# Build Personal Translator Application Using Language Translation API With IBM Cloud

By Ritika Sarkar

## 1. INTRODUCTION

### 1.1 Overview

Language translation means converting a piece of text in a given language to another language of the user's choice. In this project, I have created a translator application using APIs. Here, the RAPID API service has been utilized for creating the Translator app, with HTML, CSS and JavaScript for the frontend and Python's Flask module for the backend.

### 1.2 Purpose

The project will help me get an in-depth understanding of the working of APIs, the working of Flask module and deploying a finished web application on IBM Cloud.

## 2. LITERATURE SURVEY

### 2.1 Existing solutions

In [1], the authors have evaluated the accuracy and usefulness of Google Translate in translating common English medical statements. Treatment of local or foreign patients who do not speak English or the dominant language may cause improper treatment because of differing interpretation of medical terms by the patient. The Translate feature has proved fairly accurate in an applied treatment of a patient, hence validating the study.

[2] discusses the application of Translate features in student education. It speculates that students may rely on translation apps to complete their assignments in a second language. Based on the statistics, the perfomance of the translator services keep improving with respect to the grammatical errors and context, hence the paper argues that this technologu will have a profound imapct on the teaching og Languages for Academic Purposes.

The paper [3] analyses the use of Google Translate as a supplementary tool for helping international students at Universiti Sains Malaysia to learn the Malay language. The studies conducted on 16 international students report that the students could optimally use Google Translate as a self-learning tool to learn the language.

[4] highlights the impact of translation tools and advancing technology on the coaching

of human translators. It remarks that automation has influenced the translation trainers to incorporate these tools in their classroom teaching and also seeks to evaluate the importance translator agencies give to translation technology versus human translators.

### 2.2 Proposed solution

The solution proposed in the project harnesses the power of RAPID API's Language translation V2 service. A flask application is made using Python to handle the backend routes and form handling and the user interface for interacting with the web app is made using HTML, CSS and JavaScript languages. The user selects the target language and inputs the text in the user interface, and after the button is clicked, the submitted POST request is handled by custom function in the flask app, which makes the payload for the API call and renders the translated text on a new page upon receiving the response. Initially, a preliminary GET request is made to fetch the languages supported by the API and it is stored in the form of a json file in the 'static' directory. It is later on accessed using AJAX and the contents are appropriately passed on to the correct html tag in order to be displayed to the user in the form of target language options. The language of the text to be translated is detected by the API before translating it to the chosen language. Finally the solution is deployed to IBM Cloud using Cloudfoundry CLI.

