
**NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS**

**SCHOOL OF SCIENCE**
**DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION**


**BSc THESIS**

# Modern NLU Systems &

# A Practical Approach to Chatbots


**Theodoros G. Anagnostopoulos**

**Petros D. Morfiris**

**Supervisors:** **Manolis Koubarakis,** Professor UoA

**Stavros Vassos,** Artificial Intelligence Architect



**ATHENS**

**MARCH 2020**


**ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ**

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ**
**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**


**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

# Μια πρακτική προσέγγιση στα μοντέρνα Συστήματα Κατανόησης Φυσικής Γλώσσας και τα Τσατμποτ


**Θεόδωρος Γ. Αναγνωστόπουλος**

**Πέτρος Δ. Μορφίρης**

**Επιβλέποντες:** **Μανόλης Κουμπαράκης,** Καθηγητής ΕΚΠΑ

**Σταύρος Βάσσος,** Αρχιτέκτονας Τεχνητής Νοημοσύνης



**ΑΘΗΝΑ**

**ΜΑΡΤΙΟΣ 2020**

**Bsc Thesis**

# Modern NLU Systems &<br>A Practical Approach to Chatbots

**Theodoros G. Anagnostopoulos**

**S.N:** 1115201400009

**Petros D. Morfiris**

**S.N:** 1115201300106

**SUPERVISORS:** **Manolis Koubarakis,** Processor UoA

**Stavros Vassos,** AI Architect

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

# Μια πρακτική προσέγγιση στα μοντέρνα Συστήματα Κατανόησης Φυσικής Γλώσσας και τα Τσατμποτ

**Θεόδωρος Γ. Αναγνωστόπουλος**

**S.N:** 1115201400009

**Πέτρος Δ. Μορφίρης**

**S.N:** 1115201300106

**ΕΠΙΒΛΕΠΟΝΤΕΣ:** **Μανόλης Κουμπαράκης,** Καθηγητής ΕΚΠΑ

**Σταύρος Βάσσος,** Αρχιτέκτονας Τεχνητής Νοημοσύνης

# ABSTRACT

The main theme of this Thesis are chatbots. We explore popular NLU (Natural Language Understanding) systems, which parse human made requests, and bot frameworks, which are commonly used for the development of agents and the integration of those into commonly used messaging platforms. Our main goal is to develop our own proof of concept chatbot, DiBot (Department of Informatics), whose main functionality is to help undergraduates in their university life, by answering frequently asked questions and providing them with helpful information related to the university itself and its courses.

For the bot development, Python was used as the server language, Google Dialogflow was used as the NLU system and Discord was used as the platform of choice for the chatbot deployment (the code can be found on GitHub [1]). The bot receives messages through the Discord API (Application Programming Interface) and sends the user's input to Dialogflow, so that their intent can be identified. After identifying the user's intent, a rule-based logic is followed, meaning a certain "if-then" rule is triggered, and if not, a default message is returned. The chatbot covers 6 use-cases through these rules, which are:

1. Frequently asked questions, information is retrieved from the website of our university www.di.uoa.gr
2. Announcements of courses, from www.eclass.uoa.gr
3. Deadlines of assignments, from www.eclass.uoa.gr
4. Grade for a certain course, from www.my-studies.uoa.gr
5. Average grade or GPA, from www.my-studies.uoa.gr

After a rule is triggered, a headless virtual browser is created by the server in order to log into the online platform (di.uoa.gr, eclass or my-studies in our case), the page is navigated and parsed, and the actual data are retrieved.

Afterwards, a response containing the answer is constructed and is sent back to the user.



**SUBJECT AREA:** Chatbots, Natural Language Understanding, Virtual Assistants
**KEYWORDS:** NLU, Python, Discord, Asynchronous Programming, Selenium

# ΠΕΡΙΛΗΨΗ

Το κύριο θέμα αυτής της Πτυχιακής Εργασίας είναι τα τσατμποτ. Εξετάζουμε δημοφιλή συστήματα ΚΦΓ (Κατανόησης Φυσικής Γλώσσας) τα οποία αναλύουν ανθρώπινα ερωτήματα. Επιπλέον, ρίχνουμε μια ματιά στα πιο διάσημα πλαίσια λογισμικού ανάπτυξης μποτ τα οποία χρησιμοποιούνται ευρέως για την ανάπτυξη πρακτόρων (agents) και την ενσωμάτωση αυτών σε δημοφιλείς πλατφόρμες ανταλλαγής μνημάτων.

Ο κύριος στόχος μας είναι να αναπτύξουμε ένα δικό μας proof of concept μποτ, ή αλλιώς ένα μποτ που θα εξακριβώσει τη δυνατότητα υλοποίησης της προτεινόμενης λύσης. Αυτό το μποτ λέγεται DiBot, έχοντας πάρει το όνομά του από το Τμήμα μας, το Τμήμα Πληροφορικής και Τηλεπικοινωνιών, και θα έχει ως κύριο στόχο να βοηθήσει τους προπτυχιακούς φοιτητές στη φοιτητική τους ζωή, απαντώντας σε συνήθεις ερωτήσεις και παρέχοντάς τους με πολύτιμες πληροφορίες σχετικά με αυτό και τα μαθήματά του.

Για την ανάπτυξη αυτού του μποτ και για το σέρβερ κομμάτι της εφαρμογής, έγινε χρήση της προγραμματιστικής γλώσσας Python. Ως σύστημα ΚΦΓ χρησιμοποιήθηκε το DialogFlow της Google, και ως πλατφόρμα ανταλλαγής μηνυμάτων για την ενσωμάτωσή του μποτ επιλέχθηκε το Discord. Το μποτ δέχεται μηνύματα μέσω της διεπαφής προγραμματισμού εφαρμογών (API) του Discord και στέλνει αυτό που εισήγαγε ο χρήστης στο Dialogflow, έτσι ώστε να αναγνωριστεί η πρόθεση του μηνύματος αυτού. Μετά την αναγνώριση της πρόθεσης του μηνύματός του, ακολουθείται μια λογική ικανοποίησης κανόνων για να αποφασιστεί ποια από τις 6 εξής περιπτώσεις χρήσεις θα ικανοποιηθεί:

1. Συνήθεις ερωτήσεις, πληροφορίες από την ιστοσελίδα του τμήματός μας www.di.uoa.gr
2. Ανακοινώσεις μαθημάτων, από την ιστοσελίδα www.eclass.uoa.gr
3. Προθεσμίες εργασιών, από την ιστοσελίδα www.eclass.uoa.gr
4. Βαθμός για ένα συγκεκριμένο μάθημα, από την ιστοσελίδα www.my-studies.uoa.gr
5. Μέσος Όρος μέχρι στιγμής, από την ιστοσελίδα www.my-studies.uoa.gr

Μετά την ικανοποίηση μιας από τις παραπάνω περιπτώσεις χρήσης, δημιουργείται ένας «ακέφαλος» εικονικός φυλλομετρητής από τον διακομιστή, έτσι ώστε να γίνει η σύνδεση στη σωστή ιστοσελίδα (di.uoa.gr, eclass και my-studies στην περίπτωσή μας), να διαβαστεί η κατάλληλη πληροφορίες και να παρθεί.

Τέλος, μια απόκριση η οποία θα περιέχει την απάντηση στην ερώτηση του χρήστη δημιουργείται και στέλνεται πίσω σε αυτόν.

**ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ:** Τσατμποτ, Κατανόηση Φυσικής Γλώσσας, Εικονικές Βοηθοί
**ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ:** ΚΦΓ, Python, Discord, Ασύγχρονος Προγραμματισμός, Selenium

# CONTENTS

# LIST OF TABLES

# LIST OF IMAGES

# PREFACE

This thesis was carried out in the context of completion of our undergraduate studies. We explored and took a look at popular tools for chatbot development and made the first step for people that might want to dive deeper in the world of chatbots and AI in general. We used the powerful IDE PyCharm from Jetbrains as the development environment and Github as the versioning control system.

Last but not least, we want to thank our supervisors, Manolis Koubarakis for the opportunity, and Stavros Vassos for his invaluable guidance throughout the duration of the writing of our thesis.

# 1. INTRODUCTION

## 1.1 What is a chatbot?

A chatbot is an artificial person, animal or other creature which holds conversations with humans. This could be a text based (typed) conversation, a spoken conversation or even a non-verbal conversation [2]. A chatbot can run on local computers and phones, though most of the time it is accessed through the internet. It is typically perceived as an engaging software entity which humans can talk to. It can be interesting, inspiring and intriguing. It appears everywhere, from old ancient HTML pages to modern advanced social networking websites, and from standard computers to fashionable mobile devices.

Chatbots talk in almost every major language. Their language (Natural Language Processing, NLP) skills vary from extremely poor to very clever, intelligent, helpful and funny. The same counts for their graphic design, sometimes it feels like a cartoonish character drawn by a child, and on the other hand there are photo-realistic 3D animated characters available, which are hard to distinguish from humans. And they are all referred to as ‘chatbots’.

The term chatbot is closely related to ‘chat bot’ and ‘chatterbot’. Chatterbot is more popular in relation to chatbots who talk a lot, and is not necessarily very intelligent in processing user input. Chat bot is used by technical people who consider the word ‘bot’ as a normal term for ‘robotised actions’, and for them ‘chat bot’ is a special kind of bot. The term chatbot is actually the most popular amongst these three terms and has the broadest meaning.

## 1.2 NLP, an Integral Part of Chatbots

Natural Language Processing is a subfield of computer science, in particular artificial intelligence, in conjunction with linguistics [10] [11] [9]. As its name suggests, it is used to describe multiple complex algorithms, models and techniques that are used to parse and analyze human language. This results in computers being able to somewhat understand and draw insights from natural/non-binary human languages, and even use them a form of communication.

In order for chatbots to function and provide a fitting response, they need to understand the context behind the user’s request. That is where NLP, and more specifically Natural Language Understanding– a subset of NLP, comes in. NLU focuses only on the way computers can understand human language and represent it in their own language (bytes). In general, NLU is the "reading" aspect of NLP. Semantically, NLU looks for the meaning behind the words by comparing them to similar examples. That is what, in the end, “helps” chatbots differentiate the meaning of different words and understand the context behind them.

Natural languages are often ambiguous and quite messy, as there are a lot of ways to express something, when things such as: synonyms, word order, unnecessary symbols/words/punctuation, are taken into account. This actually is a common practical

difficulty faced in the whole field of NLP. A popular model/method utilized in NLU is word2vec [4], which produces a multidimensional vector space from input string (text), so a machine can create relations between words and grasp their meaning. The latter is almost always accomplished with clustering and classifying the numeric vectors produced, via machine learning, more specifically neural networks [5].

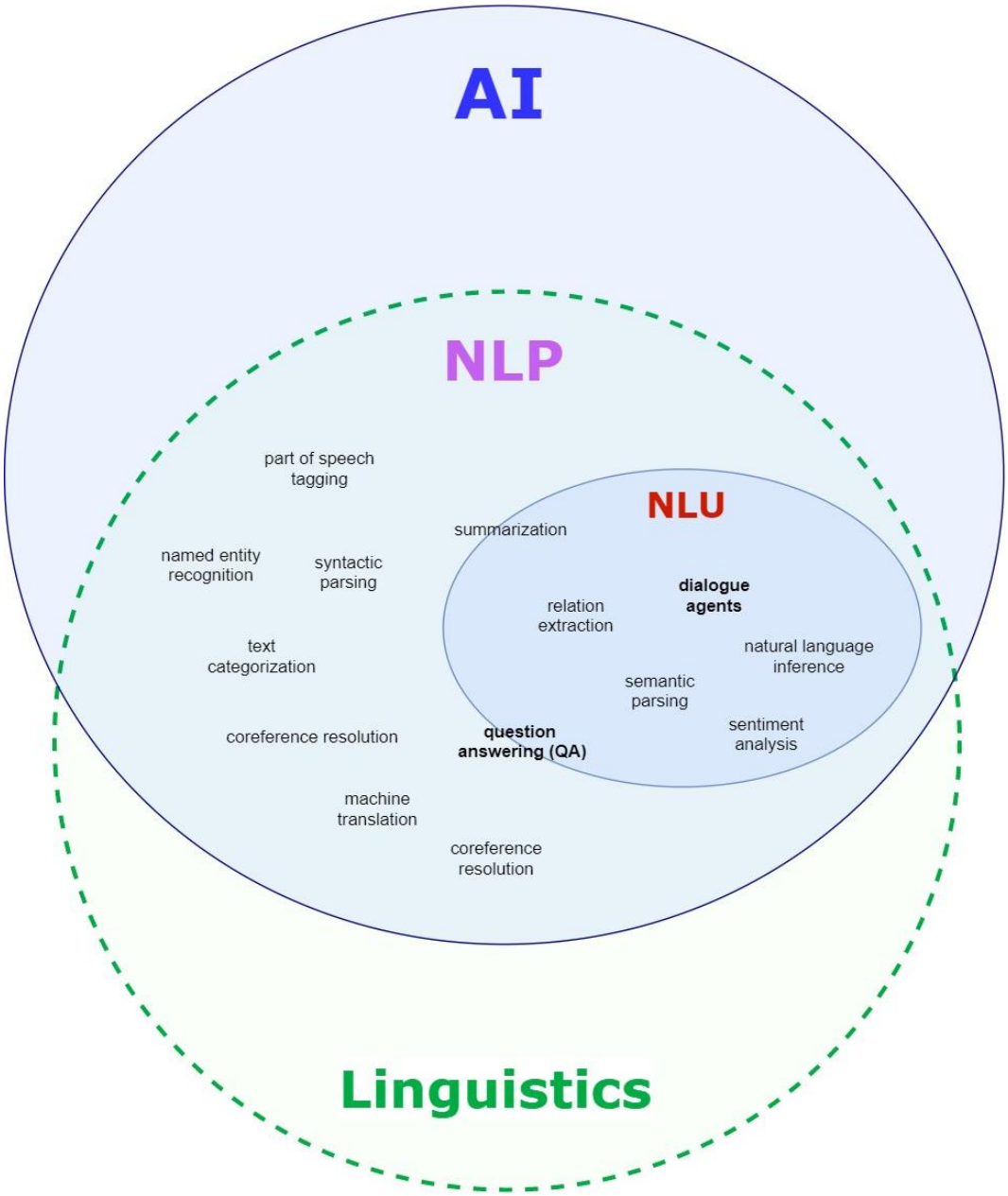


**Illustration 1: AI, NLP, NLU and linguistics**

## 1.3 Chatbots as Virtual Assistants

Chatbots, when they play their roles as virtual representatives of enterprises, are widely used by businesses globally, including the US, the UK, the Netherlands, Germany and Australia, to name a few. Additionally, the usage of this term is quite popular amongst amateur AI enthusiasts willing to spend vast amounts of time on their own intelligent creations (with diverse outcomes).

