



Πανεπιστήμιο Πειραιώς
University of Piraeus

Department of Digital Systems
M.Sc. in Big Data & Analytics

Thesis

“Coaching Medical Chatbot in Facebook”

ME1919

Menelaos Ampartzakis

Thesis Supervisor

Maglogiannis Ilias

Piraeus 2021

ABSTRACT

A **chatbot** is a piece of software that conducts a conversation with users via auditory or textual methods. A **medical** chatbot facilitates the job of a **healthcare provider** and helps improve their performance by interacting with users in a human-like way. There are countless cases where intelligent medical chatbots could help doctors or the patients. They can step in and minimize the amount of time they spend on tasks like providing information to the doctor or guidance to the patient. It's important to note that even though chatbots can offer valuable facts and symptoms, they aren't qualified to give an official diagnosis. The main premise behind these talking or texting algorithms, is to become the first point of contact before any human involvement is needed.

The objective of the current thesis is the implementation, integration and training of a medical chatbot about **Chronic Inflammatory Lung Disease (COPD)** in **Facebook Messenger**. Chatbot will offer the patient an immediate way to be informed for COPD and ways to avoid it. Mainly, a user will be able to do a questionnaire with points, that will inform user about the effect of COPD in its life. If the user completes five questionnaires, chatbot will both offer the user a prediction and will inform him about the biggest improvement of his symptoms or the symptoms that seems to get worse.

Chatbot will be trained using the **WIT.AI** framework in **Greek**. Wit is a natural language interface for applications capable of turning sentences into structured data.

For the implementation of the chatbot's questionnaire, a **MongoDB** database in **cloud** will be used. MongoDB offers limited free storage in the cloud and provide many ways of connectivity like APIs with multiple programming languages or an Analytics application. The connectivity that will support us the most is **PyMongo** Library, since the whole chatbot will be developed in **Python**. The framework that will be used for the communication between Python, WIT, MongoDB and Facebook will be **Flask**. The final application will be built, run and operate in **Heroku**, a cloud Application Platform.

Finally, a **Tableau** Dashboard will be set up, so that the chatbot's Administrator will be able to track and monitor each user, provide the Dashboard to the user's doctor and give the patient custom advices and recommendations.

Keywords: Chatbot, Medical, COPD, Facebook Messenger, WIT.AI, MongoDB, Python, Flask, Tableau

Acknowledgements

This thesis would not have been possible without the support of many people.

I would like to thank my professor Mr. Ilias Maglogiannis of the Department of Digital Systems for his constant guidance, for giving me the opportunity to do this research and providing invaluable support and motivation.

Finally, I would like to express my endless love and gratitude to my family and my wife for their patience, understanding, love and continuing support during this thesis.

Contents

ABSTRACT	
Acknowledgements	
Image Contents	
Table Contents	
1. Introduction	1
1.1. Medical Chatbot & Motivation	1
2. Background	3
2.1. Retrospection & Theory of Chatbots	3
2.1.1. History & Related Work	3
2.1.2. What is Chatbot?	4
2.1.3. NLP & Mechanisms	6
2.2. Applications & Frameworks	7
2.2.1. MongoDB	7
2.2.2. Facebook	11
2.2.3. Wit.ai	20
2.2.4. Tableau	24
2.2.5. Heroku	29
2.2.6. Rest Python Libraries	32
3. Implementation	34
3.1. Database & ODBC	34
3.2. Facebook Connectivity	39
3.3. Wit Setting & Connectivity	48
3.4. Questionnaire Flow & Methods	54
3.5. Tableau Dashboards & Connection	58
3.6. Heroku Setting & Deployment	62
4. Results	67
4.1. Expected Chatbot Conversation	67
4.2. Monitor the user	93
5. Outcome	94
5.1. Conclusions & Future Works	94
6. References	95

Image Contents

Figure 2.1 : How an AI chatbot works	5
Figure 2.2 : Structure of MongoDB Database	8
Figure 2.3 : BI Connector Components	9
Figure 2.4 : ODBC and Connections	10
Figure 2.5 : Logo of Facebook	12
Figure 2.6 : JSON format of payload	16
Figure 2.7 : JSON format of quick_reply	16
Figure 2.8 : Deconstruction of a sentence	21
Figure 2.9 : Precision and Recall of an Intent	22
Figure 2.10 : Usage Statistics	24
Figure 2.11 : New Worksheet	26
Figure 2.12 : New Dashboard.....	27
Figure 2.13 : Working space of a Worksheet.....	27
Figure 3.1 : Database to store chatbot data	34
Figure 3.2 : Methods to connect to database.....	35
Figure 3.3 : Connection via Python	36
Figure 3.4 : Python Library for MongoDB	36
Figure 3.5 : MongoDB ODBC Driver	37
Figure 3.6 : Configuration of ODBC Driver	38
Figure 3.7 : Facebook App creation	39
Figure 3.8 : Permissions of the Facebook App.....	40
Figure 3.9 : Creation of Facebook Page	40
Figure 3.10 : Created Facebook Page	41
Figure 3.11 : Settings of the Facebook App (1).....	42
Figure 3.12 : Settings of the Facebook App (2).....	43
Figure 3.13 : Access Token of Facebook Page	44
Figure 3.14 : Python Library for Facebook Messenger	44
Figure 3.15 : Webhook Verification through Python.....	44
Figure 3.16 : Token Verification on Facebook App	45
Figure 3.17 : Subscriptions of Facebook App.....	45
Figure 3.18 : Quick Reply variable.....	46
Figure 3.19 : Chatbot's welcome screen.....	47
Figure 3.20 : Let's Begin Button & Response	47
Figure 3.21 : Wit.ai App Creation.....	48
Figure 3.22 : New App's info	48
Figure 3.23 : Training the Wit App	49
Figure 3.24 : Token of the Wit App	52
Figure 3.25 : Python Library for Wit.ai.....	52

Figure 3.26 : Response JSON of a Request	53
Figure 3.27 : Inserted Document in MongoDB	55
Figure 3.28 : Final Document that is inserted.....	56
Figure 3.29 : Connection to MongoDB for Tableau	58
Figure 3.30 : Query to get all data from MongoDB to Tableau	59
Figure 3.31 : Total Number of COPD Tests Card.....	59
Figure 3.32 : Number of Unique Users Card	59
Figure 3.33 : Results Pie	60
Figure 3.34 : AVG Points per Question	60
Figure 3.35 : Number of COPD Tests per Day	60
Figure 3.36 : Tableau Dashboards.....	61
Figure 3.37 : Creation of Project's file	62
Figure 3.38 : Installation of necessary Python Libraries	62
Figure 3.39 : Needed Python Libraries.....	63
Figure 3.40 : Procfile Creation	63
Figure 3.41 : Heroku App Creation	64
Figure 3.42 : Deployed Heroku App	64
Figure 3.43 : Message of Webhook	65
Figure 3.44 : Logs of the App via Heroku	65
Figure 3.45 : Heroku page on Facebook App.....	66
Figure 4.1 : Navigation on Facebook's Page	67
Figure 4.2 : Sample Conversation (1)	68
Figure 4.3 : Sample Conversation (2)	69
Figure 4.4 : Sample Conversation (3)	70
Figure 4.5 : Sample Conversation (4)	71
Figure 4.6 : Sample Conversation (5)	72
Figure 4.7 : Sample Conversation (6)	73
Figure 4.8 : Sample Conversation (7)	74
Figure 4.9 : Sample Conversation (8)	75
Figure 4.10 : Sample Conversation (9).....	76
Figure 4.11 : Sample Conversation (10).....	77
Figure 4.12 : Sample Conversation (11).....	78
Figure 4.13 : Sample Conversation (12).....	79
Figure 4.14 : Sample Conversation (13).....	80
Figure 4.15 : Sample Conversation (14).....	81
Figure 4.16 : Sample Conversation (15).....	82
Figure 4.17 : Sample Conversation (16).....	83
Figure 4.18 : Sample Conversation (17).....	84
Figure 4.19 : Sample Conversation (18).....	85
Figure 4.20 : Sample Conversation (19).....	86

Figure 4.21 : Sample Conversation (20).....	87
Figure 4.22 : Sample Conversation (21).....	88
Figure 4.23 : Sample Conversation (22).....	89
Figure 4.24 : Sample Conversation (23).....	90
Figure 4.25 : Sample Conversation (24).....	91
Figure 4.26 : Sample Conversation (25).....	92
Figure 4.27 : Monitoring a User (1).....	93
Figure 4.28 : Monitoring a User (2).....	93

Table Contents

Table 2.1: Description of Facebook Tokens	15
Table 2.2 : Parameters of GET requests.....	17
Table 2.3 : Webhook events	18
Table 2.4 : Tiers for a Heroku app.....	32
Table 3.1 : Buttons, Payloads and Description	46
Table 3.2 : Intents & Entities.....	50
Table 3.3 : Intents & Answers	51
Table 3.4 : Questions of Questionnaire	55

1. Introduction

1.1. Medical Chatbot & Motivation

In recent years, every time people need information about a problem their first thought is to search about it in the internet. That action is not limited to general topics, but it also extends to medical issues. However, for such issues that tactic is not recommended since it can lead to undesirable effects. There are essentially two bad ways this can go: Either you overestimate your symptoms and end up taking the wrong medication or engaging in the wrong self-treatment, or you underestimate your symptoms and let a condition worsen. There's also the risk of developing "health anxiety," real condition that involves excessive worrying that you are sick. In fact, there is an actual term for it-- "cyberchondria". It is the tendency of self-diagnosing yourself with medical conditions by searching for symptoms online, resulting in serious anxiety. Case in point, just look for any symptom online and it is bound to be linked with some form of tumor or cancer. Eventually, it can make you feel sicker than you actually are. Furthermore, anyone can publish content online. Nobody can vouch for the quality and credibility of the content available online. While it can provide you information from credible sources, it can also get you links of some sketchy websites that have bogus information.

Based on that needs, people start developing several technologies & applications to help people get more accurate results regarding their disease. One of them is the creation of *yes/no* questionnaire systems based on symptoms. However, some diseases have almost the same symptoms as another, so we can't trust a system build like this. Moreover, there are confirmed websites and forums that provide information and answers about a disease to the users [3]. But those answers come from certified doctors that they are not always available, and thus it can take hours or days to reply on the queries of the patients.

The need for a reliable and accurate diagnosis, available 24/7/365, wakes the rise and need of a new healthcare technology called **Medical Chatbot**. That will help people get information about their symptoms or disease and give them a more accurate diagnosis with the tap of a button, as soon as they need it [7]. Artificial intelligence (AI) is the pillar for this computer software consisting of a complex mathematical algorithm that processes input information to produce any specific pre-defined outputs, which lead to relevant outcomes. AI systems,

can be designed to enhance decision-making and analytical processes while imitating human cognitive functions.

However, several official Medical Chatbots operate under platforms that are not popular, hard to find or platforms that the user has to pay a subscription to use those available technologies. Fortunately, one of the world's most popular and free to use platform allows the hosting and installation of chatbots. That platform is Facebook and in particular Facebook Messenger. Setting a Medical Chatbot in Facebook Messenger will allow the target audience to have it available all the time and use it for free with just a few clicks.

2. Background

2.1. Retrospection & Theory of Chatbots

2.1.1. History & Related Work

Early approach of chatbots began in 1950 with Alan Turing considering the idea of machines thinking intelligently. He proposed an imitation game called Turing test in which through written responses, a player must determine if the response comes from a person or a machine. [15] In 1966, Joseph Weizenbaum developed the program ELIZA, which aimed at tricking its users by making them believe that they were having a conversation with a real human being. ELIZA was designed to imitate a therapist who would ask open-ended questions and even respond with follow-ups. [20] In 1972 Kenneth Colby mimicked ELIZA with PARRY, by designing a program to simulate a person with a "paranoid mind". Colby used a variation of Turing test on a group of psychologists, fooling half of them that PARRY was a real patient. "PARRY and ELIZA 'met' several times. One of the most famous encounters of these chatterbots took place at the [ICCC 1972](#) [\[7\]](#), where PARRY and ELIZA were hooked up over [ARPANET](#) [\[7\]](#) and 'talked' to each other". In 1988, Rollo Carpenter creates Jabberwacky and in 1997 goes online. Its stated aim, is to "simulate natural human chat in an interesting, entertaining and humorous manner". It is an early attempt at creating an artificial intelligence through human interaction. It adds user responses from previous conversations to database and uses them to increase vocabulary. The evolved version of Jabberwacky is set as chatterbot web application called Cleverbot. Since coming online in 1997, Cleverbot has engaged in about 65 million conversations with Internet users around the world, who chat with it for fun via the [Cleverbot website](#). In 1992, Dr. Sbaitsos developed for Microsoft DOS personal computers as an AI speech synthesis software. Made to showcase digitized voice that sound cards were able to synthesize. The software was bundled with some sound cards manufactured by Creative Labs. 1995 A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) is a free software chatbot created in AIML (Artificial Intelligence Markup Language), an open, minimalist, stimulus-response language for creating bot personalities like A.L.I.C.E. AIML uses two categories: pattern and template. Pattern is matched with input message, while templates are used to construct the response. 2001 SmarterChild, a popular chatbot that was available for Short Message Service (SMS) networks, AOL Instant Messenger, and MSN Messengers. Combines fast

information delivery, like news, sports, weather, with entertaining answers based on personality. [11] IBM's 2006 Watson is a system capable of answering questions in natural language. Originally the system was designed to compete and win in game show Jeopardy. In 2010s many intelligent virtual assistants became available as voice activated assistants. In 2010 Siri [5], a virtual assistant of Apples iOS interface, is created and uses natural language to answers users queries and to perform web service requests. In 2012 Google Now, a mobile application for android and iOS offers predictive cards with information and daily updates developed by Google to automatically answer users' questions. Also, it can be used to make recommendations and to perform actions by submitting queries to a set of web services. In 2015 Cortana, a Windows Speech Platform acts as a personal assistant recognizing NLP commands and using "Bing" to answer user questions. Finally, in 2016 Facebook will now allow businesses to deliver automated customer support, e-commerce guidance, content and interactive experiences [through chatbots](#) way. Through the [Messenger Platform's new Send/Receive API](#), bots can send more than just text. They will be able to respond with [structured messages](#) that include images, links and call to action buttons. These could let users make a restaurant reservation, review an e-commerce order and more. You can swipe through product carousels and pop out to the web to pay for a purchase [6]. Chatbots suddenly become the biggest thing in tech by unlocking the ability to provide personalized and interactive communication.

2.1.2. What is Chatbot?

A chatbot is an artificial intelligence (AI) software that can simulate a conversation (or a chat) with a user in natural language through messaging applications, websites, mobile apps or through the telephone. Chatbots interpret and process user's words or phrases and give an instant pre-set answer. They inhabit platforms like – FB Messenger, Whatsapp, Skype, Slack, Line, Kik, Wechat or even a website. Similar to regular apps, chatbots have application layer, a database, APIs and Conversational User Interface. There are three main types of Chatbots.

- Rule-based chatbots

This is the simplest type of chatbots today. People interact with these bots by clicking on buttons and using pre-defined options. To give relevant answers these chatbots require people to make a few selections. As a result, these bots

have longer user journey, and they are the slowest to guide the customer to their goal. These bots are great when it comes to qualifying your leads. The chatbot – asks questions, and people answer them with buttons. The bot analyzes collected data and gives a reply. But, for more advanced scenarios with many conditions or factors, these chatbots aren't always the best solution.

- **Intellectually independent chatbots**

These bots use Machine Learning (ML) which helps the chatbot learn from user's inputs and requests.

“ML is the ability of the computer to learn by itself from the data, recognize patterns and decide with minimal human interference.”

Intellectually independent chatbots are trained to understand specific keywords and phrases that trigger bot's reply. With the time they train themselves to understand more and more questions. You can say they learn and train from experience.

For example, you write to a chatbot: “I have a problem with logging into my account”. The bot would understand the words “problem” “logging” “account” and would provide a pre-defined answer based on these phrases.

- **AI – Powered Chatbots**

AI-powered bots combine and keeps the best features from Rule-based and Intellectually independent. AI-powered chatbots understand free language, but also have a predefined flow to make sure they solve user's problem.

They can remember the context of the conversation and the user's preferences. These chatbots can jump from one point of conversation scenario to another when needed and address random user request at any moment.

These chatbots use Machine Learning, AI and Natural Language Processing (NLP) to understand people.

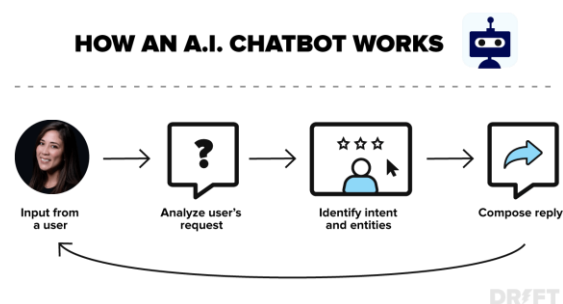


Figure 2.1 : How an AI chatbot works (Source drift.com)

2.1.3. NLP & Mechanisms

Modern organizations work with data that come in a variety of different forms including documents, audio recordings, emails, JSON, and many more. The most common way that such data is recorded is via text. That text is usually quite similar to the natural language that we use from day-to-day. **NLP** is the ability of the computer to understand and analyze natural textual data. For applications like chatbots, NLP is used as the main technique to find the right response and reply in understandable for a human language.

The goal of NLP is to make the interaction between computers and humans feel like communication between two people. With the help of NLP people can freely interact with chatbots asking a question. Some of the most common techniques of NLP to extract information from a text are the following:

- a. **Tokenization** – splitting a sentence into different parts, words or “tokens”. In the process, we throw away punctuation and extra symbols too.
- b. **Part of speech tagging** – determines which words are nouns, verbs, adjectives, etc.
- c. **Stemming** – is the technique for cleaning up text data for processing. Stemming is the process of reducing words into their root form. The purpose of this is to reduce words which are spelled slightly differently due to context but have the same meaning, into the same token for processing.
- d. **Named entity recognition** – finds entities in the text that the user has typed.

The developed chatbot of current thesis, will be an AI – powered Chatbot that will understand free language and will have predefined patterns like specific questions-answers in questionnaire. The Framework that will be used for NLP with all the techniques embedded, will be the WIT.AI Framework.

2.2.Applications & Frameworks

Chatbots are composite applications. Firstly, they are hosted to one of more platforms that each has its own requirements, specifications and available features. Afterwards, if it's necessary for chatbot to save the data, a database is required. Finally, a chatbot or NLP framework is usually used to handle answers and responses.

All these applications, platforms and framework must bind together. Often, all of them are tied up using a programming language. But not all programming languages support every framework and application and not all frameworks are supported to all platforms.

For that reason, the developed chatbot of the current thesis, will be a chatbot developed in Python using as main library and framework the *FLASK*. It will be hosted in Facebook Messenger and the connection with Python will established using the library PyMessenger. The Framework that will be used for NLP with all the techniques embedded, will be the WIT.ai. The library pywit will be used to use that framework in Python. A MongoDB database will set up in MongoDB Atlas, the cloud platform of Mongo, that will provide us with 24/7 support and functionality. The PyMongo library will be needed to use CRUD commands with Python and to use the aggregation operations. Code, requirements and every needed file will make up the application that will be built, run and operate in Heroku, a cloud Application Platform. Finally, a Tableau Dashboard will be created to monitor chatbot and users using a live connection with MongoDB in localhost.

2.2.1. MongoDB

In the world of the database, the most common and well popular database systems are RDBMS (Relational Database Management Systems). Now, if we want to develop an application which deals with a large volume of data, then we need to choose one such database which always provides a high – performance data storage solutions. So that, we can achieve the performance in the solution in terms of the data store and data retrieval with accuracy, speed and reliability. Now, if we categorized the database solutions then there are mainly two types of database category available i.e. RDBMS or Relational Database like SQL Server, Oracle etc. and another type is NoSQL database like MongoDB, CosmosDB etc.

The NoSQL database are being an alternative of conventional SQL Database and this type of database provides mostly all types of features which are normally

available in the RDBMS systems. Now-a-days, NoSQL databases become much popular in comparison to the past due to the simple design, provision for both horizontal and vertical scaling and for easy and simple control over the stored data [29]. This type of database basically breaks the normal tradition of data storage structure of the relational database. It gives the provision to the developers to store data in the database as per the actual requirement of their program. This type of facility we can't achieve by using the traditional RDBMS database.

MongoDB is a document-oriented database which stores data in JSON-like documents with dynamic schema. It means you can store your records without worrying about the data structure such as the number of fields or types of fields to store values. MongoDB documents are like JSON objects. Documents are gathered together in collections. A database stores one or more collections of documents. In MongoDB, databases hold one or more collections of documents.

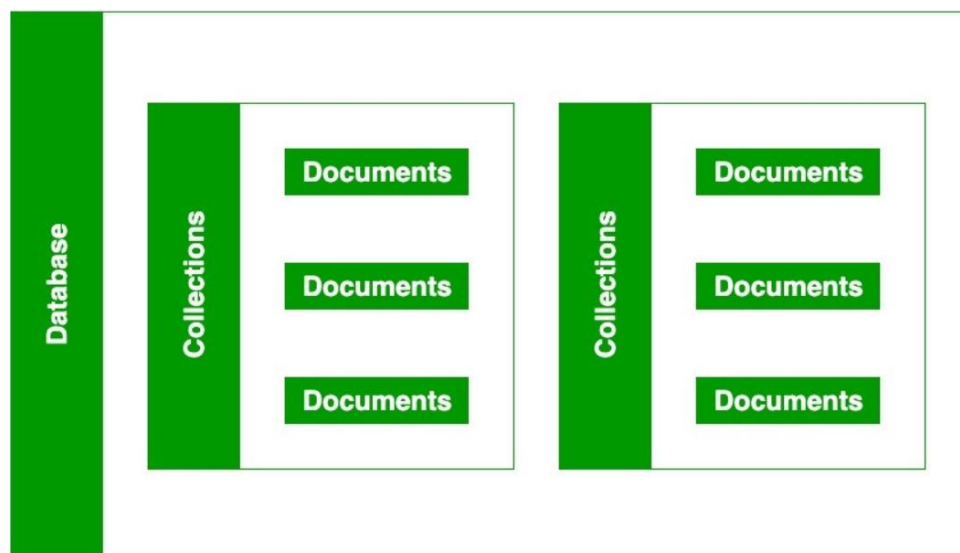


Figure 2.2 : Structure of MongoDB Database (Source geeksforgeeks.com)

The Easiest Way to Deploy, Operate, and Scale MongoDB in the Cloud is Atlas. MongoDB Atlas is a fully-managed cloud database developed by the same people that build MongoDB. Atlas handles all the complexity of deploying, managing, and healing your deployments on the cloud service provider of your choice (AWS, Azure, and GCP) [28]. MongoDB Atlas is always on and durable and also includes powerful features to enhance reliability for your mission-critical production databases, such as continuous backups and point-in-time recovery.

There are many APIs and libraries to use and interact with MongoDB in cloud. Since our application will be developed in Python a connection must be

established using a library for Mongo. PyMongo is the official Python driver that connects to and interacts with MongoDB databases. The PyMongo library is being actively developed by the MongoDB team.

Finally, to connect Mongo with a visualization application a MongoDB ODBC Driver is needed. The MongoDB ODBC Driver for BI Connector provides connectivity between a SQL client and the MongoDB Connector for BI. The ODBC driver enables users to create a Data Source Name (DSN) and connect a variety of BI tools to the BI Connector. It includes the BI Connector authentication plugin, which implements the client side of MongoDB-supported authentication mechanisms. BI Connector will be used with Tableau, a visualization software that will be explained in a later chapter.

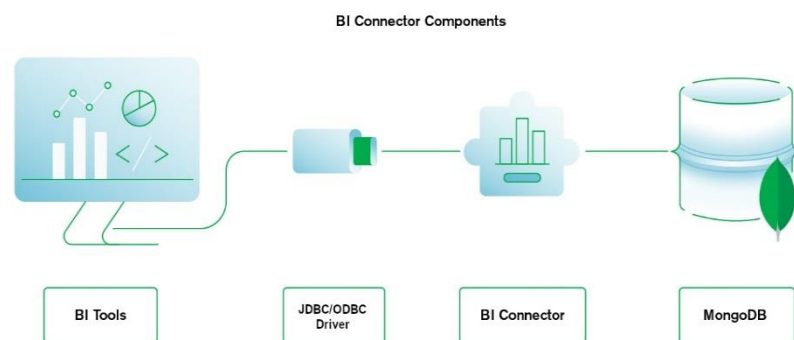


Figure 2.3 : BI Connector Components (Source docs.mongodb.com)

Open Database Connectivity (ODBC) is an open standard Application Programming Interface (API) that allows applications to access data in database management systems (DBMS) using SQL as a standard for accessing the data. Each Database system will have its own API and hence ODBC driver. An ODBC driver acts as an interface between an application and a data source, which can be anything like a file, a relational or non-relational database, or a cloud application. The architecture of ODBC-based data connectivity is as follows.

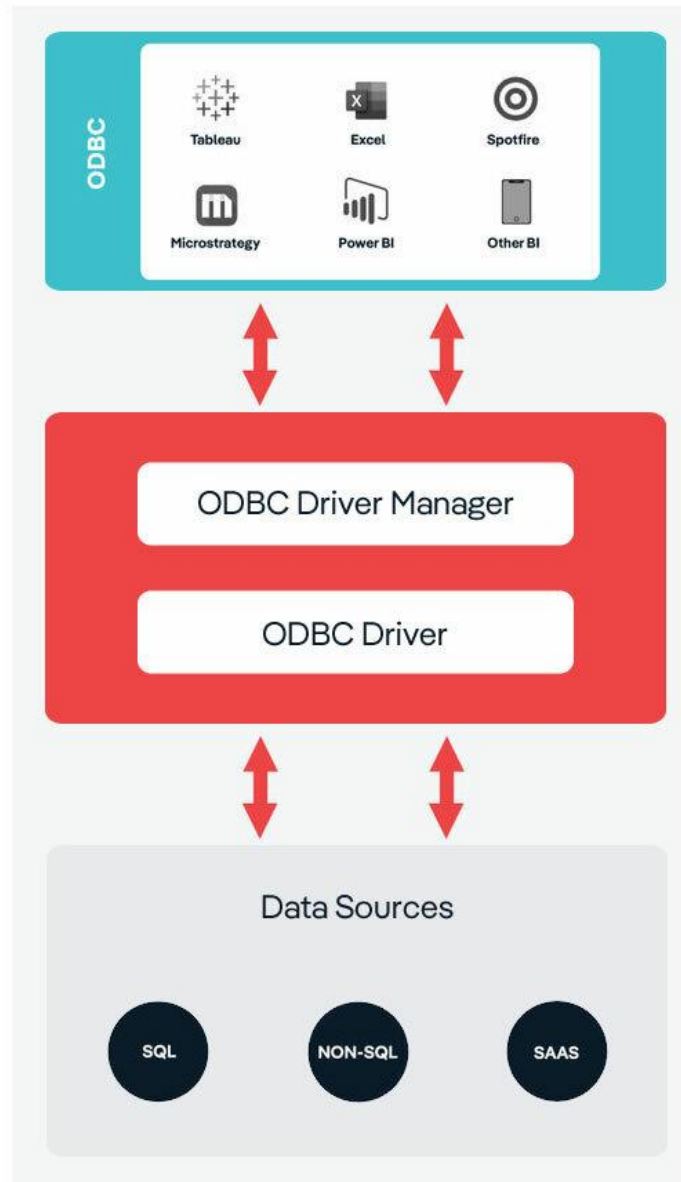


Figure 2.4 : ODBC and Connections (Source magnitude.com)

ODBC application is any compliant application that performs processing by submitting SQL statements and receiving data from the ODBC Driver Manager, in our case Tableau.

The **ODBC Driver Manager** loads and unloads ODBC drivers of the application. Windows OS comes with a default Driver Manager, while rest have the choice to use an open source ODBC Driver Manager.

The **ODBC Driver** processes ODBC function calls, submits SQL requests to a specific data source and returns results to the application. The ODBC driver may also modify an application's request so that the request fit to syntax supported by the associated database.

Data source is DBMS that is used where in our case is MongoDB.

