Design Document of Project

*Jiajun Yu 11711102*

This is the design document of the project of CS209A course. This document can be divided into three parts, which are architecture, operations, and other functions. The "architecture" part describes the brief introduction of where the resources come from, and the identification of the resources. The "operations" part describes the abstract information of each operation. The "other functions" part describes the functions that have not been stated in the "Project Description" document.

Firstly, the RESTful architecture. "REST" stands for "Representational State Transfer". According to the blog by Yifeng Ruan[1], the pattern of resources is the "representations". The resources are local text files for this project. When the client asks for operating the server, it needs a state transfer based on representations. For this project, the state transfer is implemented by the get and post command from HTTP protocol.

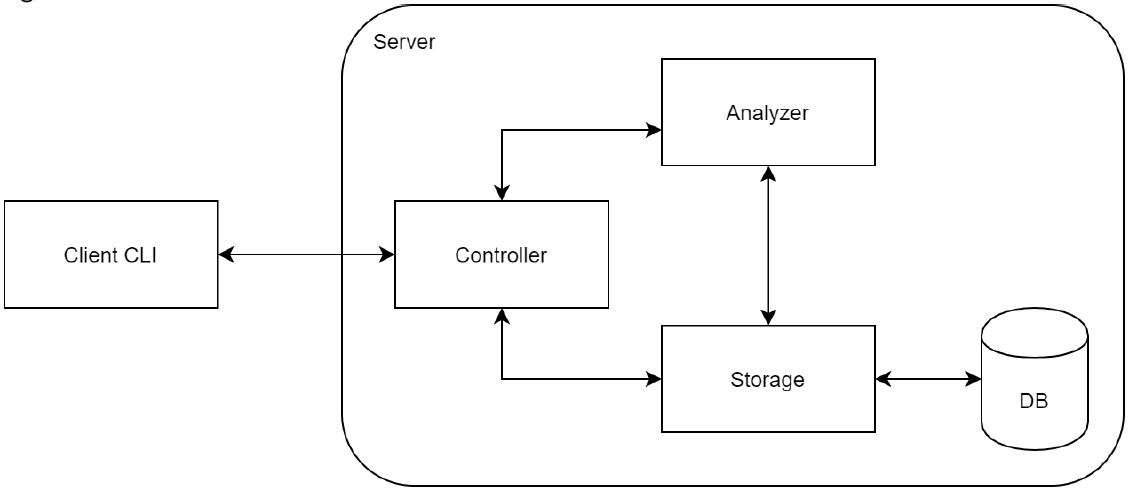


Fig.1 structure of the project

The Client is implemented in "Client.java". For the Server, "Server.java" implements the Controller part, "TextService.java" implements the Analyzer part, and "TextDao.java" implements the Storage part. The files are stored in the table "documents" in "Doc.db", and the table contains three columns that store md5, length, and content, respectively. After the "Server" class received the command sent by "Client", it would resolve the command and send the operation parameters to "TextService". Then "TextService" would ask "TextDao" to do some queries or insertions to the database. "TextDao" would later return results of the queries and insertions to the "TextService" and afterwards, the "TextService" would analyze the results and send them back to "Server". "Server" would then send the response back to "Client".

Exists command is used to check whether the file exists in the database, the client would send a get command with a header of "/files/md5\_sum/exists". The server would return the checking result. List command is used to get the file list of the database. After the client sends the get command "/files", the server would return all entries of the database.

When uploading, the client would send a "post" command, which has a header with the md5 sum of the file content and a body containing a byte array that is generated from the content of the document, to the server. After that, the server would check whether the document exists in the database and if not, the server would create a new entry in the database, then send a success response back to the client.

When downloading, the client would send a "get" command, which has a header with the md5 sum of the file content. The server would then check whether the file exists in the database and if yes, the server would send a success response with a body of the file content to the client, and then the client would create a file in the download path, and write the content to the file.

When comparing, the client would sent a "get" command that has a header like "/files/md5\_sum1/compare/md5\_sum2". Then the server would first check whether the files exist, and later return the simple similarity and Levenshtein distance. The Levenshtein distance is calculated by dynamic programming.

I also implements GUI for this project. "Client.java" is used to connect with the server as mentioned in the third paragraph. This part can be understood as the Model-View-Controller pattern. The controller part is implemented by "Controller.java", which would handle the button click for each operation, and pop up the FileChooser window. "ClientUI" would show the GUI stage as the "view" part of MVC. It would also be used to connect the controller with the model. The "Client.java" can also be the model part of MVC, for it can be used to store the result from the server.

Reference:

[1] Ruan, Y., 2011. Understanding of Restful Architecture. [online] Ruanyifeng.com. Available at: <https://www.ruanyifeng.com/blog/2011/09/restful.html> [Accessed 30 May 2020].