Those virtual representatives are called Virtual Assistants and they are artificial humanlike representations of an organization [3]. They are designed for conversations with (potential) customers, members or civilians (audiences of organizations). A virtual assistant does not only answer questions - it also tries to hold a conversation, to manage dialogues on issues relating to the companies' activities, or its offerings.

## 1.4 Classic Virtual Assistants

A classic example of a Virtual Assistant is Anna. She was a virtual assistant that assisted users in finding information about furniture in the huge IKEA online product catalogue. Created in 2003 by Artificial Solutions, she was one of the largest implementations of a Virtual Assistant worldwide. Anna resides in 20 countries, being able to communicate in 18 languages via all IKEA's country websites.


**Illustration 2: IKEAS's Anna virtual customer service agent**

The rise of branded artificial assistants such as Anna inspired YouTube user "earthtoerika" to produce a video. He also added the song "Boten Anna" by Basshunter, a song about a dialogue with a chatbot, and was convinced that Basshunter was inspired by Anna of IKEA when writing this song. Virtual Assistants assist individual people with various themes in

their life. For example with careers, housing and furniture, in travel, home entertainment, and music.

A predecessor of Anna, a well-known example of a chatbot, is A.L.I.C.E. (Artificial Linguistic Internet Computer Entity). Originally created by Richard Wallace in 1995 and extended since then. A.L.I.C.E. won many honors and awards in various NLP contests, such as Loebner prize, The Chatterbox challenge, and BCS Machine Intelligence Competition.

A.L.I.C.E. was written within the frame of Artificial Intelligence Markup Language (AIML), an open standard for creating any kind of chatbot, also developed by Wallace. Most AIML interpreters are offered under a free or open source license. Therefore, many “Alicebot clones” populate the internet, having been created based upon the original implementation of A.L.I.C.E. and its AIML knowledge base.

Since the steep rise of available hardware and software platforms lately, nowadays chatbots are available everywhere. Originally, they were very tight to computers, then exchangeable through tapes, discs and floppy discs, but since the Internet era they have been widespread.

## 1.5 Modern Intelligent Personal Assistants

The most popular intelligent personal assistants of the last decade are: Apple’s Siri, Amazon’s Alexa, Google’s Google Assistant and Microsoft’s Cortana. Having become an integral part of almost every household in the USA, they help you organize your life and easy out your daily life chores. They can keep track of your grocery list for you, remind you of your appointments, and memorize your never-ending to-do list [6].

In the past, personal assistants were a luxury only businessmen and film stars could afford. But, technology has brought it to the layman in a digital form. The first form of virtual assistant was introduced in 1994 by IBM. But, Apple is a pioneer in this game as they introduced the first modern digital virtual assistant – Siri.

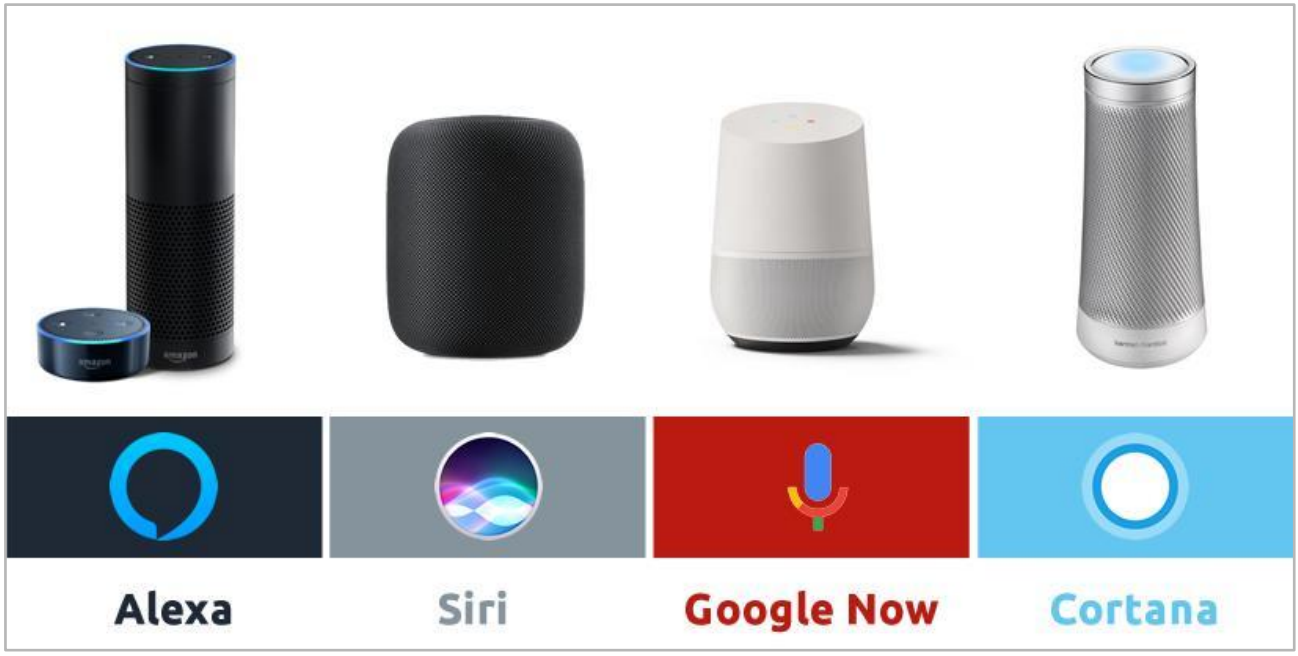


**Illustration 3: The battle between Alexa, Siri, Google Now and Cortana is big**

Siri is a virtual assistant or intelligent personal assistant or AI. A virtual assistant, like we have already explained, is a software in the smartphone or computer that acts as a personal assistant and can perform tasks like an individual. Since the introduction of Siri, there have been many new virtual assistants in the market, like Cortana, Alexa or Google Assistant.

Siri was the first ever virtual assistant to use a human-like voice. Siri responds to the wake word “Hey Siri”, and just like the other digital assistants, when asked a question, it searches the web in order to find the ideal answer. But, in comparison to the other digital assistants, it looks up other web services to gather the information, which makes the results much better in quality. As for the positives and negatives of Siri, it is very good at setting reminders, and once you have given the command to set one, it will be automatically replicated across all your iOS devices. Other good features of Siri are that it is very easy to use, has non-English language options and is very entertaining. Some key notes to take away are that, sometimes, Siri is not good at understanding accents or voice modulations and it works only on iOS devices.

Cortana is one of the most used digital assistants, considering it is integrated to Windows 10. It is very accustomed to understand and follow the colloquial language, and is very keen on asking for specifications in the tasks it is assigned and seeking confirmation before completing those tasks. Some good features of Cortana are that she is very fast, has good pronunciation, can analyze your speech, customize her language, and can be turned on or off as per user’s preference. Before using Cortana, you should be aware that your default engine will be Bing, which is the search engine it uses when it does not know the answer to

your question, and that some of your requests might be passed to Bing when they are not supposed to.

Alexa is different in outlook from its competitors due to its hardware which is not a tablet. Alexa devices can be synced with other devices – personal and home appliances, it can perform most of the tasks that other AIs can and has the ability to communicate with your family members via phone, text, or email with its 'Ask My Buddy' functionality. Some good features of it are that it is very user-friendly, and that it has a voice training feature that lets it respond to all family members. Some notes to take away are that the initial installation has to be done by an expert, the answers to general questions like for example questions for health or hobbies might not get the best results, and that it cannot send text message or email using voice command.

Even though Google Assistant was launched years after Siri, it has the advantage of being present in Android and iOS devices alike. It answers to the wake word 'OK Google' and it is so far the smartest in understanding the context of the commands and it complies by delivering the most accurate response as well. It has the easiest presence in techtown considering that Google is used by the majority of the population and it also outsmarts its contemporaries in video streaming. The best key features of Google Assistant are that it can identify different voices and it has very good sound quality. On the other hand, some features are not available in all countries and something some users might find annoying is that it cannot follow the user's voice from a distance.

**Table 1: A simple comparison between the most popular personal assistants**

| Most popular personal assistants comparison | Siri | Cortana | Alexa | Google Assistant |
|---|---|---|---|---|
| **Initial release date** | 2011 | 2014 | 2014 | 2016 |
| **Works best in** | Apple iOS devices – iPad, iPhone and iPod Touch | All Microsoft devices | Amazon Fire tablets, Fire Sticks, & the Echo devices | Google apps, Chromecast, Google Home, & other Google devices |
| **Doesn't work well with** | Gmail | YouTube | YouTube | Works well with all third-party apps |
| **Internet connection** | Wi-Fi or Mobile data | Wi-Fi or Mobile data | Only Wi-Fi | Wi-Fi or Mobile data |
| **Ways to interact** | Type out and voice commands | Type out and voice commands | Type out (in Silent Echo) and voice commands | Type out and voice commands |

# 2. DECIDING ON YOUR BOT'S DESIGN

## 2.1 First, find your bot's purpose

Before thinking about developing your own bot, for it to be great, it is going to need a purpose, personality and functionality [7]. You have to find its reason for being, meaning you have to ask yourself a simple question: *Why* do you need a bot? Your bot must fulfill a purpose, and if you cannot answer that question, it means that you have to rethink if you are really in need of one.


**Illustration 4: Find your bot's purpose**

- The users are using your chatbot for one reason, and one reason only: to seek an answer to one of their problems, UI Designer [16].

## 2.2 Choose between “rule-based” or “NLP”

After answering the why, you have to ask yourself how. Most chatbot platforms call their bot “artificial intelligence” (AI), no matter if it uses smart self-learning algorithms or just sticks to simple “IF-THEN” metrics. When you are choosing a platform, the words you are going to have to look out for are “rule-based” and “NLP”. These two categories specify how smart your bot operates within a conversation.

Rule-based bots chat based on a defined decision tree. Like a flowchart, conversations are mapped out to anticipate what a customer might ask and how the chatbot should respond. What these bots do is they match the user’s input with some predefined labels and they answer back to the user according to the label that is satisfied.

IF user's input-question contains GPA or 'average’

THEN send message with average score

In chatbots that are not just simple rule-based statements, Natural Language Understanding, a subfield of NLP, plays a crucial role as the first step of communication. For instance, when a user interacts with a bot, the role of the bot's NLU mechanism is to grasp the essence of what the user said to it, even if specific keywords were not mentioned in a fixed form. Chatbot systems only need the core meaning, that one specific case-scenario that it needs to be matched with the user's input, who had a particular interest/purpose in his mind, when interacting with them; hence the use of NLU.

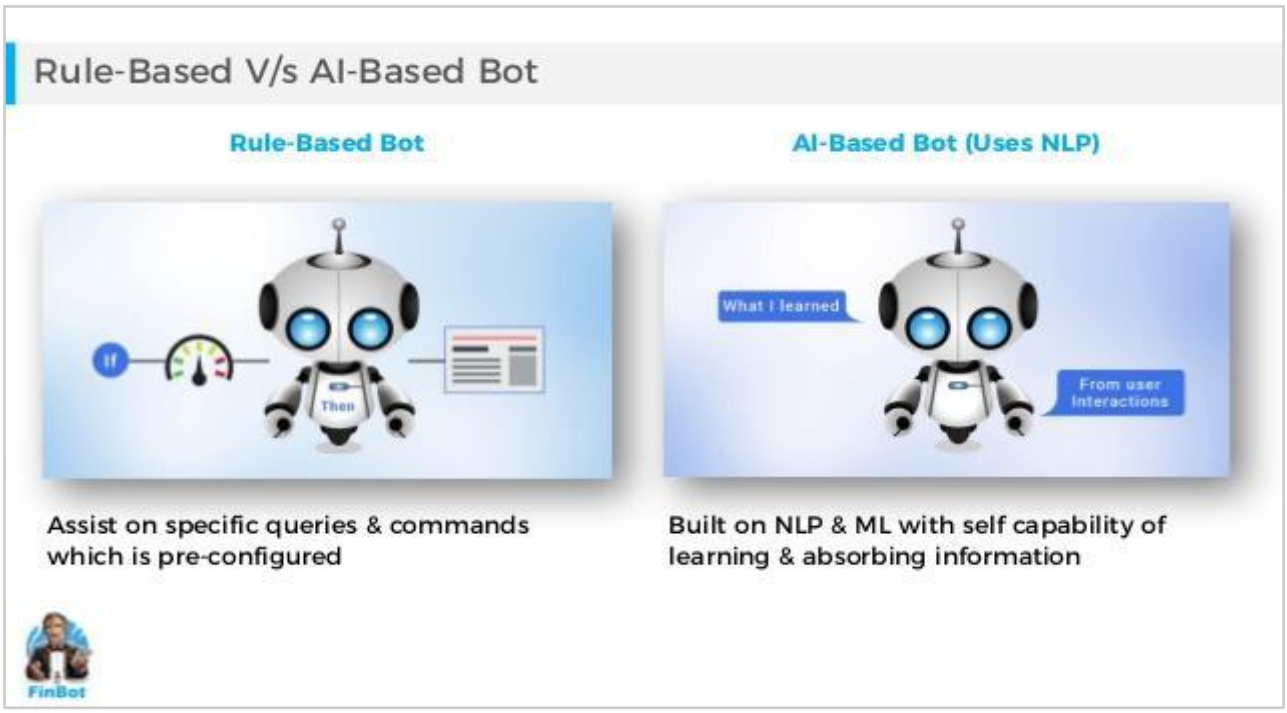


**Illustration 5: Rule-based bot versus AI-based bot**

## 2.3 Define your bot's personality and set a tone

Your bot is a representative of your brand and often the first one to greet your customers when they visit your website or application. It must be welcoming, not pushy, and it should make the customers feel at ease when chatting with it.

