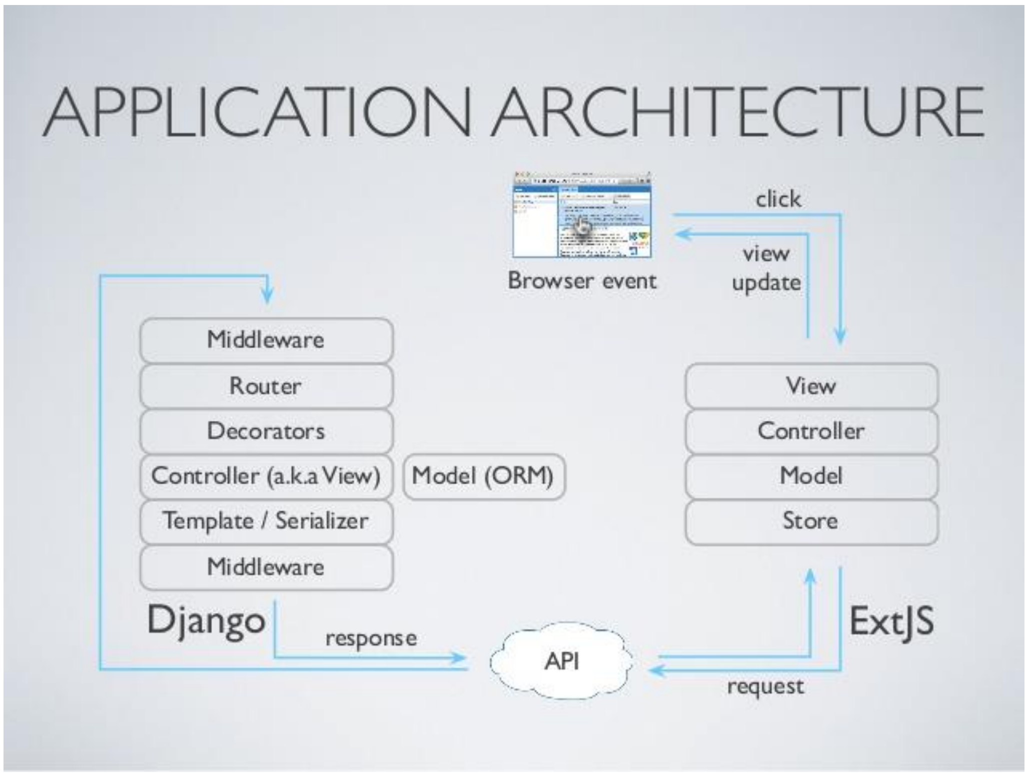


Figure 3 – Architecture of the web application

Also included in the core framework are:

- a lightweight and standalone web server for development and testing;
- a form serialization and validation system that can translate between HTML forms and values suitable for storage in the database ;
- a template system that utilizes the concept of inheritance borrowed from object-oriented programming ;
- a caching framework that can use any of several cache methods support for middleware classes that can intervene at various stages of request processing and carry out custom functions ;

- an internal dispatcher system that allows components of an application to communicate events to each other via pre-defined signals an internationalization system, including translations of Django's own components into a variety of languages;
- a serialization system that can produce and read XML and/or JSON representations of Django model instances;
- a system for extending the capabilities of the template engine;
- an interface to Python's built-in unit test framework.

2.3 Data Storage

For the storage of data, SQLite database solution was chosen. SQLite application database is easily embedded. SQLite is a software library that provides a relational database management system. The lite in SQLite means light weight in terms of setup, database administration, and required resource. SQLite has the following noticeable features: self-contained, serverless, zero-configuration, transactional.

Normally, an RDBMS such as MySQL, PostgreSQL, etc., requires a separate server process to operate. The applications that want to access the database server use TCP/IP protocol to send and receive requests. This is called client/server architecture. SQLite does NOT work this way. SQLite does NOT require a server to run. SQLite database is integrated with the application that accesses the database. The applications interact with the SQLite database read and write directly from the database files stored on disk.

The following Figure 4 illustrates the SQLite server-less architecture:

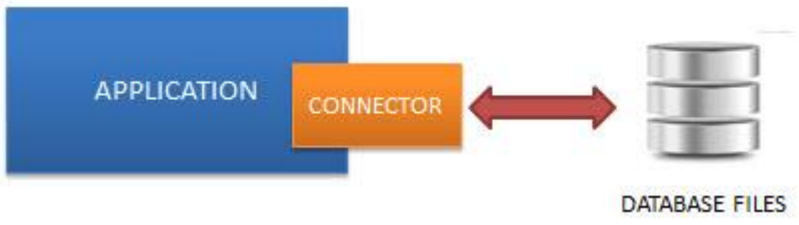


Figure 4- SQLite architecture

SQLite is self-contained means it requires minimal support from the operating system or external library. This makes SQLite usable in any environments especially in embedded devices like iPhones, Android phones, game consoles, handheld media players, etc. SQLite is developed using ANSI-C. The source code is available as a big sqlite3.c and its header file sqlite3.h. If you want to develop an application that uses SQLite, you just need to drop these files into your project and compile it with your code.

Because of the serverless architecture, you don't need to "install" SQLite before using it. There is no server process that needs to be configured, started, and stopped. In addition, SQLite does not use any configuration files.

All transactions in SQLite are fully ACID-compliant. It means all queries and changes are Atomic, Consistent, Isolated, and Durable. In other words, all changes within a transaction take place completely or not at all even when an unexpected situation like application crash, power failure, or operating system crash occurs.

SQLite uses dynamic types for tables. It means you can store any value in any column, regardless of the data type.

SQLite allows a single database connection to access multiple database files simultaneously. This brings many nice features like joining tables in different databases or copying data between databases in a single command.

SQLite is capable of creating in-memory databases which are very fast to work with.Since this system is based on files, particularly in comparison to network Data Base Management System, it offers a big variety of tools to work with it. Calls are made specifically to files once functioning with this DBMS (data is stored on those files), rather than ports and sockets in DBMS network. And that is why SQLite is both very fast and powerful due to having to serve library technology. The architecture of connection of database with Python is shown below on Figure 5:

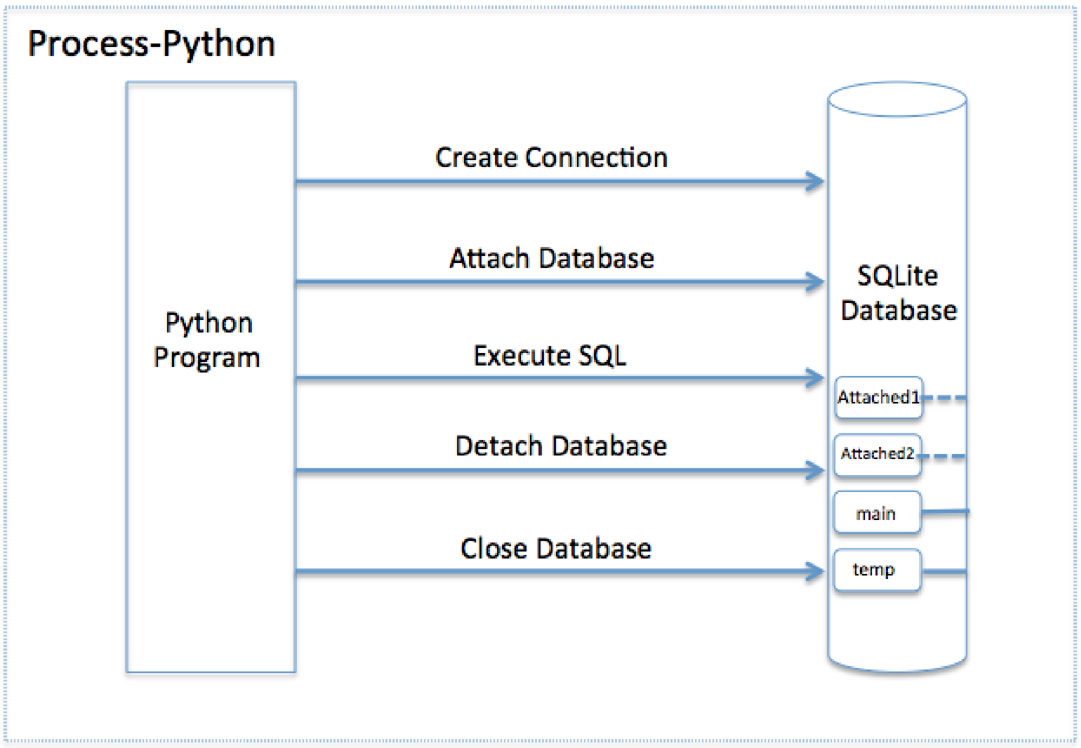


Figure 5 – Connection of SQLite with Python

SQLite advantages:

