

E-Commerce Chat-Bot Application

ARTIFICIAL INTELLIGENCE DRIVEN CUSTOMER SUPPORT

**Bachelor Project – PROJECT REPORT**

Alin Stanescu | 208218

Cristian Ciolea | 189591

Rami Jibreel | 208233

**Supervisor**

Stephan Erbs Korsholm

December 15th 2016

Contents

[Abstract](#_gjdgxs)

[1. Introduction](#_30j0zll)

[1.1. Background Description](#_1fob9te)

[1.2. Purpose.](#_3znysh7)

[1.3. Problem Formulation](#_2et92p0)

[1.4. Delimitation](#_tyjcwt)

[1.5. Choice of Model and Method](#_3dy6vkm)

[1.6. Time Schedule](#_1t3h5sf)

[2. System Development](#_4d34og8)

[2.1. Analysis](#_2s8eyo1)

[2.1.1. Requirements](#_17dp8vu)

[2.1.1.a. Functional (what the system should do – change those below!!)](#_3rdcrjn)

[2.1.1.b. Non-functional (how will the system do that?)](#_26in1rg)

[2.1.2. Use Cases](#_lnxbz9)

[2.2. Design](#_35nkun2)

[2.2.1. Technical Choices](#_1ksv4uv)

[2.3. Implementation](#_44sinio)

[2.3.1. Source Code](#_2jxsxqh)

[2.4. Testing](#_z337ya)

[3. Results and Discussion](#_3j2qqm3)

[4. Conclusion](#_1y810tw)

[5. References](#_4i7ojhp)

[6. Appendices](#_2xcytpi)

[6.2. Appendix A: List of Abbreviations](#_1ci93xb)

[6.3. Appendix B: User Guide](#_3whwml4)

# Abstract

E-Shop Online is a small start-up business in Horsens. They have recently purchased a warehouse where they keep all their mobile devices for sale. Since it’s a new business, their budget is limited so they can’t afford to have a customer support employee answering questions related to products, shipping etc. This is why they have decided to invest in an online chat bot application which could handle client requests 24/7. The application would serve as an extra employee replying to all the redundant questions that customers ask.

# Introduction

## 1.1. Background Description

Going online and buy goods is a common thing in our days because you can purchase almost anything in a matter of seconds with a few clicks if you know exactly what you want. Being one step ahead of their clients and trying to help them out during this process is not an easy task because most of the time the user is “on his own” in the e-commerce website and looking through items that he considers to buy or not at that moment.

On one hand, many online companies ignore the customer support chat or consider this option a secondary (or non-vital) solution for customer communication. On the other hand, upgrading the website with a robot that can chat with customers has its advantages as well.

For the client to have an informed decision about his purchase and get answers to his questions regarding payment, shipping, warranty, return policy and other related issues, some online e-commerce website offer a real-time chat support but only during the working hours because behind the service it has to be a real person which has to interact with the client.

No one wants to leave the products page just to go on another page to read about things that someone can answer for them in no-time, if that someone was all the time available.

Taking into consideration that every single company has as the main purpose the increase of the profit while having reduced costs, implementing an e-commerce Artificial Intelligence driven Chat-Bot can be beneficial for their business because someone is there 24 hours, 7 days/week, 365 days/year and serving their new and existing customers as a real person from the staff would do.

Besides the *profit increase* (which can also be translated as cost reduction, meaning that at least one monthly salary will be saved, whenever the Chat-Bot satisfies the customer), there are some other advantages worth to mention: *boosting online sales* (companies are leaning fast towards the online chat sales, as it can offer a higher interaction level between the company and the customer), *ensuring a higher customer satisfaction* (the chat is an efficient and comfortable mode for providing information for the customer; it is also used) and *increasing the online brand image* (which is essential for generating web traffic and attract valuable customers).

## 1.2. Purpose.

The purpose is to create an Artificial Intelligence Chat-Bot that can interact with a company’s online customers and answer back to them relevant information in real-time 24H/day.

## 1.3. Problem Formulation

The focus of the Chat-Bot is to reduce the amount of hours a person will spend answering customer’s questions, as well as working faster and more efficient than a person. The Chat Bot is created to retrieve information from various sections of the company’s website, which means the customer will not spend that time making research on his own.