But how do you create a human-like personality for your bot? You first must define the role your bot plays, then you create a little background story: What's your bot's motivation, what's its specialty, what are its striking character traits? And last but not least, what name fits best its personality?

**Illustration 6: Defining your bot's personality is very important**

## 2.4 Design the flow, and keep it flowing

When chatting, your bot should be able to keep the visitors engaged, helping them resolve their request quickly and efficiently. Identifying all the possible conversation scenarios, and defining how it will handle off-topic questions and unclear commands, can be the biggest challenge. If the user is not able to communicate what they want to do, your bot should come up with suggested tasks for them to perform.

But what is really happening under the hood of your usual chatbot? What do the popular assistants, like Siri, use behind the scenes to bring you the most suitable answer to your question. In the next chapter, we are going to cover up some of the bot development steps and procedures, explaining: what is a NLU system, what is the difference with a NLU system and a QA system, what is a Bot Framework and what you can do with one, and much more.

# 3. TOOLS FOR BOT DEVELOPMENT

## 3.1 Popular Bot Frameworks and NLU Platforms

Chatbot Frameworks and platforms provide tools to build, test, deploy, and manage intelligent bots, all in one place. Through the use of modular and extensible framework provided by the SDK, tools, templates, and AI services developers can create bots that use speech, understand natural language, and handle questions and answers, and more.

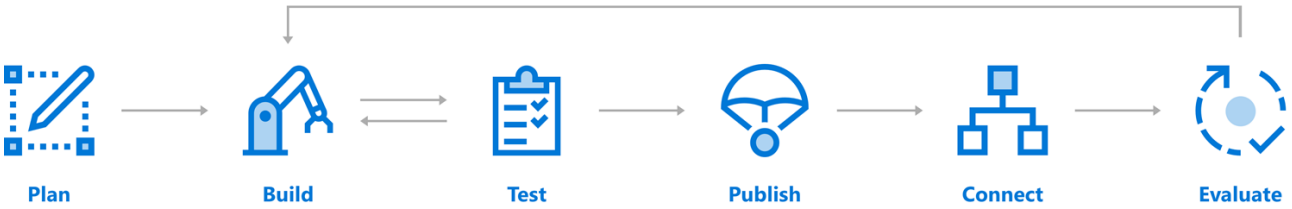


**Illustration 7: Build, test, deploy and manage intelligent bots, all in one place**

In the next chapter, we take a look and explore the core features, the most noteworthy key points and the pricing of some of the top Bot Frameworks in the industry.

### 3.1.1 Microsoft Bot Framework

The Bot Framework by Microsoft is a set of services, tools and SDKs that give a rich foundation or framework to the AI chatbot developers. Using this framework, we can build and connect intelligent bots. These enterprise-grade bots provide you with the complete ownership and control of your data [8].

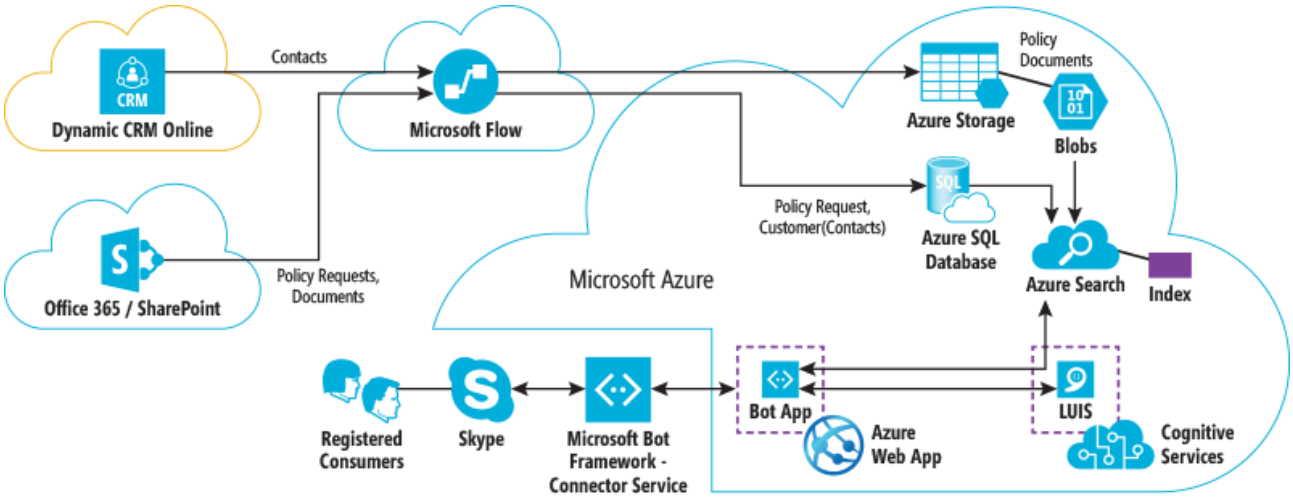


**Illustration 8: Microsoft Bot Framework**

**Features**:

- SDKs for multiple computer languages
- Prebuilt entities
- Machine learning speech to text
- Technical computer support
- Multilingual

**Notes**: Have to choose either Node.js or C# platforms for development.

**Integrate with**:

- Skype
- Slack
- Facebook Messenger
- Your own website
- Cortana
- Microsoft teams
- Kik

**Pricing**:

There are both free and paid versions in the Microsoft Bot Framework. The paid version works on the basis of pay as you use. It charges $0.50 per 1,000 messages that are exchanged via the premium channel using the AI chatbot. Flexible support plans begin at $29 per month.

### 3.1.2 RASA

Rasa is an open-source framework that has 2 major components: Rasa NLU and Rasa Core. Rasa NLU is responsible for natural language understanding while Rasa core helps create intelligent, conversational chatbots. Rasa Stack is actually leading in the open-source machine learning toolkits that help developers create better AI chatbots with minimal training data.

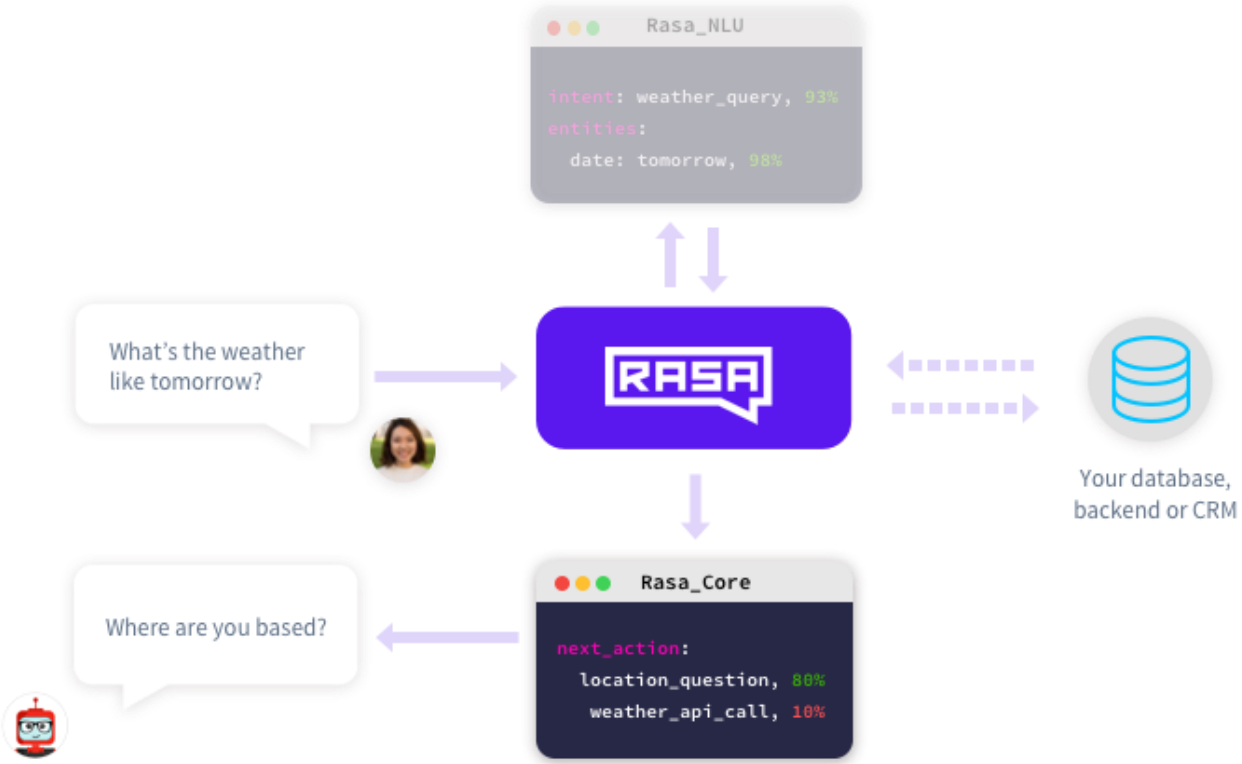


**Illustration 9: RASA: Open Source conversational AI**

**Features**:

- It can be deployed on your own server by keeping all the components in-house.
- Highly customizable, so it allows developers to create a chatbot with desired features.
- It allows multiple environments for development, staging and production.
- Rasa offers analytics for various data that allows you to understand customers better.
- This AI chatbot framework works on the basis of interactive learning, hence it keeps learning on its own as it interacts with people.

**Notes**:

Rasa is not exactly suitable for beginners so new developers could be a little intimidated by this AI chatbot framework.

**Integrate with**:

- Facebook Messenger
- Rocket.Chat
- Slack
- Telegram
- Twilio

**Pricing**:

Although open-source, Rasa also has a paid version for its AI chatbot framework. It is called Rasa platform and it is available for enterprises that need heavy functionalities and better performance. The prices are not displayed on their website and one needs to have a word with their customer representative executives to understand the pricing structure.

### 3.1.3 Wit.ai

Wit.ai has been acquired by Facebook and is free for commercial usage. This NLP platform lets the developers create the entities and intent. It is an open-source chatbot framework that began as a Y Combinator startup. Since it is still an open-source project with open tools, the developers can easily create bots with human-level intelligence without having to teach the bot the basics of human conversations.


**Illustration 10: Wit.Ai Chatbot Platform**

**Features**:

- Easy to deploy it on Facebook Messenger, so if you want to create a bot just for that platform, this is the best AI chatbot framework.
- It is open-source which means that it has a large developer community.
- The NLP engine in Wit.ai's AI chatbot framework is undeniably amongst the best and can hold its own against bigger bot-building tools.
- It offers SDKs in iOS, Node.js, Python, Ruby.
- Wit.ai supports around 80+ languages from around the world, allowing developers to easily translate data worldwide.

**Notes**:

Some of the developers have stated that it might be harder to retrieve missing parameters in Wit.ai as compared to other AI chatbot frameworks.

**Integrate with**:

A chatbot created using Wit.ai can be integrated into:

- Your app
- Your website
- Slack
- Facebook Messenger
- Wearable devices
- Home automation

**Pricing**:

Free of cost

### 3.1.4 Dialogflow

Dialogflow is an AI chatbot framework that is a subsidiary of Google. It comes with machine learning capabilities, built-in NLP features and integrations with many other popular communication platforms. With Google Dialogflow, developers are able to create highly intelligent chatbots that can understand the nuances of language and keep improving over time. This is because it is supported by Google's Cloud Natural Language, making it easier to train the bot to understand human emotions and sentiment.

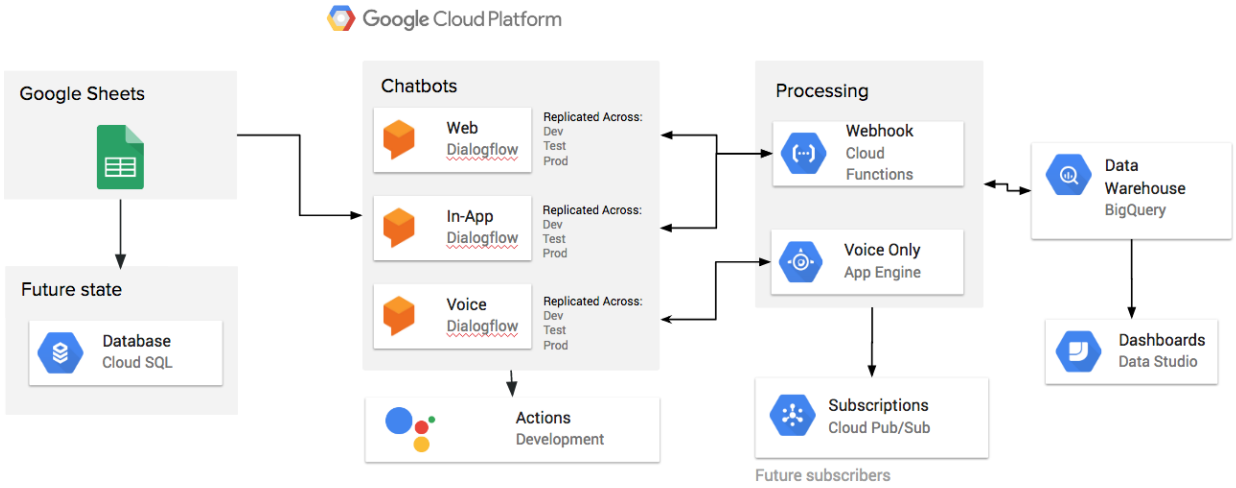


**Illustration 11: Google's DialogFlow**

**Features**:

- Supports both voice-based and text-based assistants.
- Easy to understand and learn, even for beginners.
- Best quality and rich conversations using natural language.
- It offers SDKs for more than 14 platforms.
- Google Dialogflow supports more than 20 languages from around the world.
- It can carry out various tasks such as jokes, event search and payment handling.
- With an in-line editor, the coding becomes speedier.
- It helps you do sentiment analysis for every query.
- Allows IoT integration to add an extra layer of intelligence for home automation.

**Notes**:

Fine control over dialogue processing is not available to the programmer.

**Integrate with**:

- Facebook Messenger

- Google Assistant
- Slack
- Alexa
- Cortana
- And many more

**Pricing**:

The standard edition is free of cost but you can switch to a paid version if you have to handle a lot of queries on a regular basis. Starting at $0.002 per text request, the prices vary and can go up to $0.075 per minute for all the phone calls that you process.

That is all for the Bot Frameworks. You do not necessarily have to use one to make your own chatbot, but it would certainly make your life and your development a lot easier.

