Documentation of Transliteration API

Continuing from the work already finished by my predecessor I obtained the code from Dr. AGR sir. Some of the major issues that I observed was that the code was done in two parts or in two separate ways like Tamil has been transliterated in a particular fashion and the other languages in another. Second issue was the total number of if-else conditions was too much for a particular language and I felt that it would reduce the overall efficiency of the plugin so I felt that the best solution to both the issues is to build an API that can be used to transliterate any word from any source language to any destination language. Below I have briefly explained the API that I have built. As I have done only from Telugu it has only one set of stand alone code as Telugu and Kannada have one to one mapping. When I start doing other languages like to Tamil which requires some preprocessing then we can give separate functions to each language pair.

1. **Mapping of Unicode**: the chrome plugin built by my predecessor worked on adding a particular offset to a character belonging to one particular script in order to convert it to a character in another script. But this gives rise a number of anomalies which have to be handled that require we put a separate if-else for it. Hence my logic is that we can have a database which acts like a lookup table which acts like a map from a source language to any destination language. The structure of this database is like the first column is the source language and the following columns belong to the destination language and for each character in the source script I query out the corresponding character in the destination script. This removes the need of having too many if-else conditions and hence we just need to take care of the inter-language anomalies which takes care of the first issue.
2. **How to request the API**: in order to access the API, we need to send a request and to it we would get a corresponding response. As with any API there must a standard way in which we need to request it for my API which is nothing but a simple named php file called translator.php running on my local server can be accessed with the following structure:

http://localhost:8888/translator.php/<source\_language>/<destination\_language>/<request\_unicode>

eg: consider that I want to convert from Telugu to Kannada then in place of source language and destination language would be replaced by Telugu and Kannada respectively. To make the API language independent the conversion of the word to its Unicode sequence is done locally and then this sequence is sent along the request URL. The Response to this is in JSON(JavaScript Object Notation).

1. **Advantages of having an API**: one of the major advantages of having the API is that of reusability the entire business logic can be put into the php file and can be uploaded to the server and can be accessed from any language source. It also future proofs the logic. Like if an android app has to be done for this transliterator then we don’t need to paste the entire business logic into the app locally but instead make the required API calls.

Furthur details of implementation of them I have added as comments in the code. Please tell if any anomalies or changes have to be made.