When it comes to the technical part and implementation, there are some relevant questions which require an answer. These questions are described in the Choice of Model and Method and are stated in the following section:

* How can a company respond to its customers non-stop, via chat?
* How to extract data (from the website’s API) and present it to the customer?
* How can the system respond and display information, into a readable format?
* Can the Chat-Bot give answers as accurate and precise as a person?

## 1.4. Delimitation

* The Chat-Bot will not reply to any questions which are not related to the company’s activity area.
* The chat will not handle personal user’s data (like bank account and other sensitive information).
* The chat will be active as long as the company’s Internet connection is active.
* The chat will be temporary disabled whenever the system will be maintained (improvements/updates, bug fixes).

## 1.5. Choice of Model and Method

|  |  |  |
| --- | --- | --- |
| **Problem Statement** | **Problem’s relevance** | **Solution to the problem** |
| How can a company respond to its customers non-stop, via chat? | Customers will need support, at some point, which has to be provided without any problem, but a company will lose more money if a person is hired non-stop | The Chat-Bot can be a possible solution to this problem, but it comes with a limited “knowledge” pack. |
| How to extract data (from the website’s API) and present it to the customer? | Web chat is a necessity and customer’s needs are various, so the website’s API must have enough complexity that it should answer every question within the company’s area. | When the Chat-Bot is created, it will connect to an API and he will be programmed to answer questions based on the API data. |
| How can the system respond and display information, into a readable format? | If the answers are too short formulated, there is a risk that the customer is not satisfied; if the answers are too long and vague, the customer might be confused. So the system needs to provide precise and sharp answers, straight to the point. | When the implementation is done, the Bot will be programmed in such way that the answers will be as accurate as possible, so that the customer can receive a fast and precise response. |
| Can the Chat-Bot give answers as accurate and precise as a person? | One of the aspects not included in the Chat-Bot, is the intuition, life experience and personal opinions that a person can have. | The chat will be limited to a specific area of answers; for example, if the customers is asking “What do you think/suggest about…?” the Bot will automatically make a suggestion based on the customer’s previous question. |

## 1.6. Time Schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Time Schedule** | | | | |
| **Date** | 15.08.16 -15.09.16 | 16.09.16-16.10.16 | 17.10.16-16.11.16 | 17.11.16-05.12.16 | 14.12.16 |
| **Task** | Technical research and backlog creation | Implementation of user stories | Prioritizing/Adding/Cutting features | Final debugging and testing | Project Hand-In |

The time schedule is based on **four** sprints, which are the basis for SCRUM and AUP frameworks.

# System Development

## Analysis

### Requirements

DON’T MENTION TECHNOLOGY CHOSEN IN THE DESIGN PART – SEE TECHNICAL CHOICES

THE REQUIREMENTS SHOULD BE ONE – BECAUSE THERE IS ONLY ONE USE CASE

### 2.1.1.a. Functional (what the system should do – change those below!!)

The developing team established that the program will be based on the following requirement: the system must respond to the user’s requirements, using a chat-bot.

* *~~As a website user I want to find answers to product-related questions 24 h/day.~~*
* *~~As a website user want to get answers to the FAQ in real time, so that I won’t need to find the appropriate link and read them by myself.~~*
* *~~As a website user I want to know if the website offers a specific product and how much does it cost.~~*
* *~~As a website user I want to know when will the product be delivered.~~*
* *~~As a website user I want to get relevant answers about the company, like:~~*

*~~- When was the company founded?~~*

*~~- Who are the founders of the company?~~*

*~~- What is the physical address of the company?~~*

*~~- What is the company’s e-mail address?~~*

*~~- What is the company’s telephone number?~~*

*~~- What is the company’s official Facebook account/link?~~*

*~~- What is the company’s official Twitter account/link?~~*

*~~- Who are the main competitors?~~*

*~~- What reasons do I have to become their customer?~~*

*~~- What is the company’s annual revenue?~~*

*~~- What is the company’s total number of employees?~~*

### 2.1.1.b. Non-functional (how will the system do that?)

A functional requirement describes what a software system should do, while non-functional requirements place constraints on how the system will do so.