## 3.2 Bot Society

Botsociety is the ultimate online design tool for conversational agents. It is used for prototyping virtual agents and even voice assistants' conversations with a user, practically presenting via pictures, videos and gifs, the functionalities of a project of that kind. In particular, using Botsociety, one can create a mockup of his project on a lot of popular platforms and apps such as Facebook Messenger, Alexa, Google Assistant, Slack, etc.

Even if this tool is more design oriented, it also helps a user with his first steps into chatbot development by exporting the "engineering part" of an assistant (intents, entities, flow) as it integrates with common bot frameworks and tools.

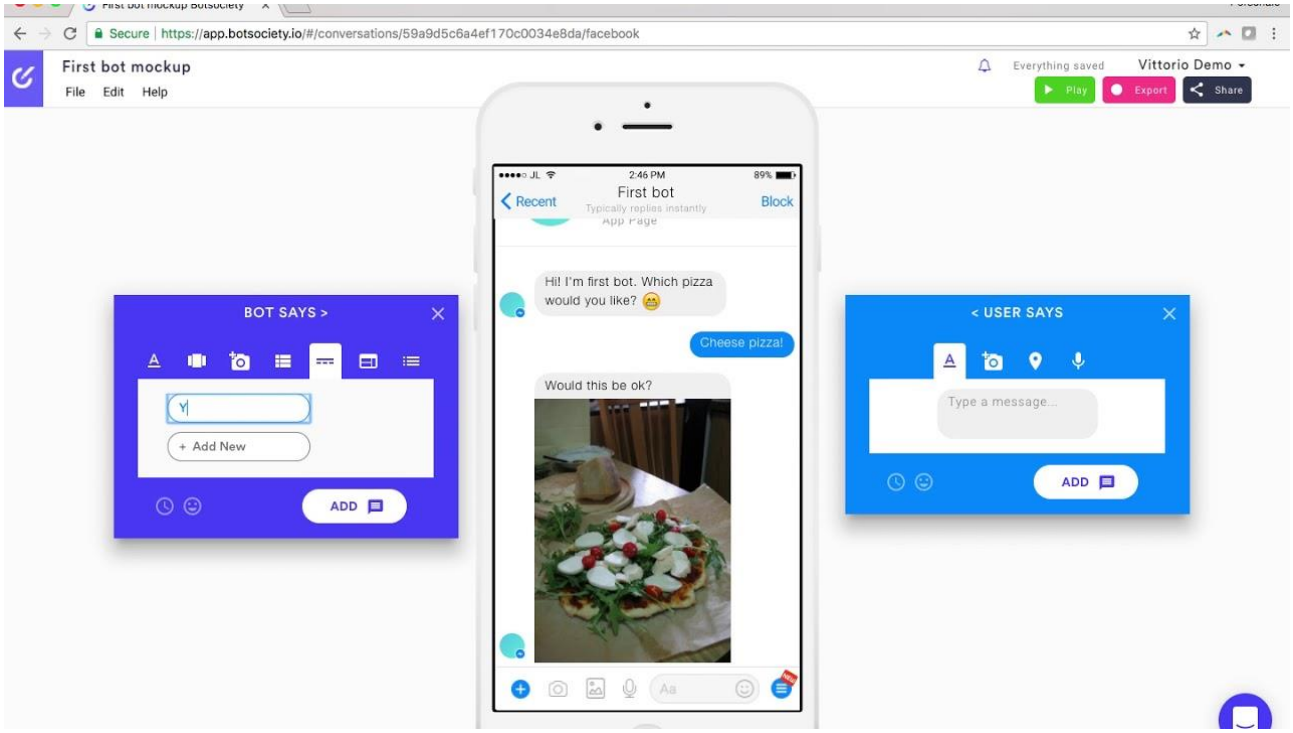


**Illustration 12: Botsociety lets you design your own dialogs and conversations**

## 3.3 Evaluation Guidelines

One important part of the chatbot development is the evaluation of it after or while the bot is being developed. Similarly to other online, desktop or mobile applications, some of the metrics that the chatbot can be tested on are [15]:

1. Its comprehension capabilities. Good comprehension capabilities should ensure a good texting and error free experience for the user.

2. How well it engages with the users. A good chatbot should be capable of initiating conversation with the users and interacting with them to share information.

3. Its response time, meaning how much time it takes for it to send back an answer to the user.

4. The functionalities it is providing. A good chatbot should be created with a variety of well-designed functionalities.

5. Its interoperability, simply meaning the ability of computer systems or software applications to exchange and make use of information. A well-designed chatbot should be deployed in such a manner that it would be capable of supporting multiple channels such as Bing, Cortana, Facebook Messenger, Kik, and Slack.

6. Its scalability, meaning how well it handles multiple users at the same times.

All the metrics that were mentioned above can be measured through means of questionnaires or even through interviews with users. A group of people can be picked to use the chatbot and be given a set of questions to answer based on if they were satisfied with it or not. Other metrics that can be used to judge the performance of a chatbot could be through automated measurements, like checking how fast its response time is, how many messages it exchanges with a user per session or how accurate its responses are.

Last but not least, the metrics that apply on Human-Computer Interaction field of study, also apply here too, because a chatbot is also a form of application.

# 4. DESIGNING AND DEVELOPING OUR BOT

## 4.1 What, Why and How?

Our chatbot was created with the idea that it could provide undergraduate students with information that they seek to find daily in their university lives. Its role is to give the students a way to find accurate and fast information, all in one place, instead of having to navigate multiple online platforms in order to retrieve that information.

## 4.2 Content and Workflows

The images below describe the 6 use cases that are covered by the chatbot and what the conversation between the user and the bot looks like.