A running [mongosqld](#) instance with authentication enabled using `--auth` must be active for the connection with Tableau. `mongosqld` accepts incoming requests from a SQL client and proxies those requests to a `mongod` or `mongos` instance. `mongod` is the primary daemon process for the MongoDB system. It handles data requests, manages data access, and performs background management operations. Finally, for a sharded cluster, the `mongos` instances provide the interface between the client applications and the sharded cluster. The `mongos` instances route queries and write operations to the shards. From the perspective of the application, a `mongos` instance behaves identically to any other MongoDB instance.

2.2.2. Facebook

Facebook is a free social networking website where users can post comments, share photographs, and post links to news or other interesting content on the web, chat live, and watch short-form video.

Shared content can be made publicly accessible, or it can be shared only among a select group of friends or family, or with a single person.

Facebook is user-friendly and open to everyone. Even the least technical-minded people can sign up and begin posting on Facebook. Although it started out to keep in touch or reconnect with long-lost friends, it rapidly became the darling of businesses that were able to closely target an audience and deliver ads directly to the people most likely to want their products or services. Its success can be attributed to its ability to appeal to both people and businesses and its ability to interact with sites around the web by providing a single login that works across multiple sites.

Facebook makes it simple to share photos, text messages, videos, status posts and feelings on Facebook. The site is entertaining and a regular daily stop for many users. Unlike some social network sites, Facebook does not allow adult content. When users transgress and are reported, they are banned from the site. Facebook provides a customizable set of privacy controls, so users can protect their information from getting to third-party individuals.



Figure 2.5 : Logo of Facebook (Source Facebook.com)

Facebook Messenger is an instant messaging service owned by Facebook, launched in August 2011, replacing Facebook Chat. The application and website are an instant messaging service that connects to the Facebook database and has replaced the in-app Facebook messaging service. To use Facebook Messenger, you must first have a Facebook account. You can sign up through the Facebook mobile application or by visiting [Facebook.com](https://www.facebook.com). If you already have a Facebook account, or just created one, you can continue by downloading the Facebook Messenger application from [Google Play](#), [App Store](#), [Windows Phone Store](#), and [Blackberry World](#), or by visiting [Messenger.com](https://messenger.com).

It allows Facebook users to connect with each other and send instant messages, emoji, photos, videos, and perform other light tasks. Additionally, Messenger provides features like send money to friends, get the weather and place product orders. In 2016, Mark Zuckerberg, CEO of Facebook, announced the arrival of Messenger chatbots [26]. Facebook now allow you install Messenger chatbots on your business page. Chatbots allow you to have an automated conversation with people who click on your Facebook Messenger to start a conversation. Companies identified their value to automate many low-level customer service functions. Part of their benefits are the reduced workload of frontline staff, allowing them to focus on more complicated customer issues, 24/7/365 availability and the instant access to basic information quickly and easily.

Messenger permits you to deploy an app like chatbots if you fulfill three requirements. You need to have:

- **Facebook Page:** Make sure that you have a Facebook Page since it represents your business identity when connecting with people on Messenger. To create a new Page, visit <https://www.facebook.com/pages/create>, you can also set up a test Page to start.

- **Facebook Developer Account:** Required to create new apps, which are the core of any Facebook integration. You can register as a developer by going to the [Facebook Developers website](https://developers.facebook.com/) and clicking the "Get Started" button.
- **Facebook App:** Contains the settings for your app, including access tokens.

Having the above requirements satisfied, grant developers the permission to create their app [4]. To do so, visit <https://developers.facebook.com/> and click on *Create App* button.

Note! During the Covid-19 Pandemic period (March 2020 – May 2021) the only app type that could be created was *Business Type App*.

When your new Facebook Page and App are ready you must connect them together. That connectivity will be achieved using Flask and Pymessenger. Let's start with Flask.

Flask is a web framework, it's a Python module that lets you develop web applications easily. It's has a small and easy-to-extend core: it's a microframework that doesn't include an ORM (Object Relational Manager) or such features.

It does have many features like URL routing, template engine. It is a WSGI web app framework. The Web Server Gateway Interface (Web Server Gateway Interface, WSGI) has been used as a standard for Python web application development. WSGI is the specification of a common interface between web servers and web applications. Werkzeug is a WSGI toolkit that implements requests, response objects, and utility functions. This enables a web frame to be built on it. The Flask framework uses Werkzeug as one of its bases. jinja2 is a popular template engine for Python. A web template system combines a template with a specific data source to render a dynamic web page.

Using Flask, we can create an endpoint – a fancy way of referring to a website URL. When a user sends us a message, Facebook will send that data to our endpoint, where we will send a response back to Facebook to show the user.

First, we imported the Flask class. An instance of this class will be our WSGI application. Next, we create an instance of this class. The first argument is the name of the application's module or package. If you are using a single module (as in this example), you should use `__name__` because depending on if it's started as application or imported as module the name will be different (`'__main__'` versus the actual import name). This is needed so that Flask knows where to look for templates, static files, and so on. For more information have

a look at the Flask documentation. We then use the *route()* decorator to tell Flask what URL should trigger our function.

Functions will be triggered through some HTTP methods. HTTP methods are the standard way of sending information to and from a web server. To break it down, a website runs on a server or multiple servers and simply returns information to a client (web-browser). Information is exchanged between the client and the server using an HTTP protocol that has a few different methods. We will be discussing the most commonly used ones called POST & GET. GET is the most commonly used HTTP method and it is typically used to retrieve information from a web server. POST is commonly used to send information to a web server. It is more often used when uploading a file, getting form data and sending sensitive data. POST is a secure way to send data to a web server. In Flask, to specify that a page works with both POST and GET requests we need to add a method argument to the decorator.

To run the application, execute the *app.run()*. Notice also that we are setting the debug parameter to true. That will print out possible Python errors on the web page helping us trace the errors. However, in a production environment, you would want to set it to False as to avoid any security issues. Port 80 is the default http port (Apache).

To handle sending messages back to a user who communicates with our bot, we'll be using the PyMessenger library to handle sending responses to users. PyMessenger is a Python Wrapper for Facebook Messenger Platform. It uses the Facebook's API called Graph. It has a single Bot class, that is used for sending different type of messages. Most of the function calls have inside hardcoded dictionaries to be converted to json. PyMessenger provides functions to send messages, create and use buttons and all the needed capabilities for chatbot. The functions return the full JSON body of the API call to Facebook.

To get the JSON and use the Bot class, the Page Access Token of the Application's Facebook page must be parsed as argument. An access token is an opaque string that identifies a user, app, or Page and can be used by the app to make graph API calls. When someone connects with an app using Facebook Login and approves the request for permissions, the app obtains an access token that provides temporary, secure access to Facebook APIs. Access tokens are obtained via a number of methods.

The token includes information about when the token will expire, and which app generated the token. Because of privacy checks, the majority of API calls on Facebook need to include an access token. There are different types of access tokens to support different use cases:

Access Token Type	Description
User Access Token	This kind of access token is needed any time the app calls an API to read, modify or write a specific person's Facebook data on their behalf. User access tokens are generally obtained via a login dialog and require a person to permit your app to obtain one.
App Access Token	This kind of access token is needed to modify and read app settings. It can also be used to publish Open Graph actions. It is generated using a pre-agreed secret between the app and Facebook and is then used during calls that change app-wide settings. You obtain an app access token via a server-to-server call.
Page Access Token	This kind of access token is similar to user access tokens, except that they provide permission to APIs that read, write or modify the data belonging to a Facebook Page. To obtain a page access token you need to start by obtaining a user access token and asking for the Page permission or permissions you need. Once you have the user access token you then get the page access token via the Graph API.
Client Token	The client token is an identifier that you can embed into native mobile binaries or desktop apps to identify your app. The client token isn't meant to be a secret identifier because it's embedded in apps. The client token is used to access app-level APIs, but only a very limited subset. The client token is found in your app's dashboard. Since the client token is used rarely, we won't talk about it in this document. Instead it's covered in any API documentation that uses the client token.

Table 2.1 : Description of Facebook Tokens (Source developers.facebook.com)

Page access tokens are used in Graph API calls to manage Facebook Pages. To generate a page access token, an admin of the page must grant your app the Page permission or permissions needed. Once granted, you can retrieve the Page access token using a user access token with the required permissions. With a Page Access Token, you can make API calls on behalf of a Page. For example, you could post a status update to a Page (rather than on the user's timeline) or read Page Insights data. Page access tokens are unique to each Page, admin, and app.

After parsing the Access Token, the functions of the Bot Class can be used. One of the main functions that will be used is `send_message(sender_id, message)` which takes 2 arguments. The 1st one is `sender_id`. Each time a user applies a request to our application (sends a message), in the incoming JSON, the id of the user is also included. That id is named as `recipient_id` which is Page Scoped User ID (PSID) of the message recipient. The user needs to have interacted with any of the Messenger entry points to opt-in into messaging with the Page. Note that Facebook Login integrations return user IDs are app-scoped and will not work with the Messenger platform. The 2nd argument is message which is the message that the application will send to the user.

Another main function is `send_raw(payload)`. This function will be used to send messages that include buttons. Buttons are a main functionality of the application. The argument of the function is Payload which is in JSON format.

```

payload = {
  "recipient":{
    "id":recipient_id
  },
  "messaging_type": "RESPONSE",
  "message":{
    "text": text,
    "quick_replies":quick_reply
  }
}

```

Figure 2.6 : JSON format of payload

The main fields that need to be filled are:

- `recipient_id`. The recipient's id received from user's message.
- `text`. The text that chatbot wants to include in message.
- `quick reply`. That field contains a nested JSON that set up the button and must be filled. These fields are:
 - `content_type`. An irrelevant field that will be left as `"text"`
 - `title`. That field represents the name that appear in the Facebook Messenger UI button. It's the text that is written in the button.
 - `payload`. The field that our application reads to identify the button that is clicked on Facebook.
 - `image_url`. The icon of the button. An 24x24 size icon can be placed in the button using the URL of the icon.

```

{
  "content_type":"text",
  "title":"Τερματισμός Τεστ", ##title of the button
  "payload":"termatismostest",
  "image_url":"https://www.gannett-cdn.com/uxstatic/usatc
}

```

Figure 2.7 : JSON format of quick_reply

Except from the connection using the Access Token to use the above functions, there is one more connection verification that must be done. It will be used to link Facebook page and application. That verifications occurs with Webhook. **Webhooks** are automated messages sent from apps when something happens. The Messenger Platform sends events to your webhook to notify you when a variety of interactions or events happen, including when a person sends a message. Webhook events are sent by the Messenger Platform as POST requests to your webhook.

Setting up a callback URL for webhook events requires verification that you control or own the domain that hosts your application.

1. Developer provides the webhook URL along with a developer generated verify token via the app dashboard.
2. The Messenger Platform will try to verify your webhook by sending a GET request to the callback URL with the parameters listed below.
3. The callback URL responds with the value of the **hub.challenge** sent. The URL should validate that the **hub.verify_token** matches with the token that was entered in the app dashboard.

The parameters that are provided in the GET request are the following.

Parameter	Description
<code>hub.mode</code>	Set to <code>subscribe</code>
<code>hub.verify_token</code>	The custom verify token that the developer provided
<code>hub.challenge</code>	Generated by the Messenger Platform. Contains the expected response.

Table 2.2 : Parameters of GET requests (Source developers.facebook.com)

The available webhook events are listed below. All webhook events are optional, so select those that are most relevant for the experience you are trying to create.

Webhook Event	Description
<code>messages</code>	Subscribes to Message Received events
<code>messaging_account_linking</code>	Subscribes to Account Linking events
<code>message_deliveries</code>	Subscribes to Message Delivered events
<code>message_echoes</code>	Subscribes to Message Echo events
<code>messaging_game_plays</code>	Subscribes to Instant Game events
<code>messaging_handovers</code>	Subscribes to Handover Protocol events
<code>messaging_optins</code>	Subscribes to Plugin Opt-in events
<code>messaging_policy_enforcement</code>	Subscribes to Policy Enforcement events
<code>messaging_postbacks</code>	Subscribes to Postback Received events
<code>message_reactions</code>	Subscribes to Message Reaction events
<code>message_reads</code>	Subscribes to Message Read events
<code>messaging_referrals</code>	Subscribes to Referral events
<code>standby</code>	Subscribes to Handover Protocol Standby Channel events

Table 2.3 : Webhook events (Source developers.facebook.com)

All webhook events have a common structure that includes information you will need to process and respond to the event, including the PSID of the sender and recipient of the event. In addition to the properties shown below, each event also has its own event-specific properties. Note that entry is an array and may contain multiple objects, so ensure your code iterates over it to process all events.

When you receive a webhook event, you must always return a 200 OK HTTP response. The Messenger Platform will resend the webhook event every 20

seconds, until a 200 OK response is received. Failing to return a 200 OK may cause your webhook to be unsubscribed by the Messenger Platform.

A delivery receipt will automatically be sent to the user after your webhook has acknowledged the message. You can also use Sender Actions to notify that the message has been seen. Your webhook should meet the following minimum performance standards:

- Respond to all webhook events with a 200 OK.
- Respond to all webhook events in 20 seconds or less.

If your webhook fails to meet either of the above requirements for more than 15 minutes a 'Webhooks Failing' alert will appear in the 'Alerts' page of your app settings. If your webhook continues to fail for 1 hour, you will receive a 'Webhooks Disabled' alert, and your Facebook app will be unsubscribed from receiving webhook events for the Page. Once you have fixed the issues with your webhook, you must set up your webhook again and every connection will be ready.

Since connections are ready means that now we can send requests to our application. The first request concerns the 'get started' button of the welcome screen which starts a conversation with your Messenger bot when it is tapped. When the button is tapped, your webhook will receive a `messaging_postbacks` event that contains a string specified by you in the `get_started` property of your bot's Messenger profile. This postback should be used to trigger your initial welcome message, such as a set of quick replies, or a text message that welcomes the person [25].

To set the postback payload, send a POST request to the [Messenger Profile API](#):

```
curl -X POST -H "Content-Type: application/json" -d '{
  "get_started": {"payload": "<postback_payload>"}
}'
"https://graph.facebook.com/v2.6/me/messenger_profile?access_token=<PAGE_ACCESS_TOKEN>"
```

Some of the best Practices that Facebook suggests are:

- **Do** communicate next steps to encourage a response in your welcome message. You can use buttons to add structure to your message and call out specific actions people can take.

- **Do** share basic commands in your welcome message. Communicate which keywords or terms people can use to ask for help, get updates, etc., so they find what they want more quickly.
- **Do** change your onboarding experience when your bot experience changes. Revisit your greeting text and welcome message as you update your capabilities to make sure they're still relevant.
- **Don't** forget everything on the screen works together. The context you provide in your Messenger Greeting should complement the "Get Started" button.
- **Don't** be too generic. Try addressing people by name to make the message feel personal and treating it as an opportunity to teach them how to use and control the experience.

The greeting text on the welcome screen is the first opportunity to tell a person why they should start a conversation with our Messenger bot. Some things we might include in our greeting text might include a brief description of what our bot does, such as key features, or a tagline. This is also a great place to start establishing the style and tone of our bot. The Greeting text of our Chatbot greets the user and let him know about what the chatbot is for. Three buttons are also sent to the user to make his interaction easier. Those buttons are about COPD information, COPD questionnaire and COPD prevention tips and they will help user to navigate along chatbot.

2.2.3. Wit.ai

Wit.ai is a free open NLP Framework for developers, acquired by Facebook, that helps you to build applications that uses chatbots or conversational tools. It provides an easy-to-use interface to create your app in the language of your choice, train and improve your app and to monitor every process regarding the NLP mechanism. Wit 's APIs are really powerful and are being supported by multiple programming languages. For Python, the message API is called **wit** and it will be used, to identify the way our chatbot "understands" the user's inputs. You need to have a Facebook Account to access and use Wit.ai. Logging with your Facebook Account you will be able to create your first application.

The next step is to train the application. Training will allow application to understand what the end-user tries to say to the chatbot. Since chatbot, will be used for information and guidance about COPD, the inputs of the user will focus on questions like:

- What is COPD?
- What is the best prevention for COPD?
- I need to take the COPD test.

Each of the above questions is an utterance. **Utterances** are examples of what a user says. In other words, utterances are examples of sentences that are possible to be questioned by users.

In each of the above questions, the user has an intention or **Intent**. The intent of a user is the aim or goal they expect to achieve when launching a conversation. Intent is a critical factor in chatbot functionality because the chatbot’s ability to parse intent is what ultimately determines the success of the interaction. For example, when a user asks, “*What is the best prevention for COPD*”, the user’s intention is *prevention*.

Also, each sentence or utterance contains some entities. **Entities** are data points or values that defines the type of information that will determine the intent of the utterance. For example, when a user asks, “*What is the best prevention for COPD*”, the entities of that utterance are “*prevention*” and “*COPD*”.

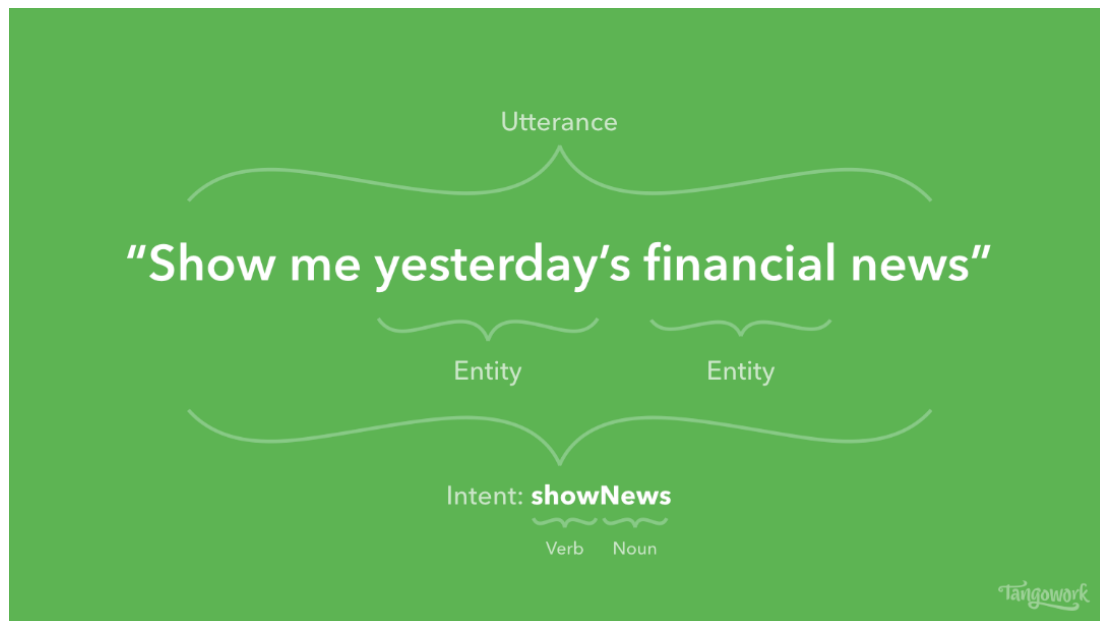


Figure 2.8 : Deconstruction of a sentence (Source aidev.co.kr)

Furthermore, after the successful training of the application the connection with Python must be established. That will be accomplished using the Python library of Wit.ai called pywit and the main class that will be used is **Wit()** and the main function is **message()**.

Finally, we need to better understand and iterate upon the performance of the Wit application and thus improve it. To make the NLP API more accurate, reliable and scalable we can find the Insights section on the Intents page in the Wit Application console by clicking any intent. Having validated enough data with the bot, this section will show how the Wit model perform at different confidence thresholds. To start, we will be able to measure precision and recall for the Wit entities, allowing us to assess the confidence levels for both the accuracy and frequency at which you are able to identify an entity.

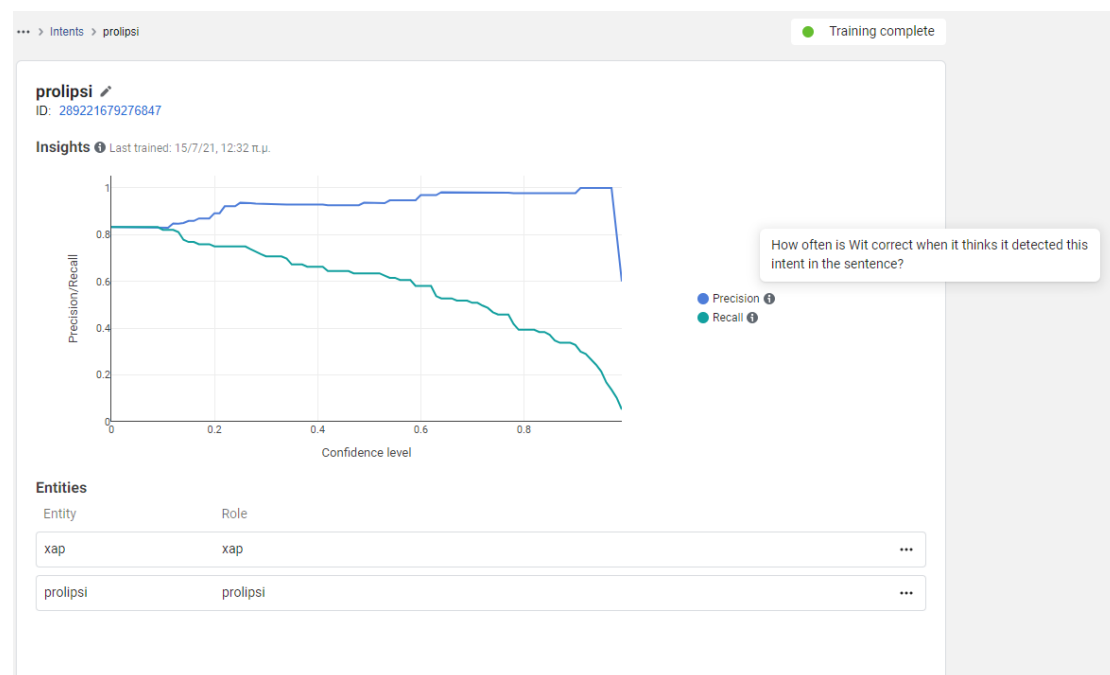


Figure 2.9 : Precision and Recall of an Intent (Source Wit.ai)

Precision-Recall is a useful measure of success of prediction when the classes are very imbalanced. In information retrieval, precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned.

The precision-recall curve shows the tradeoff between precision and recall for different threshold. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

A system with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system

with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high precision and high recall will return many results, with all results labeled correctly.

This allows us to pick the confidence threshold depending on what we want to optimize.

In the Insights section you can find:

- The App Traffic chart that shows the usage of your app. This is a measure of total API calls from users to your app over time.
- The Intent Usage chart that shows the usage of your app on a per-intent basis. This is a measure of total API calls from users to each intent of your app over time. Intent usage will shed light on how people are using your app and whether the app is behaving the way you expect it to.
- The Entity Usage chart that shows the usage of your app based on entities. This chart tracks API calls per entity over time.

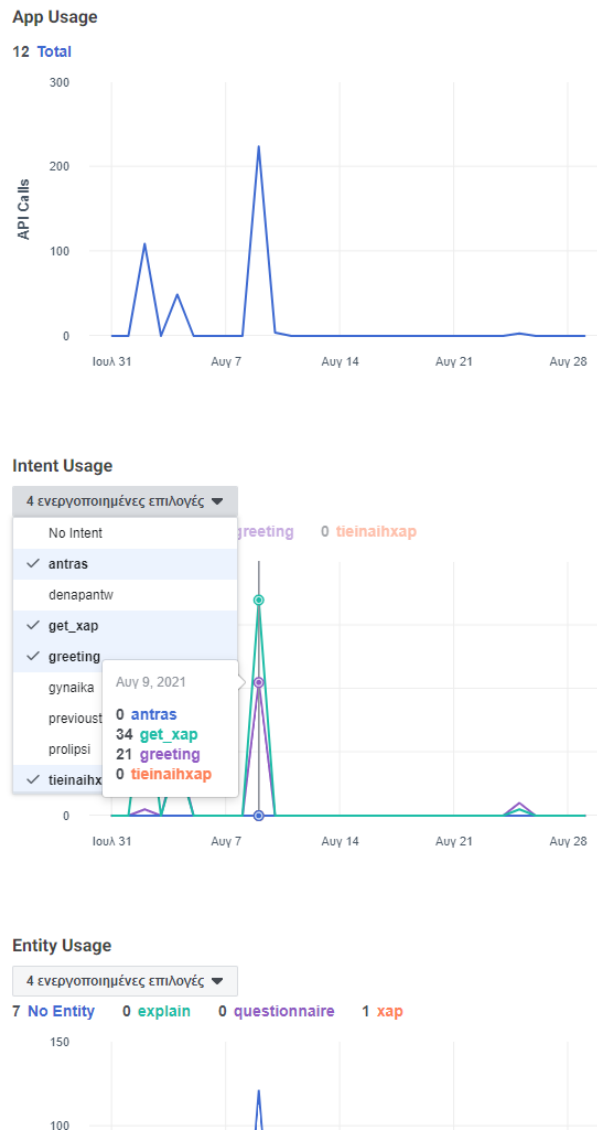


Figure 2.10 : Usage Statistics (Source Wit.ai)

Adjust the dates to get more specific metrics by selecting the calendar button on the upper right section of the Insights page. You can select the time period for which you need insights either on the calendar or from the options on the left.

2.2.4. Tableau

Tableau is a powerful and fastest growing data visualization tool used in the Business Intelligence Industry. It helps in simplifying raw data in a very easily understandable format. Tableau helps create the data that can be

understood by professionals at any level in an organization. It also allows non-technical users to create customized dashboards.

Data analysis is very fast with Tableau tool and the visualizations created are in the form of dashboards and worksheets. The great thing about Tableau software is that it doesn't require any technical or any kind of programming skills to operate. The tool has garnered interest among the people from all sectors such as business, researchers, different industries, etc.

To use all these features in our local machine the Tableau Desktop will be used. Tableau Desktop has a rich feature set and allows you to code and customize reports. Right from creating the charts, reports, to blending them all together to form a dashboard, all the necessary work is created in Tableau Desktop. For live data analysis, Tableau Desktop provides connectivity to Data Warehouse, as well as other various types of files. The workbooks and the dashboards created here can be either shared locally or publicly.

Tableau uses a workbook and sheet file structure, much like Microsoft Excel. A workbook contains sheets. A sheet can be a worksheet, a dashboard, or a story.

- A **worksheet** contains a single view along with shelves, cards, legends, and the Data and Analytics panes in its side bar. For details on the worksheet workspace, see [The Tableau Workspace](#).
- A **dashboard** is a collection of views from multiple worksheets. The Dashboard and Layout panes are available in its side bar.
- A **story** contains a sequence of worksheets or dashboards that work together to convey information. The Story and Layout panes are available in its side bar.

About sheets. Each workbook can contain different types of sheets: views (also known as worksheets), dashboards, and stories.

- A worksheet is where you build views of your data by dragging and dropping fields onto shelves.
- A dashboard is a combination of several views that you can arrange for presentation or to monitor.
- A story is a sequence of views or dashboards that work together to convey information.

The sheets display along the bottom of the workbook as tabs. In this section you'll learn how to create, open, duplicate, hide, and delete sheets. You'll also learn how to organize sheets in a workbook.

Within a workbook, you can create new sheets, clear an entire worksheet, duplicate sheets, hide or show a worksheet, and delete a sheet. Tableau has several ways to view and organize the sheets in your workbook.

Create new worksheets, dashboards, or stories

There are several ways to create new sheets in a workbook, dashboard, or a story. You can create as many sheets in a workbook as you want.

To create a new worksheet, dashboard, or story, click the **New Worksheet, New Dashboard,** or **New Story** button at the bottom of the workbook.

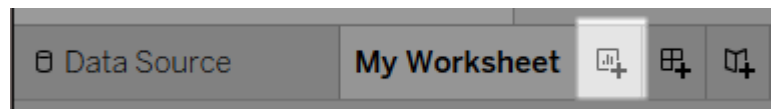



Figure 2.11 : New Worksheet (Source tableau.com)

*The **New Worksheet** button.*

To rename a new worksheet, dashboard, or story, right-click (Ctrl-click on a Mac) the tab and then select the **Rename** command.

More ways to create new worksheets

Create a new worksheet by doing one of the following:

- Select **Worksheet > New Worksheet.**
- Right-click any open tab in the workbook and select **New Worksheet** from the menu.
- On the toolbar, click the drop-down arrow on the **New Worksheet**  button and then select **New Worksheet.**
- Press Ctrl + M on your keyboard (Command-M on a Mac).

