**Restful API frame**

The REST is the abbreviated from of (Representational State Transfer), and the RESTful API describes an API which supports the RSET features. A RESTful API allows the client to instruct the server to take actions on the resource (with URI as the identifier). According to its fundamental characteristics we can found that, the definition highly overlaps with the HTTP protocols. Actually, in RESTful frame, HTTP is the one used to transfer commands.

For this project, I plan to use the Javalin frame provided by SA to build the Restful API for it was based on a very light weight structure and the convenience of loading in. I designed six functions for the API: List, Exist, Compare, Upload, Download and Delete. The functions will be mapped with different HTTP commands (List, Exist, Compare, Download for the “GET”, Upload for the “POST”, and Delete for the “DELETE”), and provide the function name in the URI string, provide the parameters in the body array, UPI parameter or the header.

**Server System design**

Once the main function of Server Class (the central controller which also work as the server of Javalin) receive the command from the client, it will invoke the corresponding method in the TextService class. The TextService is the analyzer of the system, it handles the command types and do preparatory or follow-up works like executing primary exams or read in the file according to the files. In particular, if it handles with the list or the exist case, it may directly invoke the storage program (specific methods in TextDao class); if it handles with the compare case, it may calculate the specific number of simple distance and Levenshtein distance and return a failure case if there are no such files in the database; if it handles with the upload case, it should read in the context from the file path and check whether the md5 code matches the context; if it handles with the download method it should also different failure case depends on whether the file exist in the database or whether the output path is available; if it handles with the delete case, it will check whether the command demands to delete all the files or only a particular file, and return failure case if where is no such file in the database.

After the analyzer complete the preparatory works, the storage program (TextDao class) will be awaken. The program is constructed only for simple interactions (select, insert and delete)

For list command, it will return a list containing the Document-Abstract objects; for exist command it will use the select function to return a boolean to describe whether the file exists in the database; for compare function, it will return a string array containing the contexts of the compared files or a null with no such files in the database; for upload function, it will return true when successfully insert the file into the database, false when the file already exist in the database; for download function, it will return the text if the file is successfully download to the directory, otherwise it may return null for no such file or a failure case string describing the error as “output path not available”.

The response of the storage program will be edited by the analyzer and sent back to client.

**Client design**

I have developed two sets of client systems: a sample client based on scanner; and a more stylish client system with GUI.

The basic client is based on the scanner, the user should first type the name of the operation, and type the md5 code or the file path into the terminal (the parameters should be separated. Users can also type the context directly, but a key word “type” in front of the command.

The advanced GUI client will built a main menu on the left side of the platform basing on a ClientGUI.fxml, with RootController as its controller. Users could press button the button to choose the functions. Once the function was chosen, the right side of the GUI will be replaced by the operation interface, and controller authority will be transferred to the specific controller. The user could directly type the md5 code or the context into the text-field. While if the user wants to read file from the internal storage of the client, the user may press the file path button to open the file chooser window. In the end, the result will directly shows on the bottom of the operating window.