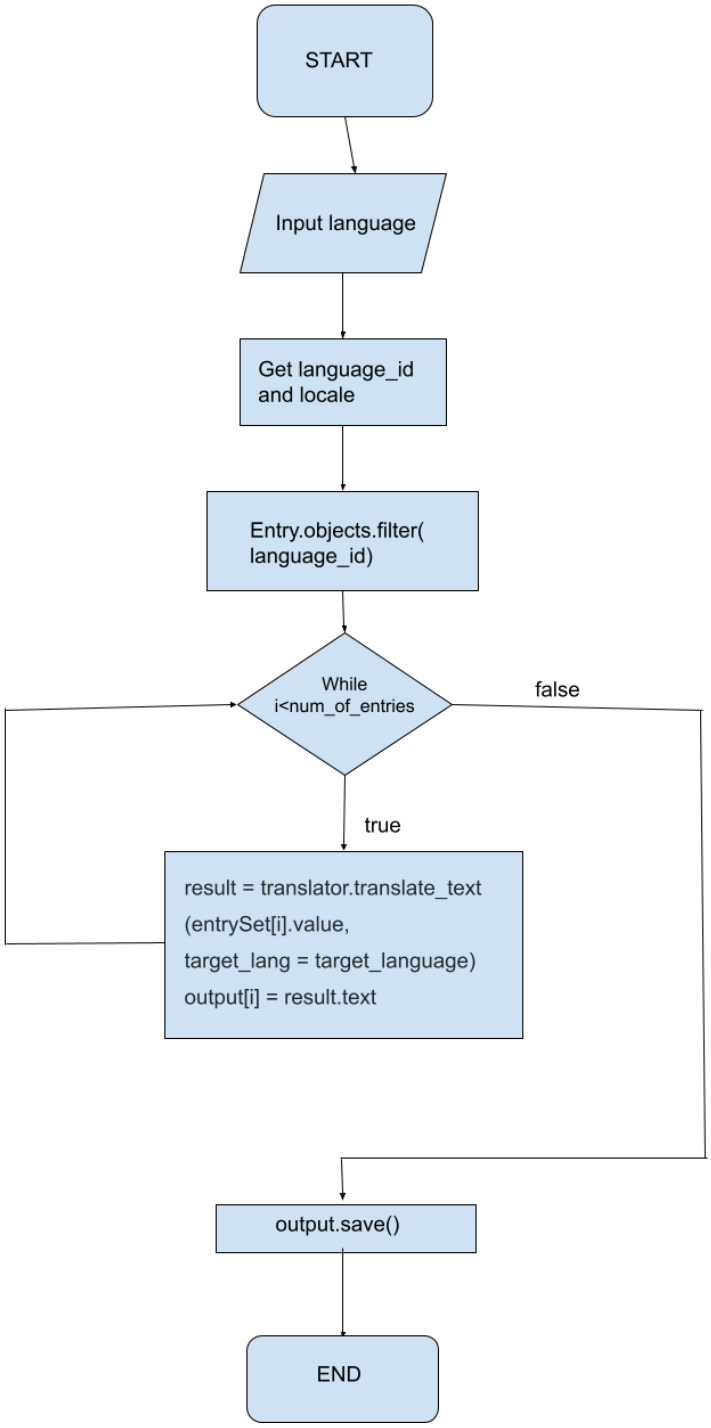
Table translation conceptual solution

Program flow chart

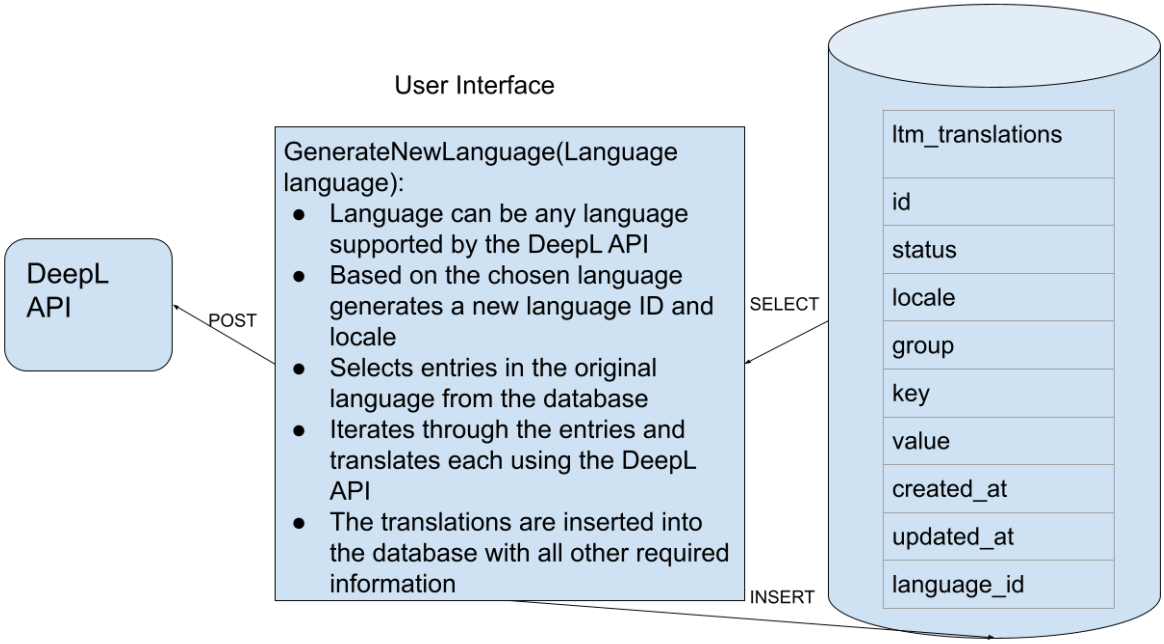


Tech stack

The below described solution would be implemented using the following tech stack:

- MySQL database (as defined in the provided sql file)
- Python backend with the Django framework which provides functions to select entries from the database and insert new entries to the database (shown in the program flow chart)
- DeepL API - integrated python client which provides functions to translate text to a target language (shown in the program flow chart above)

Solution overview



The functionality of translating the data to a new language is offered to developers through the GenerateNewLanguage function. This function accepts one argument which defines which language should be added to the database. The type of this argument is a Language enum which contains all languages which are supported by the DeepL API.

The GenerateNewLanguage function adds a new language to the database in a few steps:

- Fetches all entries which are in the original language from the database as follows:

```
Entry.objects.filter(language_id = 5)
```

    - Possible consideration: if the database becomes very large, we could select the data in chunks and process each chunk separately as described below
- The selected language is used to retrieve the locale and language_id from a dictionary in which the keys are Language enum values, and the values are dictionaries that have locale and language_id as keys and their respective values (example shown below).

```
class Language(enum.Enum):
    GERMAN = 1
    FRENCH = 2
    ITALIAN  = 3

locale_and_id_dict = {Language.GERMAN:{'locale':'de_DE','language_id':1},
Language.FRENCH:{'locale':'fr_FR','language_id':2},
Language.ITALIAN:{'locale':'it_IT','language_id':3}}
```

  Alternatively, a lookup table could be added to the database which would contain the language_id, and the locale for every language in the database.
- Iterates through the selected entries, and translates each entry using the DeepL API
- Constructs a new entry by generating other needed data (group, key, created_at, updated_at, status, id). Group, key and id are copied from the selected entry in the original language, while the created_at and updated_at fields are taken from the current timestamp while creating the entry.
- Inserts the newly generated entries into the database using Django functionality: output_list.save()