## 3. THEORETICAL ANALYSIS

### 3.1 Block diagram

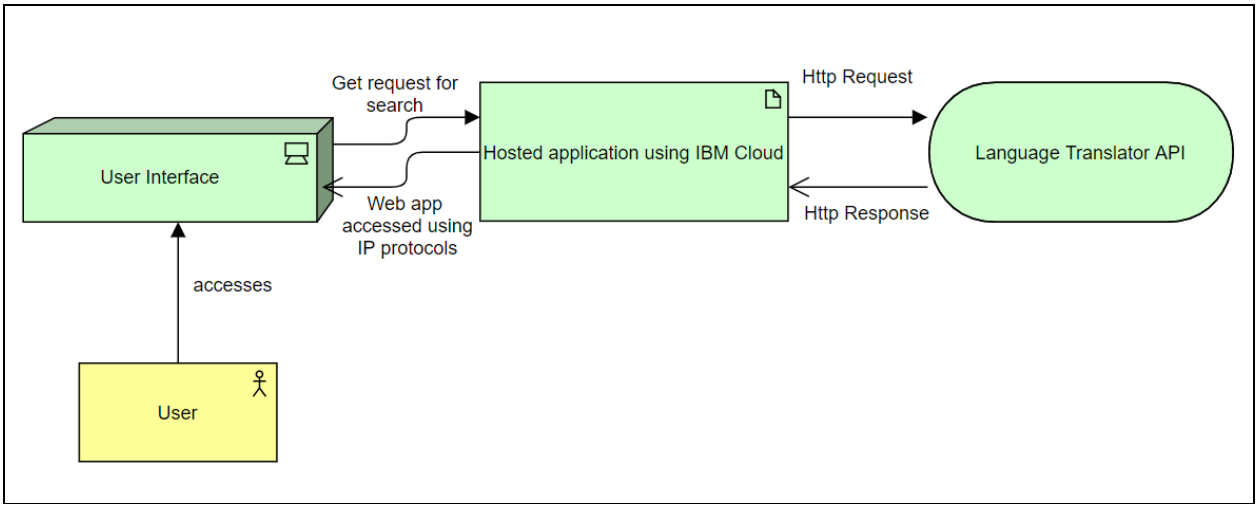



Fig 1. Overview of the project

3.2 Hardware / Software designing

1. Hardware requirements

There are no specific hardware requirements. A standard Windows operating system type architechture is given below for reference:

| | | |
|---|---|---|
| 1. | Processor | AMD Ryzen 5 4500U with Radeon Graphics 2.38 GHz |
| 2. | Installed RAM | 8.00 GB (7.37 GB usable) |
| 3. | System type | 64-bit operating system, x64-based processor |

2. Software requirements

The software modules required for the project are:

i. Python 3
ii. Flask module
iii. gevent module
iv. VSCode (or any other text editor)
v. Cloudfoundry CLI (for deploying to IBM)
vi. IBM Cloud account
vii. Bootstrap
viii. Jquery (Ajax)
ix. HTML, CSS, JavaScript

## 4. EXPERIMENTAL INVESTIGATIONS

While implementing the project, some issues were found with the rendering of official Bootstrap styles at first, but the correct positioning of the JQuery CDN inside the head tags fixed the issue. The next troubling issue was the testing of the submitted form contents in the translator page. It was fixed after identifying a mistake in not having included the name attribute of input tags, which is a very essential part of forms. The storing of the supported languages by the API returned as a json response, as a json file required careful planning as to how the contents will be accessed. Finally, the display of the target language options in the select > option tags was a challenging task as well. The very last straw was the deployment to cloud, which required a trial account in IBM. Also the pushing of the local files to IBM cloud's Flask app using Cloudfoundry CLI required some setting up in existing files and troubleshooting and inclusion of new files as well, like manifest.yml.

After the successful running of the app on the cloud server, it was observed there was some noticeable latency between the request sent for translation and the loaded page with the received response. Apart from that, all other functionalities were working as described in the

proposed methodology.