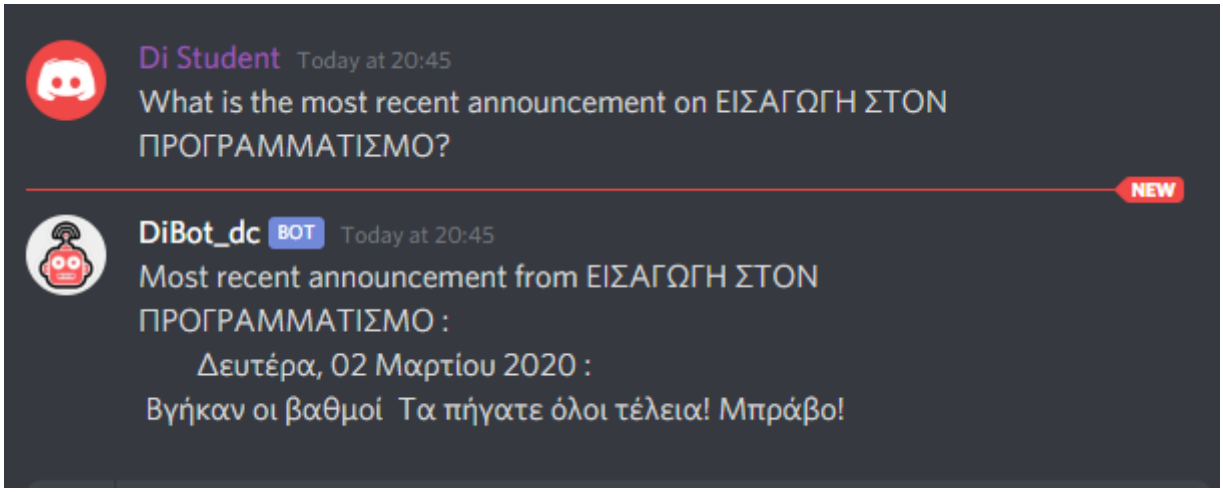


**Illustration 13: The most recent announcement of a course from www.eclass.uoa.gr**

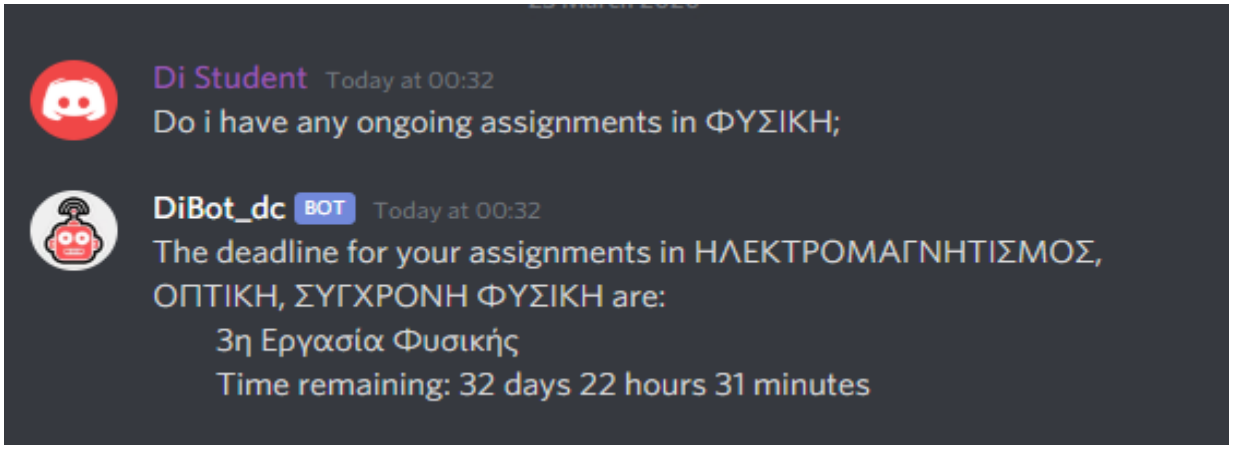


**Illustration 14: The deadlines of a course from www.eclass.uoa.gr**

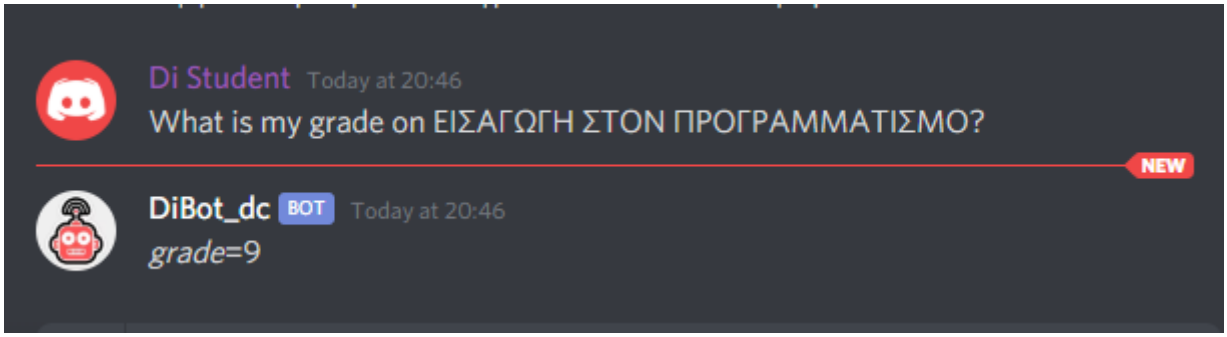


**Illustration 15: The grade of a course from www.my-studies.uoa.gr**

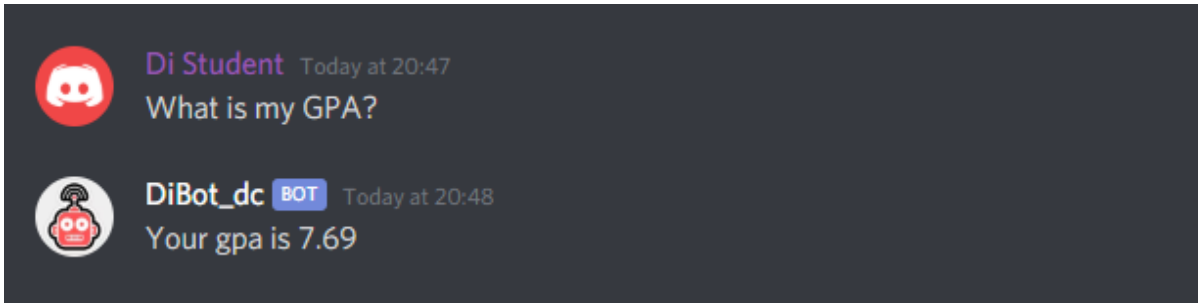


**Illustration 16: Grade point average from www.my-studies.uoa.gr**

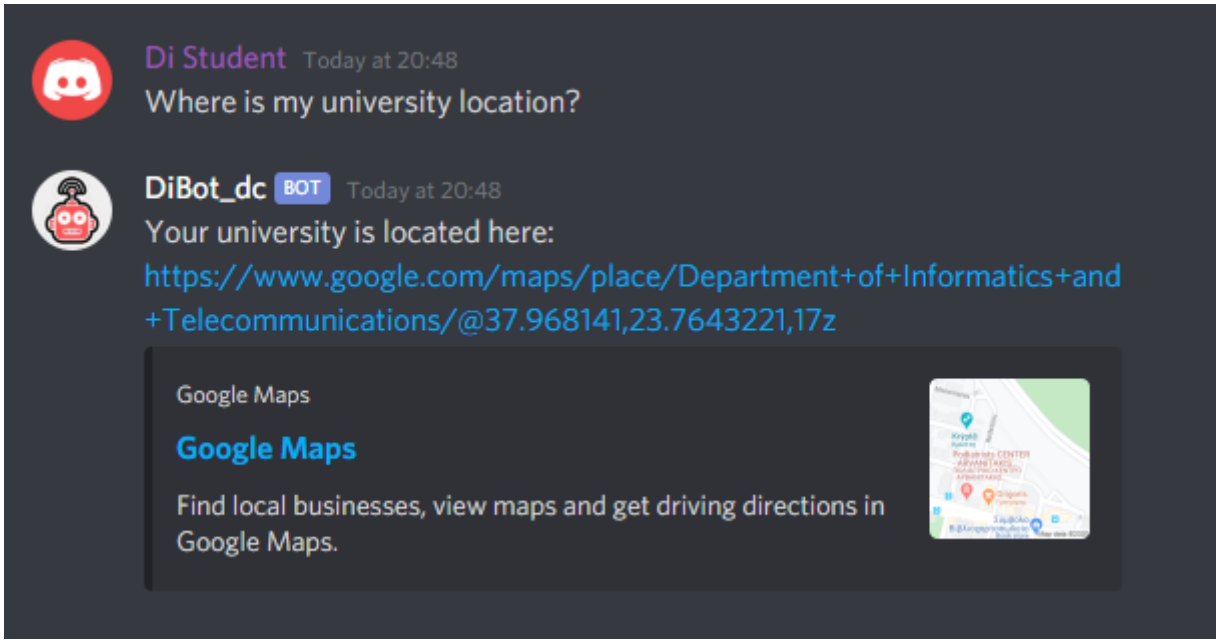


**Illustration 17: A frequently asked question: what is our university's location?**

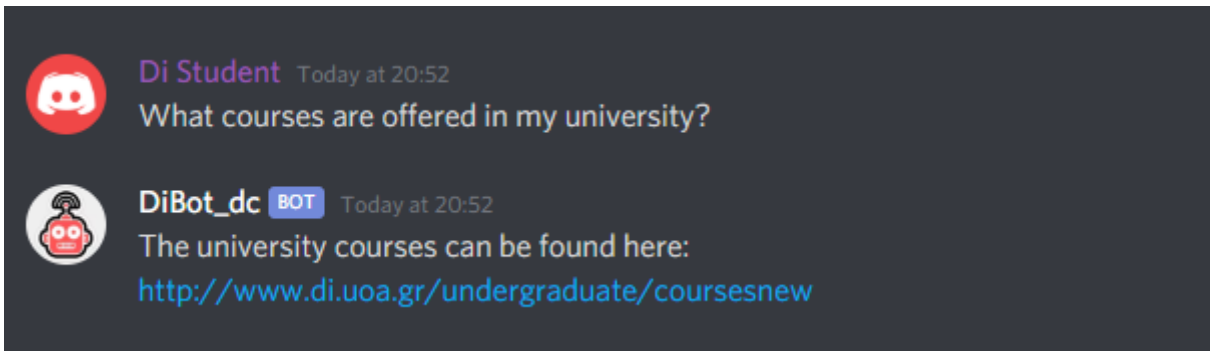


**Illustration 18: Another faq: What courses are offered in our university?**

## 4.3 Architecture

For the development of the Chatbot (DiBot), the Python programming language was used. A multi-threaded application was built which handles multiple user requests asynchronously via Discord. (Discord is a proprietary freeware VoIP application and digital distribution platform designed for text, video and audio communication between users in a chat channel.)

The users send questions and the server creates a thread to handle that request. By calling the dialogflow API, the *intent* of the user's question is identified and the data are fetched and parsed either statically or dynamically from some websites (shown below) using the Selenium chrome webdriver.

Our software design (sequence diagram & architecture diagram [12]) is as shown in the figures below.

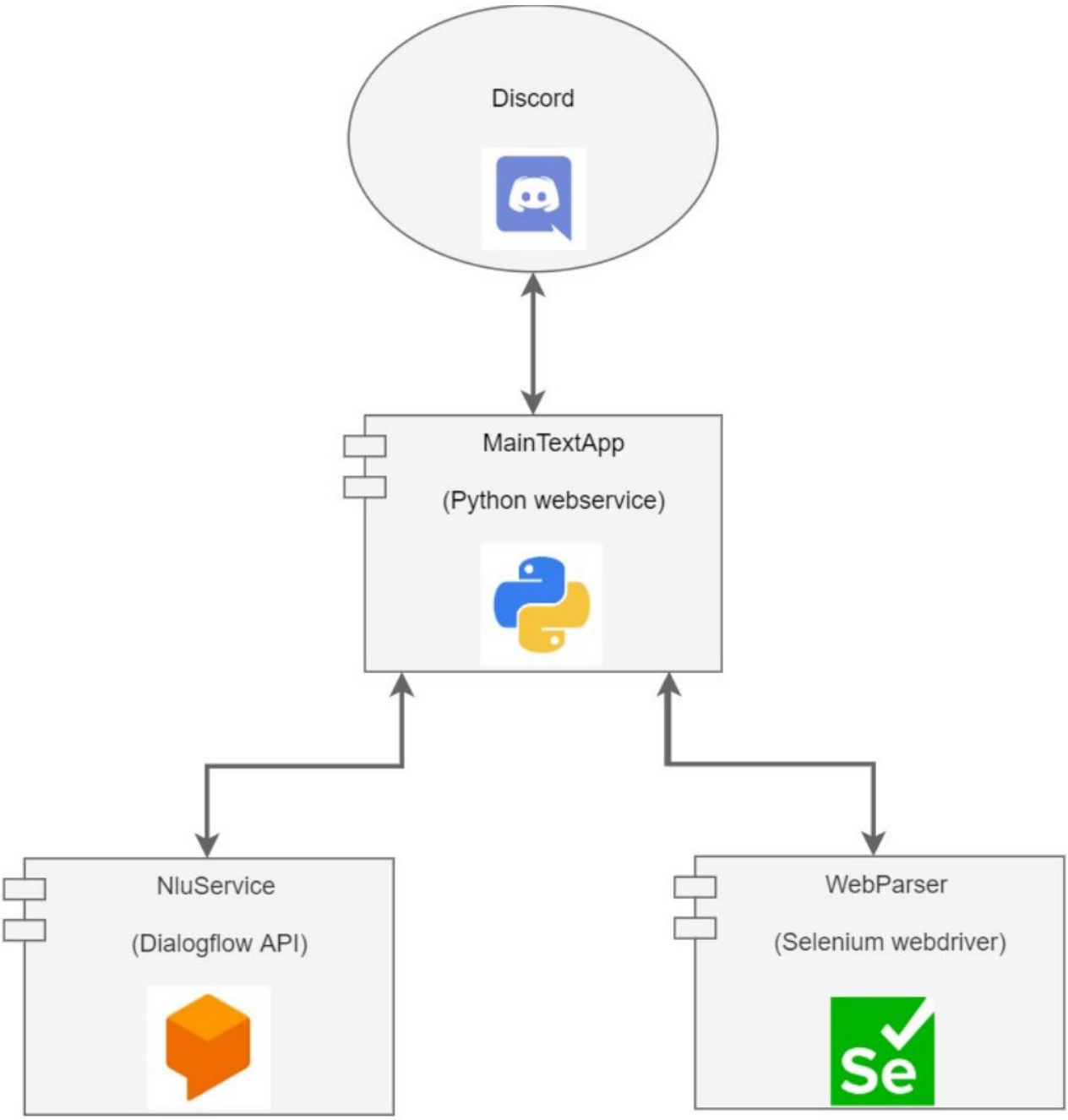


**Illustration 19: The architecture of our application**

## 4.4 Development

We followed an object oriented approach using the new Python 3(code can be found in GitHub [1]). Python's built-in asynchronous and multithreading modules and features proved to come in very handy during the whole development process.

That is because each class/component was used abstractly, only using each other's functionalities wherever and whenever it was needed. The way those classes/components interact with, and depend on each other can be shown in the following sequence diagram.

The documentation for our classes/components is as follows:

**MainTextApp**: This file is the main entry point of the application. It instantiates and runs the discord bot, which is configured according to official instructions [14].

The server runs asynchronously, meaning that the application is non-blocking, allowing multiple requests to be served at the same time. When a question arrives, a request object is created and is handled by a thread which is free at that time.

Dependencies: asyncio, ThreadPoolExecutor, discord, Request.py.

**Request**: This class contains the user request details: the input, the intent, and the answer. It also contains the function responsible for formulating the final answer, *handle_request*, which gets the intent of the user's question using the NluService. Then, switching on different cases depending on the intent, it either statically returns an answer (F.A.Q), or dynamically, by parsing online websites with WebParser.py. In the end, it formulates the final answer to be sent back to the user, as a reply.

Dependencies: WebParser.py, NluService.py.

**NluService**: This class handles the communication with DialogFlow. Every time a new user input arrives, a new session is created, if not already created for that specific user, a request is sent to the google dialogflow API and the intent and the parameters are received.

Dependencies: dialogflow_v2.

**WebParser**: Using the chrome web-driver with Selenium, an automated framework for testing websites, a headless virtual browser is created in order to log into an online platform (eclass and my-studies in our case), those pages are navigated and parsed to retrieve the actual data.

Dependencies: selenium.

**Keys_and_strings**: Contains important information that is used throughout the program, such as keys, IDs and paths to the google cloud.json authorization file and chromium webdriver executable.

Dependencies: -

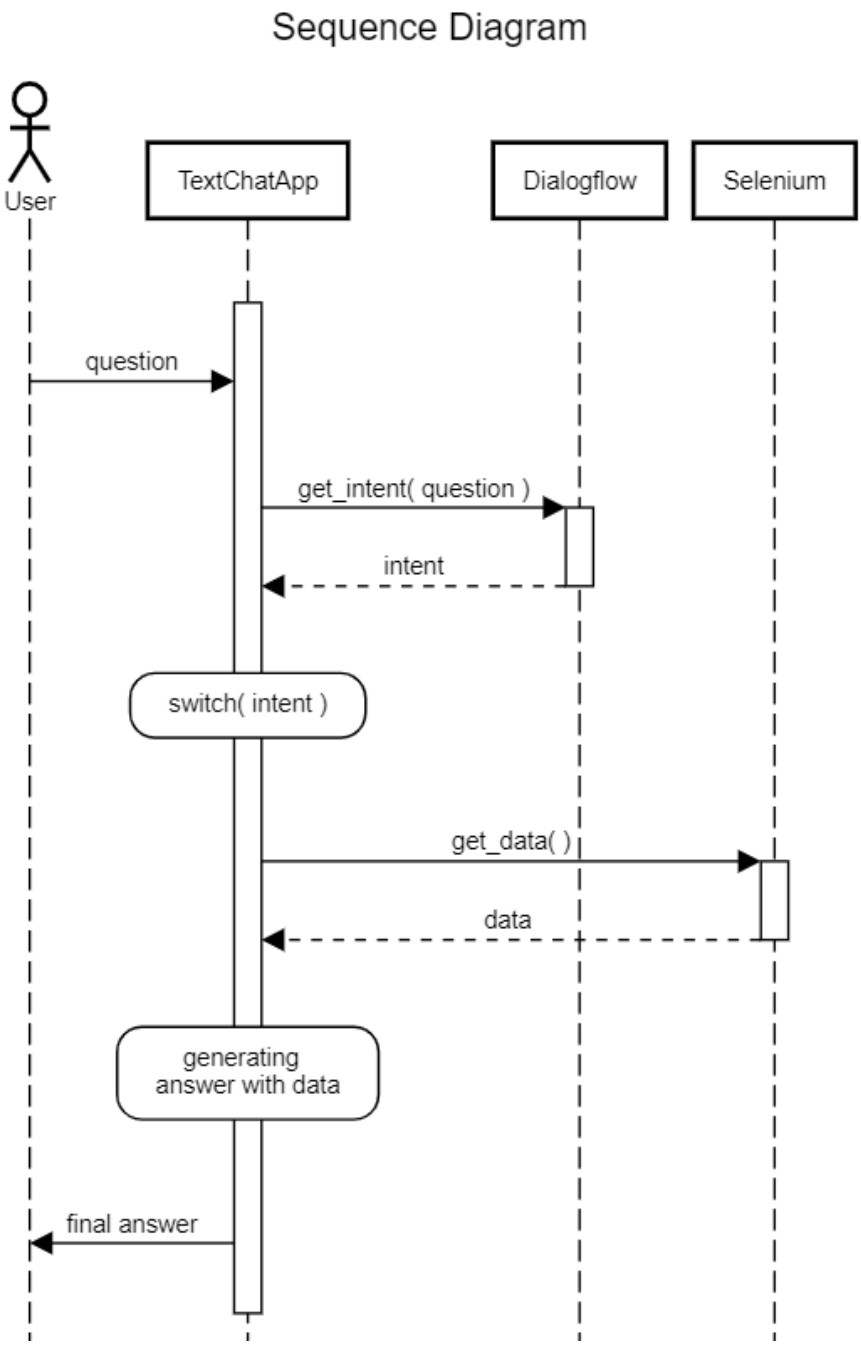


**Illustration 20: A sequence diagram of our application**

Here, we will go more into depth on what pages are navigated, parsed, and what data is searched, retrieved and how, and then are sent back to the server for the response to the user's question to be constructed. More specifically, the 6 use cases that were already mentioned are:

1) Announcements of courses from [www.eclass.uoa.gr](http://www.eclass.uoa.gr), as shown below.

