# *CURRENCY TRACKING CHATBOT*

Analysis Report

*Fuad Aghazada*

*R.I.S.K Company | Internship*

# Table of Contents

# 1. Proposed System

## 1.1 Overview

In our busy lives people are always in search of efficient ways for obtaining necessary information for them. Since we are living in a world where data is growing in a dramatic way and technology is developing in a rocket speed, we are in need of properly managing the data according to our usage. Therefore, chatbots are one of the easiest ways for people in embedding efficiency to their lives. The main reason behind this 'easiness' is because of the fact that one does not need to be stuck in different mobile/web menus and UIs in order to access his/her needed information, but he/she just needs to have a conversation with a respective bot. For instance, our chat bot will be responsible for accessing currency information according to the input of the user. This would definitely help lots of people in being able to track the data about different currencies. It will be mostly a *reactive* [1] chatbot, which means it will return the data after the input of the user for the first versions. In the later versions, it would be updated to be a *scheduled* [1] chatbot, which would send currency information in a set of timeframe. To exemplify, according to the willing of the user, it would send currency information every morning. This currency information would be the best option in terms of the value of the currency or the location of the user. Shortly, this project would be very useful in terms of accessing currency information in an efficient way. In addition, in later versions of the system a database could be added in order to access the data for the previous dates.

## 1.2 Functional Requirements

- The user should be able to get the currency information by typing the name of the needed currency. For example, typing 'USD', the user should be able to return the currency information to the user. This will return the currency conversion according to the location of the user. For instance, if the user is in Turkey, it would return information of USD-TL conversion.

- Alternatively, the user directly can also enter directly base-rate information like USD-TL.

- Some commands will be accepted by the user as "/nameOfCommand":
  - /help → gives information about how to use the chatbot.
  - /list → lists all the existing commands.
  - /best → displays the most useful currency data according to the latest currency input of the user.
  - /nearest → lists the nearest banks (ATMs) according to the latest currency input of the user.
  - /schedule → schedules for informing the user with information about the latest currency input that he/she entered by asking time for informing.

- /all → returns all the information that has been found about the latest currency that has been entered by the user

### 1.3 Non-functional Requirements

- The system should be efficient both in terms of CPU and memory, because the data would be big, so it should not be a problem to parse and process the data accessed from APIs.
- The command will be adaptive and easy for users.

### 1.4 Pseudo Requirements

- The system will be implemented using Python 3 with its necessary libraries and Telegram Bot APIs.

## 2. System Models

### 2.1 Use case Diagram

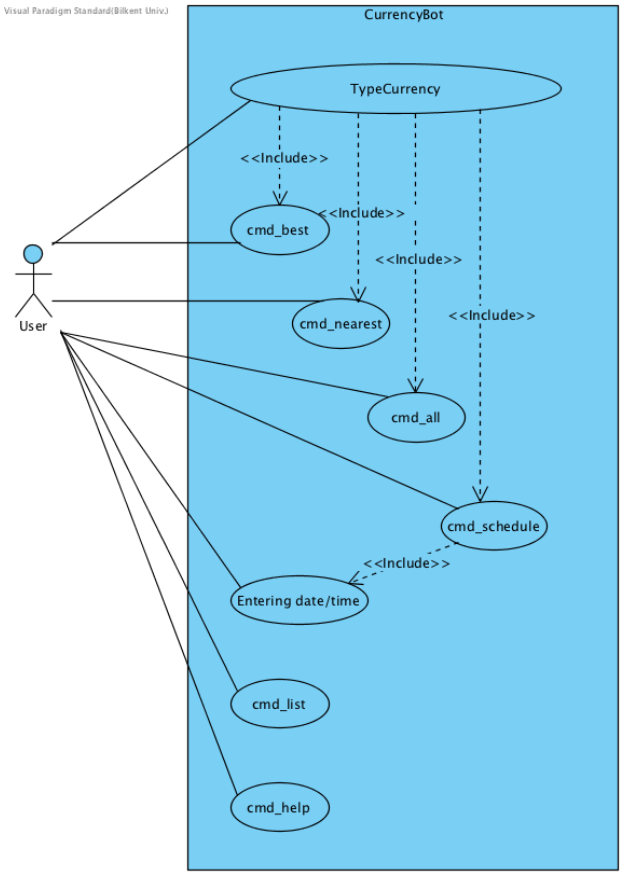


Figure 1. Use case diagram of the system

## 2.2 Use case Descriptions

### 2.2.1 Use case 1

**Use-case name:** Getting help.

**Participating actor:** Telegram user.

**Entry condition:** User opens the app and opens the chat with bot.

**Exit condition:** User exits app or moving to another chat.

**Main flow of events:**

1. User opens the app 'Telegram' and opens the chatbot.
2. User types '/help' on the text field of the chat.
3. Help information is sent by the chat.