## 5. FLOWCHART

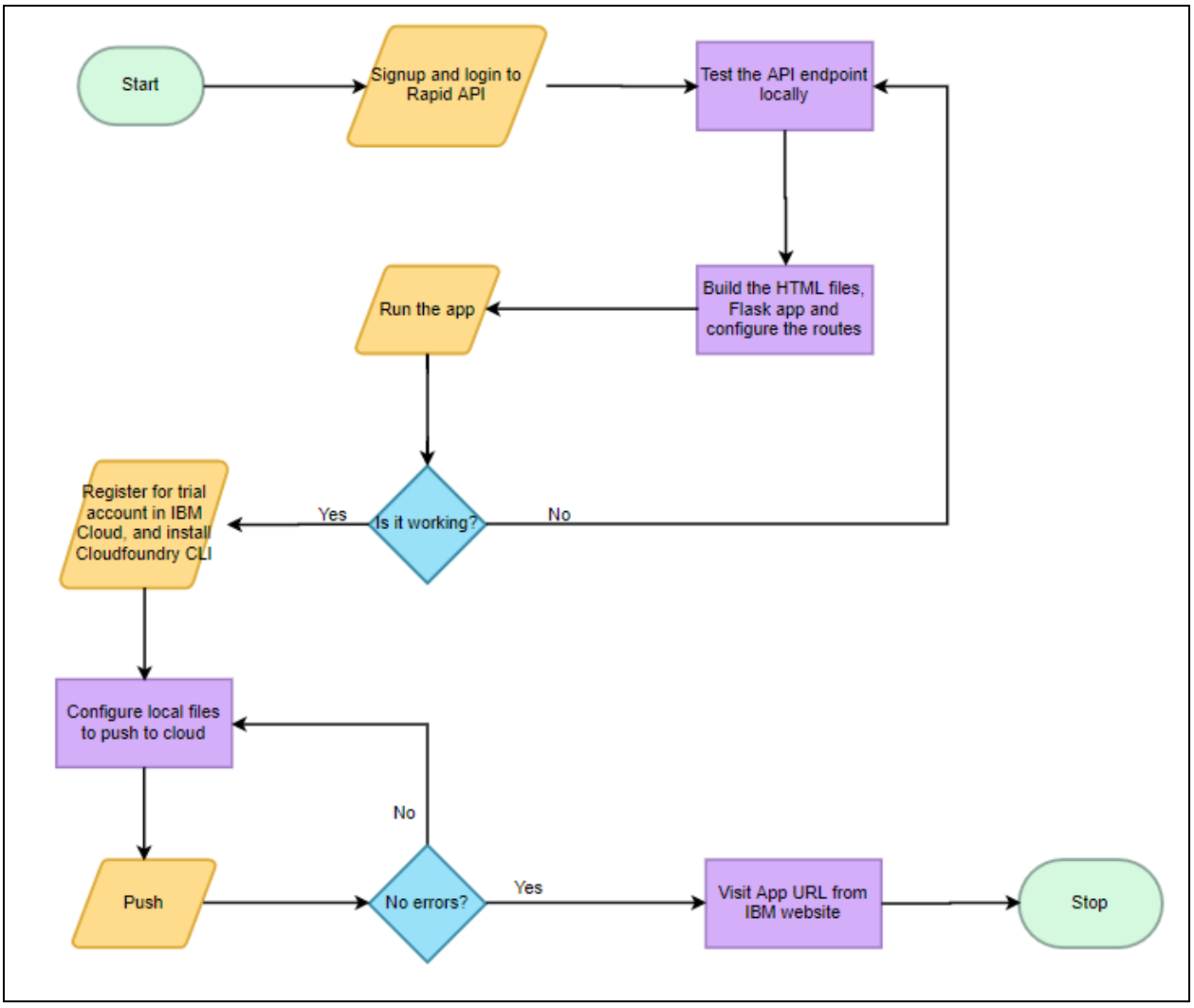
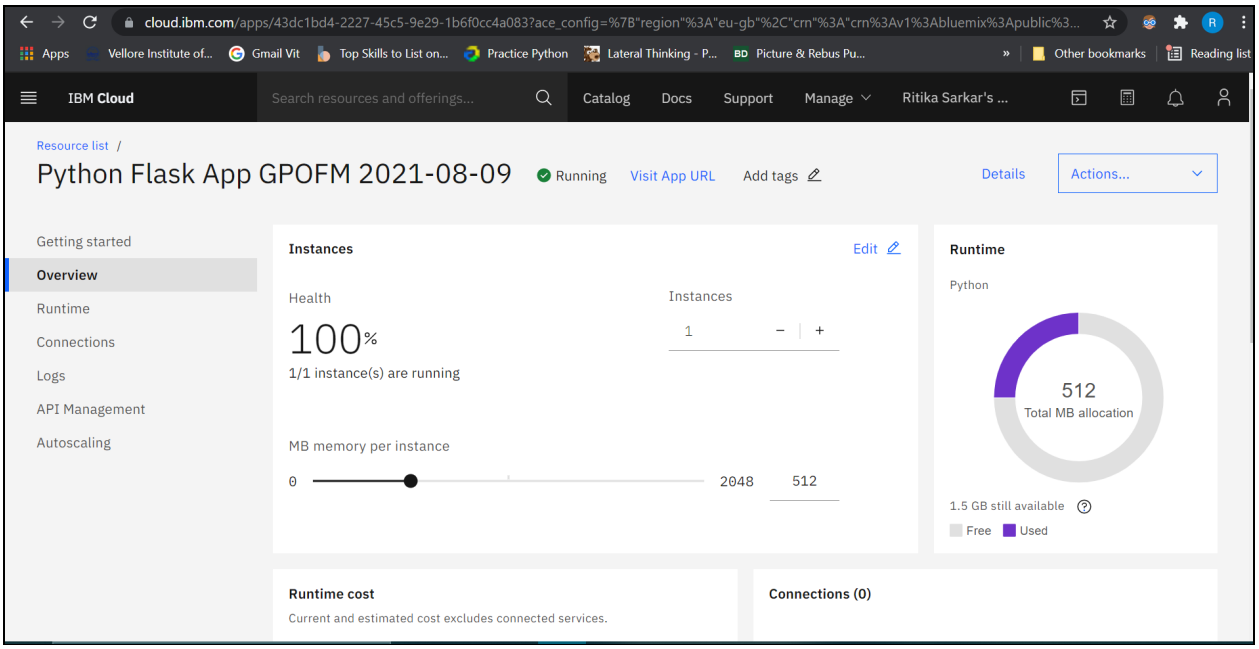


