2022

System Design Document for Translator Telegram Bot

To translate messages in native language to desired language automatically

LIU ZI'AN

Contents

[Problem Statement 2](#_Toc101027406)

[Proposed Solution 2](#_Toc101027407)

[Technologies, Tools and Methodology Used 2](#_Toc101027408)

[User Stories 3](#_Toc101027409)

[System Design and Architecture 4](#_Toc101027410)

[Design & Architecture Elaboration 5-6](#_Toc101027411)

[Class Diagram 7](#_Toc101027412)

# Problem Statement

Back when I was an intern in a Fintech company, I was assigned to a team comprised of many developers from different parts of the world like Vietnam and China. When communicating, often, developers would translate their message on Google translate to a language the receiving end could understand, copied then sent the translated output to the recipient. Such manual process took up unnecessary time and effort, affecting the communication flow among the developers.

# Proposed Solution

The proposed solution is to automate the translation process by utilizing Telegram Bot. By adding the Bot into the developers’ group chat, each developer could enter a simple command to indicate what language they would like their message to translate to. The Bot would automatically translate the sender’s text to the sender’s indicated language then send to recipients.

# Technologies, Tools and Methodology Used

The following list is the technology and tool I have decided to use, where each is coupled with a reason why:

* Python 3.5 for development mainly because there exists an open-source Telegram wrapper python library, named Telethon, available for use. This library provides high level interface to Telegram API, making it convenient and easy to develop the Bot.

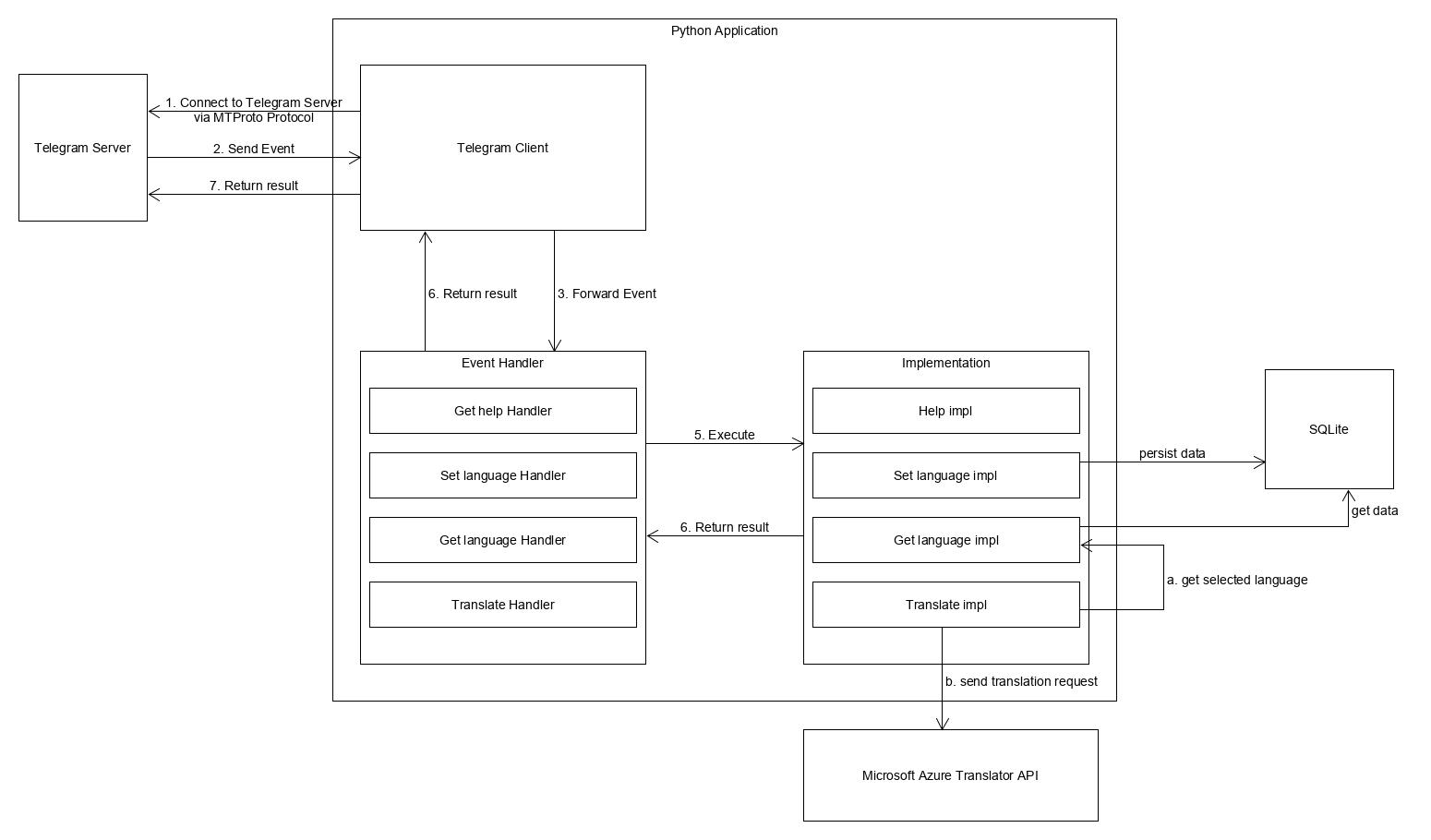
More importantly, it supports asynchronous implementation which is useful especially when our Bot interacts with external servers via REST API for example. So, while waiting for response from external servers, our Bot can still attend to other users’ request instead of waiting idly for response and blocking the entire program.