- file structure— the intire database comes from a single file, so transferring it to different machines is very easy;
- standards used. Although this database may seem primitive, but it uses Structured Query Language ery well. Some attributes (RIGHT OUTER JOIN or FOR EACH STATEMENT) are excluded, but the main features are still supported;
- great in development and testing-it is often essential to scale up along the way of application development. For this reason, SQLite provides all you need as it consists of just one file and a library written on C programming language.

When using SQLite, I do not need to create a database, because SQLite uses files of the file system to store its data. By default, the database connection parameters are defined in the project configuration file, settings.py, with the DATABASES option, which is {} by default. Base default must be defined necessarily. The dictionary can also define any number of other databases. The easiest option is to use a single SQLite database:

```
DATABASES = {
   'defaultdb': {
      'ENGINEdb': 'django.db.backends.sqllite3',
      'NAMEdb': 'webapplicationdb'
   }
}
```

2.4 Web API methods

Web API is a new environment for web applications based on instructions and patterns. The Web API enables the developer to develop simple Web API web services with just a low amount of code and configuration using a basic controller paradigm. An API is primarily an interface that enables developers to build an application using ready-made blocks. The API can transmit data in a format beyond standard HTML for web applications, making it convenient to use when writing your applications. An architecture and connection of web application with Web API is shown in the Figure 6 below:

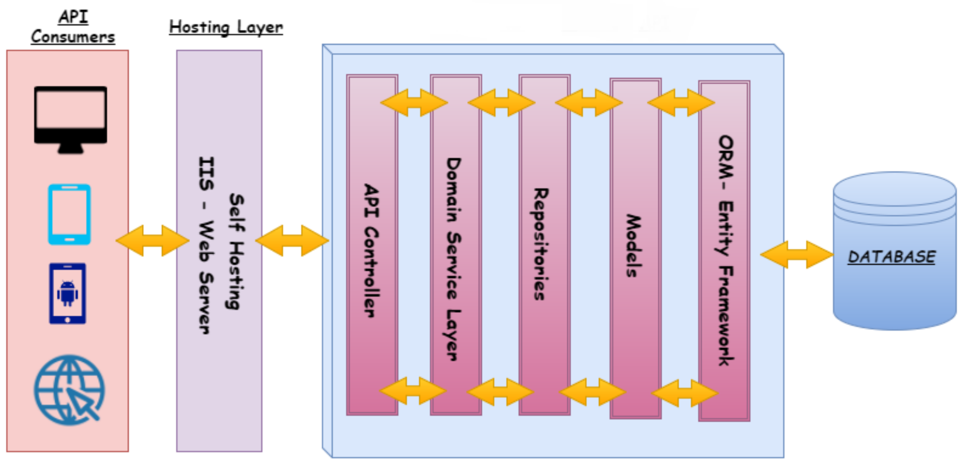


Figure 6 – Web API integration architecture

An API is data sent to other services by a service. Social networks provide third-party applications with data about users and their behavior. Independent developers are able to use this data and develop applications for various tasks: dating services, audience overview, social mail and dynamic cover management, along with games and chatbots.

API Vkontakte – This interface makes it possible to receive information from the vk.com database with the help of http requests to the server. In their architecture you do not have to know precisely how the foundation is built and the types of table and field it includes. It is sufficient that it is "known" to the API request. The service on its own specifically defines the request syntax as well as the type of data to be returned.

Telegram's development is primarily determined by a high number of bots — small service robots. Every person who is familiar with programming can develop them. Telegram API Bot is a programming interface for developing your own bot.

The API contains objects and instructions to set the Telegram bot's behavior. You can develop your own program codes using the interface, which begin to work as bots when launched in Telegram.

When developing, each bot is assigned a unique token just like in VK API. In order to receive a token, you need to sign up and fill out a form, after which the token automatically appears on your account.

2.5 REST API technology

REST is an architectural solution that organizes web applications through a network interaction. A specific REST exchange method is used if six requirements are met:

- the client-server model is being used-i.e. in the interaction of the two REST systems, one is always the client and the other is the server;
- missing state - i.e. the server "does not remember" if the client interacted earlier. For this reason, each request is processed as if the server "first sees" the requesting party (and therefore all the necessary data must be transmitted to the server in each request, for example, in order to authorize the client);
- caching - Clients can cache server-received data;
- Interface uniformity-here are four vague requirements that are best read alone;
- layers and hierarchy-the server can actually be hidden behind another "proxy" server, it's convenient for server-side load balancing. Those implies the ability to use the server hierarchy;
- on-demand code (optional limitation)-the skill to load executable files to the client from the server.

All server interaction can be reduced to 4main operations receiving server data (usually in JSON or XML), adding new data to server, modification of existing server data and deleting server data. The operation of data recovery can not lead to a change in the status of the server. The REST API architecture is shown below on the Figure 5 below:

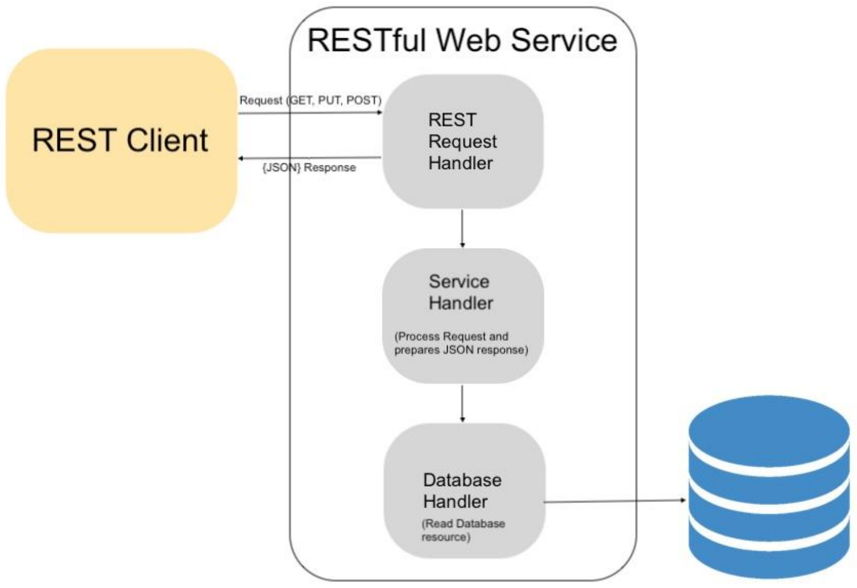


Figure 7 – REST API architecture

An HTTP request method is used for each type of operation:

- receiving – GET;
- add – POST;
- change – PUT ;
- delete – DELETE.