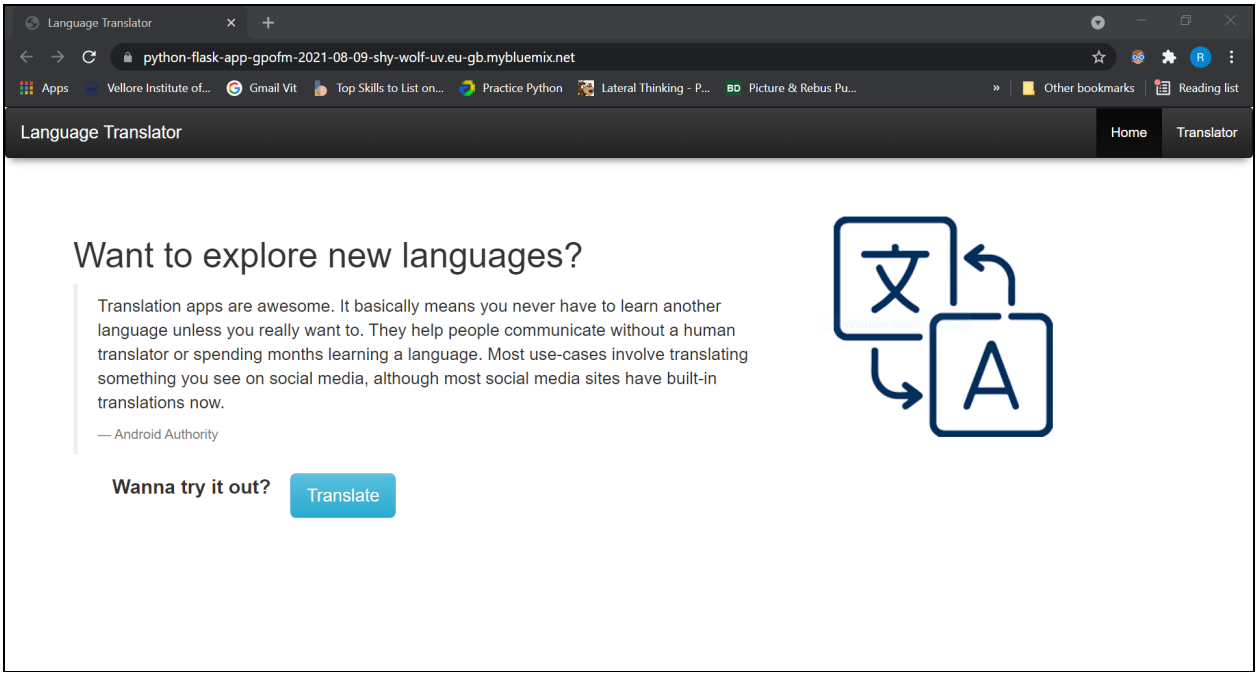

Fig 2. The flowchart describing the flow between different components of the project