Let me elaborate.

An example of a functional requirement would be:

A system must send an email whenever a certain condition is met (e.g. an order is placed, a customer signs up, etc).

A related non-functional requirement for the system may be:

Emails should be sent with a latency of no greater than 12 hours from such an activity.

The functional requirement is describing the behavior of the system as it relates to the system's functionality. The non-functional requirement elaborates a performance characteristic of the system.

The system will use Node.JS as a server.

The system should reply within 5 seconds.

The system will not require login information.

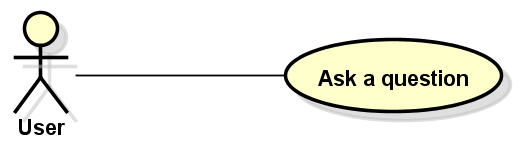
ABOUT HOW THE SYSTEM SHOULD BEHAVE

### Use Cases

In the next part, the Use Case will be presented; the section also contains a Use Case diagram and Use Case description. It has been decided that only one Use Case will be used for this application, due to its relevance.

As it can be seen in **Figure XX**, the only actor of the system is the User. The user can interact with the system using the chat.

2.1.2.a. Use Case Diagram



In the next section, the *Ask a question* Use Case will be described, as well as an activity diagram; the diagram is used later, as guideline, in the testing part.

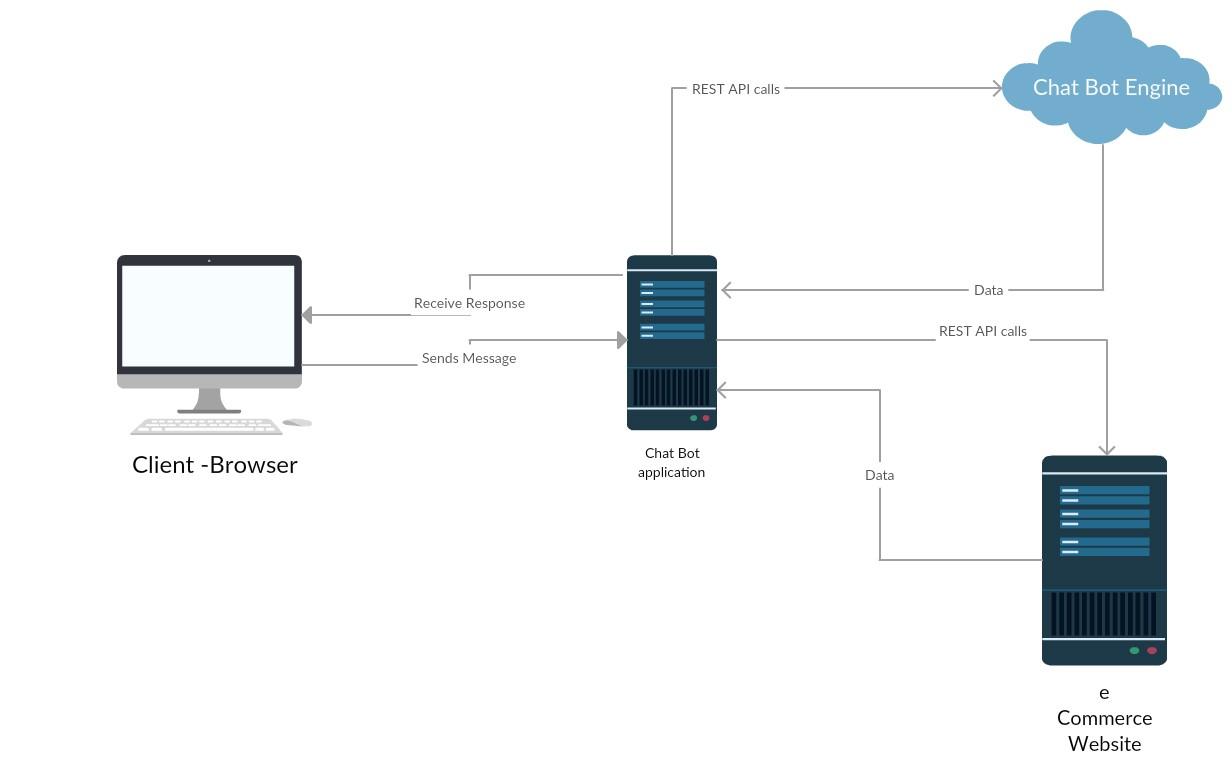
2.1.2.b. Use Case Description

The following table shows the use of *Ask a question* Use Case and its description. The flow of events shown in this table is also described in the Design part, **FIGURE XX (Diagram)**.

