Describe all the assumptions and design choices applied during the project's development

Assumptions

- Users are expected to have basic knowledge of using translation systems
- Users are expected to have access to language ISO list which values are expected to be data for “source_language” and “target_language” values
- Users are expected to have access to domains supported by translation system

Design choices

- The application is designed as a monolith because it serves as a mediator between the actual service and the client. Its primary function is to validate requests to the real service. Therefore, there's no need to adopt a microservices approach.
- In this specific scenario, there was no need to have a database since there isn't an actual API for the adapter API to connect to. Instead, Singleton classes were created to simulate the functionality of a database.
- Data transfer object (DTO) is used for data transfer between the client and the API
- The initial validation method chosen checks whether the parameters provided via the DTO are either empty or not passed at all. This method uses the Spring Validation library and include custom messages for enhanced customization. To achieve this, a “ControllerAdvice” class was developed to intercept default messages and return a suited response.
- The second validation method chosen involves verifying the presence of specified values in the database, along with checking the length of the input content.
- Error handling is managed using custom exceptions, which provide a selected message alongside the corresponding HTTP response status directly associated with each exception.
- The message content has been relocated to a collective constants class in order to centralize all messages in a single location.
- The customized string functionality was put into a single location by creating a utility class to handle string operations like character deletion and word counting.
- To improve the readability and transparency of the code inside the service classes, a significant amount of code that was initially found there has been moved to helper classes.
- Two methods have been configured to utilize a “cron” job for daily invocation of an external API, gathering information regarding available languages and domains.
- Added Aspect Logger for basic in-app logging on start, end and errors in methods

Describe what changes and additional tasks are needed to have the project and implementation fully production-ready

The future steps required for the application to go into full production-ready state are:

- To set up a database that will store the languages and domains it receives from the external API
- To change singleton repositories into real repositories that communicate with the base
- To replace the mock method with a method that calls the external API and returns its response