More ways to create new dashboards

Create a new dashboard by doing one of the following:

- Select **Dashboard > New Dashboard.**
- Click the **New Dashboard** button at the bottom of the workbook.

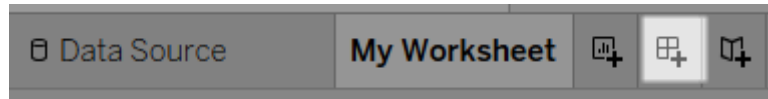



Figure 2.12 : New Dashboard (Source tableau.com)

- Right-click on any open tab in the workbook and select **New Dashboard** from the menu.
- On the toolbar, click the drop-down arrow on the New Worksheet  button and then select **New Dashboard**.

Head in the first Sheet to start building the 1st Worksheet in the workspace. The Tableau workspace consists of menus, a toolbar, the Data pane, cards and shelves, and one or more sheets. Sheets can be worksheets, dashboards, or stories. The working space appears as following.



Figure 2.13 : Working space of a Worksheet (Source tableau.com)

A. Workbook name. A workbook contains sheets. A sheet can be a worksheet, a dashboard, or a story.

B. [Cards and shelves](#) - Drag fields to the cards and shelves in the workspace to add data to your view.

C. [Toolbar](#) - Use the toolbar to access commands and analysis and navigation tools.

D. [View](#) - This is the canvas in the workspace where you create a visualization (also referred to as a "viz").

E. Click this icon to go to the Start page, where you can connect to data. For more information, see [Start Page](#).

F. [Side Bar](#) - In a worksheet, the side bar area contains the [Data pane](#) and the [Analytics pane](#).

G. Click this tab to go to the Data Source page and view your data. For more information, see [Data Source Page](#).

H. [Status bar](#) - Displays information about the current view.

I. Sheet tabs - Tabs represent each sheet in your workbook. This can include worksheets, dashboards, and stories.

To start building the visualizations, some new Calculations must be made with the retrieved data of the database. Calculated fields allow you to create new data from data that already exists in your data source. When you create a calculated field, you are essentially creating a new field (or column) in your data source, the values or members of which are determined by a calculation that you control. This new calculated field is saved to your data source in Tableau, and can be used to create more robust visualizations. But don't worry: your original data remains untouched.

You can use calculated fields for many, many reasons. Some examples might include:

- To segment data
- To convert the data type of a field, such as converting a string to a date.
- To aggregate data
- To filter results
- To calculate ratios

You create calculated fields using calculations. There are three main types of calculations you can use to create calculated fields in Tableau:

- **Basic calculations** - Basic calculations allow you to transform values or members at the data source level of detail (a row-level calculation) or at the visualization level of detail (an aggregate calculation).
- **Level of Detail (LOD) expressions** - Just like basic calculations, LOD calculations allow you to compute values at the data source level and the visualization level. However, LOD calculations give you even more control on the level of granularity you want to compute. They can be performed at a more granular level (INCLUDE), a less granular level (EXCLUDE), or an entirely independent level (FIXED) with respect to the granularity of the visualization.
- **Table calculations** - Table calculations allow you to transform values at the level of detail of the visualization only.

The type of calculation you choose depends on the needs of your analysis and the question you want to answer [31].

We are now ready to create the Dashboards that will be used to monitor the users of the chatbot so that we can offer them better guidance.

2.2.5. Heroku

Heroku is a container-based cloud Platform as a Service (PaaS). Developers use Heroku to deploy, manage, and scale modern apps. The platform is elegant, flexible, and easy to use, offering developers the simplest path to getting their apps to market.

Heroku is fully managed, giving developers the freedom to focus on their core product without the distraction of maintaining servers, hardware, or infrastructure. The Heroku experience provides services, tools, workflows, and polyglot support—all designed to enhance developer productivity. Heroku runs applications through virtual containers known as Dynos. A Dyno is a container on the Heroku platform utilized for running and scaling Heroku applications. They are fundamentally virtual Linux containers used for running code based on user commands. Applications can be scaled up to specific numbers of Dynos based on the requirements of developers. Heroku offers container management features to help users perform effortless scaling and dyno size, type and number management based on application requirements.

Dynos are the fundamental elements powering up a Heroku application. Developers can deploy their applications to Dynos and manage these units to

easily create and run scalable applications. They get freedom from having to perform infrastructure management tasks and can instead focus on the important aspects of creating and running applications.

Some of the Heroku's Advantages are:

- **Free to Start.** Users can get started with the free tier of Heroku, which does come with some restrictions and fundamental functionality. It is a suitable plan for those who intend to create a basic cloud app. Developers can opt for this plan initially before committing to the platform.
- **Easy to Use.** As a leading platform as a service offering, Heroku is known for its ease-of-use. The free version of Heroku is best suited for small development projects. Developers can also opt for several different tier packages that are more suitable for large businesses' complicated requirements. The intuitive Heroku platform dashboard helps users perform easy scaling, management, and application monitoring.
- **Developer-Centric.** Heroku focuses on helping developers hone their skills to create feature-rich applications. The experience is favorable for developers as they access some useful tools to accelerate core development processes. They are free from having to perform various repetitive tasks and can concentrate on critical development actions
- **Focus on Coding not Server.** Heroku takes away the burden of server management and gives developers access to an easy-to-use web UI. One-click deployments are also quite convenient for developers once an application is running. They can create new software releases conveniently in the form of release branches and quickly deploy branches across different environments.
- **Scalability.** Users Heroku PaaS relies on a Dyno for running the code written by developers. Each dyno is fundamentally a container encapsulating resources such as memory and CPU and application code and related dependencies. Heroku users can then choose to create any number of independent application copies once it has been prepared, a process known as horizontal scaling. Applications that receive an unprecedented influx of traffic can be handled through the one-click creation of hundreds of app instances. Heroku's auto-scaling feature helps it easily detect traffic spikes and create more Dynos accordingly.
- **Security.** Heroku offers developers a secure way of developing applications due to its set of security features. This PaaS platform

reduces developers' requirement to push constant security patches, which can be a hassle, especially in more sophisticated web applications. Heroku ensures an optimal level of security for servers, application code and prevents any possible issues.

- **Powerful CLI.** Heroku Command Line Interface or CLI is a feature that helps to develop and manage Heroku applications conveniently. The CLI is, in fact, one of the most reliable components of the Heroku PaaS.

But there are some limitations using Heroku which are:

- **Sleeping Apps.** A single web dyno assigned to a Heroku [application goes into sleep](#) when it has not received traffic for an hour. The dyno wakes up automatically for processing when the application is reaccessed. Sleeping applications can be a potential issue for some businesses. However, applications that have multiple dynos do not go to sleep.
- **Does not provide static IP addresses.** A static IP address is a fundamental requirement for most businesses and having one can prove to be quite an advantage. By default, Heroku does not offer static IP addresses within CRE (Common Runtime Environment). Users have to opt for the expressive Heroku Private Spaces to get static IP addresses that are not optimal regarding load balancing or customization. Users have to rely on third-party static IP add-on providers to use this IP type with Heroku. The complexity of utilizing this feature can be a concern for developers and business owners alike and can influence them to go for other providers.

There are 4 different Tiers that a developer can choose which are:

Type of Plan	Ideal for	Description
Free and Hobby	Minimum Viable Products Non-Commercial Applications Solo Projects	Heroku has a Free plan and a Hobby plan, both starting at \$0. These are the Starter plans, most well-suited for non-commercial app deployments.

Production	Small Business Applications	The Heroku Production plan begins from a rate of \$25 per month, but the rate can vary based on the requirements of the clients.
Advanced	Mission Critical Applications	The Heroku Advanced Plan begins from \$250 per month. It is most well-suited for businesses with more sophisticated development requirements.
Enterprise	Enterprise Grade Applications	The Heroku Enterprise package is a flexible plan that requires customers to pay depending on their needs. It is most suitable for companies with more sophisticated security and compliance requirements.

Table 2.4 : Tiers for a Heroku app (Source heroku.com)

Current Thesis chatbot application will be deployed in the Free Tier structure. But to use Heroku it's necessary to create an Account first. Visit the page <https://www.heroku.com> and Sign Up. Next, the Heroku CLI must be installed. Visit the page devcenter.heroku.com and install it. Git CLI is also needed to download to interact with git. Visit the <https://git-scm.com/downloads> and download it. Download dependencies are now fulfilled to start building the project.

2.2.6. Rest Python Libraries

Some other libraries that will be needed for the implementation of the chatbot are NumPy, datetime, os, venv, gunicorn.

Time and **Datetime** are modules of Python to work with dates and times. That modules are needed to get and use dates in our program.

The **OS** module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules.

NumPy (Numerical Python) is an open source Python library that's used in almost every field of science and engineering. It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems. Some functions of NumPy will be used to predict some numbers. For the prediction, the functions `numpy.polyfit()` and `numpy.poly1d()` of the NumPy package of Python will be used. The `numpy.polyfit()` method is used to fit our data inside a polynomial function. The `np.polyfit()` method takes few parameters and returns a vector of coefficients

p that minimizes the squared error in the order deg , $\text{deg}-1$, ... 0 . It least squares the polynomial fit. It fits a polynomial $p(X)$ of degree deg to points (X, Y) where X and Y are arrays. It returns a ndarray, shape $(\text{deg}+1,)$ or $(\text{deg}+1, K)$. `Numpy.poly1d` is a one-dimensional polynomial class. A convenience class, used to encapsulate “natural” operations on polynomials so that said operations may take on their customary form in code. The polynomial’s coefficients, in decreasing powers, or if the value of the second parameter is `True`, the polynomial’s roots (values where the polynomial evaluates to 0). For example, `poly1d([1, 2, 3])` returns an object that represents x^2+2x+3 , whereas `poly1d([1, 2, 3], True)` returns one that represents $(x-1)(x-2)(x-3)=x^3-6x^2+11x-6$.

The **venv** module provides support for creating lightweight “virtual environments” with their own site directories, optionally isolated from system site directories. Each virtual environment has its own Python binary (which matches the version of the binary that was used to create this environment) and can have its own independent set of installed Python packages in its site directories.

The **gunicorn** "Green Unicorn" (pronounced jee-unicorn or gun-i-corn) is a Python Web Server Gateway Interface (WSGI) HTTP server. It is a pre-fork worker model, ported from Ruby's Unicorn project. The Gunicorn server is broadly compatible with a number of web frameworks, simply implemented, light on server resources and fairly fast.

3. Implementation

3.1. Database & ODBC

MongoDB Atlas provides free Clusters with limited storage of 512MB. Atlas free clusters provide a small-scale development environment to host your data. Free clusters never expire and provide access to a subset of Atlas features and functionality. The procedure to create a free tier cluster is the following:

1. Create a MongoDB Account using your email.
2. Login in the MongoDB Atlas
3. Click on the *Build a Cluster* button
4. Select the *Shared Clusters* option and click *Create a Cluster*
5. Select the Cloud provider of your choice
6. Select the *M0 Sandbox* cluster tier
7. Name your Cluster and click *Create Cluster*

Creation may take up to 10 minutes and become ready to use. After the successful deployment of our project a few settings must be fixed to make database accessible and secure. Clicking on *Database Access* and adding an authentication method using password for access, will make database more secure. Now where our cluster and settings are ready, it's time to create a new database and collection for the documents we need to store.

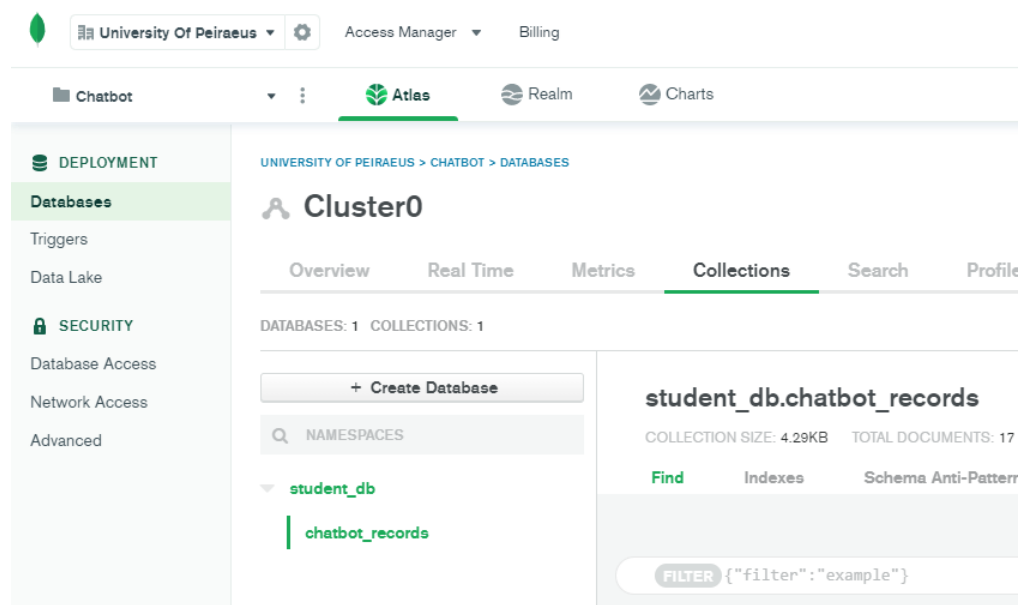


Figure 3.1 : Database to store chatbot data (Source mongodb.com)

It's necessary for the chatbot application to connect with the database and perform CRUD operations. On the *Databases/Connect* menu clicking on the

Connect your application there are multiple choices to connect the MongoDB database with the application.

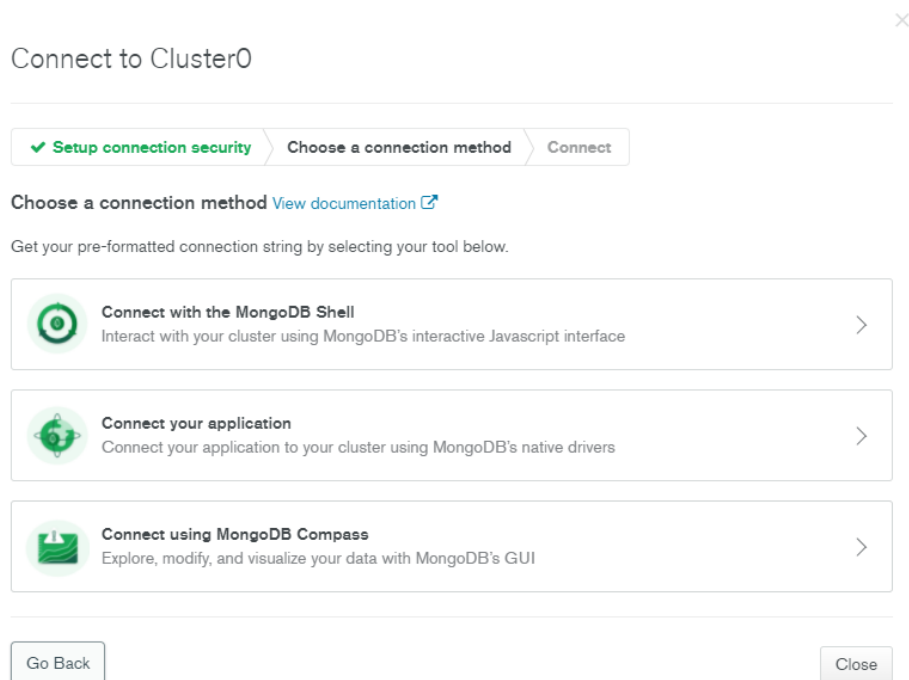


Figure 3.2 : Methods to connect to database (Source mongodb.com)

Chatbot will be developed using Python 3.7. In the *Driver* select *Python* and *Version* 3.6 or later. Below that, the connection string is created. Replace the parameter *<password>* with the password of the user and *<myFirstDatabase>* with the name of the database. Also, we must get the connection string using the driver Node.js and version 2.2.12. That connection string contains all the information of the database like uri and replicaSet name that will be needed for the connection with Tableau.

✕

Connect to Cluster0

✓ Setup connection security
✓ Choose a connection method
Connect

- 1 Select your driver and version

DRIVER

VERSION
- 2 Add your connection string into your application code

Include full driver code example

```
mongodb+srv://test:<password>@cluster0.79rb7.mongodb.net/myFirstDatabase?
retryWrites=true&w=majority
```

Replace **<password>** with the password for the **test** user. Replace **myFirstDatabase** with the name of the database that connections will use by default. Ensure any option params are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back
Close

Figure 3.3 : Connection via Python (Source mongodb.com)

From the perspective of the MongoDB Database, everything is ready to use. Next step is to use PyMongo, to connect it with our application. The connection will be established by creating a new instance of the MongoClient Class using the connection string that was acquired from the MongoDB Cloud Database. We can obtain our database by parsing the name of our database to the `get_database()` function.

```
from pymongo import MongoClient

#Connection to MongoDB database
clientMONGO = MongoClient("mongodb+srv://test: [REDACTED]@cluster0.79rb7.mongodb.net/<dbname?retryWrites=true&w=majority")
db = clientMONGO.get_database("student_db")
records = db.chatbot_records
```

Figure 3.4 : Python Library for MongoDB (Source mongodb.com)

Application is now connected with the MongoDB database and CRUD operations are available. The last connection for MongoDB that must be created is that with Tableau. To successfully connect Tableau with MongoDB an ODBC driver must be created.

To successfully create the needed ODBC Driver, the following procedure must be completed:

1. Download and install the MongoDB ODBC driver with version *1.4.1 for Windows*.
2. Add to environmental variables of the system, the bin folders of the MongoDB BI Connector.
3. Open ODBC Data Sources select *System DSN* and click *Add....*

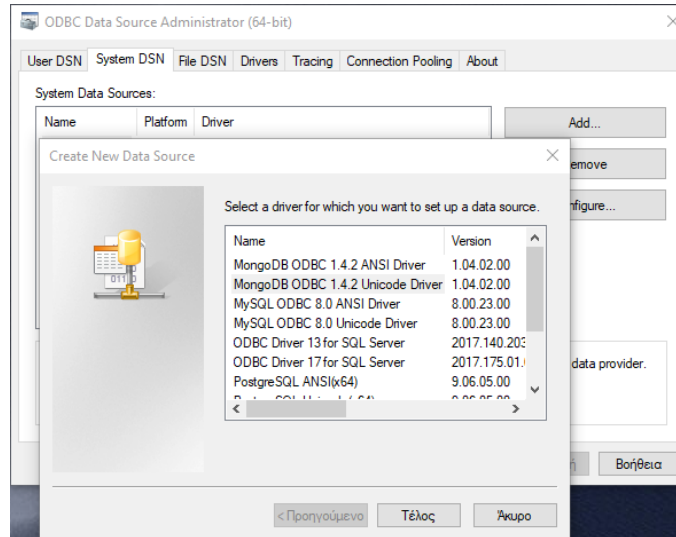


Figure 3.5 : MongoDB ODBC Driver

4. Select the MongoDB ODBC 1.4.2 Unicode Driver and fill the Connection Parameters. Data Source Name is the name of the Data source. Description is the description of the data source. TCP/IP Server is localhost since the connection is established in local machine. Port is the default port for SQL connection which is 3307. User and password fields are the user's name and password in MongoDB Cloud respectively. Database field can be filled with the database that the user needs to connect (not necessary).

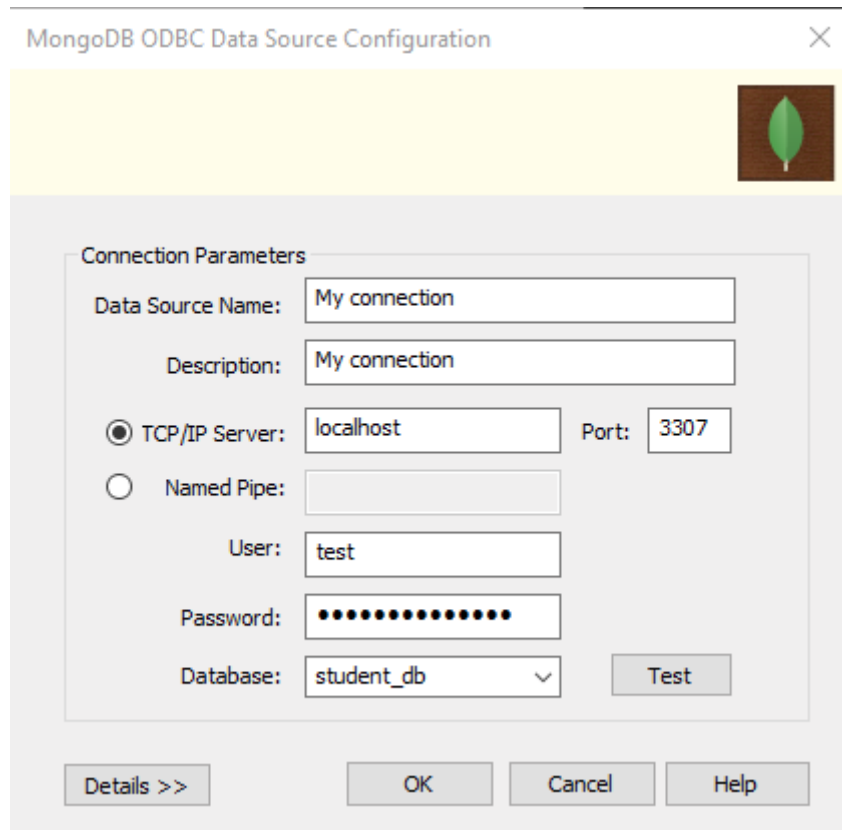


Figure 3.6 : Configuration of ODBC Driver

5. Click *OK* on the MongoDB ODBC Data Source window and click *OK* on the *ODBC Data Source* window.

Our ODBC Driver is now ready to use from the perspective of MongoDB. In the Tableau chapter, the connection between ODBC Driver and Tableau will be established and further explained.

3.2. Facebook Connectivity

To setup your app, give a name to your app and an email to let app contact you for potential policy violations, app restrictions or steps to recover the app if it's been deleted or compromised. Select if app's purpose is to access and use data on behalf of *Yourself or your own business* or *Clients*. Click Create and your app is ready.

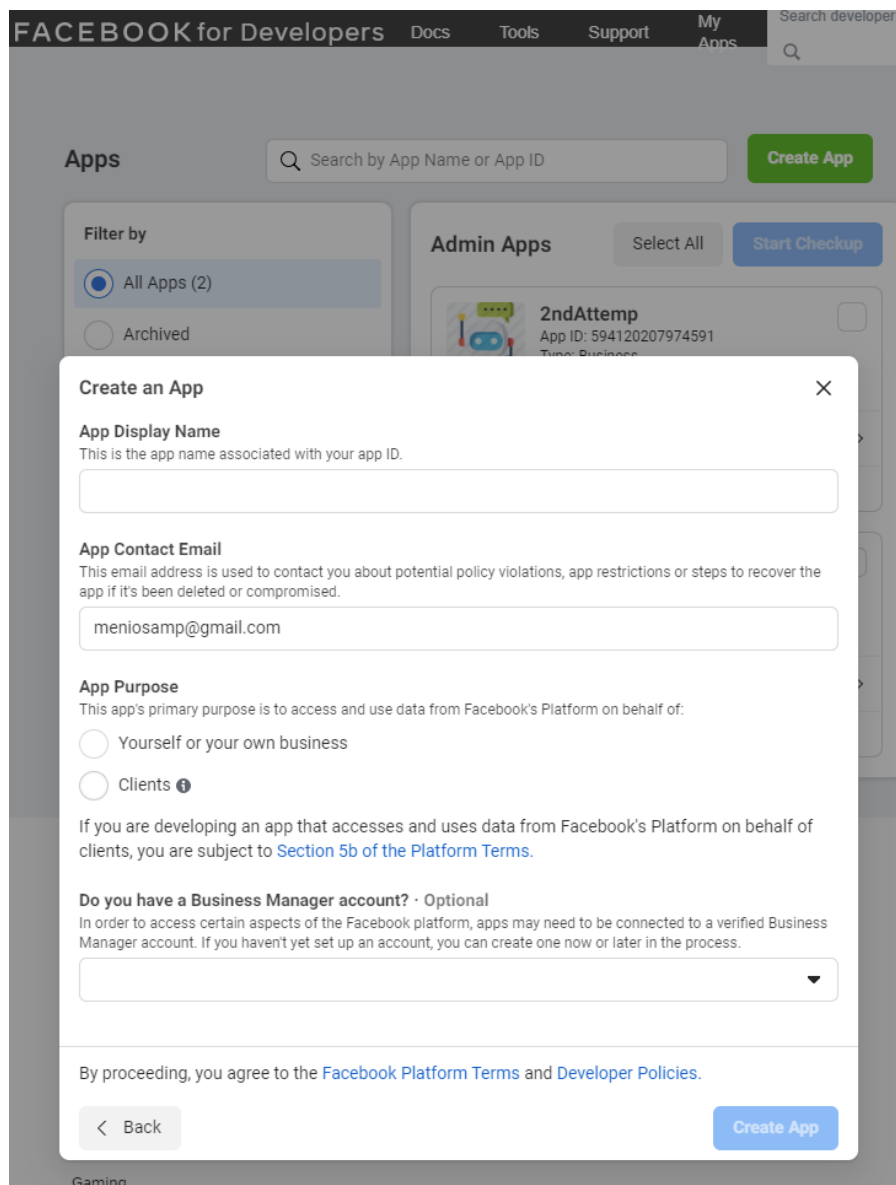


Figure 3.7 : Facebook App creation (Source developers.facebook.com)

Your newly created Facebook app is probably in development mode. Note that apps in this mode are only allowed to message people connected to the app (Admins, Developers and Testers). Once your app is ready to be public, the app needs to go through app review for the pages_messaging permission. Visit *App*

Review/Permissions and Features and find permission `pages_messaging`. Click *Request Advanced Access* and *Edit App Review Request*.

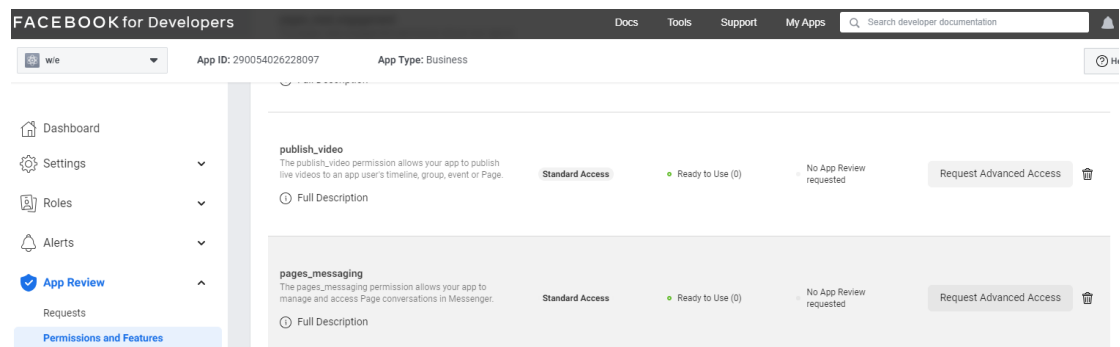


Figure 3.8 : Permissions of the Facebook App (Source developers.facebook.com)

Provide the needed verification details that are required and submit your request for review. It usually takes 1 to 5 days to approve or decline the request.

Each app must be linked to a Facebook Page. To create one visit *Settings/Advanced* and click *Create Page*.