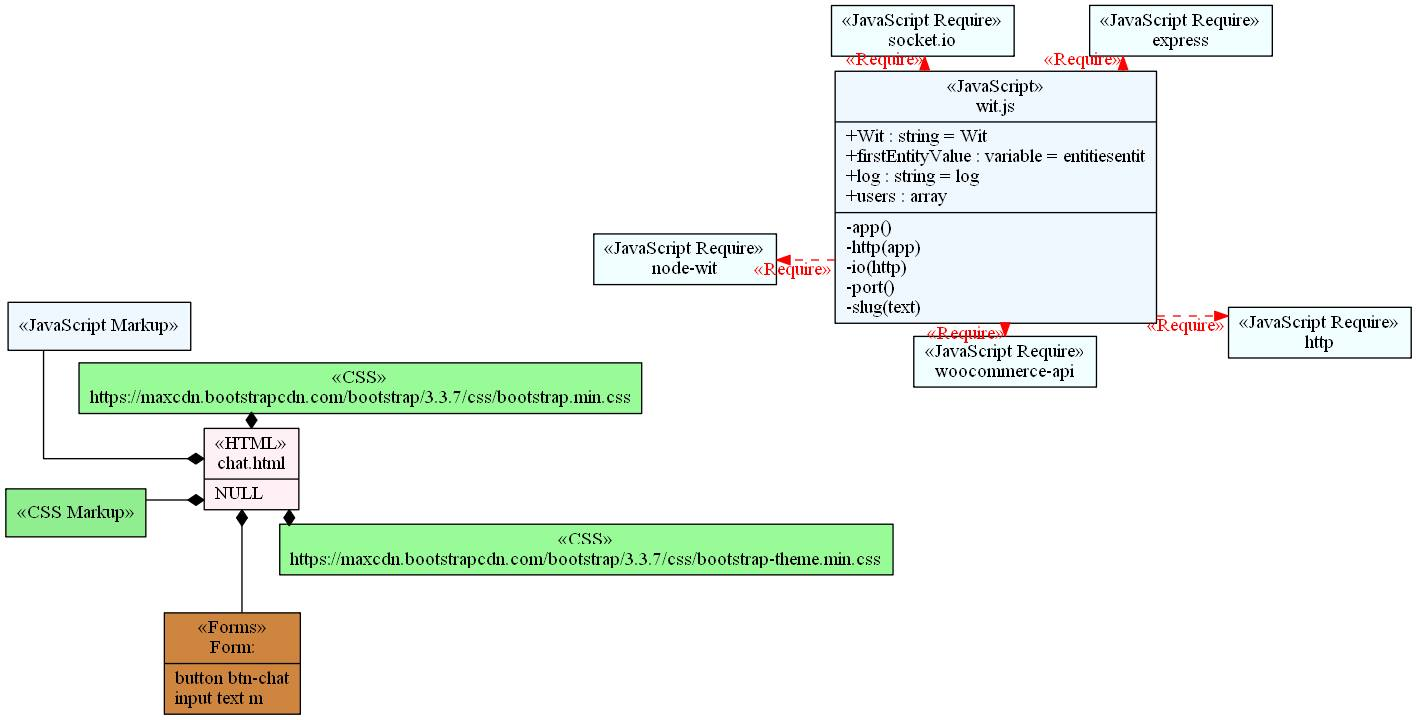
|  |
| --- |
| **Use Case name:** *Ask a question* |
| **Description:** this Use Case describes the process of interaction between the user and the system. |
| **Actors:** User. |
| **Precondition:** User must access the website **WEBSITE NAME** |
| **Flow of events:**   1. The User types in a question 2. The system checks for keywords in the User’s question and interprets the question 3. The system sends a reply to the User via chat. |
| **Activity diagram:** |

## Design

EXPLAIN THE DIAGRAM



The user accesses the website – sends a message/question – the bot is interpreting the question – sends the text/question to chatbot ENGINE to further interpreting – sends a reply to the chat bot app – the bot also sends a request to the SERVER (e-Commerce Website) and receives a reply. The functions are implemented in Chat-Bot app.