**Alternative flow of events:**

1. User opens the app 'Telegram' and opens the chatbot
2. User types anything other than '/help'.
3. Bot send a message which contains the list of commands that he can understand.

### 2.2.2 Use case 2

**Use-case name:** List commands.

**Participating actor:** Telegram user.

**Entry condition:** User opens the app and opens the chat with bot.

**Exit condition:** User exits app or moving to another chat.

**Main flow of events:**

1 User opens the app 'Telegram' and opens the chatbot.
2 User types '/list' on the text field of the chat.
3 List of the chat commands is sent by the chat.

**Alternative flow of events:**

1 User opens the app 'Telegram' and opens the chatbot
2 User types anything other than '/list'.
3 Bot send a message which contains the list of commands that he understands.

### 2.2.3 Use case 3

**Use-case name:** Best choice.

**Participating actor:** Telegram user.

**Entry condition:** User opens the app and opens the chat with bot.

**Exit condition:** User exits app or moving to another chat.

**Main flow of events:**

1. User opens the app 'Telegram' and opens the chatbot.
2. User types the name of the currency on the text field of the chat.
3. A custom keyboard, which contains commands for the typed currency, such as best choice, nearest ATMs and Schedule.
4. User clicks on 'Best Choice'.
5. Most useful currency conversion is sent by the bot.

**Alternative flow of events:**

1. User opens the app 'Telegram' and opens the chatbot
2. User types currency name wrongly.
3. Bot sends a message containing 'Did you mean …'

### 2.2.4 Use case 4

**Use-case name:** Nearest ATMs.

**Participating actor:** Telegram user.

**Entry condition:** User opens the app and opens the chat with bot.

**Exit condition:** User exits app or moving to another chat.

**Main flow of events:**

1. User opens the app 'Telegram' and opens the chatbot.
2. User types the name of the currency on the text field of the chat.
3. A custom keyboard, which contains commands for the typed currency, such as best choice, nearest ATMs and Schedule.
4. User clicks on 'Nearest ATMs'.
5. List of ATMs is sent by the bot.

**Alternative flow of events:**

1. User opens the app 'Telegram' and opens the chatbot

2 User types currency name wrongly.
3 Bot sends a message containing ‘Did you mean …’

### 2.2.5 Use case 5

**Use-case name:** Schedule

**Participating actor:** Telegram user.

**Entry condition:** User opens the app and opens the chat with bot.

**Exit condition:** User exits app or moving to another chat.

**Main flow of events:**

1 User opens the app ‘Telegram’ and opens the chatbot.
2 User types the name of the currency on the text field of the chat.
3 A custom keyboard, which contains commands for the typed currency, such as best choice, nearest ATMs and Schedule.
4 User clicks on ‘Schedule’.
5 Bot asks a date and time in a specified format.
6 User sends a date and time.
7 Bot replies that he will inform you on this date and time.

**Alternative flow of events:**

**#1**

1 User opens the app ‘Telegram’ and opens the chatbot
2 User types currency name wrongly.
3 Bot sends a message containing ‘Did you mean …’

**#2**

1 User opens the app ‘Telegram’ and opens the chatbot.
2 User types the name of the currency on the text field of the chat.
3 A custom keyboard, which contains commands for the typed currency, such as best choice, nearest ATMs and Schedule.
4 User clicks on ‘Schedule’.
5 Bot asks a date and time in a specified format.
6 User sends a date and time in an incorrect format.
7 Bot complains in his reply that ‘Please enter the date in a correct format’.

**#3**

1 User opens the app ‘Telegram’ and opens the chatbot.

2 User types the name of the currency on the text field of the chat.
3 A custom keyboard, which contains commands for the typed currency, such as best choice, nearest ATMs and Schedule.
4 User clicks on ‘Schedule’.
5 Bot asks a date and time in a specified format.
6 User sends a date and time that has already passed.
7 Bot replies that the entered time has already passed, enter another date.

## 3. References

[1] L. Smiers, "Why do we need chatbots?," Capgemini, 19 December 2016. [Online]. Available: www.capgemini.com. [Accessed 14 June 2018].