## 6. RESULT

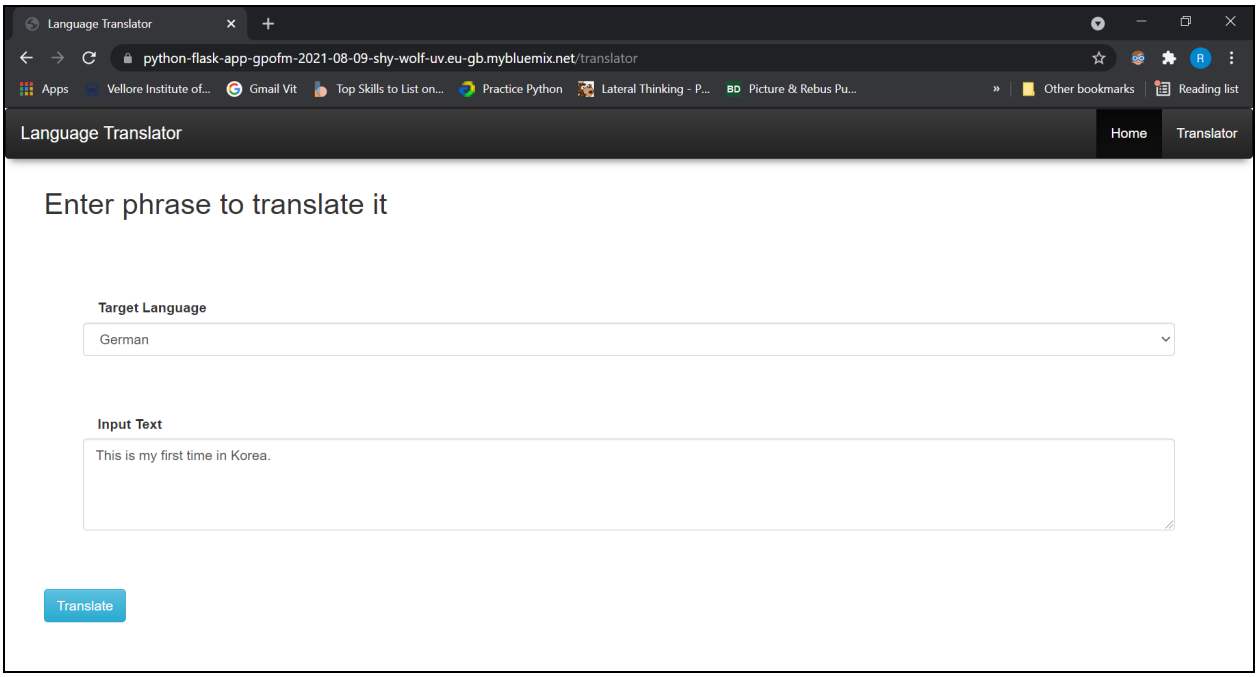
The following screenshot shows the flask app running on IBM cloud after having been deployed.