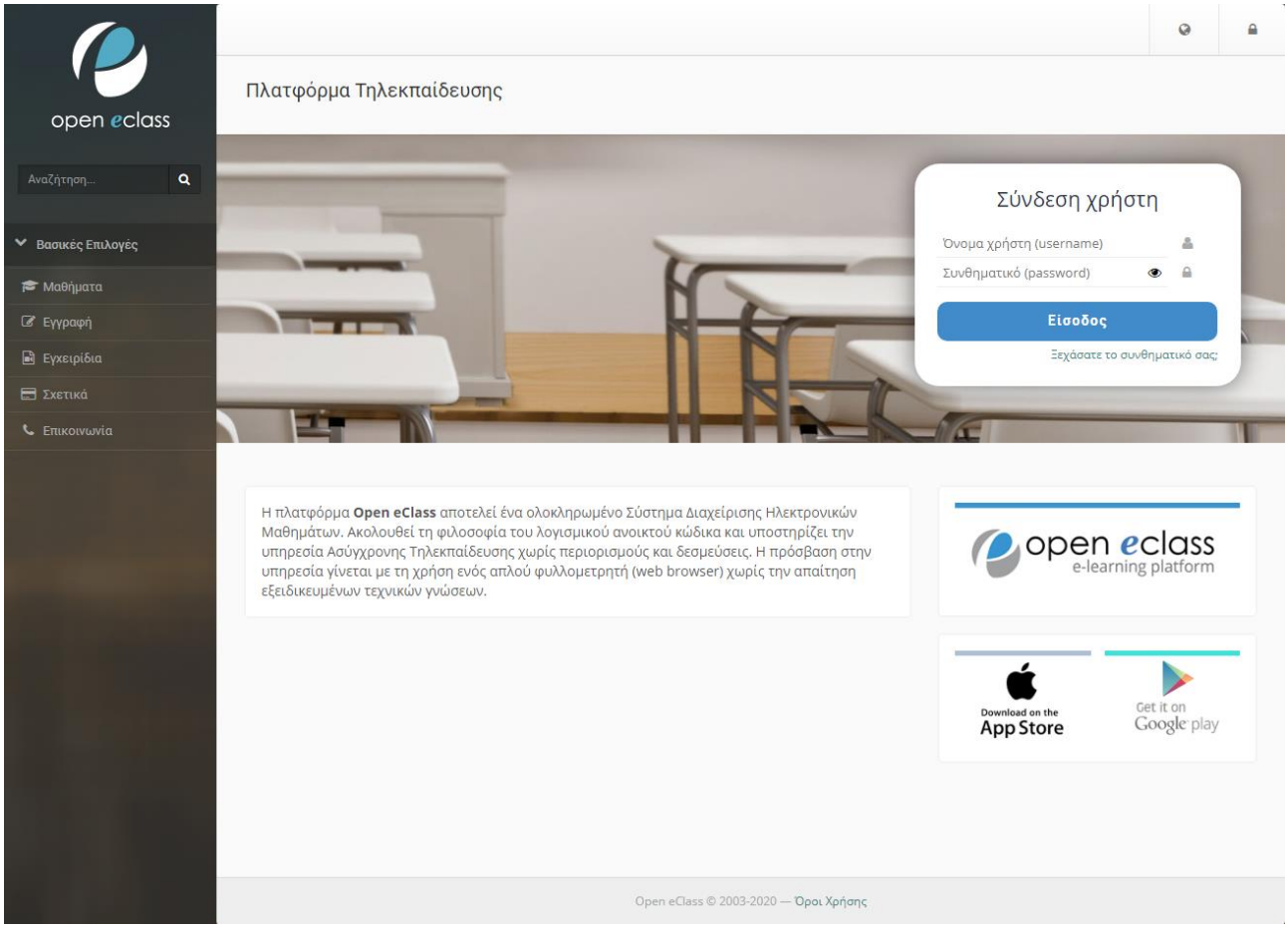


**Illustration 21: The server first receives the user's input through the Discord API, and after it identifies by using Dialogflow the user's intent, it creates a headless virtual browser that navigates and logs into www.eclass.uoa.gr**

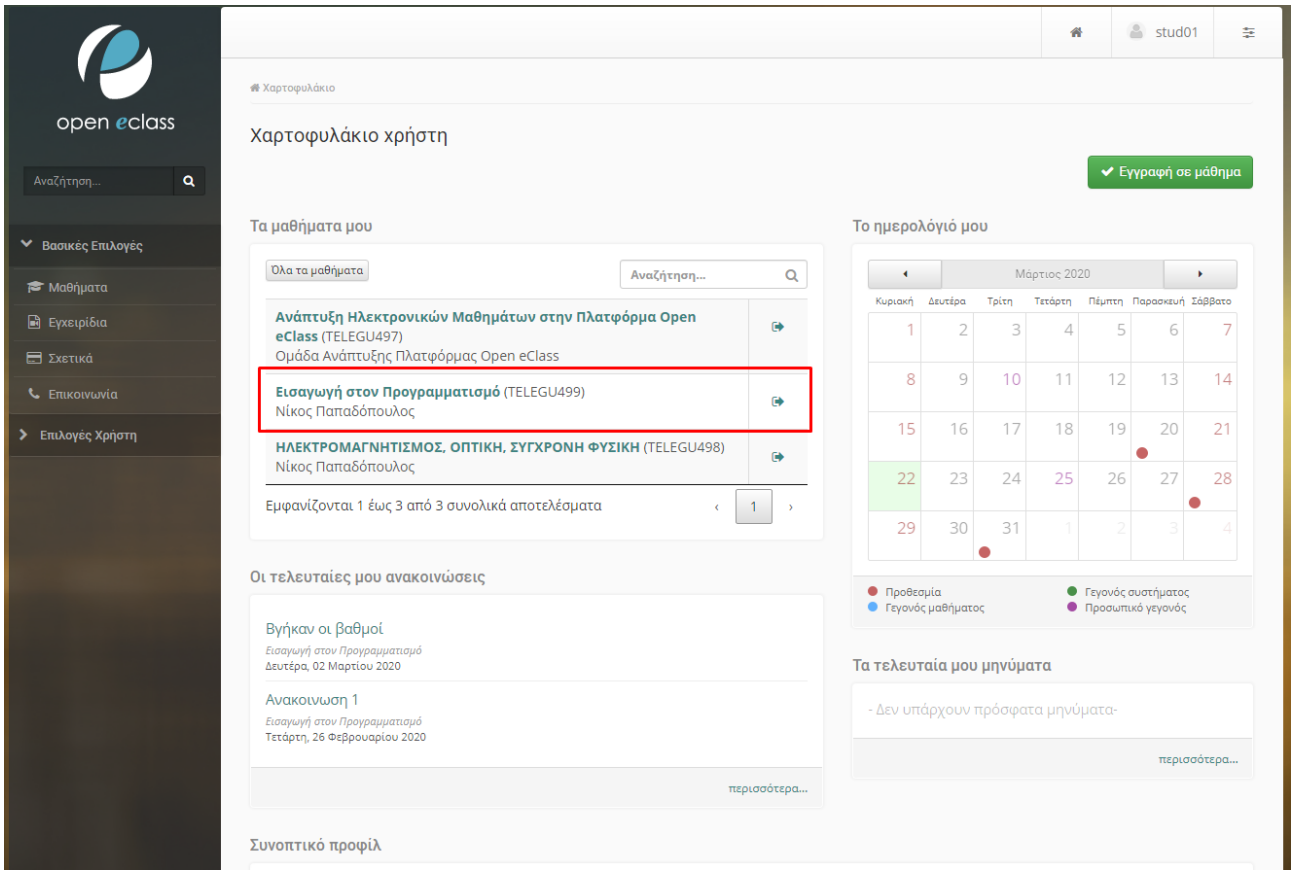


**Illustration 22: After the authentication is done, the virtual browser searches for, and then clicks on the course that the user selected to see the announcements for, in this example: "Εισαγωγή στον προγραμματισμό"**

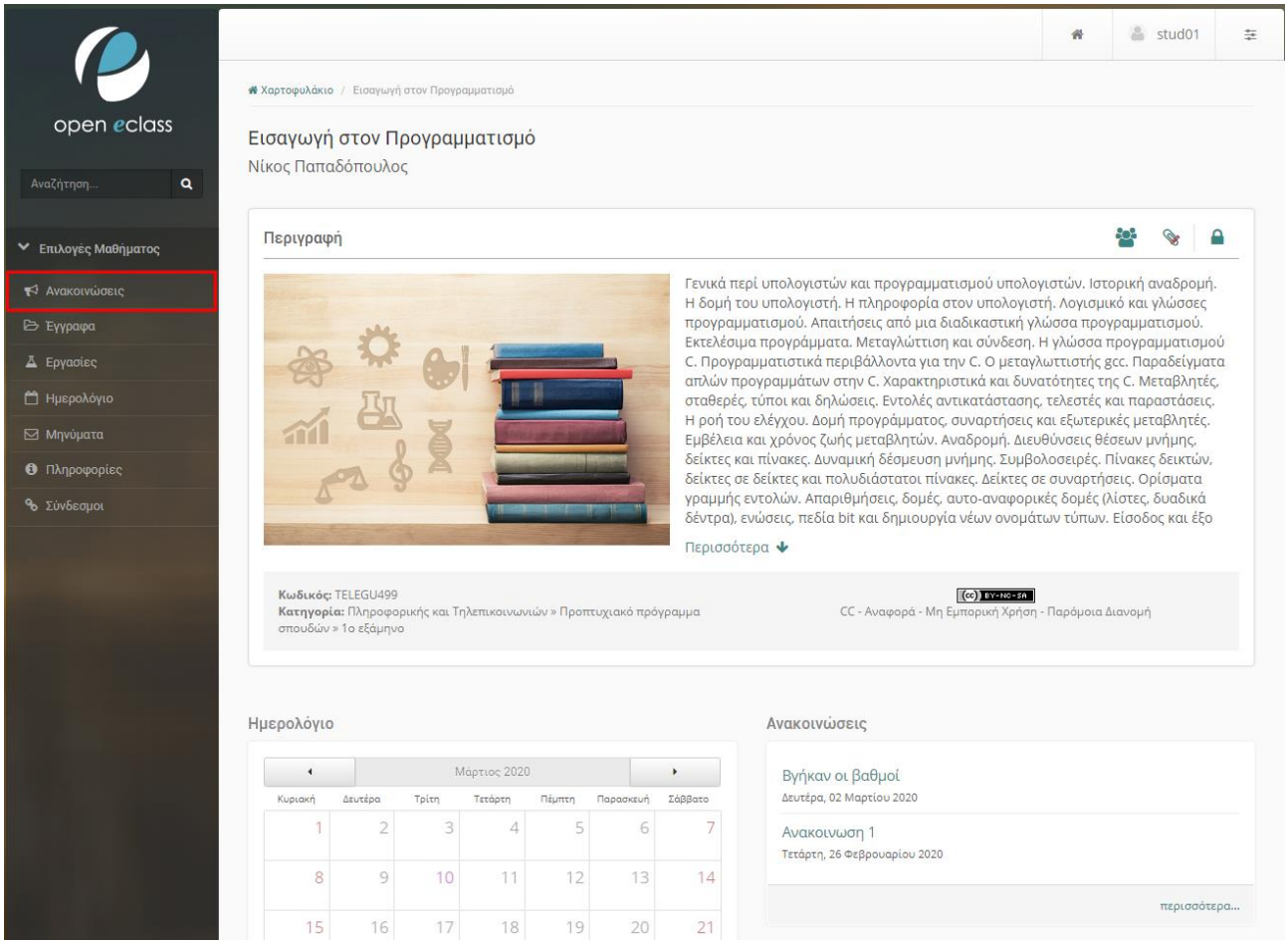


**Illustration 23: After clicking on “Εισαγωγή στον προγραμματισμό”, the virtual browser searches for and clicks on “Ανακοινώσεις” or Announcements” and it then navigates to that page**

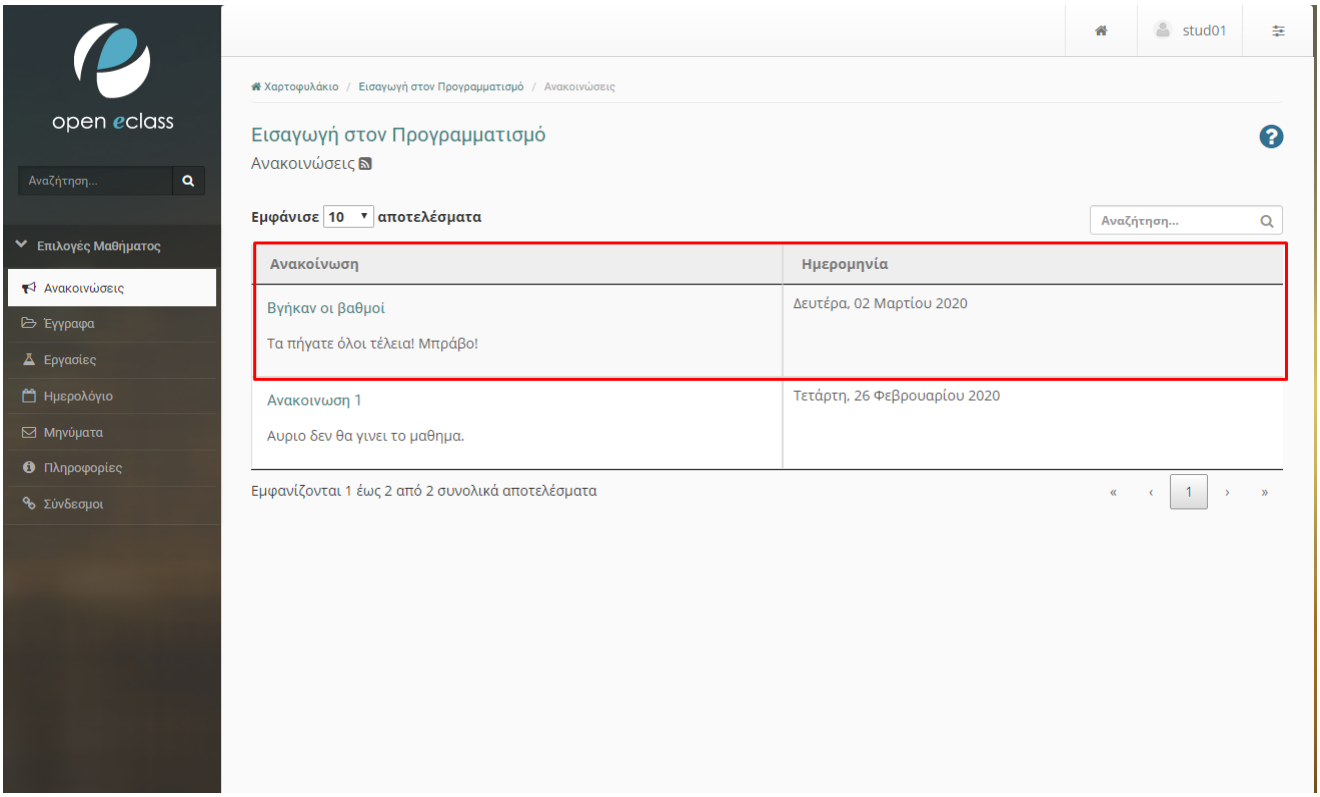


**Illustration 24: After having navigated to the announcements of course the user requested, the headless browser retrieves the most recent announcement and sends it back to the server which handles the response**

2) Deadlines of assignments of courses from [www.eclass.uoa.gr](http://www.eclass.uoa.gr)