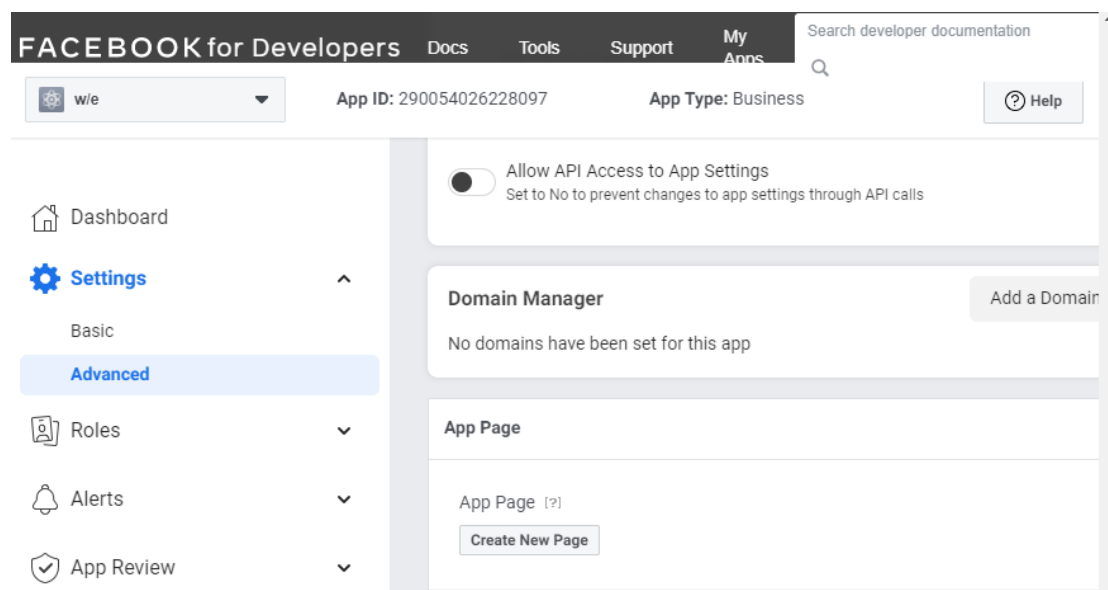


Figure 3.9 : Creation of Facebook Page (Source developers.facebook.com)

It will redirect you to your new page link. Name your page, choose the category that fits the most to the purpose of your application and finally write a description about the object of page. Click *Create Page* and the application's page is ready.

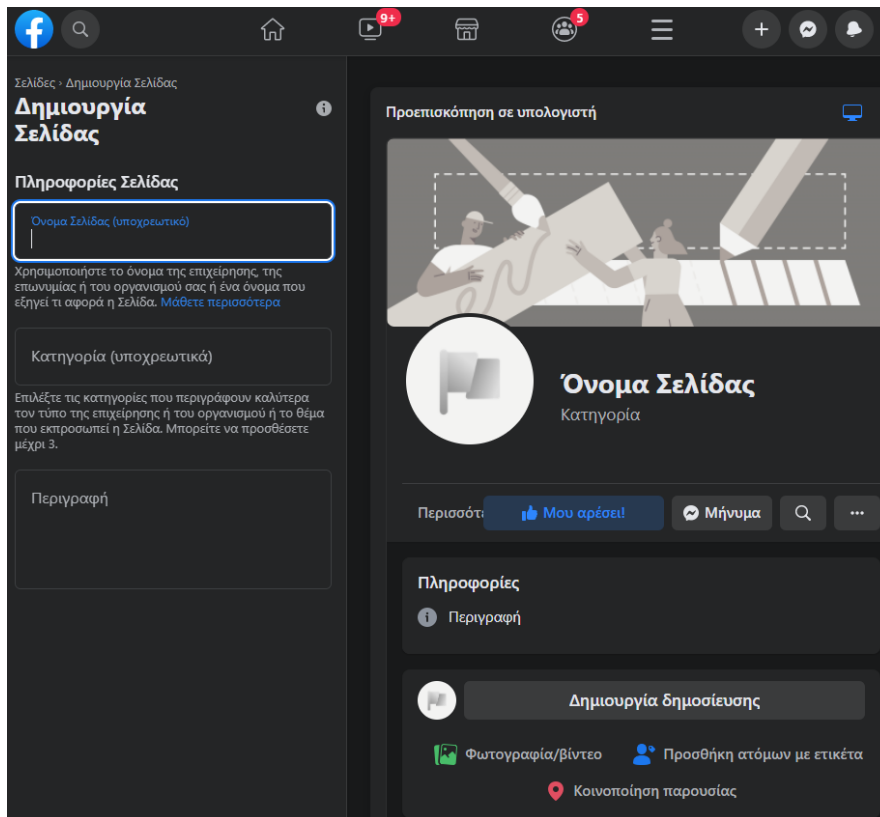


Figure 3.10 : Created Facebook Page (Source developers.facebook.com)

Since the application is a *Business Type App*, a *Business Verification* process must be done that allow Facebook Messenger to gather information about you and your business and verify your identity as business entity. To complete the process, you must move in the App Dashboard and go to Settings > Basic > Verification and click the Start Verification button.

- If your Facebook developer account is already associated with a Facebook Business account, you will be given the option to select a Business within it. If you don't have a Facebook Business account, or if your account contains no Businesses, you should create one. Connecting your app to a verified Business completes the connection process and there's nothing else to do. The Verification section should show that your app is now connected to the selected verified Business.
- If you connected your app to an unverified Business, Admin or a verified developer of the Business must complete the verification process within the Business Manager. Click Start Business Verification to load the unverified Business in the Business Manager and complete the verification process. Once you have completed verification, return to the Basic Settings panel. You should see that your app is now connected to a verified Business.

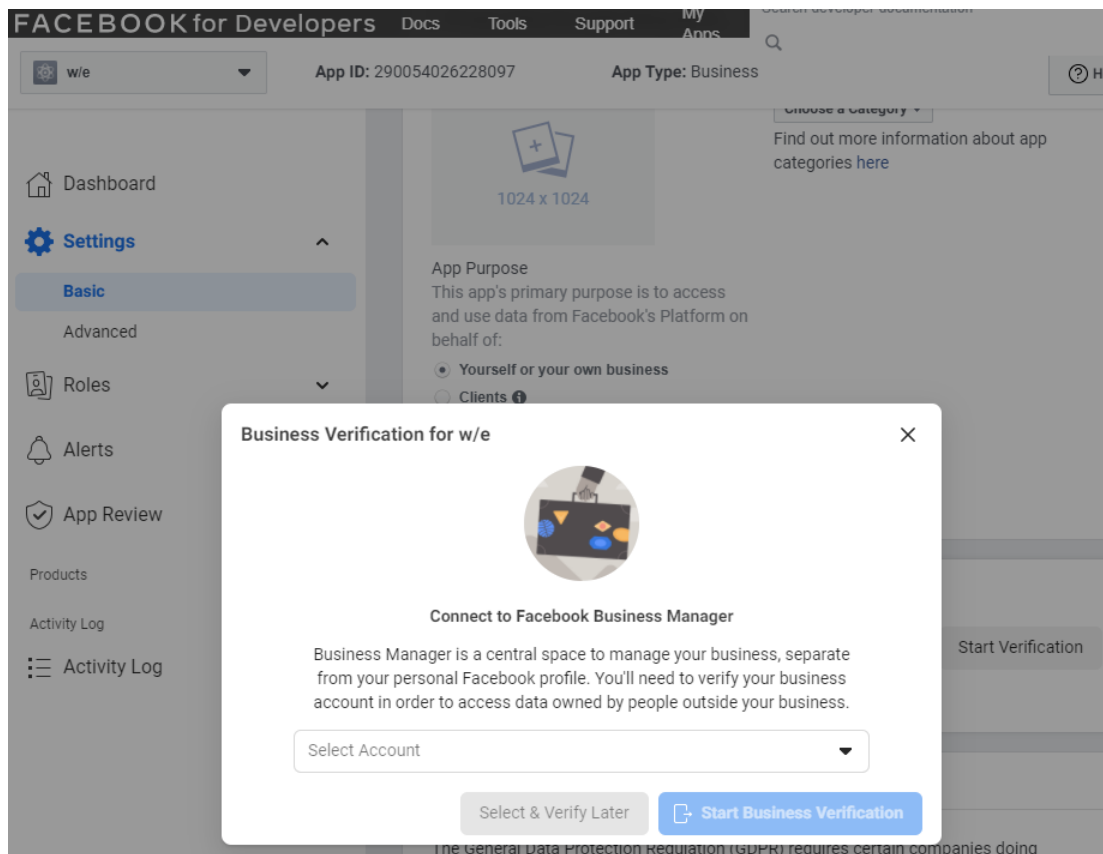


Figure 3.11 : Settings of the Facebook App (1) (Source developers.facebook.com)

The last step before getting application in public mode, is to insert the necessary information of the application in the Settings > Basic. These are:

- App Icon (1024 x 1024). An icon that fits your application's purpose
- Privacy Policy URL. You can either create a free privacy policy from various free websites or use Heroku's privacy URL.
- Category. Specify the category of your application.

Currently Ineligible for Submission

Your submission is missing data in the following fields:

- App Icon (1024 x 1024)
- Privacy Policy URL
- Category

<p>App ID</p> <input style="width: 95%;" type="text" value="290054026228097"/>	<p>App Secret</p> <input style="width: 95%;" type="password" value="••••••"/> Show
<p>Display Name</p> <input style="width: 95%;" type="text" value="w/e"/>	<p>Namespace</p> <input style="width: 95%;" type="text"/>
<p>App Domains</p> <input style="width: 95%;" type="text"/>	<p>Contact Email</p> <input style="width: 95%;" type="text" value="meniosamp@gmail.com"/>
<p>Privacy Policy URL</p> <input style="width: 95%;" type="text" value="Privacy policy for Login dialog and App Details"/>	<p>Terms of Service URL</p> <input style="width: 95%;" type="text" value="Terms of Service for Login dialog and App Details"/>
<p>App Icon (1024 x 1024)</p> <div style="border: 1px solid #ccc; padding: 10px; text-align: center; width: 100px; margin: 0 auto;"> <p>1024 x 1024</p> </div>	<p>Category</p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;"> Choose a Category ▾ </div> <p>Find out more information about app categories here</p>

App Purpose
 This app's primary purpose is to access and use data from Facebook's Platform on behalf of:

Yourself or your own business
 Clients ⓘ

If you are developing an app that accesses and uses data from Facebook's Platform on behalf of clients, you are subject to [Section 5b of the Platform Terms](#).

Figure 3.12 : Settings of the Facebook App (2) (Source developers.facebook.com)

After the successful completion of the above fields, the application is now ready to run in public mode.

Afterwards, to connect page with Python and use it with PyMessenger the Access Token of the page must be retrieved. In the Developers Facebook page, in the application's page at Messenger>Settings you can find the Access Token of the page.

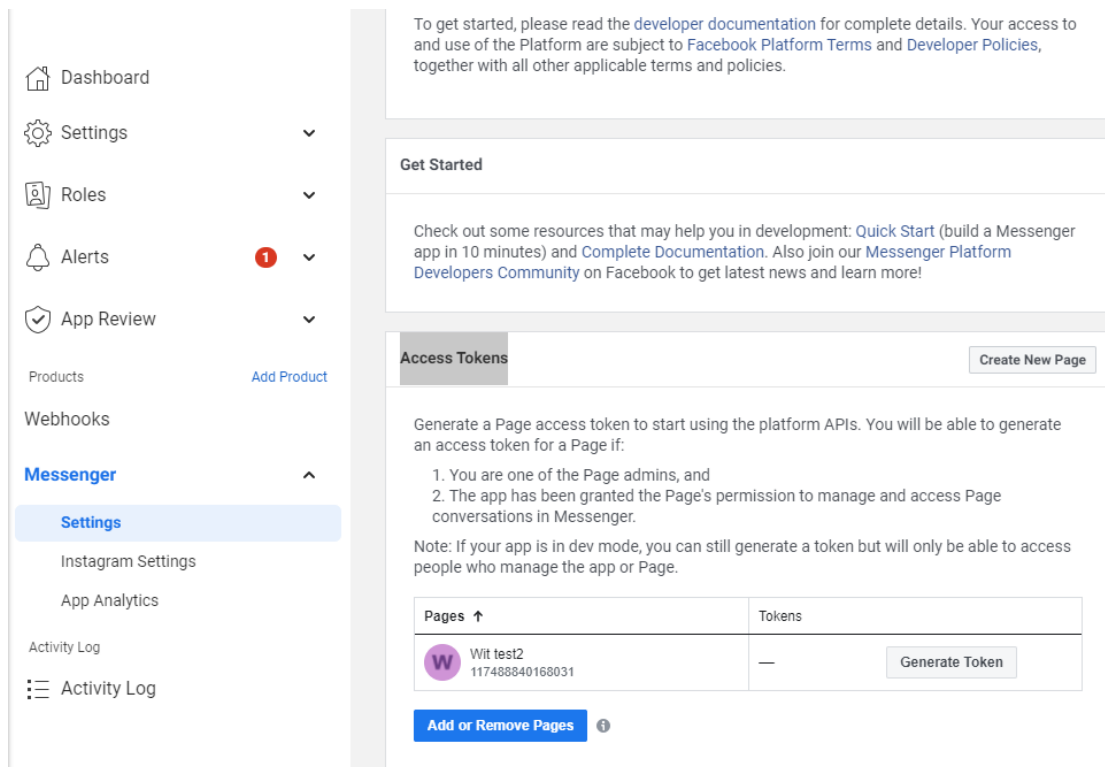


Figure 3.13 : Access Token of Facebook Page (Source developers.facebook.com)

Click the “*Generate Token*” button and your application’s token is generated and ready to be used. As explained, it will be used as a connection string in the Python code that will be handled by the Bot Class of the PyMessenger Library.

```
from pymessenger import Bot

#Connection to Facebook Messenger
PAGE_ACCESS_TOKEN = "EAAIcWWPzQL8BAIQm2P7pGPxgZl"
bot = Bot(PAGE_ACCESS_TOKEN)
```

Figure 3.14 : Python Library for Facebook Messenger (Source developers.facebook.com)

Webhook Verification will be achieved with the *verify()* function. The set token is “hello”.

```
@app.route('/', methods=['GET'])
def verify():
    # Webhook verification
    if request.args.get("hub.mode") == "subscribe" and request.args.get("hub.challenge"):
        if not request.args.get("hub.verify_token") == "hello":
            return "Verification token mismatch", 403
        return request.args["hub.challenge"], 200
    return "Hello world", 200
```

Figure 3.15 : Webhook Verification through Python (Source developers.facebook.com)

To set the same token on the application's Facebook page, go to Messenger>Settings. On the field "Verify Token" fill the Token "hello". Facebook will echo it back as part of callback URL verification. On the Callback URL the URL of the Heroku's Application will be filled. Heroku will further explained in another chapter. On the Webhook's Page select all fields for the chatbot.

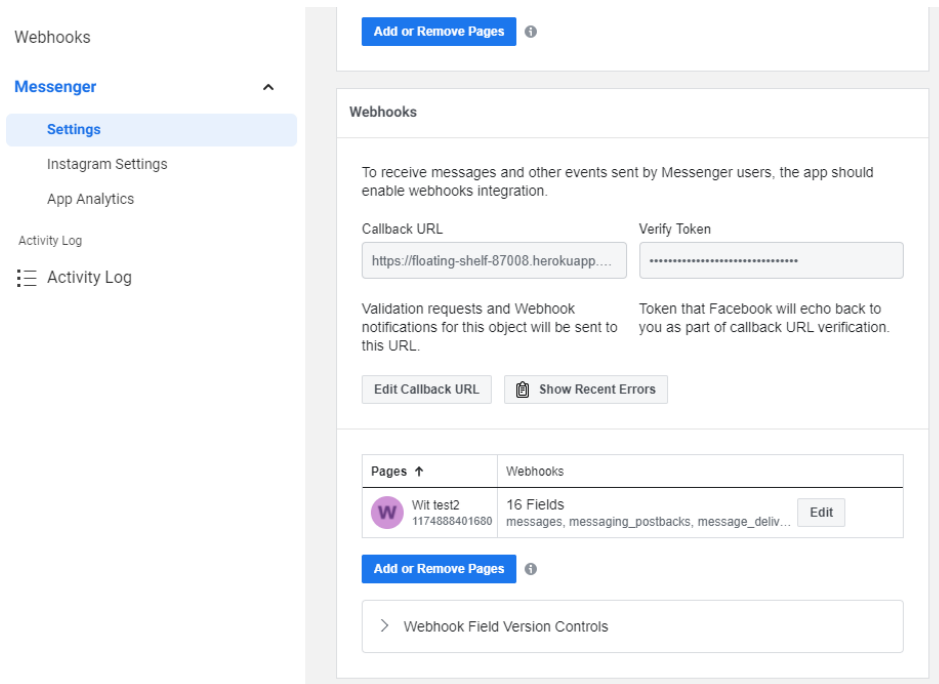


Figure 3.16 : Token Verification on Facebook App (Source developers.facebook.com)

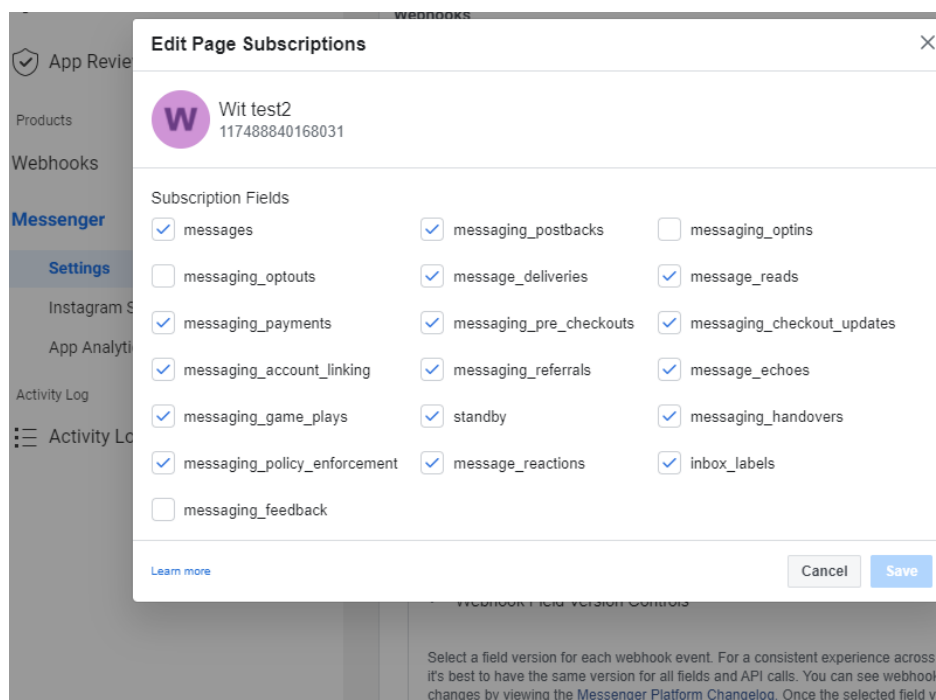


Figure 3.17 : Subscriptions of Facebook App (Source developers.facebook.com)

Connection is now verified, and the functions `send_raw(payload)` and `send_message(sender_id, message)` of the Bot Class can be used.

The next step is to create the needed buttons of the application using the above functions. As mentioned, buttons are recognized by the `payload` of the field `quick_reply` that is used on the variable `payload` of `send_raw()` function. Each time a user presses a button, the application “understands” that the user has pressed it if the `quick_reply` field is included in the JSON body. Then, to identify which button is pressed, the application checks the `payload` field inside the `quick_reply` field. `Payload` is the field that the application handles on code to reply on request.

Based on function `send_raw()` a new function is created to help us parse the arguments easier. The function `send_quick_reply(recipient_id, text, quick_reply)` gets 3 arguments. The recipient id, the text message that the application sends and the quick reply which is the argument to set up the button.

```
quick_reply = [
  {
    "content_type": "text",
    "title": "Τερματισμός Τεστ", ##title of the button
    "payload": "termatismostest",
    "image_url": "https://www.gannett-cdn.com/uxstatic/usatoday/usat-we
  }
]
send_quick_reply(sender_id, "Ποιο είναι το όνομά σας;", quick_reply)
```

Figure 3.18 : Quick Reply variable

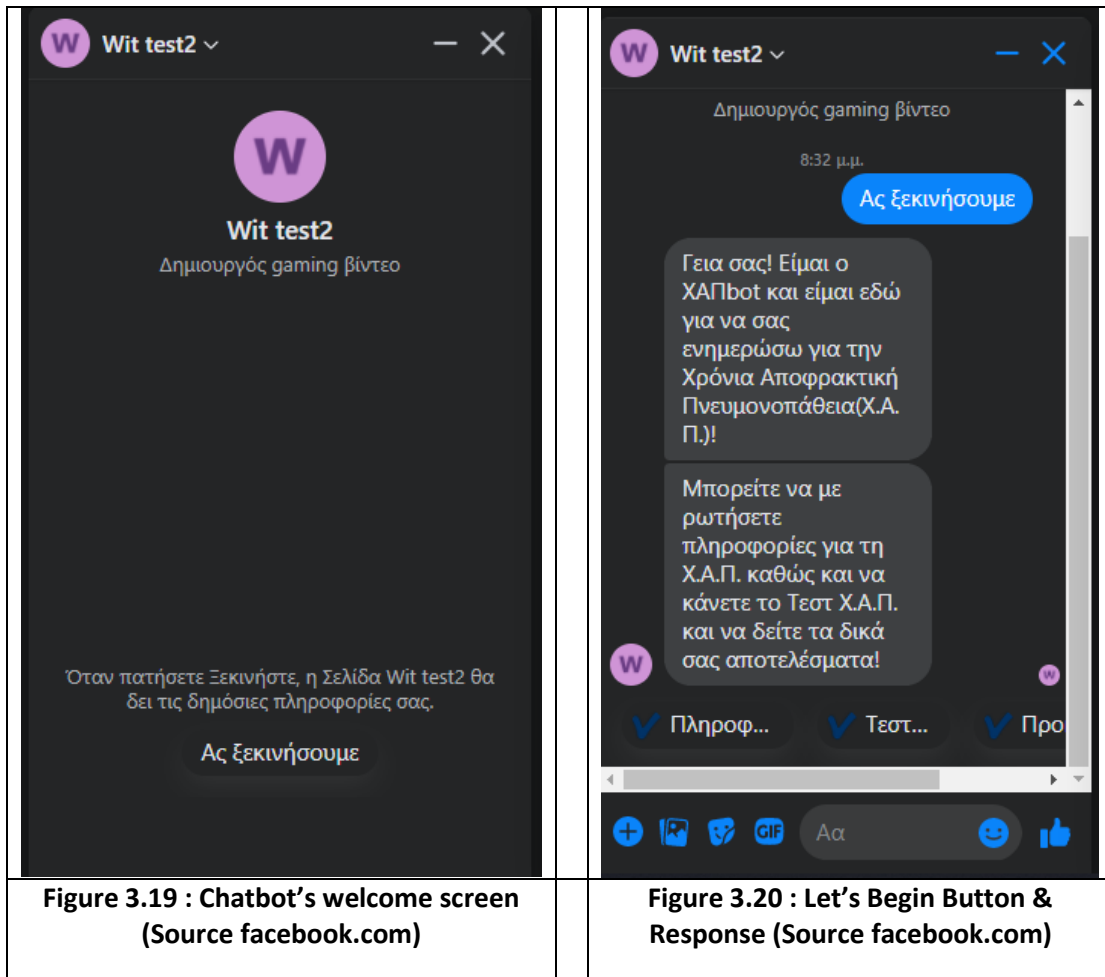
We have assigned 9 buttons in total for the chatbot application. The buttons titles, their payloads and their description are presented below:

Button Title	Payload	Description
Πληροφορίες	tieinaixap	Button to request information about COPD
Προηγούμενα Τεστ	previoustest	Button to request previous completed COPD questionnaires
Πρόληψη	prolipsi	Button to request ways to prevent COPD
Νέος Χρήστης	neosxristis	Button that gives the input "New User"
Γυναίκα	gynaika	Button that gives the input "Woman"
Άντρας	antras	Button that gives the input "Man"
Δεν απαντώ	denapantw	Button that gives the input "No Answer"
Τερματισμός Τεστ	termatismostest	Button to request to end the existing COPD questionnaire
Τεστ ΧΑΠ	testxapinit	Button to request to start a new COPD questionnaire

Table 3.1 : Buttons, Payloads and Description

Each button is used in multiple requests and multiple buttons can appear in a response. There is one more button that we need to setup in the Welcome Screen of Messenger. The welcome screen is the first thing people see when

they encounter your Messenger bot and includes information that allows a person to learn about your bot and what it offers. The welcome screen displays the name and responsiveness of your bot, the profile picture and cover photo from your Facebook Page, an optional greeting message, and the 'get started' button.



3.3. Wit Setting & Connectivity

The first step a developer needs to do to use Wit.ai Framework, is to create a new application by clicking on the “+ New App” button.

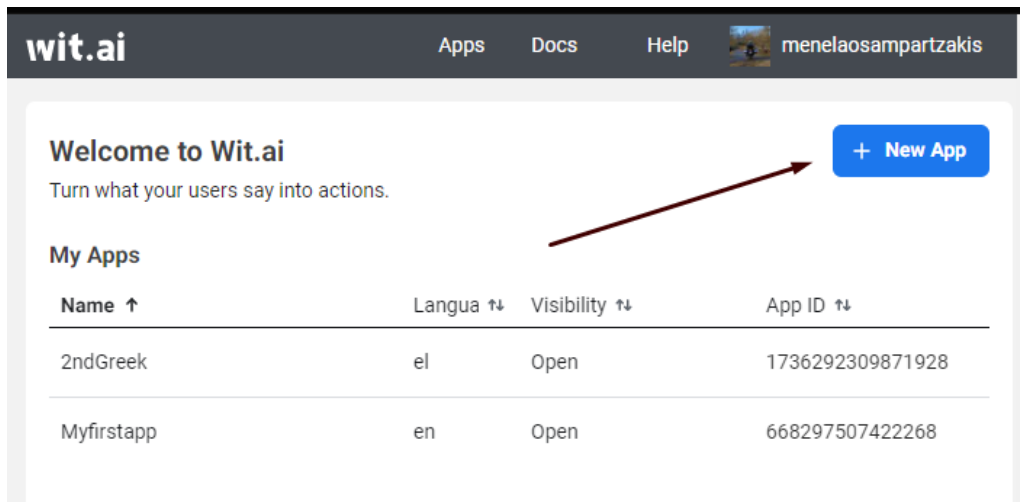


Figure 3.21 : Wit.ai App Creation (Source Wit.ai)

The new application needs a name and it's necessary to select the language that the application will use for the NLP mechanism. For chatbots that are used in Public applications, the “*Visibility*” selection must always be “*Open*”. Click “*Create*” and your application is ready.

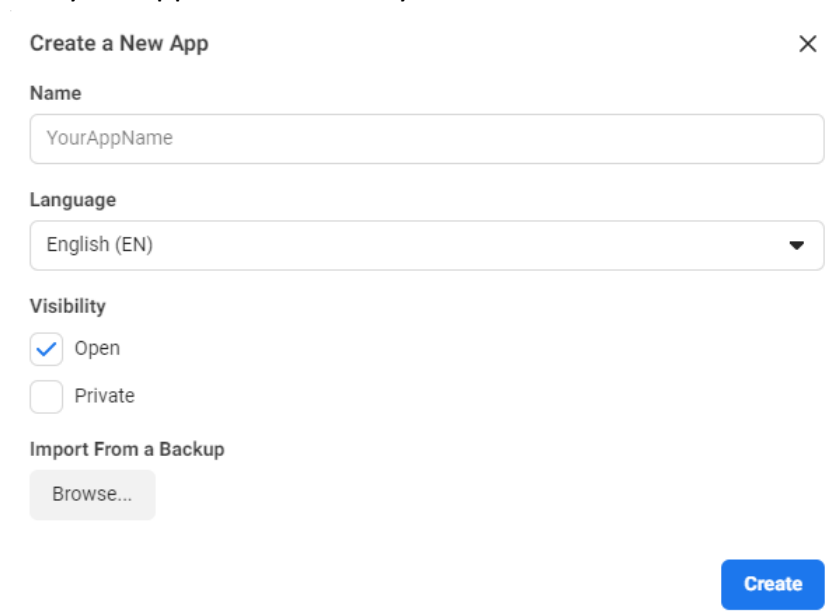


Figure 3.22 : New App's info (Source Wit.ai)

Therefore, in Wit.ai the training process is the following. Insert the user's expected utterance. Then define the intent of that utterance. After, the entities

of that utterance must also be declared. Click the “Train and Validate” button and the training process of the application will start.