### Technical Choices

Javascript

Node.JSWit.AI

WordPress

WooCommerce

Node js vs. other programming languages

## Implementation

### 2.3.1. Source Code

## Testing

The test plan was designed during the development of the application functionality. This happened in the fourth sprint. Testing the app requires testing the different components that the app uses it and observe how those components can be broken.

2.4.1. Preparing a test plan

The developer team established a testing plan, which consisted of several steps to follow; they are explained in the following lines:

1. Form input – if the user enters a valid question, the app should respond with an answer to that.

2. Form input – if the user enters invalid question, the app should reply with “I don’t understand”.

3. Form input – if the user enters a valid search term, the app should reply back with the link to that searched item.

4. Form input – if the user enters a invalid search term, the app should reply that the item is not available.

5. API data – if the data returned from the online e-commerce shop is valid the app should work.

6. API data – if the data returned from the online e-commerce shop is invalid the app should return a error.

7. Route – if the app is working correctly a 200 http status code should be returned.

8. Route – if there is a server error a 500 http status code should be returned.

9. Real User Interaction tests – understand how well the user understands how to interact with the application. If the app is easy to use and intuitive ???

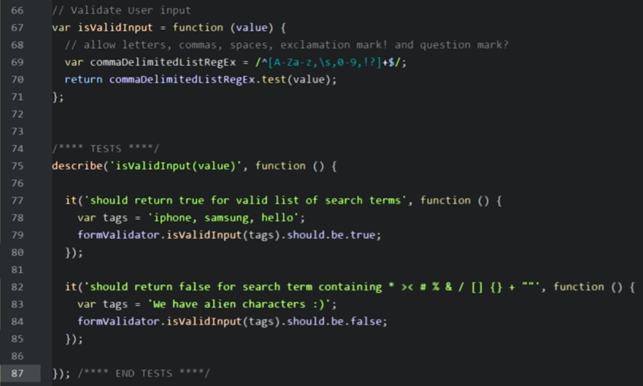
10. Test how fast a response is generated after the user asks a question.

11. Test how fast a response is generated after the user asks for an item.

2.4.2. Test types

The Chat-Bot application includes more types of testing: Unit tests, integration tests and API testing. I’m using the assertion library should.js for clean and readable tests.

2.4.2.1. Unit Tests



2.4.2.2. Integration Tests for Routs

Node web application routes can be tested using the Supertest library. Supertest can start up a node app, make a http requests to routes, and run assertions on http responses. It can pass parameters to routes and run assertions on things like the response http status code, content type, and execute regular expressions against the response body.

**NEED TO ADD AN IMAGE**

2.4.2.3. API Tests

Being able to test an external service that the application uses can be very helpful. If the external services are the main focus and tests are available for them, then when the app breaks it easier to determine if the problem is with the code base or the external service itself. It has been decided that two platforms will be used during the testing part, and each of them will be shortly described in the following section.

The test platforms [Mocha](https://mochajs.org/) and [Chai](http://chaijs.com/) makes it easily to asynchronously test an API endpoint. These platforms are used to test the application’s backend.

Chai is a BDD/TDD assertion library for node and the browser that can be delightfully paired with any JavaScript testing framework.