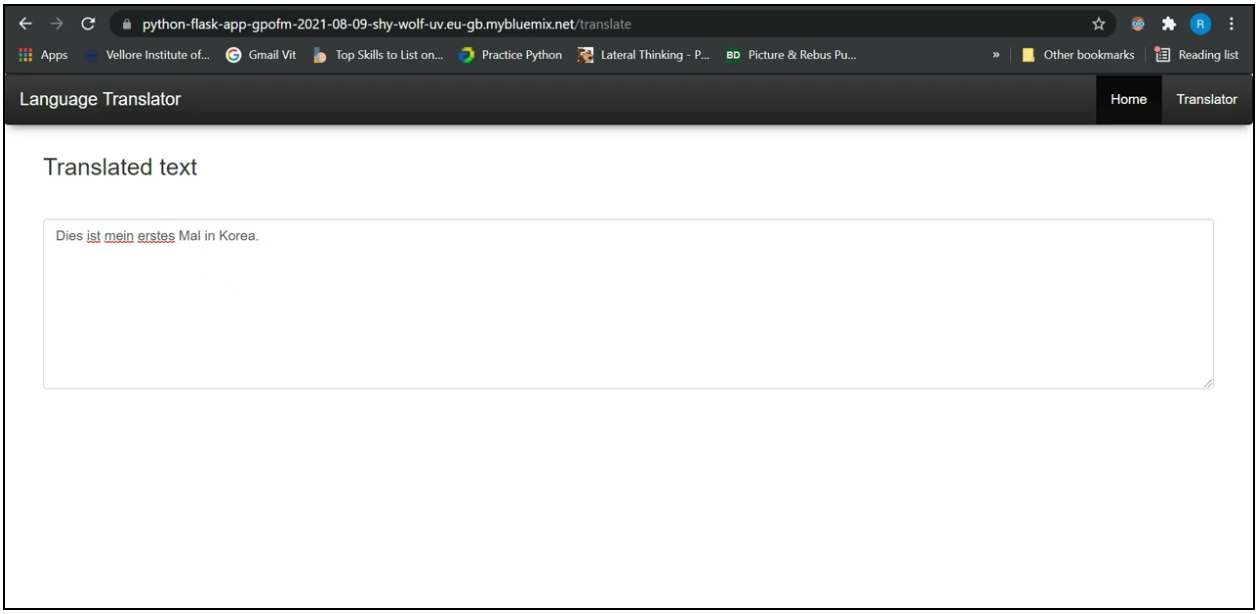

The following screenshot shows the home page of the web app. It opens in root("/") route of the hosted url. You can click on translate button or Translator in the navigation bar to move on to the translator page.




Below you can observe the translator page where you can enter your text and target language. Clicking on Translate button in this page sends the form data as a POST request to the translate page.




In the following image, you can observe that the translated text as been displayed inside the textarea. You can copy it and use it in any of your other applications.




## 7. ADVANTAGES & DISADVANTAGES OF THE PROPOSED SOLUTION

The advantages are:

1. The API has perfect translation (so far) and the solution works properly as proposed

above.

2. The API detects the language of the input text properly, and all the routes work properly in the page.

The disadvantages are:

1. The application currently can handle only 10 API calls/day, as it is based on the pricing plans of the API, so cannot apply to a larger user base.
2. There is no way to control how the translation is done by the API, because here the main focus is on integrating the API with the solution. If there are grammatical errors by the user or sentences without context, the behavior of the API is unknown. In that case, we may have to go for more complex custom implementation using deep learning.

## 8. APPLICATIONS

The language translator APIs can be integrated into applications, websites, tools, or other solutions to provide multi-language user experiences. The solution can be applied in the tourism websites, healthcare industry for foreign patients, and for business purposes in larger organizations.

## 9. CONCLUSION

In this project, a web application has been created using Flask, HTML, and Web APIs. The application successfully takes input from the user and also presents a list of languages to convert the input to, and uses the above-mentioned API to translate the input text and displays the accurately translated text on a fresh page to the user. The creation of the application proved to be a good learning experience for exploring a new api interface and integrating it with modules and languages like Flask and HTML. Further work is proposed to create a mobile application and release it as chrome or firefox extension.

## 10. FUTURE SCOPE

The future works planned include implementing language translation using Neural Networks, and creating mobile application and/or releasing the application as a chrome or firefox extension.

## 11. BIBILOGRAPHY

### References for Literature Review


1. Patil, S., & Davies, P. (2014). Use of Google Translate in medical communication: evaluation of accuracy. Bmj, 349.
2. Groves, M., & Mundt, K. (2015). Friend or foe? Google Translate in language for academic purposes. English for Specific Purposes, 37, 112-121.
3. Bahri, H., & Mahadi, T. S. T. (2016). Google translate as a supplementary tool for learning Malay: A case study at Universiti Sains Malaysia. Advances in Language and Literary Studies, 7(3), 161-167.
4. De Cespedes, B. R. (2019). Translator education at a crossroads: the impact of automation. Lebende Sprachen, 64(1), 103-121.


## APPENDIX

### A. Source Code

The source code has been uploaded in the following GitHub repository:
https://github.com/smartinternz02/SPS-11419-Build-Personal-translator-application-using-Language-Translation-API-with-IBM-Cloud