Regarding learning REST API. REST is a software architecture style for building distributed, scalable web services that use HTTP requests .

Hypertext Transfer Protocol (HTTP), one of the protocols of the TCP / IP stack (Transmission Control Protocol / Internet Protocol), was originally designed for publishing and receiving HTML (Hyper Text Markup Language) pages and is now used for distributed information systems. HTTP is used on the World Wide Web for data transmission and is one of the most widely used application protocols.

HTTP defines a request / response protocol. When a client, such as a web browser, sends a request message to the server, the HTTP protocol determines the types of messages that the client uses to request the web page, as well as the types of messages that the server uses to respond. The three common message types are GET, POST and PUT.

POST and PUT are used to send messages that upload data to a web server. For example, when a user enters data in a form embedded in a web page, POST includes the data in a message sent to the server. PUT uploads resources or content to a web server.

Being extremely flexible, HTTP is not a secure protocol. POST messages upload information to the server in plain text that can be intercepted and read. Similarly, server responses, as a rule, HTML pages are also not encrypted. For secure communication over the Internet, the Secure HTTP Protocol (HTTPS) is used to access or publish information on a web server. HTTPS can use authentication and encryption to secure data as it travels between client and server. HTTPS defines additional rules for passing data between the Application and Transport Layers.

JSON (JavaScript Object Notation) is a simple data exchange format that is easy to read and write by both humans and computers. JSON is a text format completely independent of the implementation language, but it uses conventions familiar to programmers in C-like languages like C, C ++, C # , Java, JavaScript, Perl, Python and many others. These properties make JSON the ideal language for exchanging data.

JSON is based on two data structures:

- collection of key or value pairs. In different languages, this concept is implemented as an object, record, structure, dictionary, hash, named list, or associative array;
- an ordered list of values. In most languages, this is implemented as an array, vector, list, or sequence.

These are universal data structures. Almost all modern programming languages support them in any form. It is logical to assume that a data format independent of a programming language should be based on these structures.

In JSON notation, it looks like this: object is an unordered set of key or value pairs. An object begins with an opening brace and ends with a closing brace. Each name is followed by a colon, key or a comma separates value pairs. An array is an ordered collection of values. The array begins with an opening square bracket and ends with a closing square bracket. Values are separated by commas. The value can be a string in double quotes, a number, true, false, null, an object or an array. These structures can be nested. A string is a Unicode character set and a null terminator enclosed in double quotes, using the backslash as an escape character. A character is represented as a single character string. The number is represented in the same way as in C or Java, except that only the decimal number system is used. Spaces can be used between any tokens. Excluding some coding details, the foregoing fully describes the language. The described architecture is shown on the Figure 6 below:

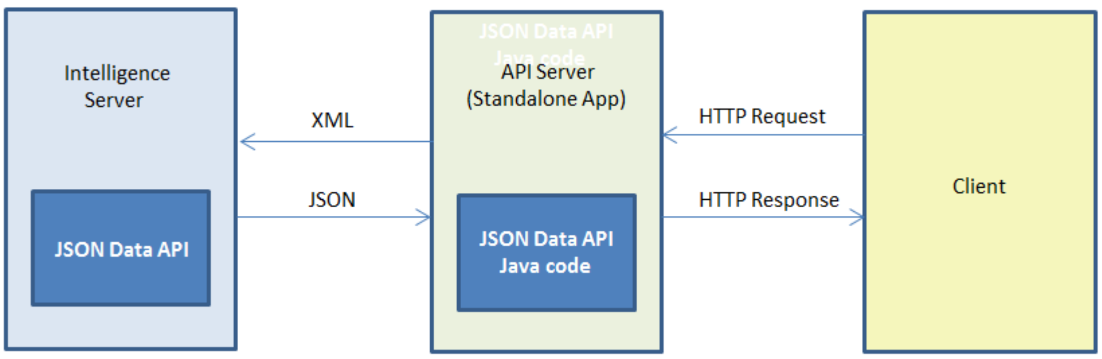


Figure 8 – JSON API architecture

As shown on the Figure 6, the general workflow for the JSON Data API works like this. The client sends an HTTP request to the JSON Data API server, using the JSON Data API syntax. The JSON Data API server consumes the incoming HTTP request, validates its syntax, converts it into an XML command that the Intelligence Server can consume, and passes the XML command to the Intelligence Server. The

Intelligence Server consumes the XML command, generates the requested results, and passes them back to the JSON Data API server as JSON. The JSON Data API server combines all of the necessary results, builds an HTTP response, and passes it back to the client.

3 Development of a system

First step of creating a web application is to set up the environment. I downloaded the database of choice SQLite, installed Python and Django.

After that, I created a project on Django framework. It consists of two main parts, which are the project itself, and its apps and uses an MVC web architecture. Then made sure that it is situated in the right directory and the project is ready for future development. After that step, I created an application called mychatbotwebapplication.

Then I also created a view, which is a set of functions and classes inside the views.py file. After that, I wrote a view function that will call our html page.

One statement is used for the app's view function, request. That object was an HttpRequestObject generated every time a page is loading. It consists of request data, including the method that can take multiple values, including GET and POST queries.

Now that the view function was created, I created the user's HTML page.

After that I needed to work with URLs so that the user could see my web application. URLs are placed in the url.py file. Wrote the following script to implement it. Then I imported the path of the html page into the url.py file to connect them. Now when I have done that if I visit the localhost server the html page will be shown.

So I had views and URLs and another main part that was missing is model. In order for the system to work properly with the model, I created database using SQLite. Firstly, I created tables auth_user_user_permissions, auth_user, auth_group_permissions, auth_group.

auth_user_user_permissions table consists of id, user_id, permission_id, permission_auth attributes.

auth_user table has attributes id, password, last_login, username, first_name, last_name, date_joined, is_active, is_staff, email.

auth_group_permissions has id, group_id, auth_permisson.

auth_group consists of id, name.

These tables are used for user authorization.

3.1 Database development

In SQLite, sqlite3 command is used to create a new SQLite database. You do not need to have any special privilege to create a database.

Following is the basic syntax of sqlite3 command to create a database is $sqlite3 DatabaseName.db Always, database name should be unique within the RDBMS.
To create a new database <webapplication.db>, then SQLITE3 statement would be as follows:

```
$sqlite3 webapplication.db
SQLite version 3.7.15.2 2013-01-09 11:53:05
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

The above command will create a file webapplication.db in the current directory. This file will be used as database by SQLite engine. If you have noticed while creating database, sqlite3 command will provide a sqlite> prompt after creating a database file successfully.

Once a database is created, I can verify it in the list of databases using the following SQLite .databases command.

```
sqlite>.databases
seq  name          file
0    main          /home/sqlite/webapplication.db
```

I will use SQLite .quit command to come out of the sqlite prompt as follows:

```
sqlite>.quit
$
```

The .dump Command

I can use .dump dot command to export complete database in a text file using the following SQLite command at the command prompt. $sqlite3 testDB.db .dump > webapplication.sql. The above command will convert the entire contents of webapplication.db database into SQLite statements and dump it into ASCII text file webapplication.sql. I can perform restoration from the generated webapplication.sql in a simple way as follows :

```
$sqlite3 webapplication.db < testDB.sql
```

At this moment my database is empty, so I can try above two procedures once you have few tables and data in your database.

Consider a case when I have multiple databases available and I want to use any one of them at a time. SQLite ATTACH DATABASE statement is used to select a particular database, and after this command, all SQLite statements will be executed under the attached database.

If I want to attach an existing database webapplication.db, then ATTACH DATABASE statement would be as follows:

```
sqlite> ATTACH DATABASE webapplication.db' as 'TEST';
```

Use SQLite .database command to display attached database.