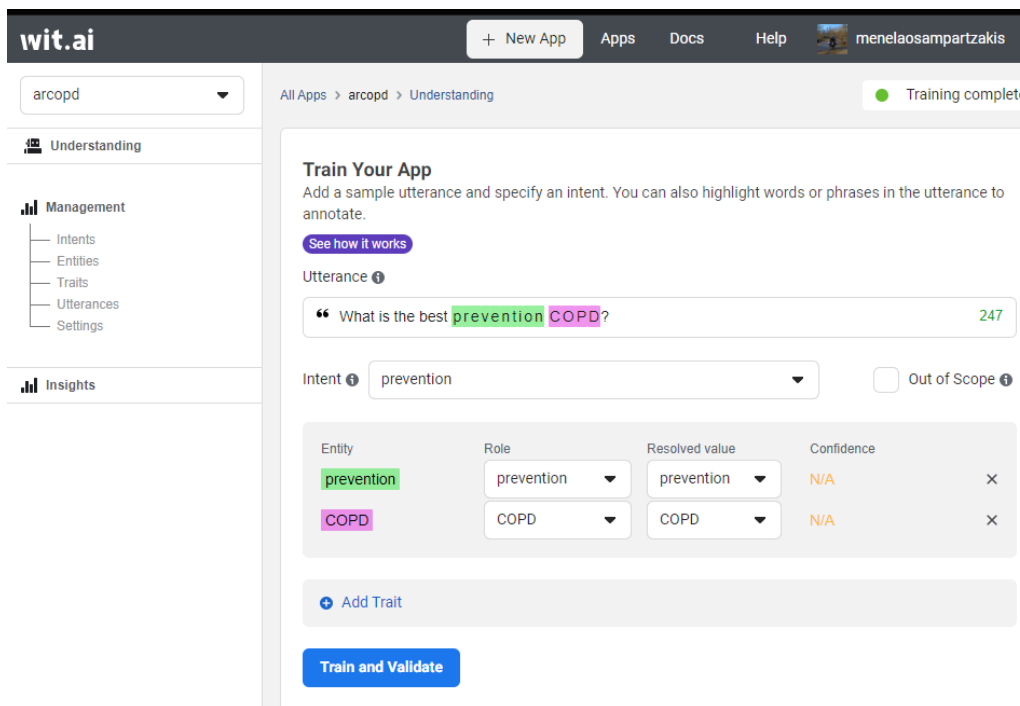


Figure 3.23 : Training the Wit App (Source Wit.ai)

The procedure of training will be continued until chatbot is fully trained and able to understand every utterance of the user. The trained intents & entities of the application are below.

intents	description	Entities
previoustest	Sentences with that intent have as intent to get the previous completed COPD questionnaires of the user	questionnaire:questionnaire, previoustest:previoustest
greeting	Sentences with that intent have as intent to greet user and navigate him through chatbot	
prolipsi	Sentences with that intent have as intent to give user ways to prevent COPD	xap:xap, prolipsi:prolipsi
what_is	Sentences with that intent have as intent to give user information about COPD	tieinaihxp:tieinaihxp, xap:xap, explain:explain
get_xap	Sentences with that intent have as intent to start a new questionnaire for the user	xap:xap, questionnaire:questionnaire

antras	Sentences with that intent have as intent to get gender male	
denapantw	Sentences with that intent have as intent to get gender unknown	
gynaika	Sentences with that intent have as intent to get gender female	

Table 3.2 : Intents & Entities (Source Wit.ai)

Each of the above intents are handled and answered by fixed answers that have been inserted in the chatbot code. Answers are presented below.

intents	Answer
previoustest	<p>If user has completed COPD questionnaire at least once: <i>"Results based on points"</i></p> <p>If user has never completed the COPD questionnaire: <i>Φαίνεται ότι δεν έχετε κάνει το Τεστ ακόμα! Μπορείτε να το δοκιμάσετε είτε πατώντας το κουμπι "Τεστ ΧΑΠ" είτε γράφοντας "Τεστ ΧΑΠ".</i></p>
greeting	<i>Γεια σας! Είμαι ο ΧΑΠbot και είμαι εδώ για να σας ενημερώσω για την Χρόνια Αποφρακτική Πνευμονοπάθεια(Χ.Α.Π.)!</i>
prolipsi	<i>Στη θεραπεία της ΧΑΠ παίζει ρόλο η κατάλληλη φαρμακευτική αγωγή, αλλά και άλλες μη φαρμακευτικές παρεμβάσεις. Η διακοπή του καπνίσματος είναι επιλογή κλειδί στην αντιμετώπιση της ΧΑΠ. Ο εμβολιασμός των ασθενών για τη γρίπη και τον πνευμονιόκοκο είναι ουσιώδους σημασίας παρέμβαση. Η πνευμονική αποκατάσταση παίζει επίσης πολύ βασικό ρόλο γιατί βελτιώνει τα συμπτώματα, βοηθά τον ασθενή να ανταποκρίνεται καλύτερα σωματικά, αλλά και συναισθηματικά, στις καθημερινές του δραστηριότητες και εν τέλει βελτιώνει την ποιότητα ζωής. Οι φαρμακολογικές θεραπείες στη ΧΑΠ παίζουν σημαντικό ρόλο στη μείωση των συμπτωμάτων, στη μείωση της συχνότητας και της σοβαρότητας των παροξύνσεων, στη βελτίωση της αντοχής στην άσκηση και συνολικά στη βελτίωση της ποιότητας ζωής. Η φαρμακευτική αγωγή στη ΧΑΠ συνήθως χορηγείται με εισπνοή και για το λόγο αυτό η τεχνική της εισπνοής από τον ασθενή παίζει ουσιαστικό ρόλο για τη σωστή λήψη της αγωγής. Συνεπώς τόσο η τεχνική εισπνοής όσο και η τακτική λήψη της, σύμφωνα με τις οδηγίες του θεράποντος ιατρού, θα πρέπει να επανεκτιμώνται τακτικά.</i>
what_is	<i>Η Χρόνια Αποφρακτική Πνευμονοπάθεια (ΧΑΠ) είναι μια προοδευτική ασθένεια που επηρεάζει τους αεραγωγούς των</i>

	πνευμόνων. Με την πάροδο του χρόνου καθιστά την αναπνοή όλο και πιο δύσκολη.
get_xap	<p>If user has never completed the COPD questionnaire: <i>Ας ξεκινήσουμε λοιπόν! Το Τεστ αξιολόγησης για τη Χ.Α.Π. αποτελείται από 6 ερωτήσεις ώστε να εκτιμήσουμε την επίδραση που έχει η Χρόνια Αποφρακτική Πνευμονοπάθεια στην καθημερινότητά σας!</i></p> <p>If user has completed COPD questionnaire at least once: <i>Φαίνεται ότι έχετε ξανακάνει το τεστ Χ.Α.Π. από αυτόν τον λογαριασμό! Θα θέλατε να συνεχίσετε σαν κάποιος από τους παρακάτω χρήστες;</i></p>
antras	<p>Occurs only inside questionnaire. Will explained in a later chapter. <i>Τέλεια, ας περάσουμε στο ερωτηματολόγιο! Ερώτηση 1:.....</i></p>
denapantw	<p>Occurs only inside questionnaire. Will explained in a later chapter. <i>Τέλεια, ας περάσουμε στο ερωτηματολόγιο! Ερώτηση 1:.....</i></p>
gynaika	<p>Occurs only inside questionnaire. Will explained in a later chapter. <i>Τέλεια, ας περάσουμε στο ερωτηματολόγιο! Ερώτηση 1:.....</i></p>

Table 3.3 : Intents & Answers (Source Wit.ai)

Now that our application’s training is finished, lead us to the next step of connection. To be able to use our trained application of Wit in our code, a token is generated to be used as a connection string. It’s available in Wit’s Application “Settings” Menu. For our application, the “Client Access Token” will be used.

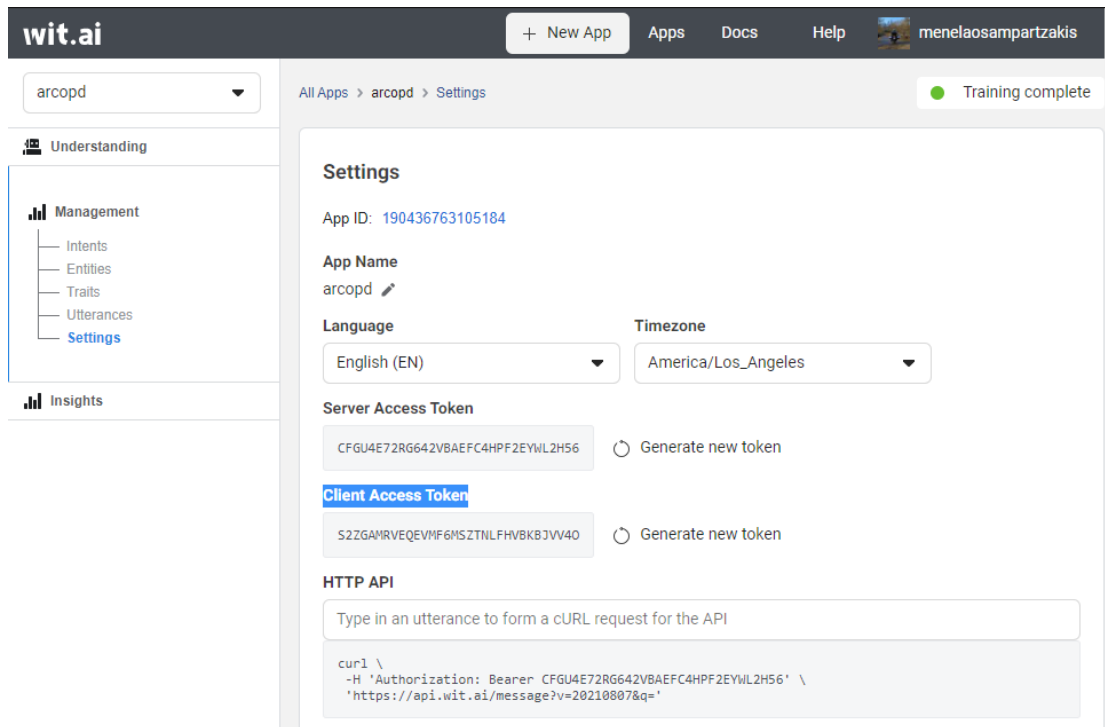


Figure 3.24 : Token of the Wit App (Source Wit.ai)

That token will be used as a connection string to let our application communicate with the Python code.

```
from wit import Wit

#Connection to Wit.ai
wit_access_token = "S2ZGAMRVEQEVMF6MSZTNLFHVBKBJVV40"
client = Wit(access_token = wit_access_token)
```

Figure 3.25 : Python Library for Wit.ai

Class `Wit` gets the “*Client Access Token*” of our application in order to use our trained *NLP* model. Function `message()` gets the user’s utterance as argument, extracts the intents and entities of the user’s sentence based on our Wit Application and returns it in a JSON format. A sample context of the `GET message` function parsing the utterance “*Πρόληψη για την ΧΑΠ*” (*Prevention of COPD in Greek*) is the above.

```

{'entities': {'prolipsi:prolipsi': [{'body': 'Πρόληψη',
                                     'confidence': 0.9208,
                                     'end': 7,
                                     'entities': [],
                                     'id': '274956244024247',
                                     'name': 'prolipsi',
                                     'role': 'prolipsi',
                                     'start': 0,
                                     'type': 'value',
                                     'value': 'Πρόληψη'}],
             'xap:xap': [{'body': 'ΧΑΠ',
                           'confidence': 0.9569,
                           'end': 19,
                           'entities': [],
                           'id': '400676767769860',
                           'name': 'xap',
                           'role': 'xap',
                           'start': 16,
                           'type': 'value',
                           'value': 'ΧΑΠ'}]},
'intents': [{'confidence': 0.8188,
             'id': '289221679276847',
             'name': 'prolipsi'}],
'text': 'Πρόληψη για την ΧΑΠ',
'traits': {}}

```

Figure 3.26 : Response JSON of a Request (Source Wit.ai)

Returned JSON contains all the necessary information to understand what the user wants to say to chatbot. Sentence “Πρόληψη για την ΧΑΠ” contains 2 entities that were created during training. The first entity is “Πρόληψη” (*Prevention*) and the second is “ΧΑΠ” (*COPD*). Each body of the entity appears in the JSON and the confidence metric. Finally, it contains 1 intent which is *prevention* with confidence 0.8188.

3.4. Questionnaire Flow & Methods

In our chatbot application, the user will have the capability to take a COPD questionnaire to learn about the COPD effect in his/her life apart from asking regular questions that are answered through regular and fixed answers from trained intents in Wit. Questionnaire consists of 10 questions. Some questions accept limited answers. Questions and their limitation are presented below. Questions are ordered to represent the flow of the questionnaire for a new user.

	Questions	Answer Limitations
Question 1	What's your name?	No Limit
Question 2	What's your surname?	No Limit
Question 3	What's your weight in kilograms?	Integer value from 30 to 300
Question 4	What's your gender?	<i>Man, Woman or No Answer.</i> These answers have been trained as intents and similar answers are also accepted. Can be also inserted as answers from buttons.
Question 5	From 1 to 10, how often do you cough? (1 = Never & 10 = Always)	Integer value from 1 to 10
Question 6	From 1 to 10, do you often feel tightness in the chest? (1 = Never & 10 = Always)	Integer value from 1 to 10
Question 7	From 1 to 10, do you gasp when climbing stairs? (1 = Never & 10 = Always)	Integer value from 1 to 10
Question 8	From 1 to 10, do you feel weakness during the day? (1 = No Weakness & 10 = Total weakness)	Integer value from 1 to 10

Question 9	From 1 to 10, do you feel insecure when you leave home because of your illness? (1 = Absolutely Secure & 10 = Totally insecure)	Integer value from 1 to 10
Question 10	From 1 to 10, how restful is your sleep? (1 = Absolutely calm & 10 = Totally anxious)	Integer value from 1 to 10

Table 3.4 : Questions of Questionnaire

Every time a user starts taking a COPD test, a document is inserted in the MongoDB database. That document has the following format:

```

new_questionnaire = {
  "user_id": sender_id,
  "start_xap": "yes",
  "init_check": "not exist",
  "name": "NA",
  "surname": "NA",
  "weight": "NA",
  "gender": "NA",
  "q1": "NA",
  "q2": "NA",
  "q3": "NA",
  "q4": "NA",
  "q5": "NA",
  "q6": "NA"
}

```

Figure 3.27 : Inserted Document in MongoDB

After the successful insertion of the above document, every time chatbot receive valid answer (based on the answers limitations) in each question, it proceeds to next question. Also, each time a valid answer is received the counterpart field of the document is updated with the answer.

The *“user_id”* field contains the user’s Facebook ID. *“Start_xap”* field indicates if the user has started the COPD test and it’s filled with *“yes”* or *“no”*. Next is *“Init_check”* field. The purpose of that field, is to check if the user has completed the COPD test ever before to autofill some of his personal information. If the user has completed the test at least once, then its value will

be filled with “*exist*” during the insertion, otherwise with “*not exist*”. In the “*exist*” case, the user must choose to continue either with one of the first name and last name that has completed the COPD test or continue as a new user. If the user continues with one of the previous first and last name, then the fields “*name*”, “*surname*” and “*gender*” are filled from the fields of the previous COPD test and the field “*Init_check*” turns to “*not exist*” to continue the questionnaire. In case that the user selects to continue as a new user, the field “*Init_check*” turns to “*not exist*” once again. In that way, multiple users can take the COPD test from the same Facebook Account. The fields “*name*”, “*surname*”, “*weight*” and “*gender*” are the user’s first name, last name, weight and gender respectively. The fields “*q1*” to “*q6*” are filled from the inputs of the user regarding Question 1 to Question 6.

After the successful completion of the COPD questionnaire, a new document is inserted in the database that has the following format and the old one is deleted.

```
_id: ObjectId("61083901e2fe221b3455eba5")
results: "results"
user_id_res: "3481400398647377"
name: "μΕν"
surname: "αμτ"
weight: 56
gender: "male"
points: 29
q1: 5
q2: 4
q3: 5
q4: 6
q5: 3
q6: 6
tr_timestamp: "02-08-2021 18:27:13"
```

Figure 3.28 : Final Document that is inserted

The field “*_id*” is the default ID field that Mongo pass to every document. The “*user_id_res*” field contains the user’s Facebook ID. The field “*points*” contains the sum of points of q1 to q6. The “*tr_timestamp*” contains the timestamp in GMT+2:00 (time zone of Greece) that lie during the completion of the COPD test. The rest fields are same as the first inserted document.

In case where the user selects to continue as an existing user then the flow of the questionnaire is limited to the Questions 3,6,7,8,9,10 of the above table (**Table 3.4**) since the rest fields are auto filled from a previous test.

The only questions where the answer can be predicted is the one regarding gender. For that reason, we trained our chatbot application with multiple inputs regarding gender, so it can also accept and understand the intent of text except from buttons. The rest questions can be answered with infinite answers. For the questions that they don't have a limitation, like name and surname it's not necessary to check the input of the user. But for the questions regarding weight and q1 to q6 there are some limitations. Those limitations **can only be checked** by scanning the raw text of the response JSON. For example, for the question *"What's your weight in kilograms?"* the answer must be an integer number from 1 to 10. The answer will be included in the field *"text"* of the response JSON (see picture JSON above **Figure 3.26**). If that field contains a text different than a number from 1 to 10 then the answer is considered as invalid and the user must answer that question again.

After the successful completion of the questionnaire, chatbot will warn user about the effect of COPD in user's life and inform user on how to prevent it. Moreover, after the completion of 5 COPD tests from the same user (same name and last name), chatbot will provide some statistical information about the user's progression. Finally, chatbot will make a prediction about the user's points of the 5th COPD test. If prediction points are between +-7 points of user's 5th actual COPD test, prediction will be considered as successful and chatbot will explain user the result of that prediction. If prediction exceeds threshold of +-7 points, prediction will be considered as failed and chatbot will explain user the result of that prediction. That's the logical flow of the questionnaire.

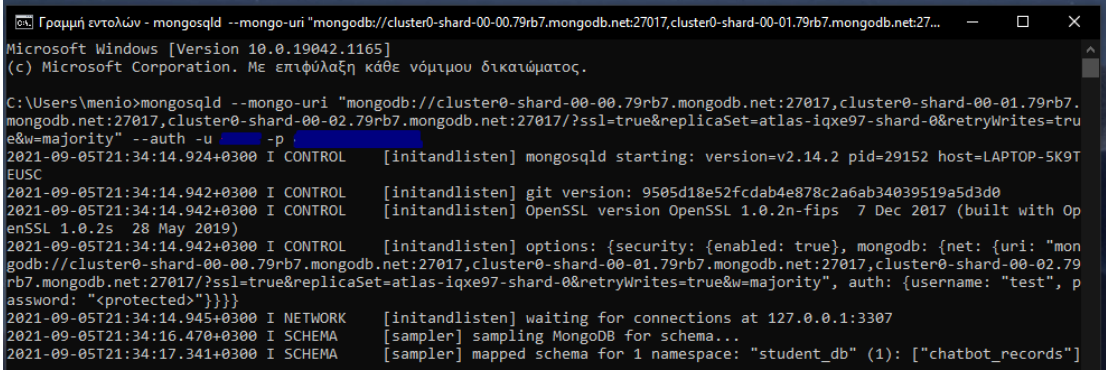
3.5. Tableau Dashboards & Connection

To retrieve the data of chatbot, create the necessary Dashboards and monitor the users, a live connection between Tableau and MongoDB must be established. that reason, an ODBC driver was created using the MongoDB Connector. To enable the connection, open a CLI and enter the following command.

```
mongosqld --mongo-uri "mongodb://cluster0-shard-00-00.79rb7.mongodb.net:27017,cluster0-shard-00-01.79rb7.mongodb.net:27017,cluster0-shard-00-02.79rb7.mongodb.net:27017/?ssl=true&replicaSet=atlas-iqxe97-shard-0&retryWrites=true&w=majority" --auth -u username -p password
```

- mongosqld will start the connection of localhost with the MongoDB Atlas Database.
- --mongo-uri takes as argument the URI string that was taken from the connection string of Node.js except username and password in the beginning. These 2 parameters will be filled in the end with the --auth
- --auth gets as argument the username and password

The CLI will look like this:



```
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. Με επιφύλαξη κάθε νόμιμου δικαιώματος.

C:\Users\menio>mongosqld --mongo-uri "mongodb://cluster0-shard-00-00.79rb7.mongodb.net:27017,cluster0-shard-00-01.79rb7.
mongodb.net:27017,cluster0-shard-00-02.79rb7.mongodb.net:27017/?ssl=true&replicaSet=atlas-iqxe97-shard-0&retryWrites=tru
e&w=majority" --auth -u [REDACTED] -p [REDACTED]
2021-09-05T21:34:14.924+0300 I CONTROL [initandlisten] mongosqld starting: version=v2.14.2 pid=29152 host=LAPTOP-5K9T
EUSC
2021-09-05T21:34:14.942+0300 I CONTROL [initandlisten] git version: 9505d18e52fcdab4e878c2a6ab34039519a5d3d0
2021-09-05T21:34:14.942+0300 I CONTROL [initandlisten] OpenSSL version OpenSSL 1.0.2n-fips 7 Dec 2017 (built with Op
enSSL 1.0.2s 28 May 2019)
2021-09-05T21:34:14.942+0300 I CONTROL [initandlisten] options: {security: {enabled: true}, mongodb: {net: {uri: "mon
godb://cluster0-shard-00-00.79rb7.mongodb.net:27017,cluster0-shard-00-01.79rb7.mongodb.net:27017,cluster0-shard-00-02.79
rb7.mongodb.net:27017/?ssl=true&replicaSet=atlas-iqxe97-shard-0&retryWrites=true&w=majority", auth: {username: "test", p
assword: "<protected>"}}}
2021-09-05T21:34:14.945+0300 I NETWORK [initandlisten] waiting for connections at 127.0.0.1:3307
2021-09-05T21:34:16.470+0300 I SCHEMA [sampler] sampling MongoDB for schema...
2021-09-05T21:34:17.341+0300 I SCHEMA [sampler] mapped schema for 1 namespace: "student_db" (1): ["chatbot_records"]
```

Figure 3.29 : Connection to MongoDB for Tableau

The connection between database and localhost is now active.

Having the CLI running, open the Tableau application, click to connect to a server and select *Other Databases (ODBC)*. Select DNS and click on the name of our connection i.e. *“My Connection”*. Click *“Connect”* and *“Sign In”* and the Tableau is now connected with on. Select our database named *“student_db”* and select the table *“chatbot_records”*. Double click on the *“New Custom SQL”* and input the query ***SELECT * FROM student_db.chatbot_records.*** That query will give us all the records of the table.

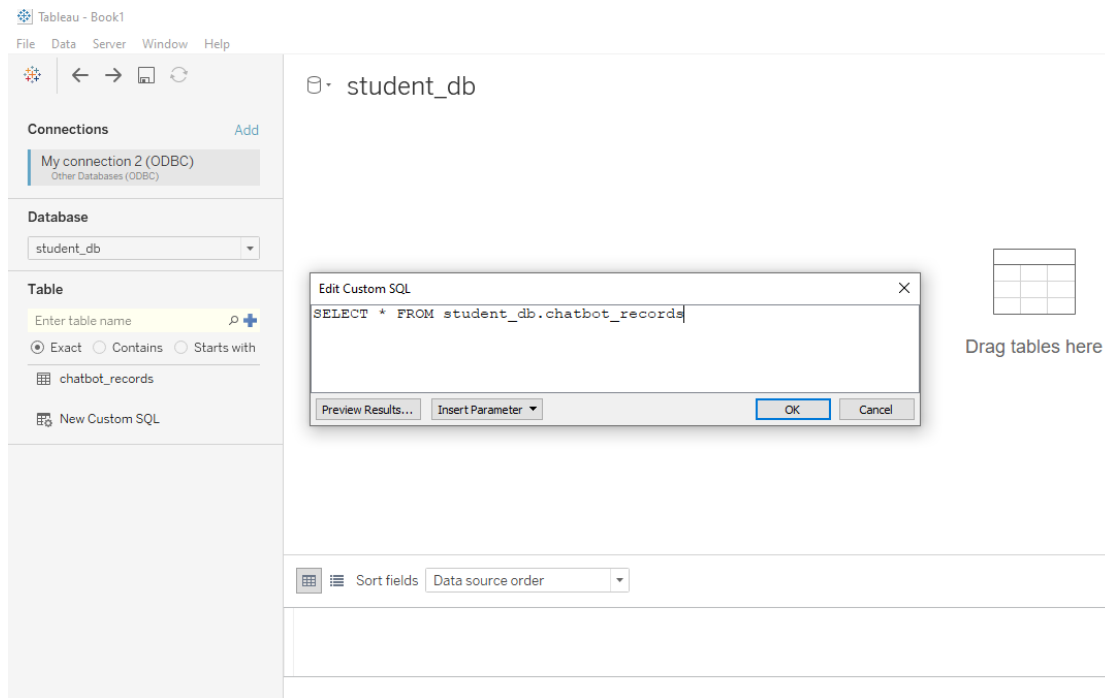


Figure 3.30 : Query to get all data from MongoDB to Tableau

We are now ready to proceed with the creation of worksheets.

The 1st Worksheet refers to the *Total COPD Tests*.

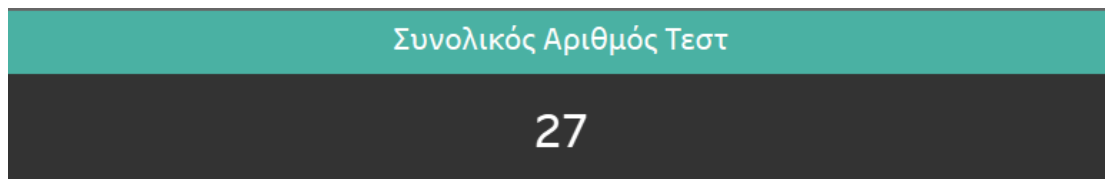


Figure 3.31 : Total Number of COPD Tests Card

The 2nd Worksheet refers to the *Number of Unique Users*.

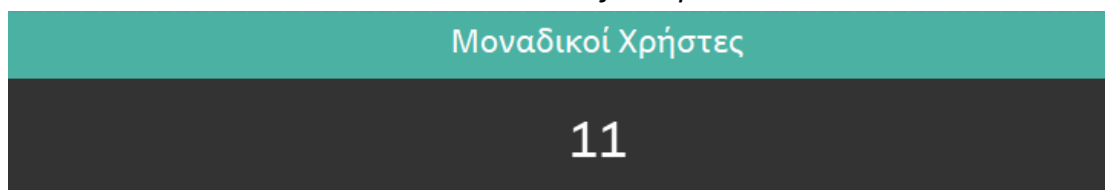


Figure 3.32 : Number of Unique Users Card

The 3rd Worksheet refers to the *Total COPD Tests per Result*.

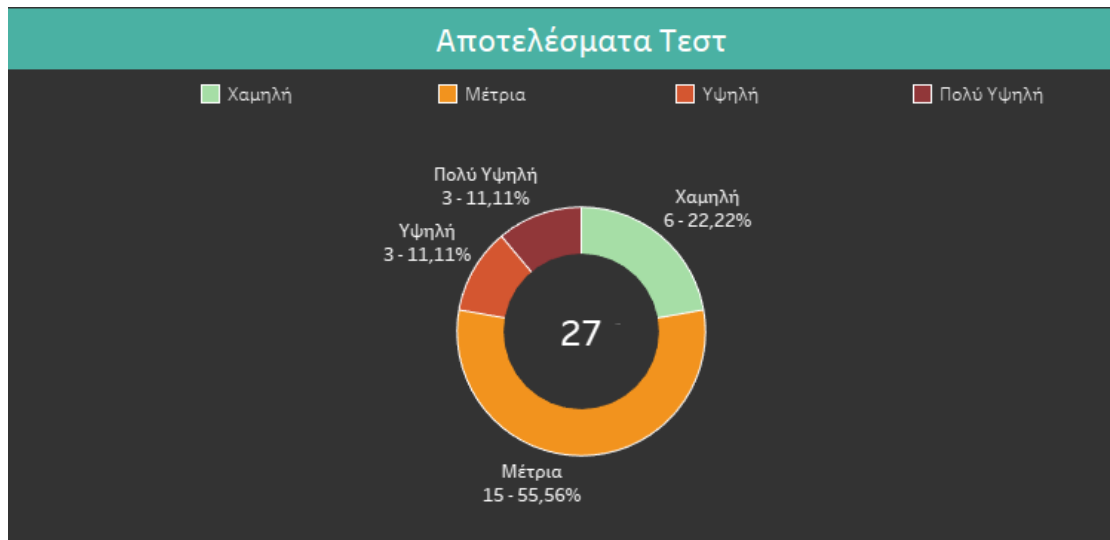


Figure 3.33 : Results Pie

The 4th Worksheet refers to the *AVG Points per Question*.