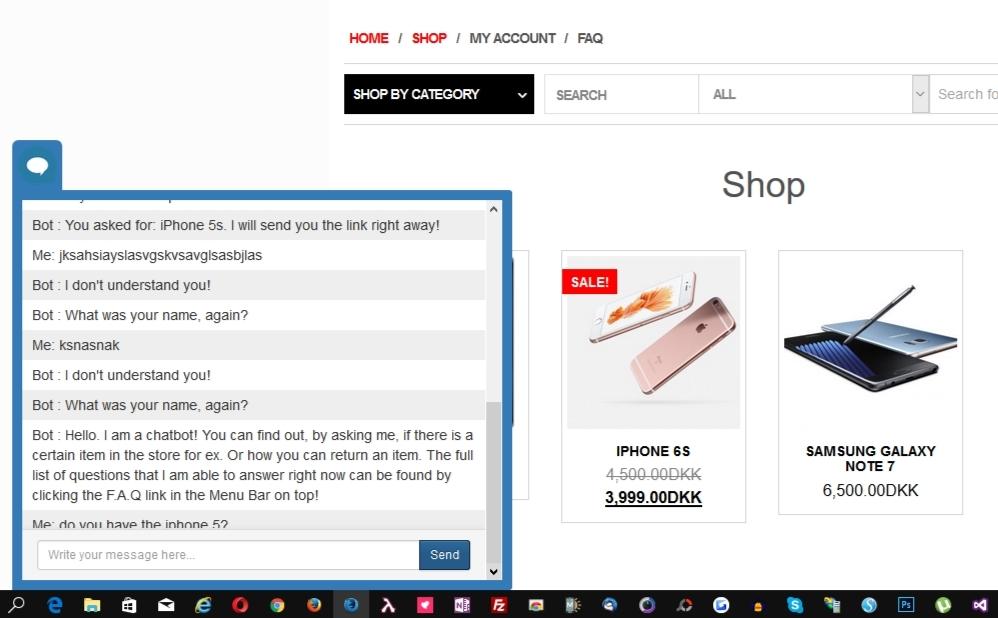
Mocha is a feature-rich JavaScript test framework running on [Node.js](http://nodejs.org/) and in the browser, keeping asynchronous testing simple. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases; it is hosted on GitHub.

**NEED TO ADD AN IMAGE**

2.4.2.4. Functional Testing

The application has been tested on the next list of browsers, running on Windows 10 64-bit Operating System. Most of the tests performed as expected, but one browser encountered a visual problem with the text format:

1. Microsoft Edge 38.14393.0.0 : OK
2. Internet Explorer 11.447.14393.0 : OK
3. Opera 41.0.2353.69 : OK
4. Google Chrome Version 54.0.2840.99 m : OK
5. Firefox 48.0.2 : Problem found->The bottom text is going under the Form field.
6. Firefox Developer Edition 51.0a2(2016-11-14)(32-bit) : Problem found->The bottom text is going under the Form field.



# Results and Discussion

# Conclusion

# References

Facebook, 2016. *Messenger Platform ᴮᴱᵀᴬ.* [Online]   
Available at: https://developers.facebook.com/docs/messenger-platform

Go Response, 2015. *Live Chat Statistics.* [Online]   
Available at: http://www.goresponse.co.uk/web-chat/

Microsoft, 2016. *Microsoft Bot Framework.* [Online]   
Available at: https://dev.botframework.com/

Reynolds., P., 2015. *Importance of Online Chat.* [Online]   
Available at: http://www.etechgs.com/blog/post/the-importance-of-online-chat.aspx

Wit AI Team., 2016. *Getting Started with Wit.ai.* [Online]   
Available at: <https://wit.ai/docs>

GitHub Inc., 2016. *BDD Style Assertions for Node.js*. - test framework agnostic. [Online]

Available at: <https://github.com/shouldjs/should.js>

Mocha js, 2016. *Mocha JavaScript Test Framework*. [Online]

Available at: <https://mochajs.org/>

Chai 2016. *Chai Assertion Library*. [Online]

Available at: <http://chaijs.com/>

# 6. Appendices

## 6.2. Appendix A: List of Abbreviations

The following abbreviations are used in this project description, and they will be explained below:

* FAQ: Frequently Asked Questions
* AI: Artificial Intelligence
* API: Application Program Interface
* BDD (assertion): Behavior-Driven Development
* JS: JavaScript
* TDD (assertion): Test-Driven Development

## 6.3. Appendix B: User Guide