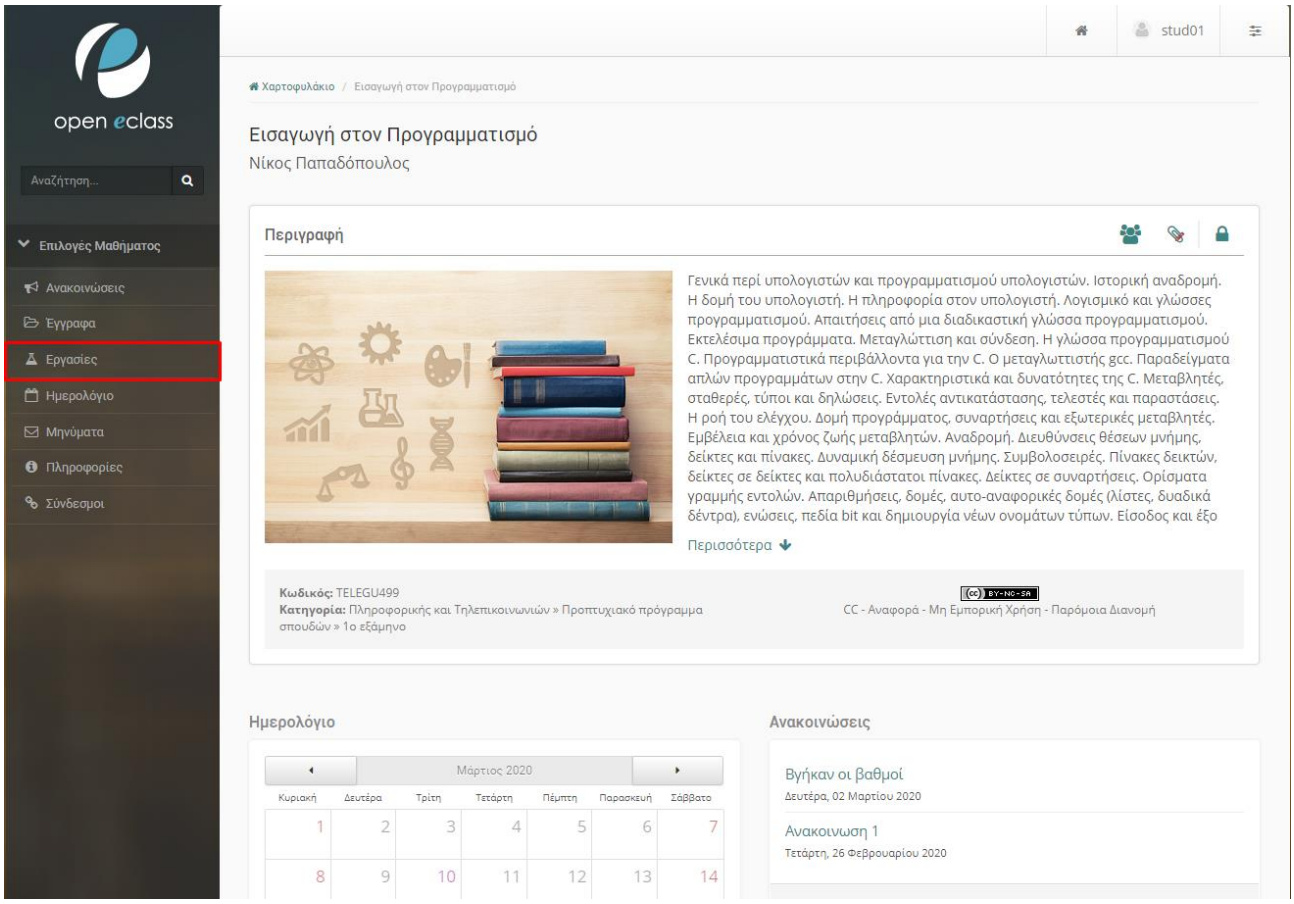


**Illustration 25: Similarly to the first use case, the first 2 steps are the same, but now instead of navigates to the "Announcements", the virtual browser searches for and navigates to "Εργασίες" or "Assignments"**

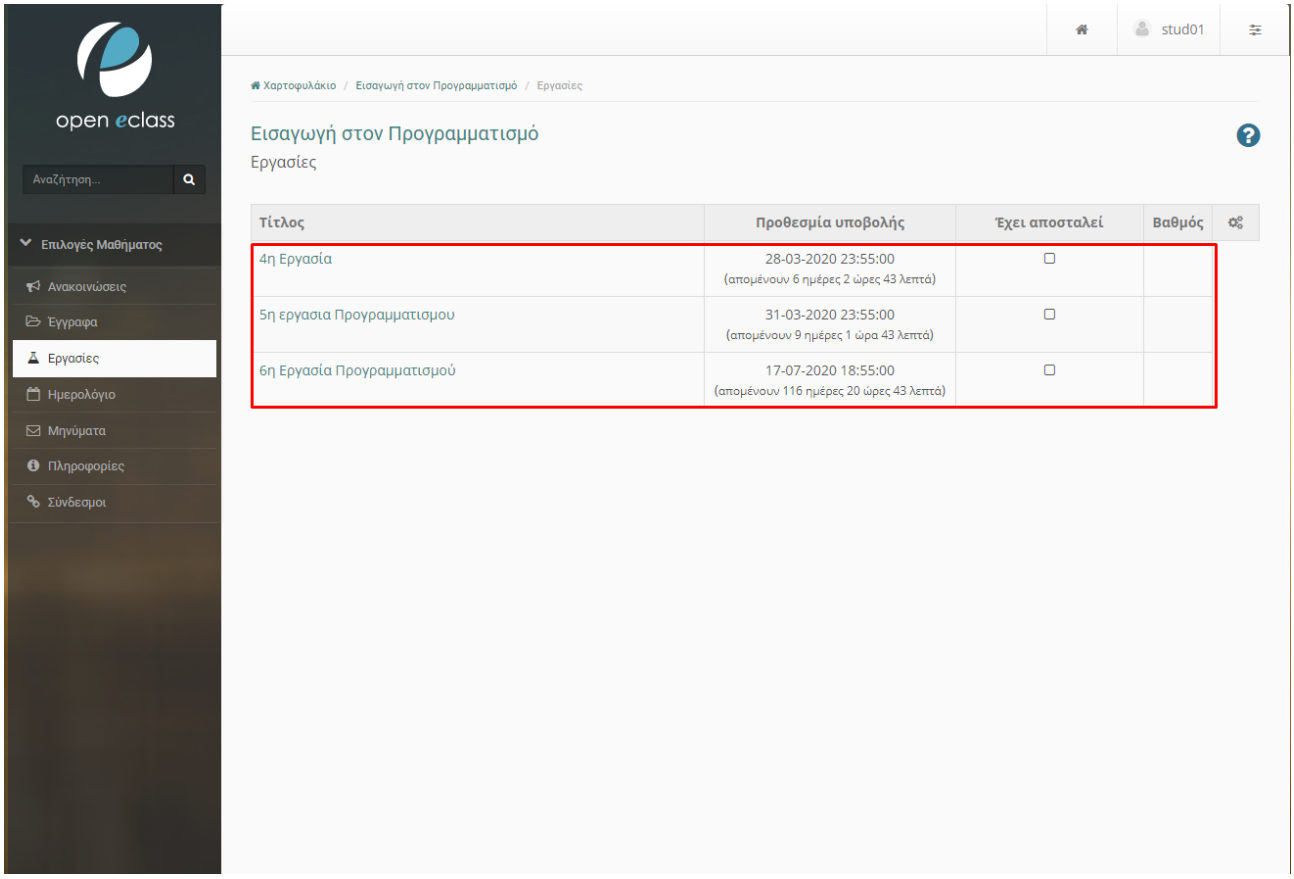


**Illustration 26: After navigating to the right page, it retrieves the future deadlines for the course and returns them to the server, which then responds to the user**

3) Grade for one course from www.my-studies.uoa.gr.

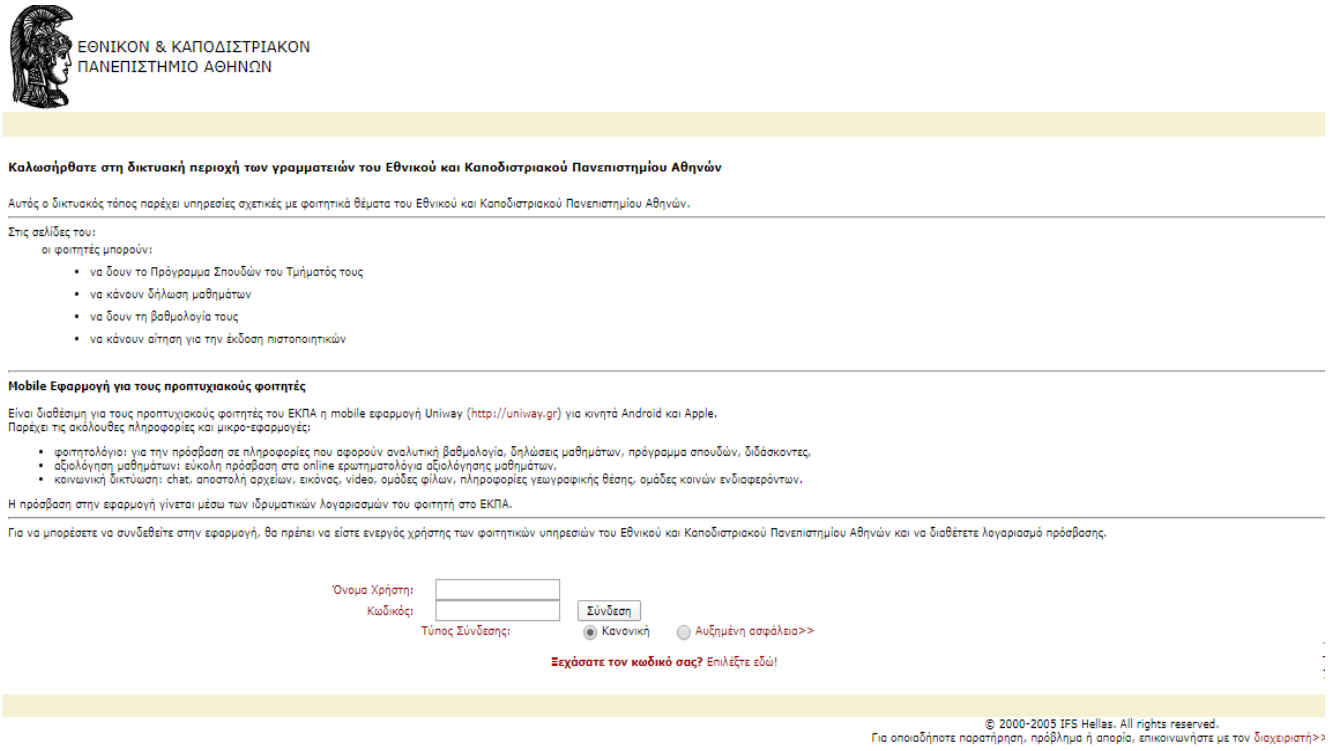


**Illustration 27: Similarly to first use case, the server receives the user's input through the Discord API, and after it identifies by using Dialogflow the user's intent, it creates a headless virtual browser that navigates and logs into www.my-studies.uoa.gr**

**Illustration 28: It then clicks on “Grades” or “Βαθμολογία”, navigating to that page**

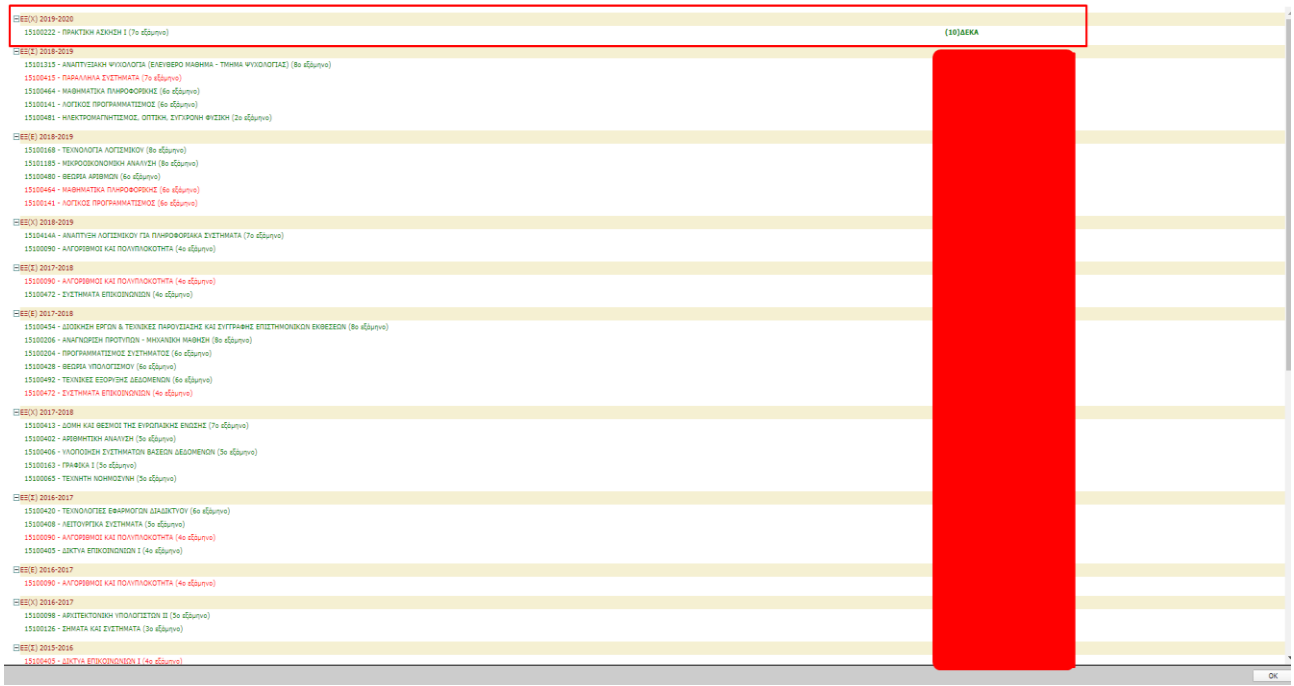

**Illustration 29: After navigating to that page, it finds the course that the user requested the grade for, retrieves it and returns it to the server for it to be sent back to the user.**

GPA or Average of Grade from www.my-studies.uoa.gr

To find the GPA of a user, the bot follows the same logic as the previous use case, but instead of retrieving the grades for one course, it retrieves the grades for all courses that are more than or equal to grade 5 and sends them back to the server. After the grades are sent back to the server, the server calculates their mean value and responds to the user's initial request.

4) Frequently asked questions with static answers.

## 4.5 Evaluation

We asked a group of undergraduate students from our university (n=10) via google forms [13] to test our chatbot's capabilities. Afterwards we asked them the following questions to determine the effectiveness and usefulness of our chatbot.