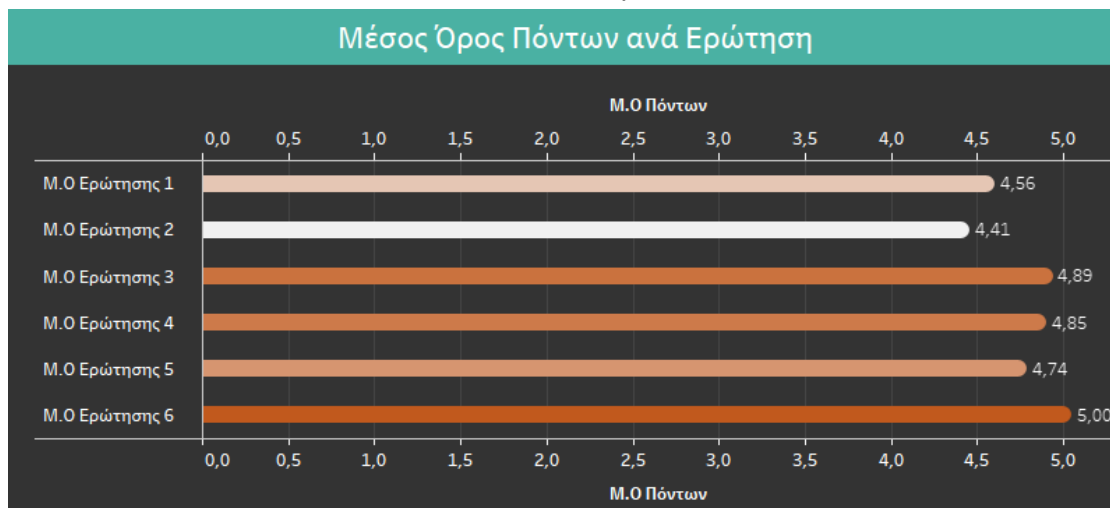


Figure 3.34 : AVG Points per Question

The 5th Worksheet refers to the *Number of COPD Tests per Date*.

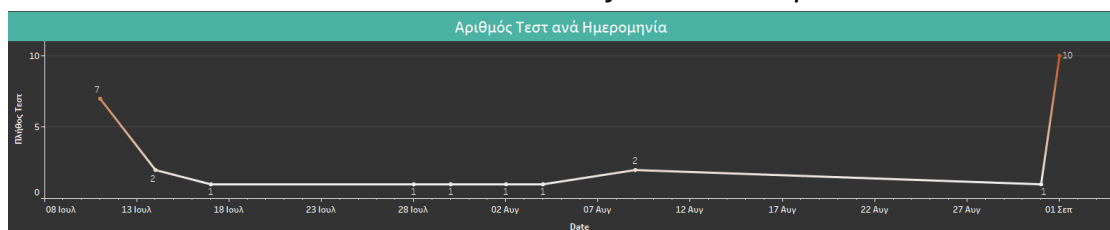


Figure 3.35 : Number of COPD Tests per Day

We are now ready to bind all Workbooks together to a Dashboard. It will contain filters like Date, Time, First Name & Last Name and Results. Filters will help us track better the information we need. More filters and parameters were about to be added but the current ODBC driver cannot support them. The final Dashboard looks like this.

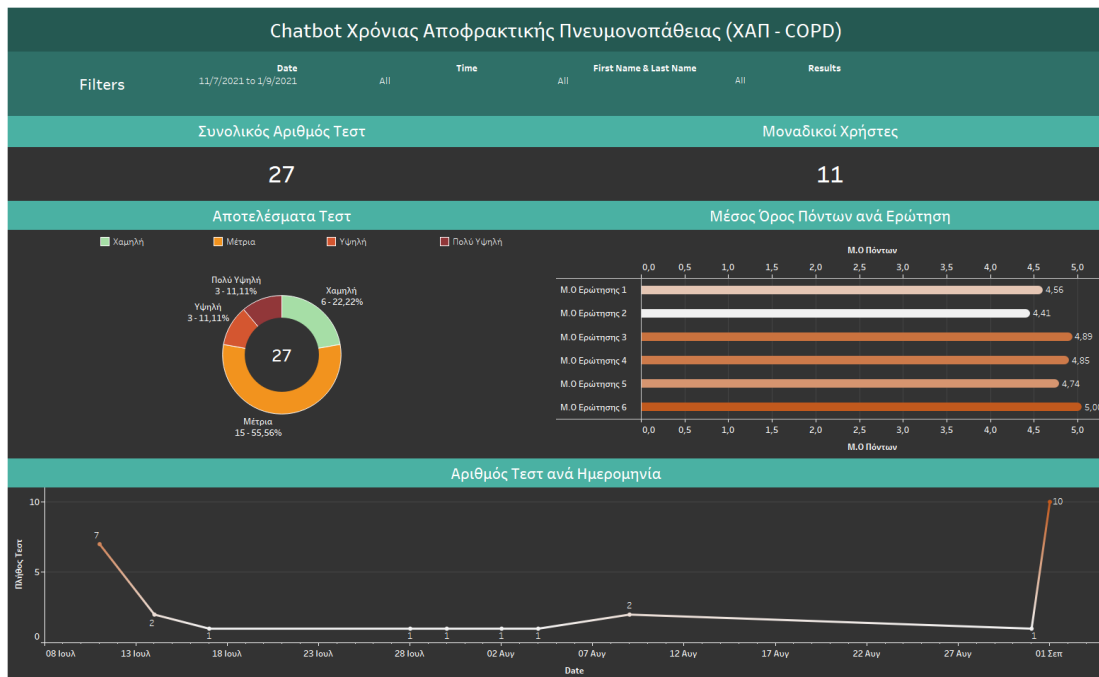


Figure 3.36 : Tableau Dashboards

For a user with no programming skills it's difficult to run the necessary commands and open the Tableau report. For that reason, a new .bat file was created. Batch files (.bat) consists of a series of commands to be executed by the command-line interpreter, stored in a plain text file. A new file was created with the Tableau file and the .bat file. We first opened a notepad, insert the following commands and the notepad was saved with extension *.bat*.

```
start mongosql --mongo-uri "mongodb://cluster0-shard-00-00.79rb7.mongodb.net:27017,cluster0-shard-00-01.79rb7.mongodb.net:27017,cluster0-shard-00-02.79rb7.mongodb.net:27017/?ssl=true&replicaSet=atlas-iqxe97-shard-0&retryWrites=true&w=majority" --auth -u username -p password
start C:\Users\menio\Desktop\thesis\Book1.twb
```

As a result, by double clicking the .bat file the above scripts will executed, and the Tableau Dashboard will open.

3.6. Heroku Setting & Deployment

To build the project's folder create a file named *facebook-messenger-chatbot*. Open a CLI and run the command *virtualenv myenv* to create the isolated environment of the application.

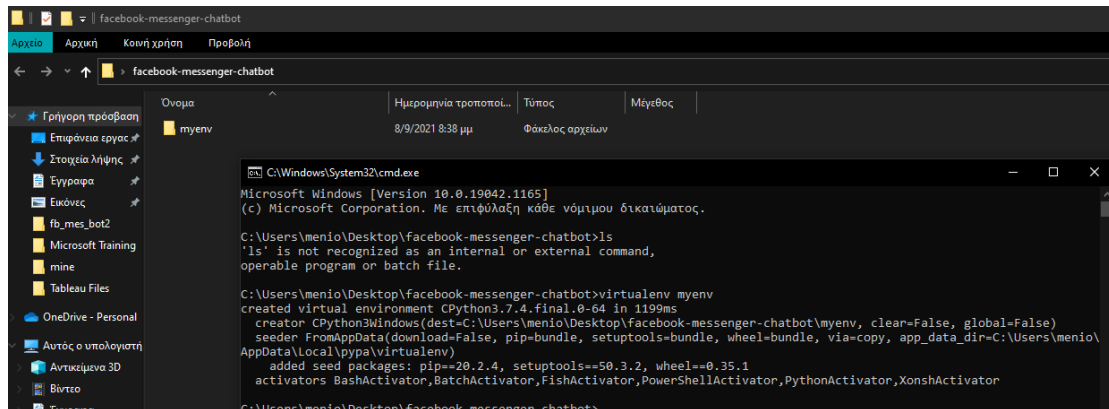


Figure 3.37 : Creation of Project's file

After the successful creation a file named *myenv* will appear in the folder. To activate the virtual environment run the command *myenv/Scripts/activate*.

Use the command *pip install pymessenger flask* to install all the necessary libraries of the project of the code including *unicorn*.

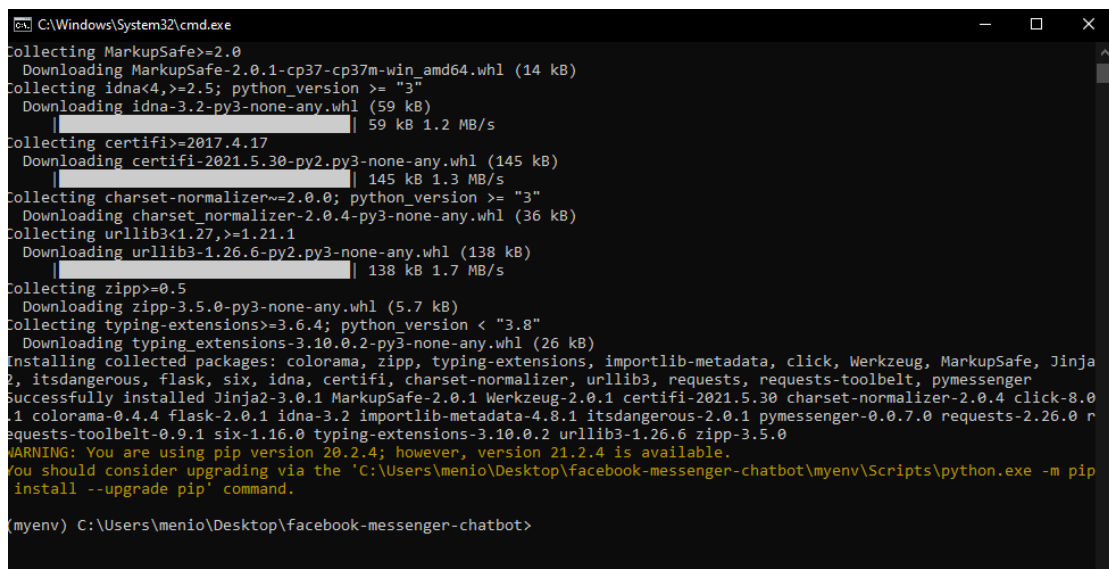


Figure 3.38 : Installation of necessary Python libraries

Create a python file named *app.py* and insert the code needed to run the application. That file will hold the code of the application.

Next, run the command *pip freeze > requirements.txt*. That command will return all dependencies of the application and their version.

```

requirements.txt - Σημειωματάριο
Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια
certifi==2020.6.20
chardet==3.0.4
click==7.1.2
dnspython==2.0.0
Flask==1.1.2
gunicorn==20.0.4
idna==2.10
itsdangerous==1.1.0
Jinja2==2.11.2
MarkupSafe==1.1.1
numpy==1.18.4
prompt-toolkit==3.0.8
pymessenger==0.0.7.0
pymongo==3.11.0
requests==2.24.0
requests-toolbelt==0.9.1
six==1.15.0
urllib3==1.25.10
wcwidth==0.2.5
Werkzeug==1.0.1
wit==6.0.0

```

Figure 3.39 : Needed Python Libraries

After, we need to create a Procfile. Procfile will tell Heroku which file contains the flask app and thus which script to run first. To create it run the command `echo > Procfile`. Edit the created Procfile by removing all contents and filling it with the command `web: gunicorn app:app`.

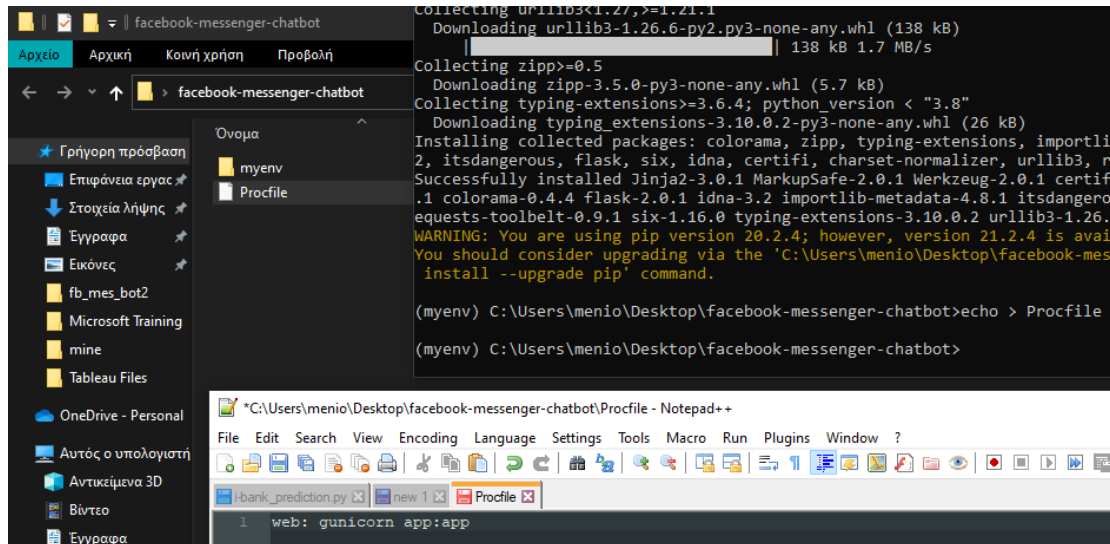


Figure 3.40 : Procfile Creation

A git repository must be made for that project and so, run the command `git init`. Running the command `git status`, one of the files that seems to be untracked is `myenv/`. But since the `requirements.txt` file is also included `myenv/` should not be added. So the next command is `echo > .gitignore` which creates the file `gitignore`. In that file remove all contents and add the text `myenv/`. Everything on that folder will be ignored from being added by git.

Next step is to add the files running the command `git add .` in the command prompt and then commit the added files with the command `git commit -m "ourproject"`.

It remains to push the application. But to use Heroku it's necessary to create an app and then push it in that app. To create a Heroku app run the command `heroku create`. A colored name will appear in the command prompt that is the name of the heroku app.

```
myenv) C:\Users\menio\Desktop\facebook-messenger-chatbot>heroku create
Warning: heroku update available from 7.45.0 to 7.50.0.
Creating app... done, ⬢ floating-woodland-66551
https://floating-woodland-66551.herokuapp.com/ | https://git.heroku.com/floating-woodland-66551.git
myenv) C:\Users\menio\Desktop\facebook-messenger-chatbot>
```

Figure 3.41 : Heroku App Creation

Now we need to tell git repository what the URL of the git repository of the heroku app is. To do that, run the command `heroku git:remote -a name_of_the_heroku_app`. Finally, run the command `git push heroku master` to push the commits on the Heroku app.

Log in Heroku in the app's page. Click on the app's name and then on the *Open app* in top of page. If the deployment was successful, the set message of the Webhook will appear on the top of that page.

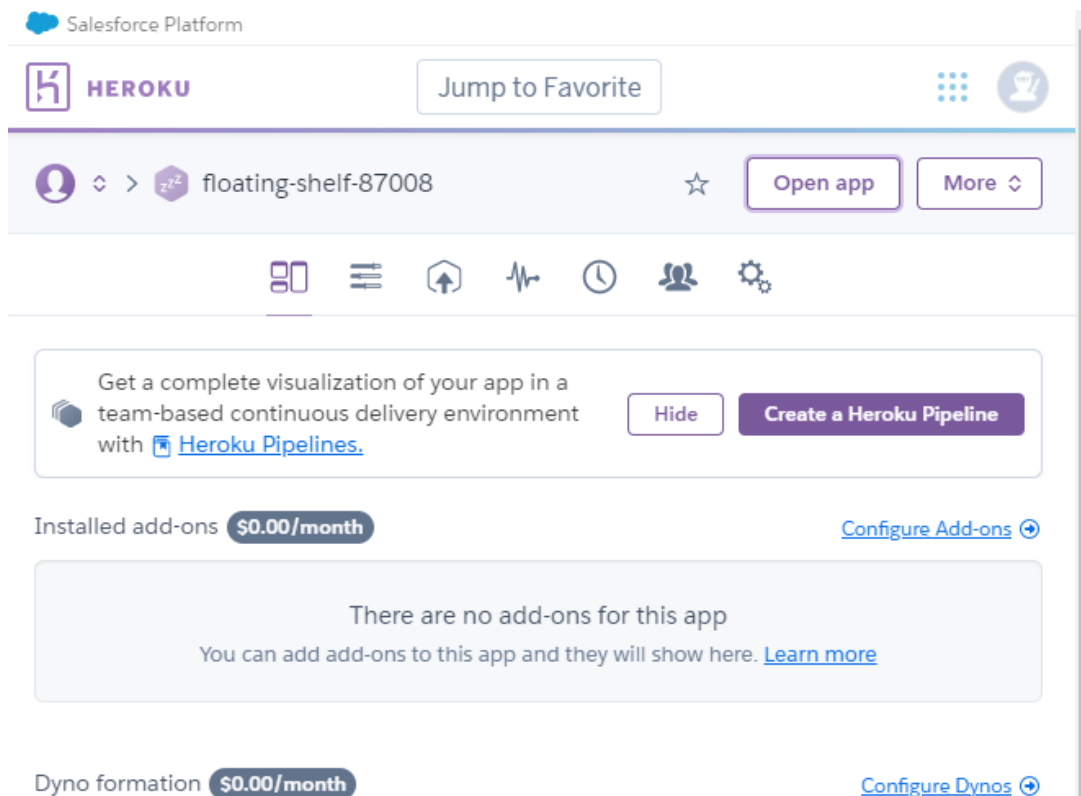


Figure 3.42 : Deployed Heroku App (Source heroku.com)

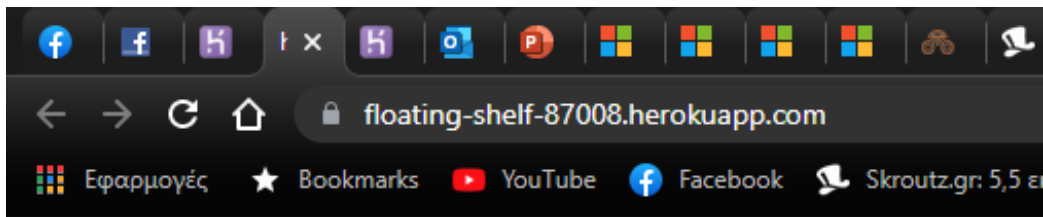


Figure 3.43 : Message of Webhook (Source heroku.com)

To monitor and debug the application click on the *More* button and *View Logs*. Each time an action happens in the chatbot, the logs will be updated with the respective actions.

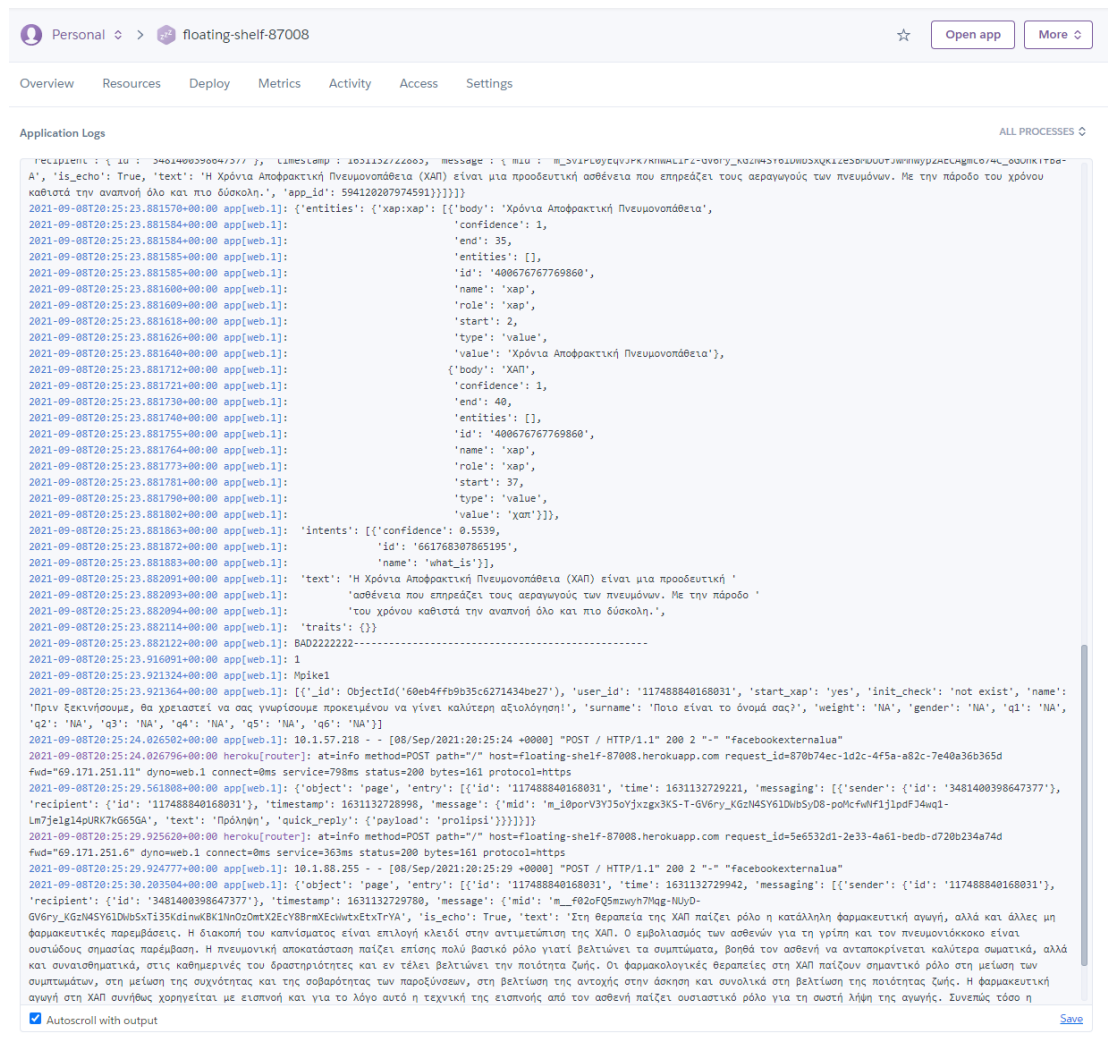


Figure 3.44 : Logs of the App via Heroku (Source heroku.com)

The URL of the application must be completed in the Application’s Developer’s Facebook page in the Messenger Settings as mentioned in the Facebook app configuration chapter.

Webhooks

To receive messages and other events sent by Messenger users, the app should enable webhooks integration.

Callback URL Verify Token

Validation requests and Webhook notifications for this object will be sent to this URL. Token that Facebook will echo back to you as part of callback URL verification.

[Edit Callback URL](#) [Show Recent Errors](#)

Figure 3.45 : Heroku page on Facebook App (Source developers.facebook.com)

The chatbot application is now ready and running.

4. Results

4.1. Expected Chatbot Conversation

Since the application is now active and in public mode, the users can now use the chatbot. The conversation between a user and the chatbot is displayed below. The Chatbot's Facebook page is located at the URL <https://www.facebook.com/Wit-test2-117488840168031/>

Click on the button *Message* to open the Starting page.

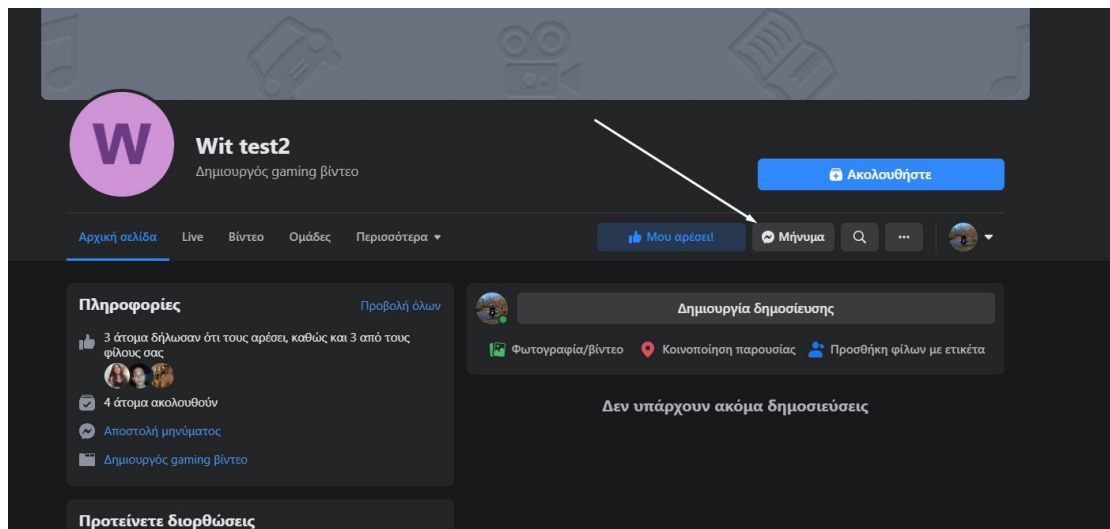


Figure 4.1 : Navigation on Facebook's Page (Source facebook.com)

← Wit test2



Wit test2

Δημιουργός gaming βίντεο

Ο χρήστης Αγγελική Γιούλη και 2 ακόμη φίλοι δήλωσαν ότι τους αρέσει αυτή τη Σ...

Όταν πατήσετε ΞΕΚΙΝΗΣΤΕ, ο χρήστης Wit test2 θα δει τις δημόσιες πληροφορίες σας.

ΞΕΚΙΝΗΣΤΕ



Figure 4.2 : Sample Conversation (1)



Wit test2

Ο χρήστης Αγγελική Γιούλη και 2 ακόμη φίλοι δήλωσαν
ότι τους αρέσει αυτή τη Σελίδα

Δημιουργός gaming βίντεο

ΠΡΟΒΟΛΗ ΠΡΟΦΙΛ

23:29

Ας ξεκινήσουμε

Γεια σας! Είμαι ο ΧΑΠbot και είμαι
εδώ για να σας ενημερώσω για την
Χρόνια Αποφρακτική
Πνευμονοπάθεια(Χ.Α.Π.)!

Μπορείτε να με ρωτήσετε
πληροφορίες για τη Χ.Α.Π. καθώς και
να κάνετε το Τεστ Χ.Α.Π. και να δείτε
τα δικά σας αποτελέσματα!



Πληροφορίες



Τεστ ΧΑΠ



Προηγούμε



Αα



Figure 4.3 : Sample Conversation (2)



W

Wit test2



ότι τους αρέσει αυτή τη Σελίδα

Δημιουργός gaming βίντεο

ΠΡΟΒΟΛΗ ΠΡΟΦΙΛ

23:30

Ξεκινήστε

Γεια σας! Είμαι ο ΧΑΠbot και είμαι
εδώ για να σας ενημερώσω για την
Χρόνια Αποφρακτική
Πνευμονοπάθεια(Χ.Α.Π.)!

W

Μπορείτε να με ρωτήσετε
πληροφορίες για τη Χ.Α.Π. καθώς και
να κάνετε το Τεστ Χ.Α.Π. και να δείτε
τα δικά σας αποτελέσματα!

Πληροφορίες

W

Η Χρόνια Αποφρακτική
Πνευμονοπάθεια (ΧΑΠ) είναι μια
προοδευτική ασθένεια που
επηρεάζει τους αεραγωγούς των
πνευμόνων. Με την πάροδο του
χρόνου καθιστά την αναπνοή όλο και
πιο δύσκολη.