```
sqlite> .database
seq  name          file
0    main          /home/sqlite/ webapplication.db
2    test          /home/sqlite/ webapplication.db
```

The database names main and temp are reserved for the primary database and database to hold temporary tables and other temporary data objects. Both of these database names exist for every database connection and should not be used for attachment, otherwise you will get the following warning message.

```
sqlite> ATTACH DATABASE webapplication.db' as 'TEMP';
Error: database TEMP is already in use
sqlite> ATTACH DATABASE webapplication.db' as 'main';
Error: database TEMP is already in use
```

SQLite DETACH DATABASE statement is used to detach and dissociate a named database from a database connection which was previously attached using

ATTACH statement. If the same database file has been attached with multiple aliases, then DETACH command will disconnect only the given name and rest of the attachment will still continue. I cannot detach the main or temp databases.

If the database is an in-memory or temporary database, the database will be destroyed and the contents will be lost. Following is the basic syntax of SQLite DETACH DATABASE 'Alias-Name' statement.

```
DETACH DATABASE 'Alias-Name';
```

Here, 'Alias-Name' is the same alias, which you had used while attaching the database using ATTACH statement.

```
sqlite>.databases
seq  name          file
0    main          /home/sqlite/testDB.db
2    test          /home/sqlite/testDB.db
3    currentDB     /home/sqlite/testDB.db
```

Let's try to detach 'currentDB' from webapplication.db using the following command.

```
sqlite> DETACH DATABASE 'currentDB';
```

Now, if I will check the current attachment, I will find that testDB.db is still connected with 'test' and 'main'.

```
sqlite>.databases
seq  name          file
0    main          /home/sqlite/ webapplication.db
2    test          /home/sqlite/ webapplication.db
```

SQLite CREATE TABLE statement is used to create a new table in any of the given database. Creating a basic table involves naming the table and defining its columns and each column's data type.

CREATE TABLE is the keyword telling the database system to create a new table. The unique name or identifier for the table follows the CREATE TABLE statement. Optionally, I can specify database_name along with table_name.

Creates a COMPANY table with ID as the primary key and NOT NULL are the constraints showing that these fields cannot be NULL while creating records in this table.

```
sqlite> CREATE TABLE COMPANY(
   ID INT PRIMARY KEY     NOT NULL,
   NAME           TEXT    NOT NULL,
   AGE            INT     NOT NULL,
   ADDRESS        CHAR(50),
   SALARY         REAL);
```

Lets create one more table, which I will use in our exercises in subsequent chapters.

```
sqlite> CREATE TABLE DEPARTMENT(
   ID INT PRIMARY KEY      NOT NULL,
   DEPT           CHAR(50) NOT NULL,
   EMP_ID         INT      NOT NULL
```

);

I can verify if my table has been created successfully using SQLite command .tables command, which will be used to list down all the tables in an attached database.

```
sqlite>.tables
COMPANY     DEPARTMENT
```

A developed database with tables and connections is shown below on the Figure 7:

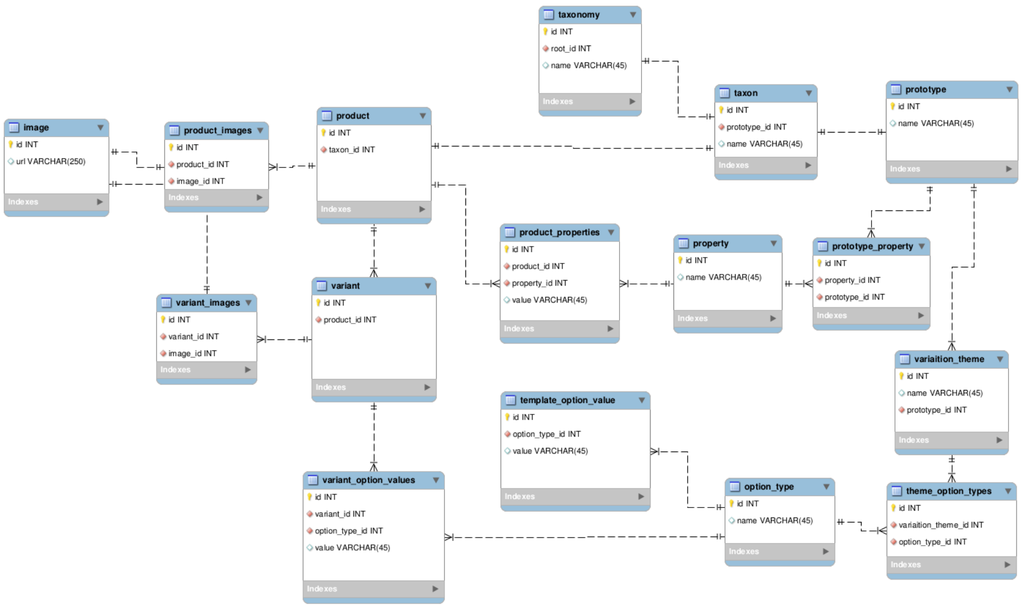


Figure 7 – Developed database for a system

Also I the database of chatbot's algorithm is hown below on the Figure 8.

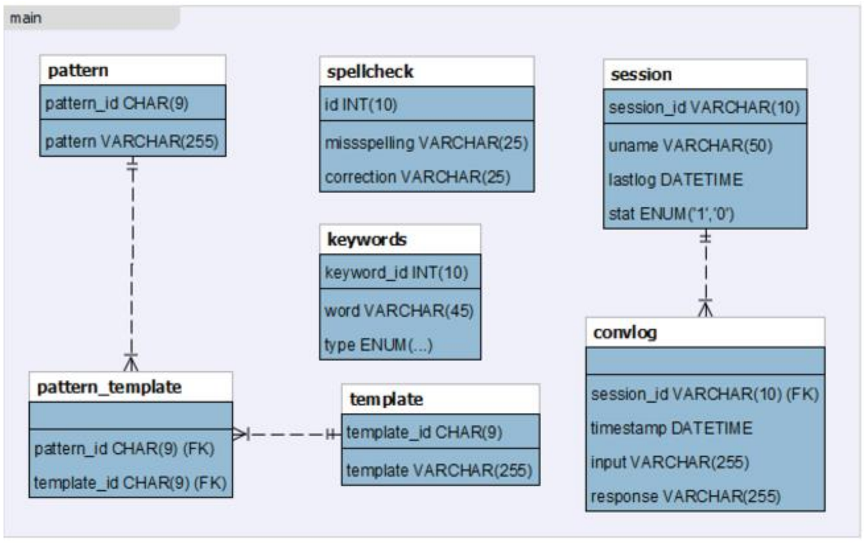


Figure 8 – Chatbot algorithm database

Here, you can see the COMPANY table twice because it is showing COMPANY table for main database and test. COMPANY table for 'test' alias created for your webapplication.db. I can get complete information about a table using the following SQLite .schema command.