* Microsoft Azure Translator API for translation service because it is the most affordable and very easy to set up.
* Visual Studio Code IDE because I can organize my codes better and the environment makes writing code more enjoyable.
* Agile methodology because I want to implement features in small and increment fashion so that development is more manageable, and testing is easier.

# User Stories

* As a user of the Bot, I want to see a brief guide on how to use the Bot so that I know how to use the Bot.
* As a user of the Bot, I want to select a language to translate my message to so that the Bot will automatically translate my message to the selected language option.
* As a user of the Bot, I want to be able to edit my selected language option so that I can make changes to my selected language options.
* As a user of the Bot, I want to be able to get my selected language choice so that I know what language I have selected.
* As a user of the Bot, I want my messages to be translated as per my selected language choice so that my messages can be automatically translated.
* As a user of the Bot, I want to have an option to disable translation so that my messages will not be translated.
* As a user of the Bot, I want the Bot to remember my selected language option so that I do not need to reselect my previous language options.

# System Design and Architecture



*Figure 1.1 shows the architecture of the entire System*

# Design & Architecture Elaboration

* **Telegram Client and Server Connection via MTProto Protocol**

MTProto protocol is designed for access to Telegram Server API from applications running on mobile devices. It is especially reliable when a mobile device experiences weak connections or handles very large data files. Since there is no HTTP connection, which means no polling or webhooks, this reduces overhead. Hence, we have chosen this protocol to connect to Telegram server for the above-mentioned benefits.

* **Event Handlers**

This bot handles 4 specific types of events:

1. when users type “/help” command,
2. when users type “/set” command to specify and select a language to translate,
3. when users type “/get” command to retrieve what language they have chosen,
4. when users send normal text messages.

When any of such event occurs, the Event Handlers will forward the events, as Objects, to their respective implementation as input.

* **Implementation**
  + **When user types “/help” command**

The command will call Help Implementation where a guide to using the Bot will return as output and then send to user. The guide will include a basic greeting, what the Bot can do, what command to set a language to translate to, and command to get selected language.

* + **When user types “/set” command**

The Bot will reply to the command with a list of available languages option for user to select. Once user has selected, the Bot will call Set Implementation where the following fields, including the selected language, will be stored to SQLite as user’s record:

1. User id to uniquely identify the user.
2. Chat id to uniquely identify the group chat the user is in.
3. Selected Language as chosen by the user.

If user does not want their messages to be translated, they have the option to select “none”, and the Set Implementation will write or update the user record in the SQLite according to user selection.

If user already has selected a language, the next selection will replace the previous one. The change in user’s record will be updated in SQLite.

* + **When user types “/get” command**

The Bot will call Get Implementation where it will query user’s record from SQLite. If no records are found, it means that user has not used the Bot before or is using the bot in a different chat. The Bot will reply by stating user record is not found and will then prompt user to call “/set” to start the translation service.

* + **When user sends normal text messages.**

Whenever a user sends a normal text message, the Bot will check if the sender has selected a language by calling the Get Implementation. If None is returned as output, the Bot will not translate.

Else, the Bot will do the following in such sequence:

1. Translate sender’s original text by calling Microsoft Azure Translator REST API.
2. Reply to user original text with the translated text.