W



Τεστ ΧΑΠ



Πρόληψη



Προηγούμενα Τ



Αα



Figure 4.4 : Sample Conversation (3)



W

Wit test2



αλλά και άλλες μη φαρμακευτικές παρεμβάσεις. Η διακοπή του καπνίσματος είναι επιλογή κλειδί στην αντιμετώπιση της ΧΑΠ. Ο εμβολιασμός των ασθενών για τη γρίπη και τον πνευμονιόκκοκο είναι ουσιώδους σημασίας παρέμβαση. Η πνευμονική αποκατάσταση παίζει επίσης πολύ βασικό ρόλο γιατί βελτιώνει τα συμπτώματα, βοηθά τον ασθενή να ανταποκρίνεται καλύτερα σωματικά, αλλά και συναισθηματικά, στις καθημερινές του δραστηριότητες και εν τέλει βελτιώνει την ποιότητα ζωής. Οι φαρμακολογικές θεραπείες στη ΧΑΠ παίζουν σημαντικό ρόλο στη μείωση των συμπτωμάτων, στη μείωση της συχνότητας και της σοβαρότητας των παροξύνσεων, στη βελτίωση της αντοχής στην άσκηση και συνολικά στη βελτίωση της ποιότητας ζωής. Η φαρμακευτική αγωγή στη ΧΑΠ συνήθως χορηγείται με εισπνοή και για το λόγο αυτό η τεχνική της εισπνοής από τον ασθενή παίζει ουσιαστικό ρόλο για τη σωστή λήψη της αγωγής. Συνεπώς τόσο η τεχνική εισπνοής όσο και η τακτική λήψη της, σύμφωνα με τις οδηγίες του θεράποντος ιατρού, θα πρέπει να επανεκτιμώνται τακτικά.

W

W



Πληροφορίες



Τεστ ΧΑΠ



Προηγούμε



Aa



Figure 4.5 : Sample Conversation (4)



W

Wit test2



Προηγούμενα Τεστ

Οι προηγούμενοι πόντοι σας, ήταν 21!
Τα αποτελέσματα για αυτούς τους πόντους είναι τα εξής:

Επίδραση: Μέτρια

Γενική Εικόνα: Η ΧΑΠ είναι ένα από τα σημαντικότερα προβλήματα που αντιμετωπίζετε. Κάποιες μέρες της εβδομάδας είναι καλές, αλλά τις περισσότερες βήχετε, ενώ περνάτε κρίσεις επιδείνωσης 1-2 φορές το χρόνο. Τις περισσότερες μέρες δυσκολεύεστε να αναπνεύσετε και συχνά ξυπνάτε με σφίξιμο στο στήθος ή δύσπνοια. Όταν κάνετε οικιακές εργασίες χρειάζεστε συχνά διαλείμματα.

Προτείνεται να:

- κόψετε το κάπνισμα
- εμβολιάζεστε κάθε χρόνο κατά της γρίπης
- αποφεύγετε την έκθεση σε παράγοντες που μπορεί να επιδεινώσουν την κατάστασή σας
- ακολουθήσετε θεραπεία, μετά από ιατρική εξέταση και επιπλέον, σε συνεργασία με τον ιατρό σας, να:
- επανεξετάσετε τη θεραπεία που ακολουθείτε



Aa



Figure 4.6 : Sample Conversation (5)



W

Wit test2



Τεστ ΧΑΠ

Φαίνεται ότι έχετε ξανακάνει το τεστ Χ.Α.Π. από αυτόν τον λογαριασμό! Θα θέλατε να συνεχίσετε σαν κάποιος από τους παρακάτω χρήστες;

1: MEN2 AMP2

2: MEN3 AMP3

3: gh gh

4: test1 test1

5: tyyt tytytyt

6: Μενεάπς Δηγηφ

7: μεν αμπ

8: μυν μυν

9: σδσδφδσδσδ δαφσδδσφδσ

Εάν ναι, παρακαλούμε δώστε τον αριθμό που αντιστοιχεί στον επιθυμητό χρήστη, διαφορετικά πατήστε το κουμπί 'Νέος Χρήστης'.



Νέος Χρήστης



Τερματισμός Τεστ



Αα



Figure 4.7 : Sample Conversation (6)



W

Wit test2



5: tyyt tytytyt

6: Μενεάπς Δηγφ

7: μεν αμπ

8: μυν μυν

9: σδσδφδσδσδ δαφσδδσφδσ

Εάν ναι, παρακαλούμε δώστε τον αριθμό που αντιστοιχεί στον επιθυμητό χρήστη, διαφορετικά πατήστε το κουμπί 'Νέος Χρήστης'.

W

Νέος Χρήστης

Ας ξεκινήσουμε λοιπόν! Το Τεστ αξιολόγησης για τη Χ.Α.Π. αποτελείται από 6 ερωτήσεις ώστε να εκτιμήσουμε την επίδραση που έχει η Χρόνια Αποφρακτική Πνευμονοπάθεια στην καθημερινότητά σας.!

Πριν ξεκινήσουμε, θα χρειαστεί να σας γνωρίσουμε προκειμένου να γίνει καλύτερη αξιολόγηση!

W

Ποιο είναι το όνομά σας;

W

✕ Τερματισμός Τεστ



Aa



Figure 4.8 : Sample Conversation (7)

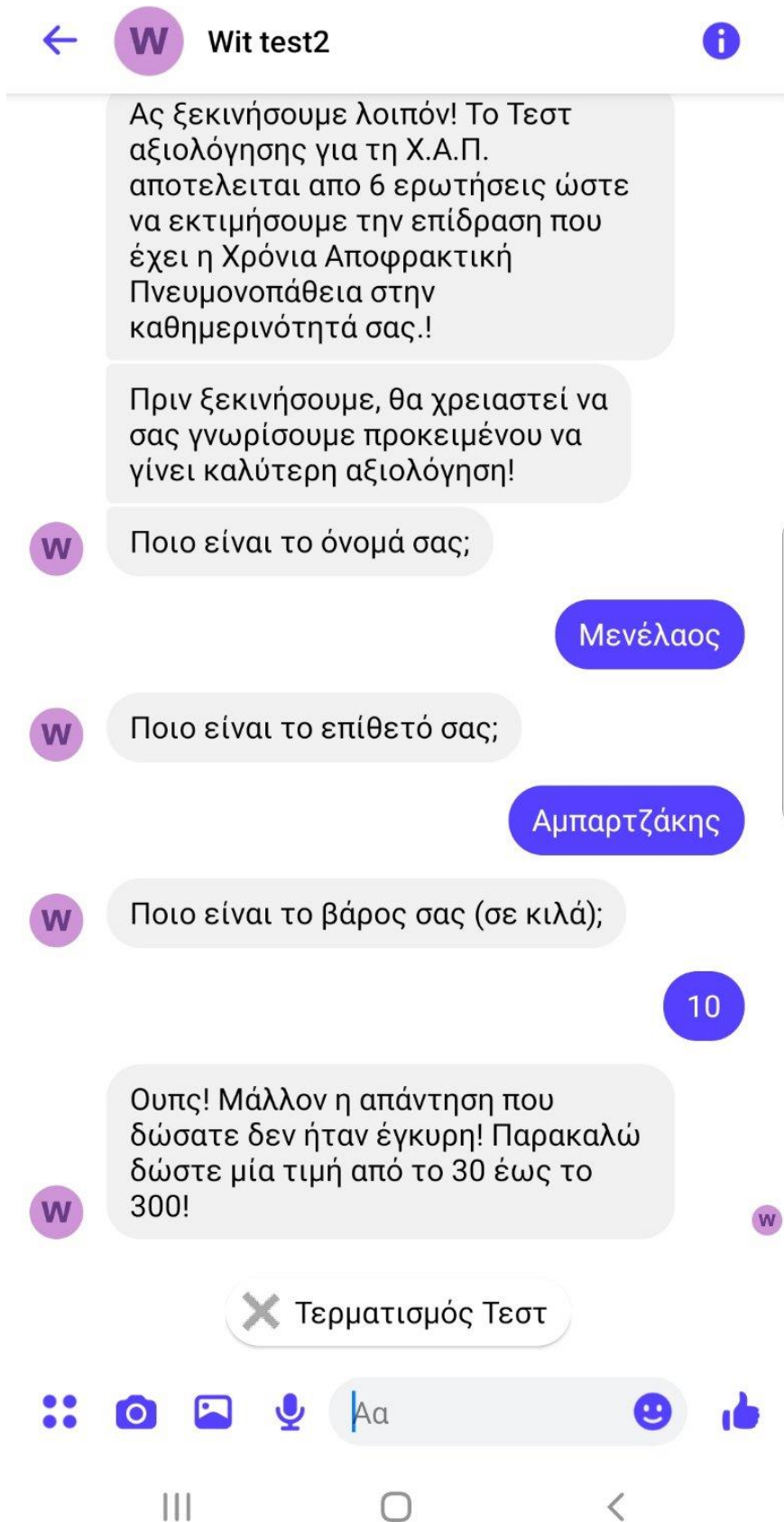


Figure 4.9 : Sample Conversation (8)



W

Wit test2



80

Ποιο είναι το φύλο σας;

W

Επιλέξτε μία από τις παρακάτω επιλογές.

Δεν καταλαβα

W

Ουπς! Μάλλον η απάντηση που δώσατε δεν ήταν έγκυρη! Παρακαλώ δώστε απάντηση 'αρσενικό', 'θηλυκό' ή 'δεν απαντώ'.

antras

Τέλεια, ας περάσουμε στο ερωτηματολόγιο!

W

Ερώτηση 1: Από το 1 έως το 10, πόσο συχνά βήχετε; (Όπου 1=Δεν βήχω ποτέ και 10=Βήχω συνέχεια)

99

W

Ουπς! Μάλλον η απάντηση που δώσατε δεν ήταν έγκυρη! Παρακαλώ δώστε μία τιμή από το 1 έως το 10!

W

✕ Τερματισμός Τεστ



Aa



Figure 4.10 : Sample Conversation (9)

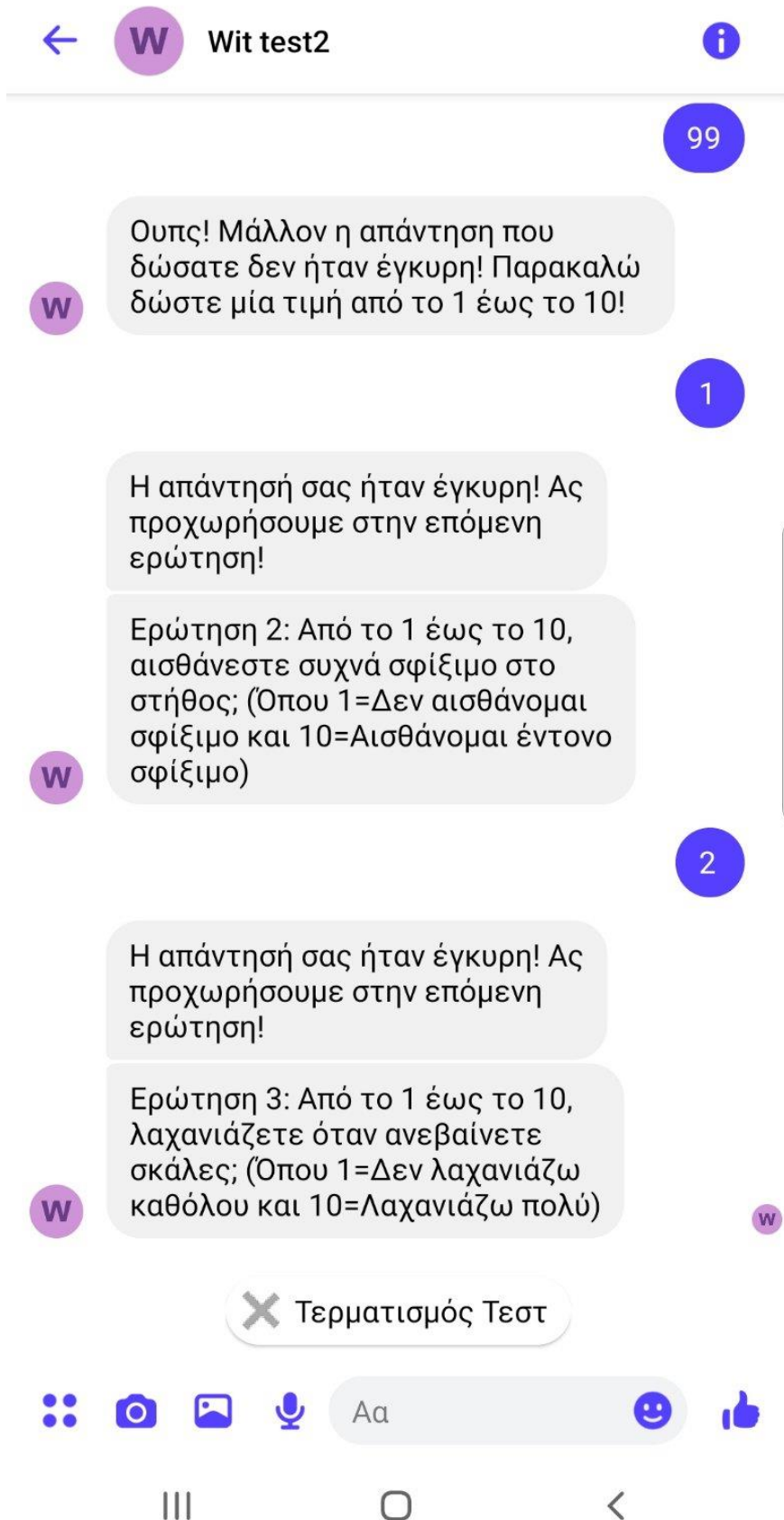


Figure 4.11 : Sample Conversation (10)

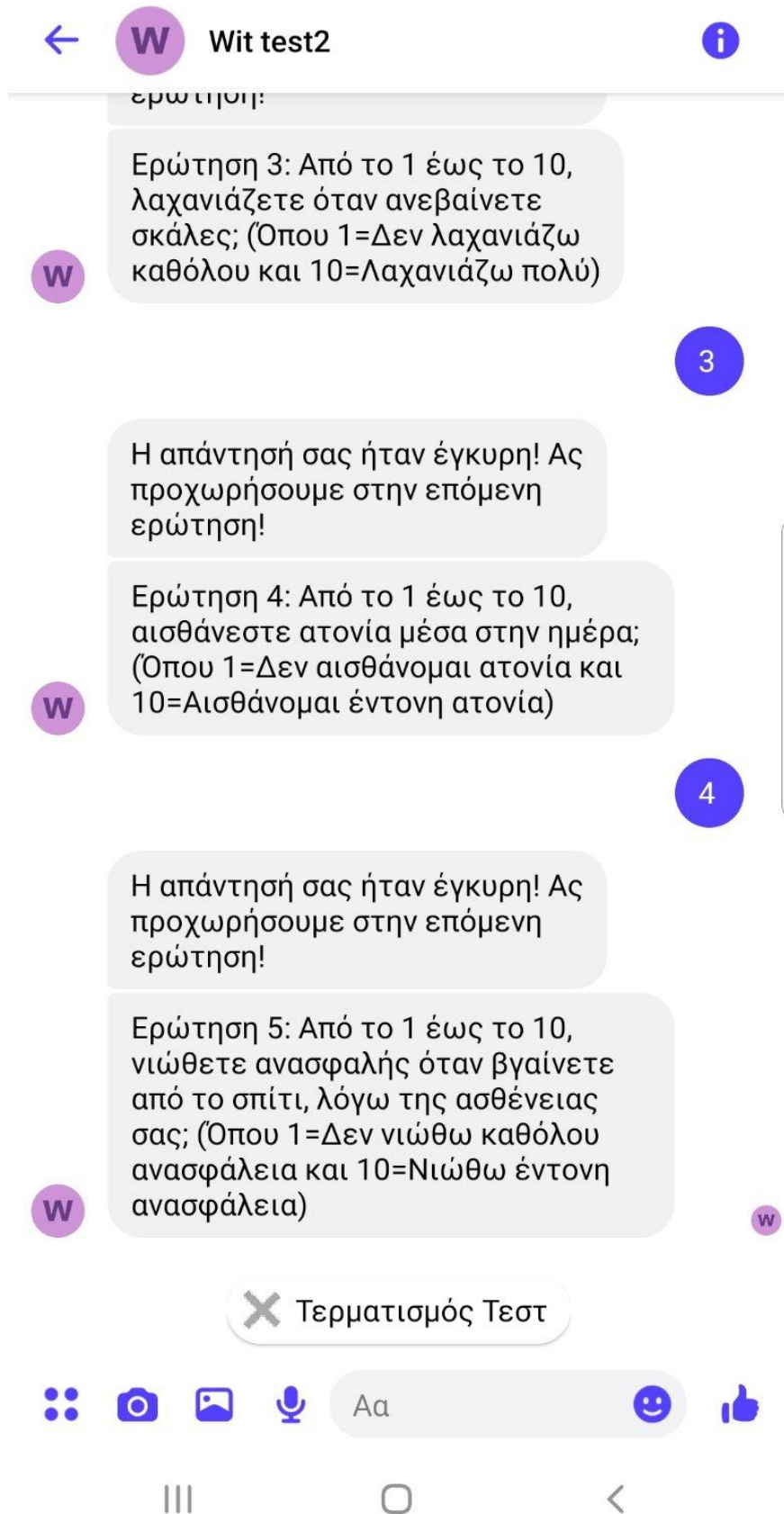


Figure 4.12 : Sample Conversation (11)

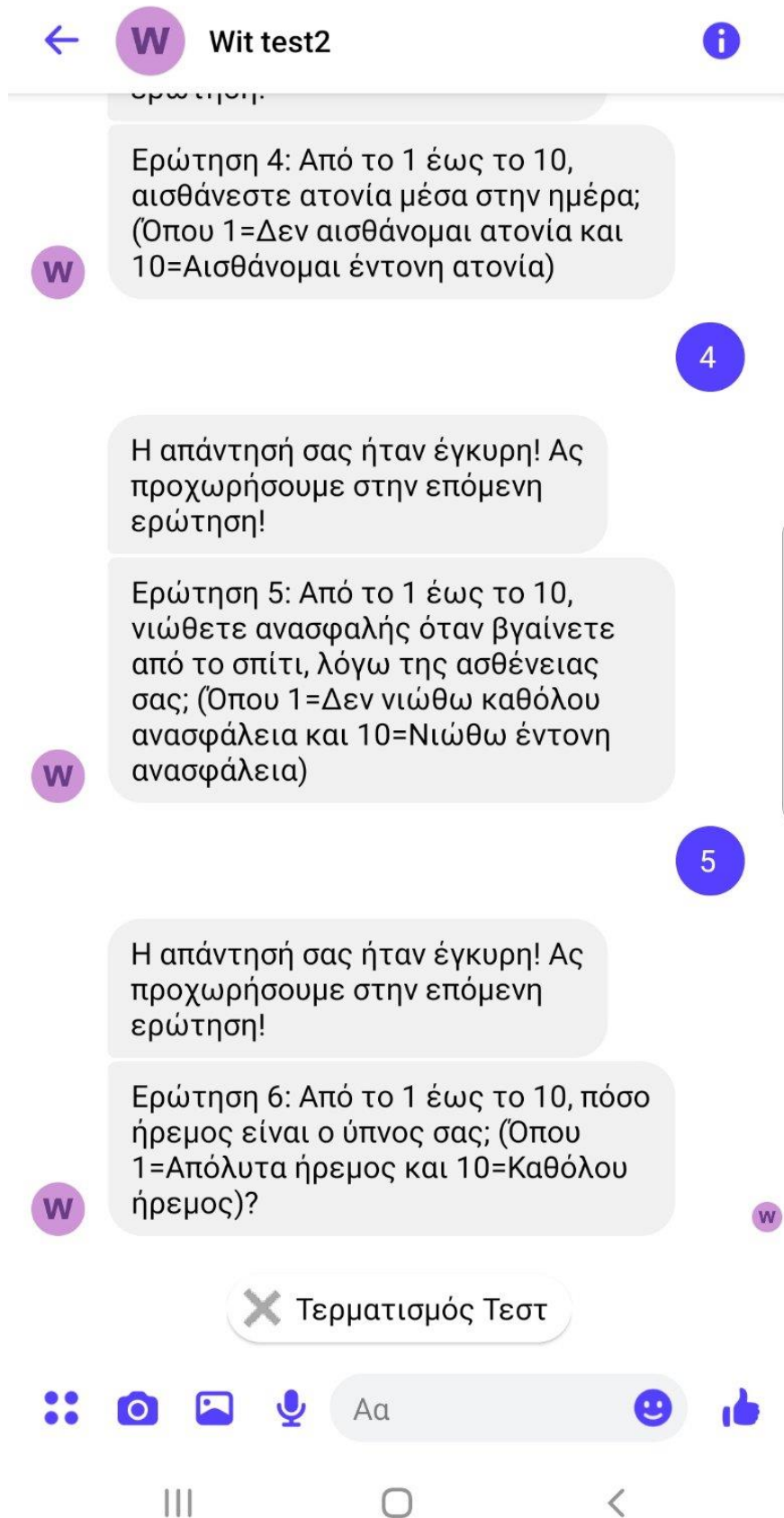


Figure 4.13 : Sample Conversation (12)

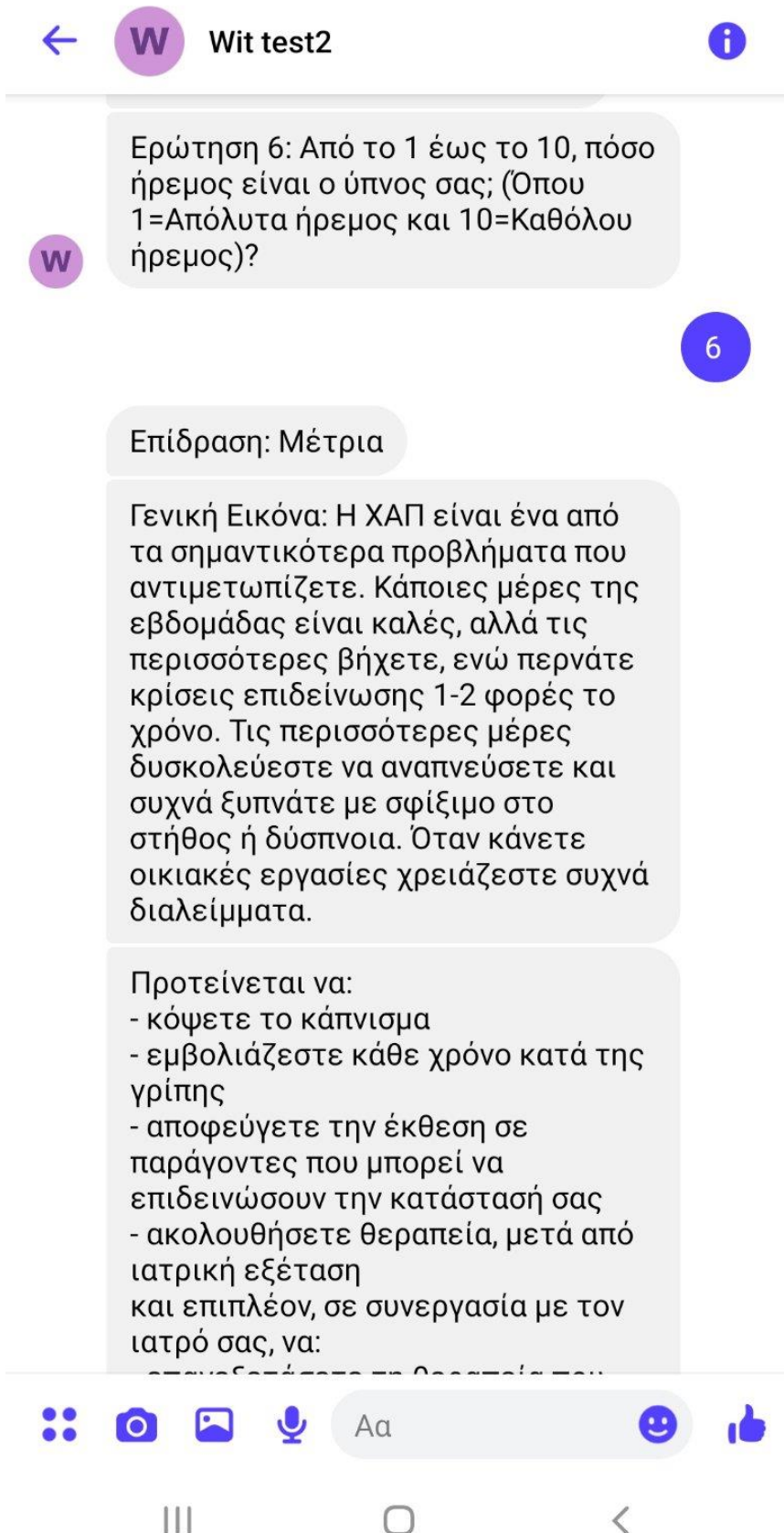


Figure 4.14 : Sample Conversation (13)



W

Wit test2



- κοφείτε το καινίωμα
- εμβολιάζεστε κάθε χρόνο κατά της γρίπης
- αποφεύγετε την έκθεση σε παράγοντες που μπορεί να επιδεινώσουν την κατάστασή σας
- ακολουθήσετε θεραπεία, μετά από ιατρική εξέταση και επιπλέον, σε συνεργασία με τον ιατρό σας, να:
- επανεξετάσετε τη θεραπεία που ακολουθείτε
- απευθυνθείτε σε ένα κέντρο πνευμονικής αποκατάστασης
- εξασφαλίστε ότι ακολουθείτε τη βέλτιστη προσέγγιση για την ελαχιστοποίηση και τη διαχείριση των κρίσεων επιδείνωσης
- επανεξετάστε τους παράγοντες που ενδεχομένως επιβαρύνουν την κατάσταση της υγείας σας

W

Θέλω πληροφορίες για τη χαπ

Η Χρόνια Αποφρακτική Πνευμονοπάθεια (ΧΑΠ) είναι μια προοδευτική ασθένεια που επηρεάζει τους αεραγωγούς των πνευμόνων. Με την πάροδο του χρόνου καθιστά την αναπνοή όλο και πιο δύσκολη.