```
sqlite>.schema COMPANY
CREATE TABLE COMPANY(
  ID INT PRIMARY KEY     NOT NULL,
  NAME           TEXT    NOT NULL,
  AGE            INT     NOT NULL,
  ADDRESS        CHAR(50),
  SALARY         REAL);
```

The developed system database is shown on the Figure 9:

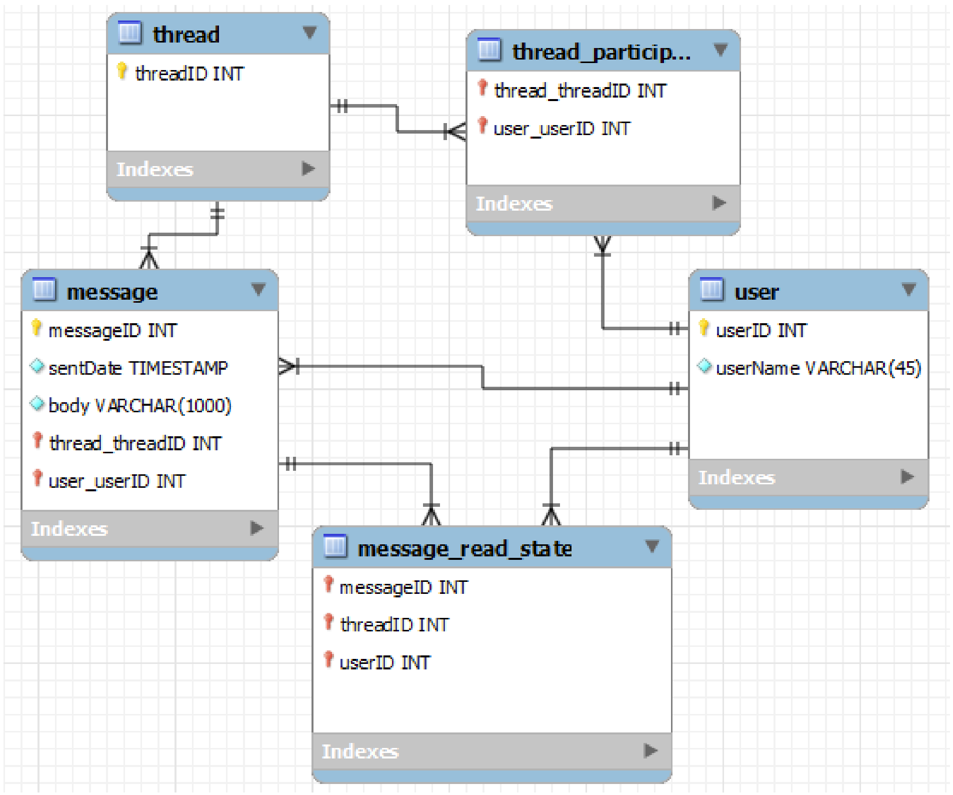


Figure 9 – Database of a messages connection

Also the database for connecting with Web API and getting messages and saving them into the database as well the replies to them is shown below on the Figure 10.

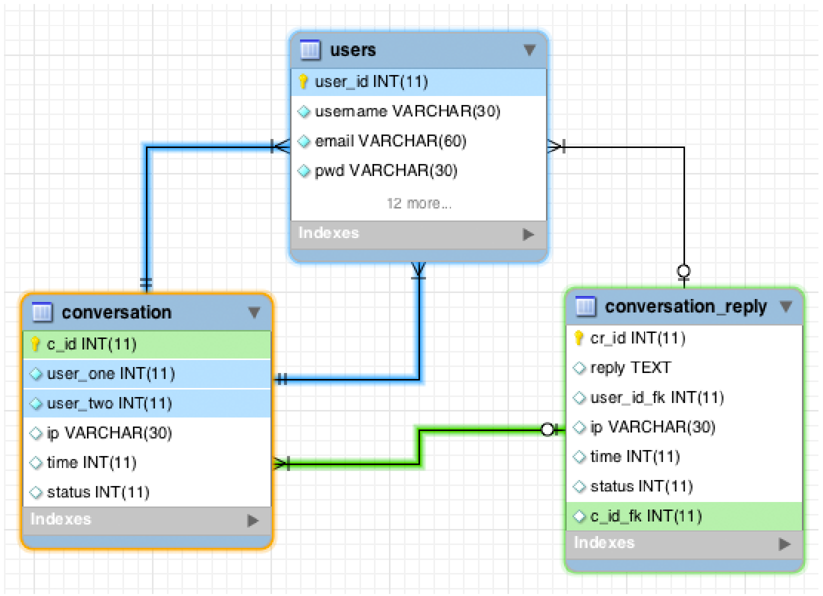


Figure 10 – Database of a saved messages and replies

3.2 Development of a chatbot

After that, I integrated my application with Vkontakte social media channel throughVK API. To get started with an API Vkontakte I got an access token, which is a special access key. Token is a string of digits and Latin characters and may refer to a user, community or application itself. After connecting the token, I can start to get data through http queries. After connecting the system with VKontakte API, I added user authorization.

Getting access token and implementing user authorization function:

```
Def authorization(events manager):
  Vk_api:
   App_ID = 586879451651561
   if entrys_logs.get( ) == '' entrys_pass.get1( ) == '':
      tkinter.messagesbox.showstheerror('Error!', 'Fill in the form!')
   else:
            Session = vk.Auths_Session(apps_id=Apps_ID,
users_login=entrys_logs.get(), users_password=entrys_pass.get(),
                   scopes='walls, messages, friends, groups, video')
         vk_apis = vk.API(Session)
         tkinters.messagesbox.showsinfo(
            'Information:','Authorization is successful!')
         auths_windows.destroy()
         mains_menus()
      except Exceptions as Error:
         tkinter.messagesbox.showserror(Information:', Authorization is not
successful!')
```

Then I decided to integrate the application with Telegram instant messenger through Telegram API. First you had to get the keys to interact with the API. An API key is a secret code that identifies a specific account and allows you to use the methods for which it is needed. In the case of a Bot API, the key acts as a path to be accessed, and in the case of a database, the API key acts as a parameter for the request.

For a bot to function, you need to create it. To do this, there is a special meta bot BotFather (@BotFather) in the telegram. It is necessary to add it through the search, in the client telegram. A list of his commands can be obtained by writing / help in the chat with him. To create a new bot, you need to write the / newbot command and in the next message send the name of the bot (must end with the word bot). In response, you will receive a message with the API key.

To get the key database API, you need to register and fill out a small form, after which the key automatically appears in your account.

After receiving the keys, tests were conducted in which simple requests were sent through the browser in order to determine in which format the data is returned. An example of the request and response is shown.

Directly writing Python code, during which our own class was developed and written to access the database, was not required to access Telegram, since the library was already written. WebHooks were chosen as the method of receiving updates, since they are more reliable than the getUpdates method.

Different requests to the base API occur almost equally, the method is selected by which the request will occur, a dictionary is formed with the parameters for this method and a GET request is sent, which returns a response in JSON format. An example of accessing the database is given below. To access the Telegram Bot API, the existing telebot library was used, and for setting up webcasts and receiving updates from Telegram was selected cherrypy web server. The class for installing webhooks are shown in the listing.

With an increase in the amount of code developed, it becomes easy to get confused in a large number of if-elif-else constructions. Handlers (or decorators) were written to solve this problem. This allows the code to be spread across various functions, which improves readability. Decorators will be checked in the order they appear in the code, that is, the first one that came up is used. Examples of decorators used are shown in the listing.

Each of the decorators performs their own part of the code, for example, the first processing start command (first input) generates three buttons with a choice and a message prompting to click on one of them. An example of processing the first entry is shown in the listing.

The callback handler divides all callbacks into three types, depending on which button was pressed. In all the buttons there is information about the mode in which it works, the page number to be opened and the field with basic information (keyword or genre). Processing modes among themselves differs little. In the general case, a search is performed on a field with information and a given page. After that, a list of the found answers and two buttons are created to go to the next or previous page. After which all this is sent to the user.

In the generated lists, in addition to the name, there are also links that, when clicked, the message is sent automatically. Such messages are processed by the following two decorators, they differ only in that one is written to search for information about the question, and the other to search for information about the answer. In each of them, a request is executed with an attached identifier, with the help of the information received, a form is filled out and sent to the user.

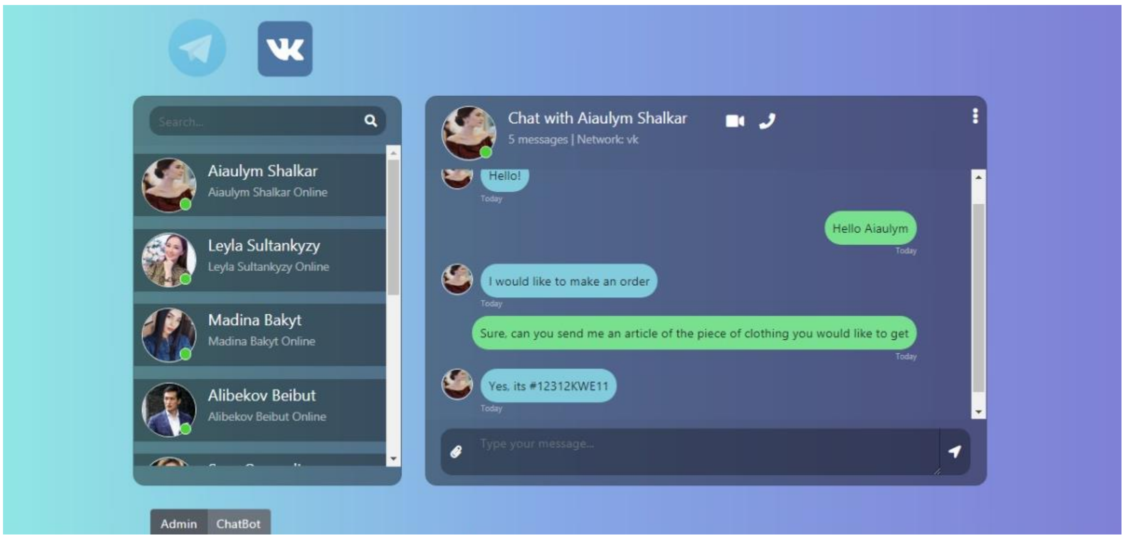
The work of the last decorator can be divided into 4 parts. The first and second is the arrival of “Question” or “Answer” messages, and in either case, a set of buttons with a callback function is formed, each button contains a genre identifier, a mode, and what needs to be searched on the first page. An example of processing a message movies is shown in the listing.

Class learn that is used for training mode and defining and initiating words and sumbols:

```
class Learn:

    def __init__(self,statements=[],falseStatements=[]):
        sentenceToWords = [s.split() for s in statements]
        if len(sentenceToWords):
            stripper = re.compile("(^[^a-z0-9]+|[^a-z0-9]+$)")
            commonWords = set(stripper.sub("",word) for word in sentenceToWords[0])
            for words in sentenceToWords[1:]:
                commonWords = commonWords.intersection(stripper.sub("",word) for word in words)
```

The developed web application chatbot interface is shown on the Figure 11 below:



Chat with Aiaulym Shalkar
5 messages | Network: vk

Hello!
Today

Hello Aiaulym
Today

I would like to make an order
Today

Sure, can you send me an article of the piece of clothing you would like to get
Today

Yes, its #12312KWE11
Today

Figure 11 – Web application chatbot interface

CONCLUSION

The central aim of this diploma work was to develop an Intellectual ChatBot system for processing customer requests from the social network channel Vkontakte and instant messenger Telegram, which would have a training mode and according to the results of the development, the goal was fully achieved. All of the previously defined objectives were implemented. The database and algorithms for a chatbot were developed and research on the integration of systems has been made. One interface system that automates customer support through chatbot for small and medium businesses were created and works as defined. The created system will allow solving a number of valuable tasks for a client, such as automating incoming and outgoing messages from social networks and instant messengers, real-time work and availability 24/7.

Also it has been found out whether chatbots are useful for the ecommerce industry and why an organisation should or should not use chatbots. For this purpose, a literature review and interviews were conducted. This examination enabled the author to gain knowledge about the usefulness and appropriateness of chatbot applications nowadays and to provide a recommendation for organisations operating in the service industry.

Looking back at the accomplished work, it can be said that the solution developed is satisfying and allows for easy deployment and development if changes need to be performed. It can be ported on virtually any system provided it can host a Python environment. Furthermore, it leaves the door open for additional languages' support and user problems. It is not a perfect solution but considering the requirements, it works well in practice.

Although chatbots have already been in use for some decades, it could be said that most of them still face some major problems concerning day-to-day business. One issue includes that users are always heterogeneous which implies that they are individuals and need answers to be personalised.

REFERENCES

1 Lana R.A, “Messenger marketing as a modern way to promote brands”, May 22, 2018. Pp 105-120.

2 Oxford University Press, “Oxford English Dictionary”, 1 August 2010. Pp 678-680.

3 Alex B.N, “Chatbots are Bots of the future”, May 2016. Pp 67-120.

4 Mazen A.T., “How Instant Translator bot is used by more than 50k users”, December 2006. Pp 25-30.

5 Ryan Kelly, “How bot attracted 1 million users without even trying”, August 2017. Pp 127.

6 Jon Bruner, Joshua Browder, “Bots that fight bureaucracy”, May 2016. Pp 28-30.

7 J. Weizenbaum, "ELIZA - A Computer Program for the Study of Natural Language Communication Between Man and Machine”, Jan 1983. Pp. 23- 28.

8 Katie McCall, “How Chatbots Benefit Brands and Customers”, May 2017. Pp 111.

9 Hussain Fakhruddin, “Why Should Businesses Start Using Chatbots” June, 2017. Pp 17-23.

10 Guttag, John V, “Introduction to Computation and Programming Using Python: With Application to Understanding Data”. 12 August 2016. Pp 11.

11 Hafsa Jabeen, “Making Web Crawlers Using Scrapy for Python”, January 11th, 2019. Pp 12.

12 Karlijn Willems, “TensorFlow Tutorial For Beginners”, Jan. 16th, 2019. Pp 13.

13 De Smedt, T. & Daelemans, W, “Pattern for Python. Journal of Machine Learning Research”, 2012. Pp 26.

14 How to start visual designing with Bokeh library on Python. URL: https://bokeh.pydata.org/en/latest/docs/user_guide.html . (visited on 18/04/2019).

15 Adrian Holovaty, “The Definitive Guide to Django: Web Development Done Right 2nd ed. Edition”, 2016. Pp 64.

16 VK Open API documentation for developers. URL: https://vk.com/dev/openapi . (visited on 21/04/2019).

17 Telegram API Bot documentation for developers. URL: https://tlgrm.ru/docs/bots/api . (visited on 21/04/2019).

18 James Higginbotham and Keith Casey Jones, “A Practical Approach to API Design”, 2014. Pp 34-56.

19 Aaron Smidt, T. & Daelemans, W, “Python for web. Introduction to Django”, 2014. Pp 38.

20 Kate Andersion, “SQLite database For Beginners”, May 1, 2006. Pp 13.

Application A

Listing of the program:

```
import sqlite3
# Create database connection to the sqlite main database
connectionObject    = sqlite3.connect("primedb.db"
#Obtain a cursor object
cursorObject        = connectionObject.cursor()
# Attach a database file
attachDatabaseSQL        = "ATTACH DATABASE ? AS billing"
dbSpec  = ("bill.db",)
cursorObject.execute(attachDatabaseSQL,dbSpec)
# Drop any existing tables with the same name
cursorObject.execute("drop table billing.cashbill")
# Create billing table in the newly attached database
createTableSQL      = "CREATE TABLE billing.cashbill(billid int, itemcode int,
rate int, quantity int, price int, taxrate int, tax int, total int)"
cursorObject.execute(createTableSQL
# Insert a row of data into the billing.cashbill table
billid      = 1
itemcode    = 256
rate        = 300
quantity    = 2
price       = 600
taxrate     = 2
tax         = 12
total       = 621
insertDataSQL       = "INSERT INTO billing.cashbill VALUES (?,
?,?,?,?,?,?,?)"
# Substitution parameters for the insert statement- ? will be replaced
by members of this tuple
insertSpec          =
(billid,itemcode,rate,quantity,price,taxrate,tax,total)
#Execute SQL insert using parameter substitution
cursorObject.execute(insertDataSQL, insertSpec)
# Commit the changes
connectionObject.commit()
# Query the cashbill tabl
querySpec = (billid,)
cursorObject.execute("SELECT * FROM billing.cashbill WHERE
billid=?", querySpec)
# print the record
print("Fetched record from the attached SQLite database table:");
print(cursorObject.fetchone())
```

```
# detach the database
detachDatabaseSQL   = "DETACH DATABASE billing"
cursorObject.execute(detachDatabaseSQL
# Close the database connection as the resource is no longer needed
     CREATE TABLE contacts (
      contact_id INTEGER PRIMARY KEY,
      first_name TEXT NOT NULL,
      last_name TEXT NOT NULL,
      email text NOT NULL UNIQUE,
      phone text NOT NULL UNIQUE
     );
     CREATE TABLE groups (
      group_id integer PRIMARY KEY,
      name text NOT NULL
     );
     CREATE TABLE contact_groups (
      contact_id integer,
      group_id integer,
      PRIMARY KEY (contact_id, group_id),
      FOREIGN KEY (contact_id) REFERENCES contacts (contact_id)
      ON DELETE CASCADE ON UPDATE NO ACTION,
      FOREIGN KEY (group_id) REFERENCES groups (group_id)
      ON DELETE CASCADE ON UPDATE NO ACTION
     ); CREATE TABLE customers (
       id INTEGER PRIMARY KEY,
       company_name TEXT NOT NULL,
       street_address TEXT NOT NULL,
       city TEXT NOT NULL,
       state TEXT NOT NULL,
       zip TEXT NOT NULL
     ); CREATE TABLE orders (
       id INTEGER PRIMARY KEY,
       customer_id INTEGER,
       salesperson_id INTEGER,
       FOREIGN KEY(customer_id) REFERENCES customers(id),
       FOREIGN KEY(salesperson_id) REFERENCES salespeople(id)
     ); CREATE TABLE order_items (
       id INTEGER PRIMARY KEY,
       order_id INTEGER,
       product_id INTEGER,
       product_quantity INTEGER,
       FOREIGN KEY(order_id) REFERENCES orders(id),
```

Continue of application A

```
  FOREIGN KEY(product_id) REFERENCES products(id)
); INSERT INTO customers VALUES (null, 'ACME, INC.', '101 Main Street', 'Anchorage', 'AK', '99501');
INSERT INTO customers VALUES (null, 'FOOBAR', '200 Foo Way', 'Louisville', 'KY', '40207');
INSERT INTO orders VALUES (null, 1, 1);
INSERT INTO orders VALUES (null, 2, 2);
INSERT INTO order_items VALUES (null, 1, 1, 5);
INSERT INTO order_items VALUES (null, 1, 2, 8);
INSERT INTO order_items VALUES (null, 2, 3, 6);
INSERT INTO order_items VALUES (null, 2, 1, 10);
connectionObject.close()
import fileinput
import tkinter
import tkinter.filedialog
import tkinter.messagebox
import tkinter.ttk as ttk
from itertools import cycle
from time import sleep
import vk
def main_auth(event):
global vk_api
App_ID = 5868794
if entry_log.get() == '' or entry_pass.get() == '':
tkinter.messagebox.showerror('Error', 'Fill the whole field')
else:
try:
Session = vk.AuthSession(app_id=App_ID,
user_login=entry_log.get(), user_password=entry_pass.get(),
scope='wall, messages, friends, groups, video')
vk_api = vk.API(Session)
tkinter.messagebox.showinfo(
'Information', 'Login is successful!')
auth_window.destroy()
main_menu()
except Exception as Error:
tkinter.messagebox.showerror('Information', Login error
')
def update(event):
def send(event):
try:
vk_api.messages.send(
```

Continue of application A

```
user_id=151911284, message=entry_message_1.get())
tkinter.messagebox.showinfo('Information', 'Message has been sent')
contacts.destroy()
except:
tkinter.messagebox.showerror('Error', 'Message hasn't been sent')
contacts = tkinter.Tk()
label_update = tkinter.Label(contacts, text='Message', font='arial 14')

label_update.grid(row=0, column=0)
entry_message_1 = tkinter.Entry(contacts)
entry_message_1.grid(row=0, column=1)
send_btn = tkinter.Button(
contacts, text='Send', width=11, height=0, font='arial 14')
send_btn.grid(row=1, column=1, sticky='e')
send_btn.bind('<Button-1>', send)
def delete_all_groups(event):
'''bot for delete groups'''
def start_delete_groups():
pb_hd_delete_groups = ttk.Progressbar(root, length=321,
orient='horizontal',
mode='determinate', max=int(find_groups[0] + 1))
pb_hd_delete_groups.pack()
pb_hd_delete_groups.start()
for Leave in find_groups[1:]:
root.update()
vk_api.groups.leave(group_id=Leave)
sleep(0.2)
pb_hd_delete_groups.destroy()
find_groups = vk_api.groups.get(count=1000)
ask = tkinter.messagebox.askyesno(
'Continue', 'Found {} users.
Delete?'.format(find_groups[0]))
if ask == True:
start_delete_groups()
tkinter.messagebox.showinfo(‘Information', 'Done')
def delete_all_friends(event):
'''bot for delete friends'''
del_friends = vk_api.friends.get()
def start_delete_friends():
pb_hd_delete_friends = ttk.Progressbar(root, length=321,
orient='horizontal',
mode='determinate', max=int(len(del_friends) + 1))
```

Continue of application A

```
pb_hd_delete_friends.pack()
pb_hd_delete_friends.start()
for delete in del_friends:
root.update()
vk_api.friends.delete(user_id=delete)
sleep(0.2)
pb_hd_delete_friends.destroy()
ask = tkinter.messagebox.askyesno(
'Continue', 'Found {} users.
Delete?'.format(len(del_friends)))
if ask == True:
try:
start_delete_friends()
tkinter.messagebox.showinfo('Information', 'Done')
except:
tkinter.messagebox.showerror('Error', Error try again')

def clear_message(event):
'''clear all your message'''
find_message = vk_api.messages.getDialogs(count=200)

def start_delete_message():
pb_hd_clear_message = ttk.Progressbar(root, length=321,
orient='horizontal',
mode='determinate', max=int(find_message[0] + 1))
pb_hd_clear_message.pack()
pb_hd_clear_message.start()
for remowe in find_message[1:]:
root.update()
vk_api.messages.deleteDialog(user_id=remowe['uid'])
sleep(0.3)
pb_hd_clear_message.destroy()
ask = tkinter.messagebox.askyesno(
'Continue', 'Found {} messages.
Delete?'.format(find_message[0]))
if ask == True:
try:
start_delete_message()
tkinter.messagebox.showinfo('Information', 'Done')
except:
tkinter.messagebox.showerror('Error', 'Error try again later')
def clear_wall(event):
```