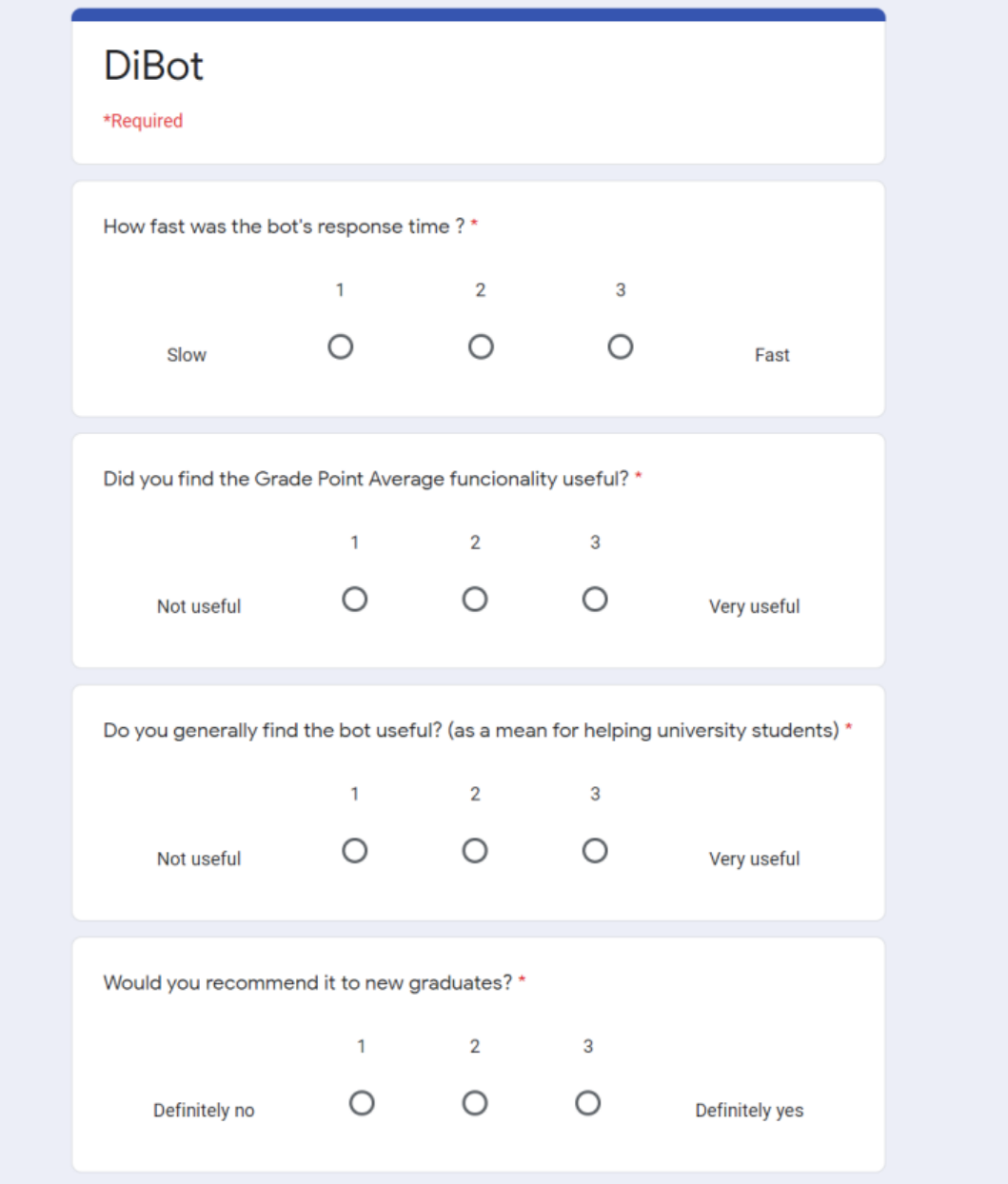


**Illustration 30: Part one of the evaluation questionnaire**

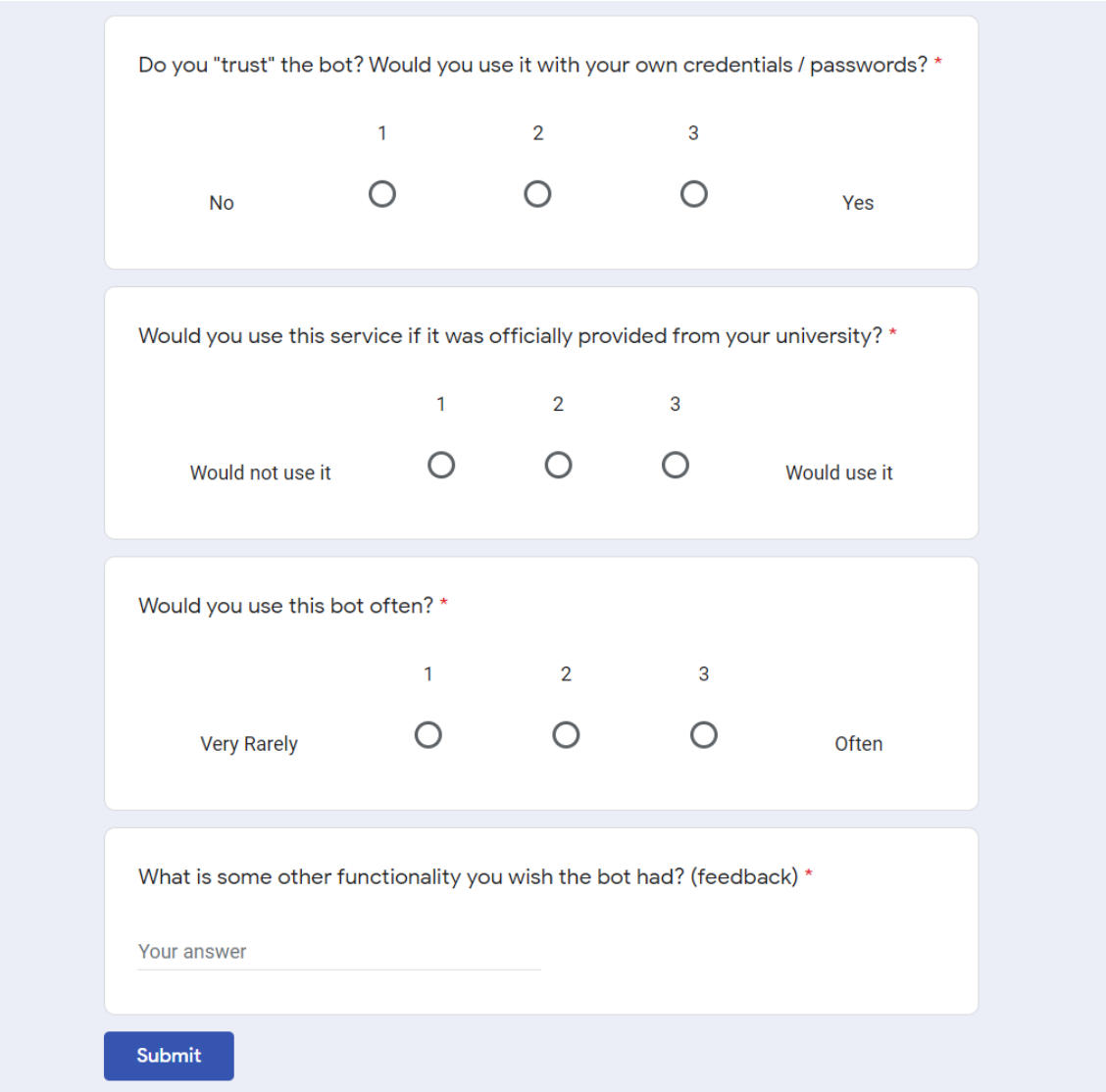


**Illustration 31: Part two of the evaluation questionnaire**

**Table 2: Table of evaluation questionnaire responses**

| Question | (Answer) Count | (Answer) Count | (Answer) Count |
|---|---|---|---|
| How fast was the bot's response time? | (Slow) 3 | (Mediocre) **4** | (Fast) 3 |
| Did you find the Grade Point Average functionality useful? | (Not useful) 0 | (Quite Useful) 3 | (Very Useful) **7** |
| Do you generally find the bot useful? (as a mean for helping university students) | (Not useful) 0 | (Quite Useful) 3 | (Very Useful) **7** |
| Would you recommend it to new graduates? | (Definitely no) 0 | (Not Sure) 3 | (Definitely Yes) **7** |
| Do you "trust" the bot? Would you use it with your own credentials / passwords? | (No) 0 | (Maybe) **7** | (Yes) 3 |
| Would you use this service if it was officially provided from your university? | (Would not use it) 0 | (Not Sure) 2 | (Would use it) **8** |
| Would you use this bot often? | (Very rarely) 1 | (Sometimes) **6** | (Often) 3 |

As for the last, free text question *("What is some other functionality you wish the bot had?")*, we thought about respondent's answers and added their ideas as future work in the conclusions chapter.

**Summary of respondent's responses**:

A far as the bot's response time is concerned, in average, the respondents said that they found the chatbot's response time slow, but that is to be expected due to the fact it creates a new virtual browser every time it needs to parse dynamic website data. But, even though they found it slow, they stated that they think its functionalities are useful, because they gather a lot of valuable information in one place.

As for the privacy aspect of it, the respondents stated that there is still work needs to be done, because sharing their personal info on a messaging app for a chatbot to parse feels quite dangerous.

Last but not least, looking at the respondent's answers on the last free text question, if an application/virtual assistant like the one we tried to build was officially provided by the university, the respondents stated that they would trust it and use it quite often.

All in all, the respondents think that a full-fledged application that can gather all the information in one place would be extremely useful, so we too hope that in the future our university makes a move to create an application like this in collaboration even with the students themselves.

# 5. CONCLUSIONS - FUTURE WORK

The focus of this Thesis was chatbots and how Natural Language Processing is an integral chatbot development. We explored popular NLU systems and platforms that are offered by big companies, such as Microsoft, Facebook and Google. It is clear that, with the help of bot frameworks and NLU Systems, an interested person can make an efficient and helpful chatbot agent. The task is not necessarily heavy on the coding aspect, so just by using and trying out the tools we compared and analyzed in this thesis, they will be able to design and develop such agents, and integrate them into commonly used messaging/chatting platforms.

As for our work, we built a chatbot whose main function was to provide the average university student with meaningful information. The data was found from sources that are officially used and supported by our university: eclass and my-studies. But, as there was no available API so that we could get that data in a programmatic way, we had to use an external service that creates a virtual browser to parse and fetch the information we needed from the aforementioned websites.

Even though the result of our effort was good, in the end, our work was only the first step of us learning how chatbots function, and how we could create a chatbot with some basic functionalities so that a person that reads this thesis in the future can start building their own chatbot, this time being a step further ahead than where we were when we first started. Some food for thought, or some ideas rather that we thought of, but we did not have the time to implement, are the following:

- Use a database to store user IDs, passwords and parsed data from target websites.
- Try to reduce the response time. Do not create a new virtual browser every time a new request arrives, but try to use some sort of cookie system or database to remember if a user has already sent a request.
- Try creating a recommendation system using Machine learning on student data and interests (i.e. grades), to suggest to a student what courses they could take.
- Create dialogs, not just single question-answer conversations.
- Enrich Dialogflow with more training phrases/questions.
- Try out different NLU systems and bot frameworks.

This thesis is meant to be only the first step and we hope that someone else might find it helpful in the future and that it helps them create their own chatbot.

# ABBREVIATIONS – ACRONYMS

| | |
|---|---|
| NLP | Natural Language Processing |
| NLU | Natural Language Understanding |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| SDK | Software Development Kit |
| QA | Question Answering |
| GPA | Grade Point Average |
| FAQ | Frequently Asked Questions |
| DIT | Department of Informatics and Telecommunications |
| ΕΚΠΑ | Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών |

# REFERENCES


[1] A chatbot that answers questions related to my university (grades, deadlines, FAQ). https://github.com/Ted96/Di_Bot [Accessed 23/3/2020]
[2] Chatbot https://www.chatbots.org/chatbot/ [Accessed 3/3/2020]
[3] Virtual Assistant https://www.chatbots.org/virtual_assistant/ [Accessed 3/3/2020]
[4] Word2Vec https://en.wikipedia.org/wiki/Word2vec [Accessed 20/3/2020]
[5] Neural Networks https://en.wikipedia.org/wiki/Neural_network [Accessed 20/3/2020]
[6] Siri vs Cortana vs Alexa vs Google Assistant vs Bixby https://reachbyte.com/siri-cortana-alexa-google-assistant-bixby/ [Accessed 17/3/2020]
[7] 11 Step Process for a Great Chatbot Design https://www.userlike.com/en/blog/chatbot-design [Accessed 29/2/2020]
[8] Top 6 AI Chatbot Development Frameworks https://www.spaceo.ca/top-ai-chatbot-frameworks/ [Accessed 18/3/2020]
[9] NLP vs. NLU: What's the Difference? https://medium.com/@lola.com/nlp-vs-nlu-whats-the-difference-d91c06780992 [Accessed 18/3/2020]
[10] NLP; NLU and NLG Conversational Process Automation Chatbots explained https://medium.com/botique-ai/nlp-nlu-and-nlg-chatbots-explained-64820bc32ad [Accessed 18/3/2020]
[11] NLP vs NLU vs NLG https://towardsdatascience.com/nlp-vs-nlu-vs-nlg-know-what-you-are-trying-to-achieve-nlp-engine-part-1-1487a2c8b696 [Accessed 18/3/2020]
[12] Online tools that were used for the drawings: https://app.diagrams.net/, https://sequencediagram.org/ [Accessed 22/3/2020]
[13] Google Forms https://docs.google.com/forms [Accessed 22/3/2020]
[14] Discord documentation for Python https://discordpy.readthedocs.io/en/latest/discord.html [Accessed 15/2/2020]
[15] 6 key metrics to measure the performance of your chatbot https://chatbotslife.com/6-key-metrics-to-measure-the-performance-of-your-chatbot-5fd0adfd0b5b [Accessed 20/3/2020]
[16] Designing a chatbot https://www.lollypop.design/blog/2018/january/designing-a-chatbot/ [Accessed 20/3/2020]