W

W



Τεστ ΧΑΠ



Πρόληψη



Προηγούμενα Τ



Αα



Figure 4.15 : Sample Conversation (14)

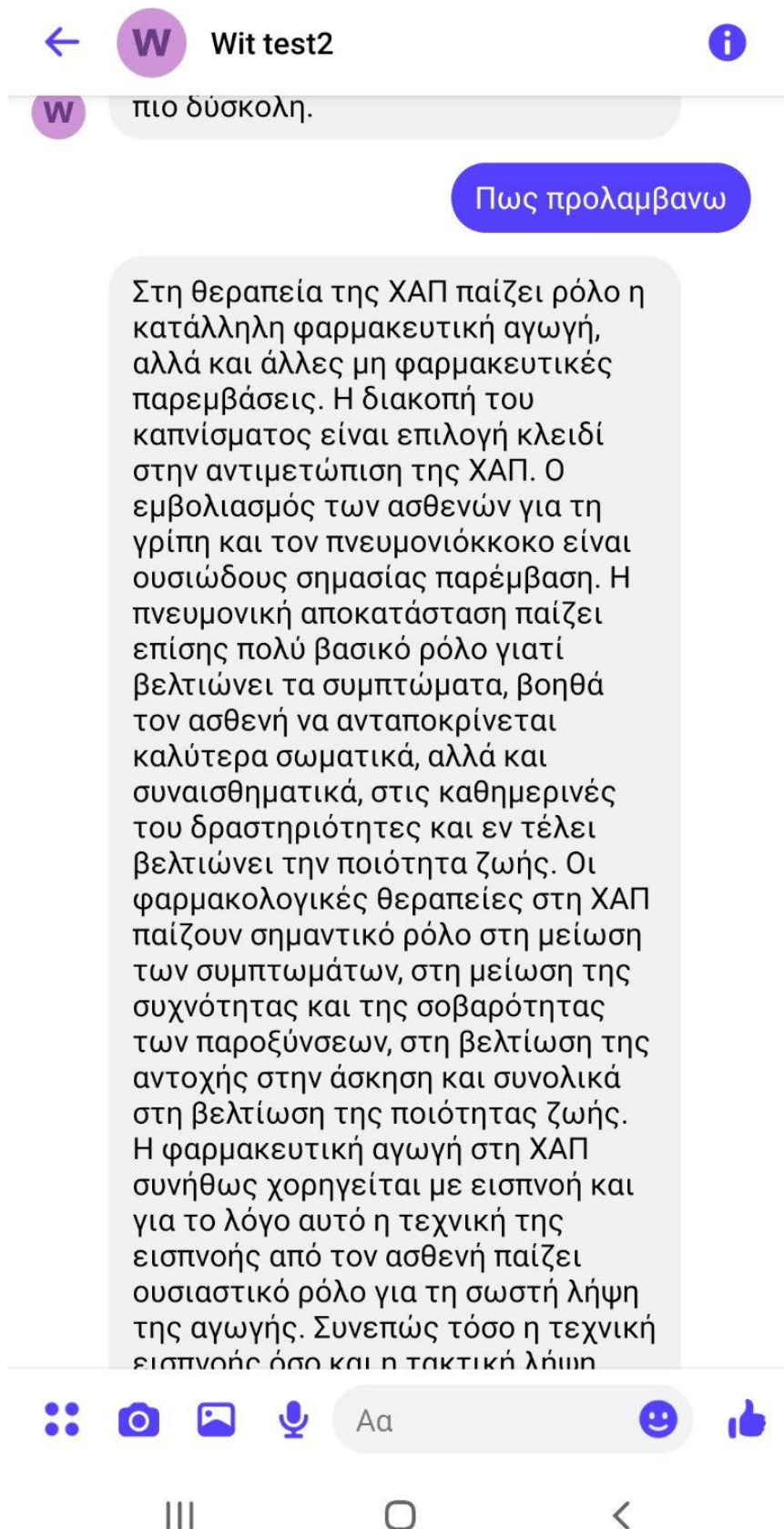


Figure 4.16 : Sample Conversation (15)

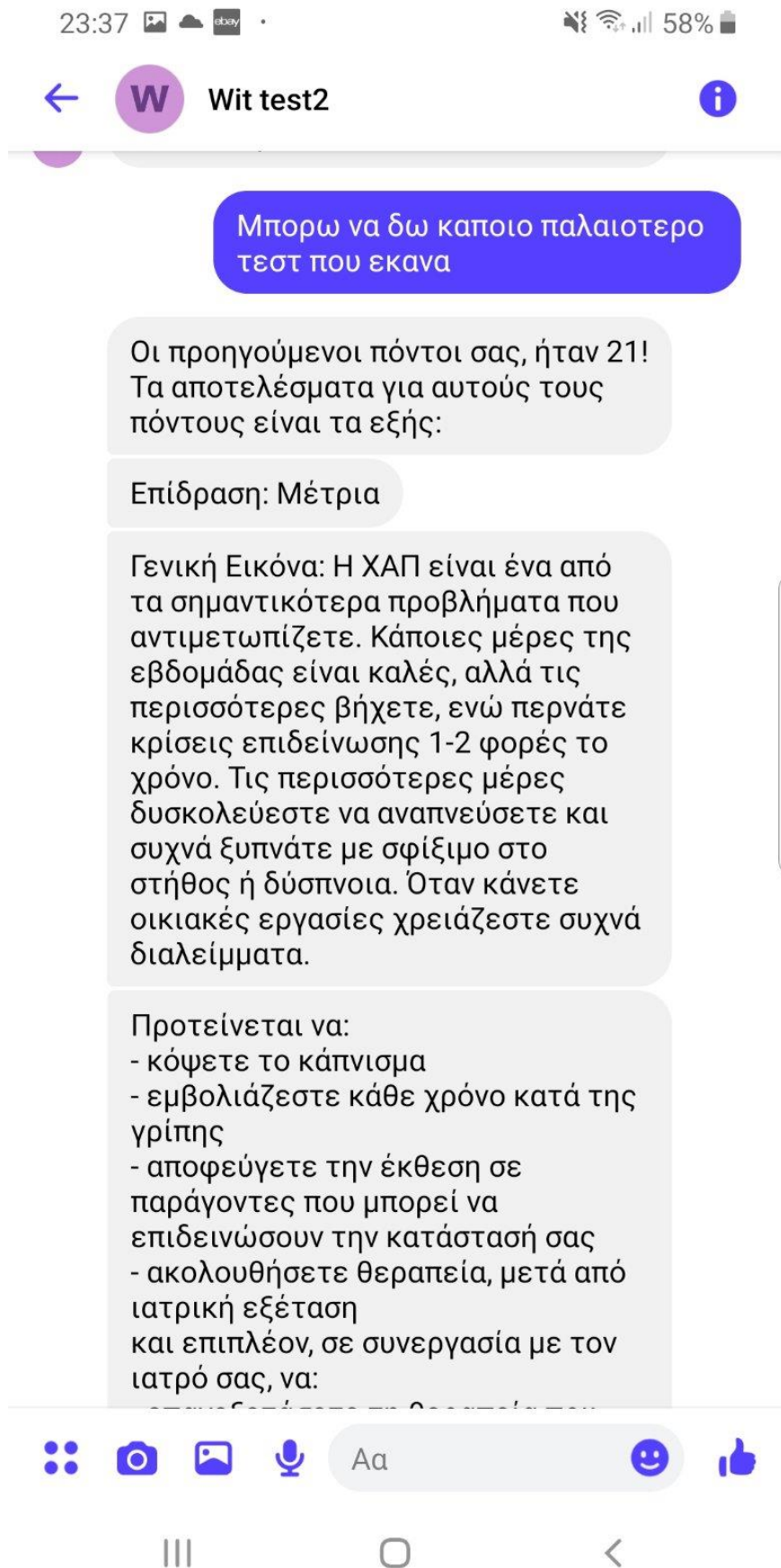


Figure 4.17 : Sample Conversation (16)



W

Wit test2



Θελω να κανω το τεστ χαπ

Φαίνεται ότι έχετε ξανακάνει το τεστ Χ.Α.Π. από αυτόν τον λογαριασμό! Θα θέλατε να συνεχίσετε σαν κάποιος από τους παρακάτω χρήστες;

1: MEN2 AMP2

2: MEN3 AMP3

3: gh gh

4: test1 test1

5: tyyt tytytyt

6: Μενέλαος Αμπαρτζάκης

7: Μενεάπς Δηγηφ

8: μεν αμπ

9: μυν μυν

10: σδσδφδσδσδ δαφσδδσφδσ

Εάν ναι, παρακαλούμε δώστε τον αριθμό που αντιστοιχεί στον επιθυμητό χρήστη, διαφορετικά πατήστε το κουμπί 'Νέος Χρήστης'.

W

W



Αα



Figure 4.18 : Sample Conversation (17)



W

Wit test2



88

Ουπς! Φαίνεται ότι δεν δώσατε σωστό αριθμό! Δώστε τον αριθμό που αντιστοιχεί στον επιθυμητό χρήστη.

1: MEN2 AMP2

2: MEN3 AMP3

3: gh gh

4: test1 test1

5: tyyt tytytyt

6: Μενέλαος Αμπαρτζάκης

7: Μενεάπς Δηγφ

8: μεν αμπ

9: μυν μυν

10: σδσδφδσδσδ δαφσδδσφδσ

Διαφορετικά, εάν θέλετε να συνεχίσετε ως νέος χρήστης, πατήστε το κουμπί 'Νέος Χρήστης'.

W

W

 Νέος Χρήστης Τερματισμός Τεστ

Aa



Figure 4.19 : Sample Conversation (18)



Wit test2



2: MEN3 AMP3

3: gh gh

4: test1 test1

5: tyyt tytytyt

6: Μενέλαος Αμπαρτζάκης

7: Μενεάπς Δηγηφ

8: μεν αμπ

9: μυν μυν

10: σδσδφδσδσδσδ δαφσδδσφδσ

Διαφορετικά, εάν θέλετε να συνεχίσετε ως νέος χρήστης, πατήστε το κουμπί 'Νέος Χρήστης'.



Καλώς ήρθατε και πάλι Μενέλαος Αμπαρτζάκης!

Πριν ξεκινήσουμε το τεστ, θα χρειαστεί να μας δώσετε κάποιες πληροφορίες ακόμη για καλύτερη αξιολόγηση.



Ποιο είναι το βάρος σας (σε κιλά);



Aa



Figure 4.20 : Sample Conversation (19)



W

Wit test2



9: μυν μυν

10: σδσδφδσδσδσδ δαφσδδσφδσ

W Διαφορετικά, εάν θέλετε να συνεχίσετε ως νέος χρήστης, πατήστε το κουμπί 'Νέος Χρήστης'.

6

Καλώς ήρθατε και πάλι Μενέλαος Αμπαρτζάκης!

Πριν ξεκινήσουμε το τεστ, θα χρειαστεί να μας δώσετε κάποιες πληροφορίες ακόμη για καλύτερη αξιολόγηση.

W

Ποιο είναι το βάρος σας (σε κιλά);

85

Τέλεια, ας περάσουμε στο ερωτηματολόγιο!

W

Ερώτηση 1: Από το 1 έως το 10, πόσο συχνά βήχετε; (Όπου 1=Δεν βήχω ποτέ και 10=Βήχω συνέχεια)

W

 Τερματισμός Τεστ



Aa



Figure 4.21 : Sample Conversation (20)



W

Wit test2



Τέλεια, ας περάσουμε στο ερωτηματολόγιο!

W

Ερώτηση 1: Από το 1 έως το 10, πόσο συχνά βήχεται; (Όπου 1=Δεν βήχω ποτέ και 10=Βήχω συνέχεια)

9

Η απάντησή σας ήταν έγκυρη! Ας προχωρήσουμε στην επόμενη ερώτηση!

W

Ερώτηση 2: Από το 1 έως το 10, αισθάνεστε συχνά σφίξιμο στο στήθος; (Όπου 1=Δεν αισθάνομαι σφίξιμο και 10=Αισθάνομαι έντονο σφίξιμο)

7

Η απάντησή σας ήταν έγκυρη! Ας προχωρήσουμε στην επόμενη ερώτηση!

W

Ερώτηση 3: Από το 1 έως το 10, λαχανιάζετε όταν ανεβαίνετε σκάλες; (Όπου 1=Δεν λαχανιάζω καθόλου και 10=Λαχανιάζω πολύ)

W

Τερματισμός Τεστ



Aa



Figure 4.22 : Sample Conversation (21)



Wit test2



Ερώτηση 3:



Ερώτηση 3: Από το 1 έως το 10, λαχανιάζετε όταν ανεβαίνετε σκάλες; (Όπου 1=Δεν λαχανιάζω καθόλου και 10=Λαχανιάζω πολύ)



Η απάντησή σας ήταν έγκυρη! Ας προχωρήσουμε στην επόμενη ερώτηση!



Ερώτηση 4: Από το 1 έως το 10, αισθάνεστε ατονία μέσα στην ημέρα; (Όπου 1=Δεν αισθάνομαι ατονία και 10=Αισθάνομαι έντονη ατονία)



Η απάντησή σας ήταν έγκυρη! Ας προχωρήσουμε στην επόμενη ερώτηση!



Ερώτηση 5: Από το 1 έως το 10, νιώθετε ανασφαλής όταν βγαίνετε από το σπίτι, λόγω της ασθένειάς σας; (Όπου 1=Δεν νιώθω καθόλου ανασφάλεια και 10=Νιώθω έντονη ανασφάλεια)



Τερματισμός Τεστ



Figure 4.23 : Sample Conversation (22)



Wit test2



Ερώτηση 4:



Ερώτηση 4: Από το 1 έως το 10, αισθάνεστε ατονία μέσα στην ημέρα; (Όπου 1=Δεν αισθάνομαι ατονία και 10=Αισθάνομαι έντονη ατονία)

8

Η απάντησή σας ήταν έγκυρη! Ας προχωρήσουμε στην επόμενη ερώτηση!



Ερώτηση 5: Από το 1 έως το 10, νιώθετε ανασφαλής όταν βγαίνετε από το σπίτι, λόγω της ασθένειάς σας; (Όπου 1=Δεν νιώθω καθόλου ανασφάλεια και 10=Νιώθω έντονη ανασφάλεια)

6

Η απάντησή σας ήταν έγκυρη! Ας προχωρήσουμε στην επόμενη ερώτηση!



Ερώτηση 6: Από το 1 έως το 10, πόσο ήρεμος είναι ο ύπνος σας; (Όπου 1=Απόλυτα ήρεμος και 10=Καθόλου ήρεμος)?



✕ Τερματισμός Τεστ



Aa



Figure 4.24 : Sample Conversation (23)

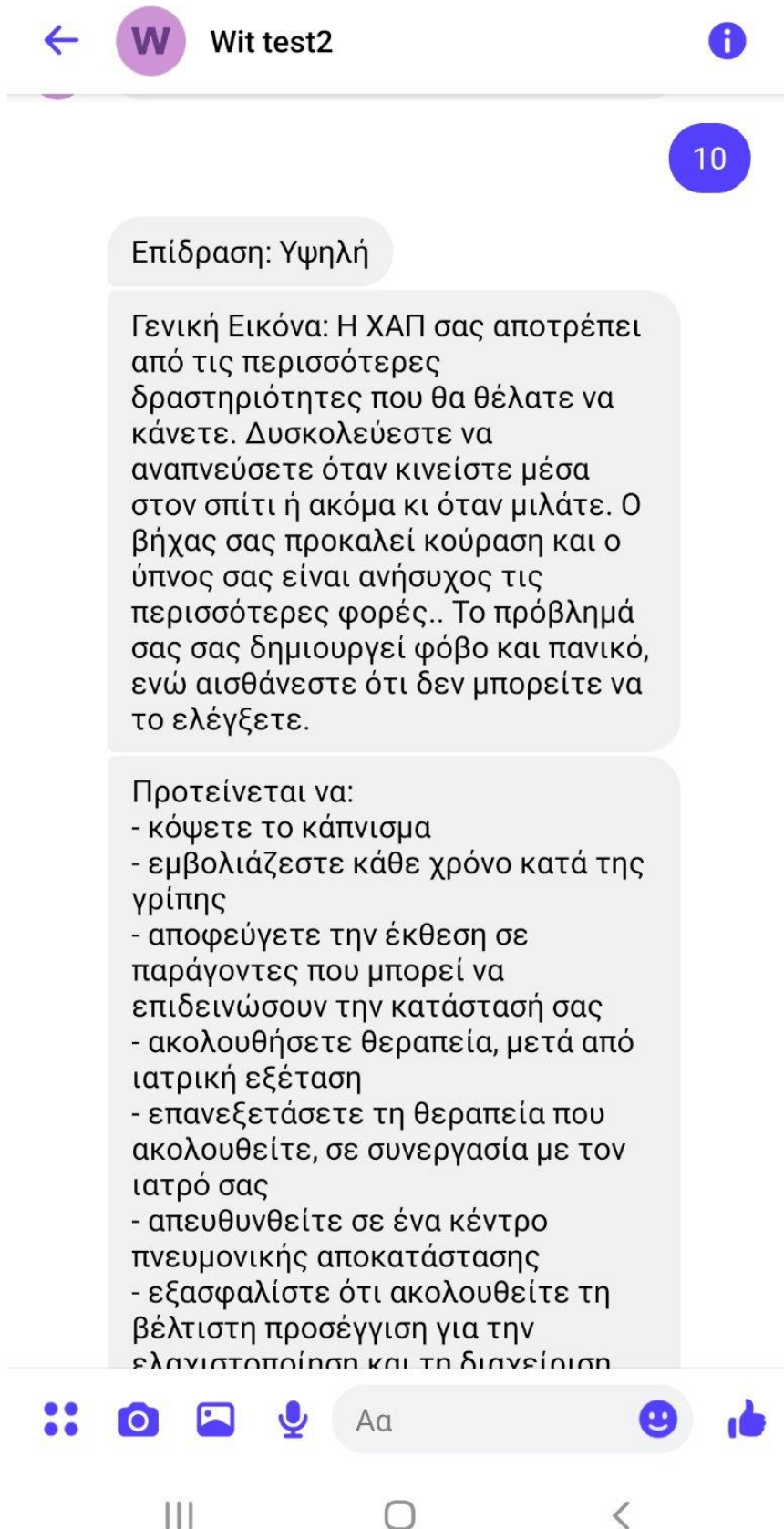


Figure 4.25 : Sample Conversation (24)



W

Wit test2



εβδομάδας είναι καλές, αλλά τις περισσότερες βήχετε, ενώ περνάτε κρίσεις επιδείνωσης 1-2 φορές το χρόνο. Τις περισσότερες μέρες δυσκολεύεστε να αναπνεύσετε και συχνά ξυπνάτε με σφίξιμο στο στήθος ή δύσπνοια. Όταν κάνετε οικιακές εργασίες χρειάζεστε συχνά διαλείμματα.

Προτείνεται να:

- κόψετε το κάπνισμα
- εμβολιάζεστε κάθε χρόνο κατά της γρίπης
- αποφεύγετε την έκθεση σε παράγοντες που μπορεί να επιδεινώσουν την κατάστασή σας
- ακολουθήσετε θεραπεία, μετά από ιατρική εξέταση και επιπλέον, σε συνεργασία με τον ιατρό σας, να:
 - επανεξετάσετε τη θεραπεία που ακολουθείτε
 - απευθυνθείτε σε ένα κέντρο πνευμονικής αποκατάστασης
 - εξασφαλίστε ότι ακολουθείτε τη βέλτιστη προσέγγιση για την ελαχιστοποίηση και τη διαχείριση των κρίσεων επιδείνωσης
 - επανεξετάστε τους παράγοντες που ενδεχομένως επιβαρύνουν την κατάσταση της υγείας σας

W

W



Πληροφορίες



Τεστ ΧΑΠ



Aa



Figure 4.26 : Sample Conversation (25)

4.2. Monitor the user

After the successful completion of COPD test from a user, a new document with all the information of the user and the test is inserted in the database. To monitor these information Tableau will be used, as mentioned in a previous chapter. Double click on the .bat file and the Tableau Workbook will open.

To monitor a specific COPD test of a user, just adjust the filters. For example, according to the above example the latest COPD test was completed by the user, *Μενέλαος Αμπαρτζάκης*. Click on the *First Name & Last Name Filter* to select the First & Last Name of the user. If you want to select the latest COPD test, filter with one more condition based on Time.

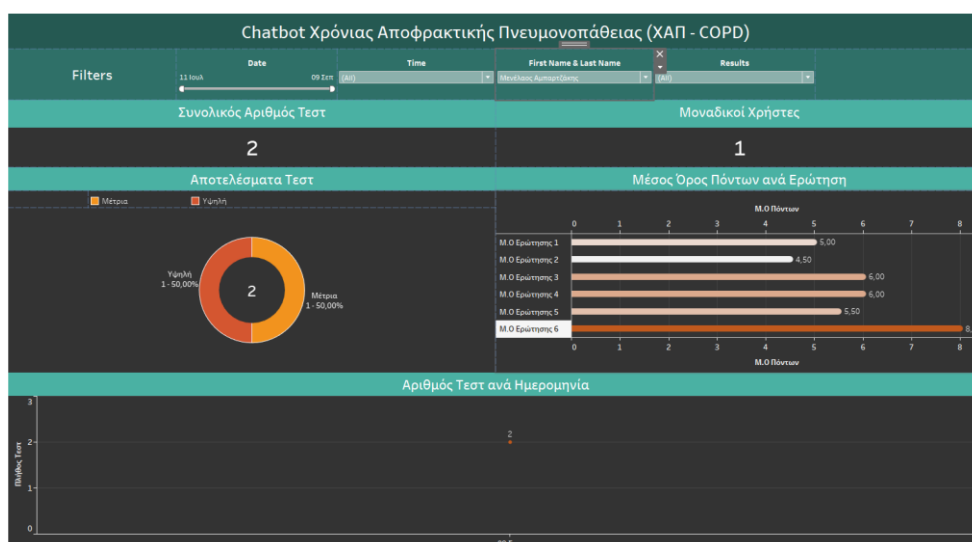


Figure 4.27 : Monitoring a User (1)

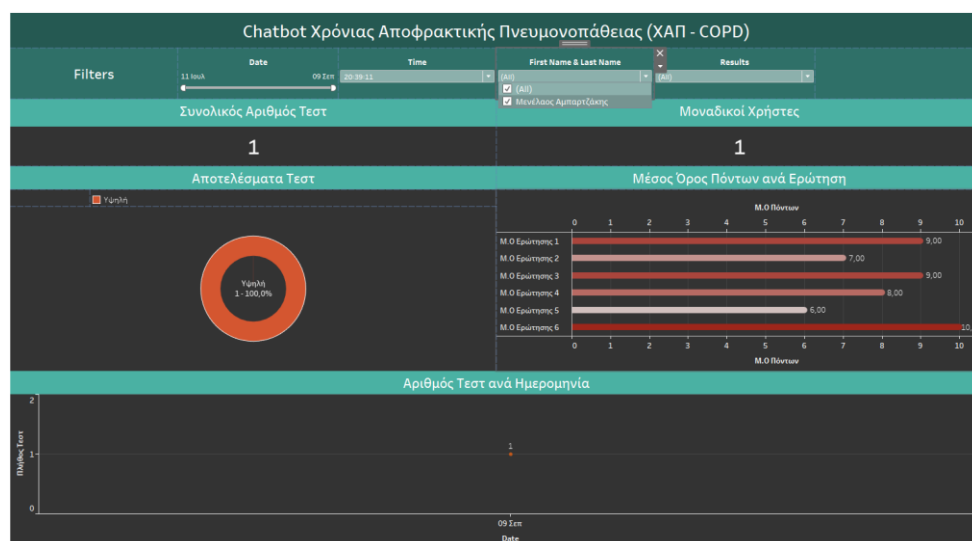


Figure 4.28 : Monitoring a User (2)

5. Outcome

5.1. Conclusions & Future Works

In the current thesis, a new approach was described to build a medical chatbot in Facebook Messenger Platform using Wit.ai in the Greek language. Later, Tableau was used to monitor the results of the users. Finally, a method to include questionnaire in the chatbot was developed using MongoDB and Python. This entire end-to-end solution can be adopted by doctors or organizations that want to provide users an instant way to identify the effect of symptoms regarding COPD.

The solution can be applied to multiple areas and not just for medical use. Thus, the logic and implementation of the questionnaire will always be the same. However, questionnaire method is not scalable, and it needs more hard coding as long as you add more questions. It's free for everyone to have that ecosystem with the chatbot trained except from Tableau. Additionally, minor operations must be applied in the computer of the administrator to have it in use.

As future works, it would be valuable to develop a method that adds all the necessary conditions in a scalable way. We also plan to improve the way the prediction is done. For example, a clustering can be applied in users and their results to find clusters that contains users with similar results and habits. Then the prediction will be accomplished using the results of the cluster. Later on, more KPIs can be added in the Dashboard to monitor even better the users. Then, the dashboard can be deployed in a Tableau Server to set alerts each time a user has a really negative result. That will reduce the need to check it constantly and eventually open it just the times when an alert occurs.

6. References

1. IBM 2018. Overview of Watson assistant. <https://console.bluemix.net/docs/services/conversation/index.html>, 2018.
2. Reifler, Erwin (February 2–5, 1960). "The solution of MT linguistic problems through lexicography". Proceedings of the National Symposium on Machine Translation.
3. Benilda Eleonor V. Comendador, , et al., (2015), "Pharmabot: A Pediatric Generic Medicine Consultant Chatbot "
4. J. Constine. Facebook launches messenger plat- form with chatbots, 2016.
5. Jerome R Bellegarda. Spoken language understanding for natural interaction: The siri experience.,. In Natural Interaction with Robots, Knowbots and Smartphones, pages 3–14, 2005.
6. Abu Shavar, B., & Atwell, E. (Eds.) 2007. Chatbots: Are they really useful?
7. Divya S et al., (2018) "Self-Diagnosis Medical Chatbot Using Artificial Intelligence"
8. Bickmore, T. W., Mitchell, S. E., Jack, B. W., Paasche-Orlow, M. K., Pfeifer, L. M., & O'Donnell, J. 2010. Response to a relational agent by hospital patients with depressive symptoms. *Interacting with Computers*, 22(4): 289–298.
9. Turing, Alan (October 1950), "Computing Machinery and Intelligence", *Mind*
10. Duffy, M. C., & Azevedo, R. 2015. Motivation matters: Interactions between achievement goals and agent scaffolding for self-regulated learning within an intelligent tutoring system. *Computers in Human Behavior*, 52: 338–348.
11. Fryer, L. K., Ainley, M., Thompson, A., Gibson, A., & Sherlock, Z. 2017a. Stimulating and sustaining interest in a language course: An experimental comparison of Chatbot and Human task partners. *Computers in Human Behavior*, 75: 461–468.
12. Ahmed Fadhil (2018). Beyond Patient Monitoring: Conversational Agents Role in Telemedicine & Healthcare Support for Home-Living Elderly Individuals
13. Gartner. Gartner Predicts a Virtual World of Exponential Change - Smarter with Gartner; <https://www.gartner.com/smarterwithgartner/gartner-predicts-a-virtual-world-of-exponentialchange/>, 04 Jan 2018.
14. V.Manoj Kumar et al., (2016) "Sanative Chatbot For Health Seekers"

15. Weizenbaum, J. (1966). ELIZA—A computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1), 36–45. Wessel, M., Adam, M., & Benlian, A. (2019). Th
16. Elkins, A. C., Derrick, D. C., Burgoon, J. K., & Nunamaker Jr, J. F. (2012). Predicting users' perceived trust in Embodied Conversational Agents using vocal dynamics. In: *Proceedings of the 45th Hawaii International Conference on System Science (HICSS)*. Maui: IEEE.
17. Krishnendu Rarhi (ORCID: 0000-0002-5794-215X), Abhishek Bhattacharya, Abhishek Mishra, Krishnasis Mandal (2018) “Automated Medical Chatbot”
18. Flora Amato, et al., (2019) “Chatbots meet eHealth: automatizing healthcare”
19. J., Ford, W. R., & Farreras, I. G. (2015). Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations. *Computers in Human Behavior*, 49, 245–250.
20. Shah, H., Warwick, K., Vallverdú, J., Wu, D.: Can Machines Talk? Comparison of ELIZA with Modern Dialogue Systems. *Comput. Hum. Behav.* 58, 278–295 (2016)
21. Lin Ni, Chenhao Lu, Niu Liu, and Jiamou Liu. 2017. Mandy: Towards a smart primary care chatbot application. In *International Symposium on Knowledge and Systems Sciences*. Springer, 38–52.
22. Webber, G. M. (2005). Data representation and algorithms for biomedical informatics applications. PhD thesis, Harvard University.
23. L. Zhang, “Building Facebook Messenger,” 2011. [Diakses: 31-Mar-2017]
24. Sameera A. Abdul-Kader, Dr. John Woods, “Survey on Chatbot Design Techniques in Speech Conversation Systems”, (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, Vol. 6, No. 7, 2015
25. Facebook for developers. April 12, 2016. Retrieved July 6, 2018 from <https://developers.facebook.com/videos/f8-2016/introducing-bots-on-messenger>
26. Facebook Messenger Platform. <https://messengerplatform.fb.com/>. Accessed 5 April 2017
27. Awidi et al., 2019 I.T. Awidi, M. Paynter, T. Vujosevic Facebook group in the learning design of a higher education course: An analysis of factors influencing positive learning experience for students *Computers & Education*, 129 (2019), pp. 106-121

28. Shady Hamouda, Zurinahni Zainol, "Document-Oriented Data Schema for Relational Database Migration to NoSQL", Big Data Innovations and Applications (Innovate-Data) 2017 International Conference on, pp. 43-50, 2017.
29. Sheffi Gupta, Rinkle Rani, "A comparative study of elasticsearch and CouchDB document-oriented databases", Inventive Computation Technologies (ICICT) International Conference on, vol. 1, pp. 1-4, 2016.
30. Jonathan de Oliveira Assis, Vanessa C O Souza, Melise M V Paula, João Bosco S Cunha, "Performance evaluation of NoSQL data store for digital media", Information Systems and Technologies (CISTI) 2017 12th Iberian Conference on, pp. 1-6, 2017.
31. Datig, I. and Whiting, P. (2018), "Telling your library story: tableau public for data visualization", Library Hi Tech News, Vol. 35 No. 4, pp. 6-8. <https://doi.org/10.1108/LHTN-02-2018-0008>