```
posts = vk_api.wall.get()
def start_clear_wall():
pb_hd = ttk.Progressbar(root, length=321, orient='horizontal',
mode='determinate',
max=int(posts[0] + 1))
pb_hd.pack()
pb_hd.start()
for post in posts[1:]:
root.update()
vk_api.wall.delete(post_id=post['id'])
sleep(0.2)
pb_hd.destroy()
ask = tkinter.messagebox.askyesno(
'Continue', 'Found {} posts. Delete?'.format(posts[0]))
if ask == True:
try:
start_clear_wall()
tkinter.messagebox.showinfo('Information', 'Done')
except:
tkinter.messagebox.showerror('Error', ''Error try again later
'')
def add_friends(event):
def start_add_friends(event):
search_users = vk_api.users.search(age_from=entry_min_age.get(),
age_to=entry_max_age.get(),
sex=0, sort=1,
online=1, count=1000)
tkinter.messagebox.showinfo(
'Information', 'Found {}
users.format(search_users[0]))
friends_bot.destroy()
status = tkinter.Tk()
status.geometry('400x340')
tx = tkinter.Text(status,
font=('times', 12),
width=62, height=15,
wrap=tkinter.WORD)
tx.pack()
btn_stop = tkinter.Button(status,
text='Отмена',
font='arial 14',
width=15,
```

Continue of application A

```
height=0)
btn_stop.pack()
btn_stop.bind('<Button-1>', lambda event: status.destroy())
for find_friends in search_users[1:]:
tx.insert(1.0, str('\nSending a request{} {} ID --> {}
\n'.format(find_friends['first_name'],
find_friends['last_name'],
find_friends['uid']) + '\n'))
tx.update()
try:
vk_api.friends.add(user_id=find_friends['uid'])
tx.insert(1.0, 'Request has been sent')
tx.update()
sleep(25)
except:
tx.insert(1.0, 'Request hasn't been sent')
tx.update()
sleep(1.5)
tkinter.messagebox.showinfo('Information', 'Done')
status.destroy()
friends_bot = tkinter.Tk()
label_min_age = tkinter.Label(friends_bot,
text='Minimal age',
font='arial 14')
label_min_age.grid(row=0, column=0)
label_max_age = tkinter.Label(friends_bot,
text='Minimal age',
font='arial 14')
label_max_age.grid(row=1, column=0)
entry_min_age = tkinter.Entry(friends_bot)
entry_min_age.grid(row=0, column=1)
entry_max_age = tkinter.Entry(friends_bot)
entry_max_age.grid(row=1, column=1)
btn_start = tkinter.Button(friends_bot,
text='Start',
width=11,
height=0,
font='arial 14')
btn_start.grid(row=2, column=1, sticky='e')
btn_start.bind('<Button-1>', start_add_friends)
def message_flood_bot(event):
def start_flood(event):
```

Continue of application A

```
Msg = entry_message.get()
search_users = vk_api.users.search(age_from=entry_min_age.get(),
age_to=entry_max_age.get(),
sex=1 if entry_sex.get() == 'Female' else 2,
online=1,
sort=1,
count=1000)
tkinter.messagebox.showinfo(
'Information', 'Found {} users'.format(search_users[0]))
message_flood.destroy()
status_send = tkinter.Tk()
status_send.geometry('400x340')
tx_1 = tkinter.Text(status_send,
font=('times', 12),
width=62,
height=15,
wrap=tkinter.WORD)
tx_1.pack()
btn_stop = tkinter.Button(
status_send, text='Cancel', font='arial 14', width=15, height=0)
btn_stop.pack()
btn_stop.bind('<Button-1>', lambda event: status_send.destroy())
for find_user in search_users[1:]:
tx_1.insert(1.0, '\nSending message {} {} ID --> {}
\n'.format(find_user['first_name'],
find_user['last_name'],
find_user['uid']) + '\n')
tx_1.update()
try:
vk_api.messages.send(user_id=find_user['uid'], message=Msg)
tx_1.insert(1.0, 'Message has been sent')
tx_1.update()
sleep(15)
except:
tx_1.insert(1.0, 'Message hasn't been sent')
tx_1.update()
sleep(1.5)
tkinter.messagebox.showinfo('Information', 'Done')
status_send.destroy()
message_flood = tkinter.Tk()
label_min_age = tkinter.Label(
message_flood, text='Minimal age', font='arial 14')
```

```
label_min_age.grid(row=0, column=0, sticky='e')
entry_min_age = tkinter.Entry(message_flood)
entry_min_age.grid(row=0, column=1)
label_max_age = tkinter.Label(
message_flood, text='Maximum age', font='arial 14')
label_max_age.grid(row=1, column=0, sticky='e')
entry_max_age = tkinter.Entry(message_flood)
entry_max_age.grid(row=1, column=1)
label_message = tkinter.Label(
message_flood, text='Message', font='arial 14')
label_message.grid(row=2, column=0, sticky='e')
entry_message = tkinter.Entry(message_flood)
entry_message.grid(row=2, column=1)
label_sex = tkinter.Label(message_flood, text='Пол', font='arial 14')
label_sex.grid(row=3, column=0, sticky='e')
width=11,
height=0,
font='arial 14')
btn_start.grid(row=2, column=1, sticky='e')
btn_start.bind('<Button-1>', start_add_friends)
def message_flood_bot(event):
def start_flood(event):
Msg = entry_message.get()
search_users = vk_api.users.search(age_from=entry_min_age.get(),
age_to=entry_max_age.get(),
sex=1 if entry_sex.get() == '' else 2,
online=1,
pb_hd.start()
```

Application B

Spellcheck program listing:

```
from spellcheck import *
def unit_tests():
assert correction('speling') == 'spelling'              # insert
assert correction('korrectud') == 'corrected'           # replace 2
assert correction('bycycle') == 'bicycle'               # replace
assert correction('inconvient') == 'inconvenient'       # insert 2
assert correction('arrainged') == 'arranged'            # delete
assert correction('peotry') =='poetry'                  # transpose
assert correction('peotryy') =='poetry'                 # transpose + delete
assert correction('word') == 'word'                     # known
assert correction('quintessential') == 'quintessential' # unknown
assert words('This is a TEST.') == ['this', 'is', 'a', 'test']
assert Counter(words('This is a test. 123; A TEST this is.')) == (
Counter({'123': 1, 'a': 2, 'is': 2, 'test': 2, 'this': 2}))
assert len(WORDS) == 506747
assert sum(WORDS.values()) == 1642669
assert WORDS.most_common(10) == [
('the', 79811),
('of', 40026),
('and', 38315),
('to', 28767),
('in', 22026),
('a', 21124),
('that', 12513),
('he', 12404),
('was', 11411),
('it', 10684)]
assert WORDS['the'] == 79811
assert P('quintessential') >0
assert 0.04 < P('the') < 0.08
return 'unit_tests pass'
def spelltest(tests, verbose=False):
"Run correction(wrong) on all (right, wrong) pairs; report results."
import time
start = time.clock()
good, unknown = 0, 0
n = len(tests)
for right, wrong in tests:
w = correction(wrong)
good += (w == right)
if w != right:
```

```
unknown += (right not in WORDS)
if verbose:
print('correction({}) => {} ({}); expected {} ({})'
.format(wrong, w, WORDS[w], right, WORDS[right]))
dt = time.clock() - start
print('{:.0%} of {} correct ({:.0%} unknown) at {:.0f} words per second
.format(good / n, n, unknown / n, n / dt))
def Testset(lines):
"Parse 'right: wrong1 wrong2' lines into [('right', 'wrong1'), ('right', 'wrong2')] pairs."
return [(right, wrong)
for (right, wrongs) in (line.split(':') for line in lines)
for wrong in wrongs.split()]
if __name__ == '__main__':
print(unit_tests())
spelltest(Testset(open(path.join(path.dirname(path.abspath(__file__)),'spell-testset1.txt'))))
spelltest(Testset(open(path.join(path.dirname(path.abspath(__file__)),'spell-testset2.txt'))))
```