Independent Study Final Report

Haoxuan Li

Yicheng Tu, Mou Chengcheng

CIS 4900 Summer 2020

Contents

[Overview 3](#_Toc44873221)

[Software Design Description 3](#_Toc44873222)

[General setup 1 3](#_Toc44873223)

[General setup 2 6](#_Toc44873224)

[General setup 3 6](#_Toc44873225)

[Test cases 7](#_Toc44873226)

[User system 7](#_Toc44873227)

[Receiving data 7](#_Toc44873228)

[Reorganizing data and write to output file 7](#_Toc44873229)

[Future enhancement 8](#_Toc44873230)

# Overview

The purpose of this project is to receive data input from web browsers and reorganize them into a JSON file. Users can submit their queries by entering plain text or uploading local files. All inputs are categorized by user ID, timestamp. Queries are put into an array and each query can be accessed by array index. Each metadata(queries.json, buffer.json,all output file) is in the form of JSON file where each element in the JSON file is an object and can be accessed by index.

Users have to create account and log in to use the web. Users can only see their own submissions and results. The server is capable of handling multiple inputs from different users within a short time period.

* The code is written in JavaScript
* Communication tool: socket.io
* User system is implemented by using passport and PostgreSQL.

# Software Design Description

## General setup 1

**Each query must end with a ‘ ; ’ otherwise the code won’t work**

Each input(file/text) is limited to 150KB About 574 queries

All text inputs will be written to queries.json first. queries.json and output file are limited to 1M. When queries.json is full, new inputs will be written to buffer.json. There is no limit on buffer.

**Queries.json and buffer.json and output files** are formatted as follow: A screenshot of a computer screen

Description automatically generated

In JSON file, every element is a detailed object with UID, time stamp, etc. In the above example, id 10 and all its data is in index 0, id 8 and all its data is in index 1.

Every user has a unique user ID. Each input will have their own time stamp(when the text is submitted). Queries is an array that holds all the queries from one submission.

When a user submits multiple queries within 15s and the total size is less than 1M, all the queries in the **queries.json** will be arranged in the following manner: A screenshot of a cell phone

Description automatically generated

In this object, queryInformation is the big array that holds submission information. Queries is the small array that holds all the queries. Each submission will have their own timestamp and can be accessed by array index of queryInformation. Now user 8 has 3 submissions and each submission has 512 queries.

All uploaded files are written to upload folder. Named with UID and timestamp(when the file is uploaded) A picture containing clock, light

Description automatically generated

Output files are generated inside output folder, names with timestamp(this timestamp only describes when the file is created, it has nothing to do with its content’s timestamps.)

A picture containing object, clock, sitting, orange

Description automatically generated

## General setup 2

When generating output file:

**Check**

1. if buffer is empty

2. if folder is empty

3. if some buffer is older than files

4. if some files are older than buffer

The code will pick one from the buffer and one from the upload folder to avoid starvation

and write them to the output file in the order of their timestamps.

**Now there should be one output file generated inside /output**

**then**

Transfer data from queries.json

Since we put the 1M limit on queries.json, we can simply move the data in queries.json to output file

Clear queries.json for next iteration

**Now there should be another output file generated inside /output**

**Repeat the process every 15s**

## General setup 3

To display results, we have processing area which shows all submitted input with timestamp. Each submission is counted as one input, regardless of its size.

Now, the code will automatically pop one submission from processing area and add it to result area. *In further implementation, this should be replaced by a function such that only when the processing item is finished then it would be moved to result area.*

Completed results will have finished time and duration. Now I use X to indicate finished time since I don’t have access to the actual result. Users can retrieve their result by clicking on the link.

A screenshot of a cell phone

Description automatically generated

# Test cases

This section lists some tests I have done.

### User system

1. Multiple users register/login
2. Receive submission with correct user id

### Receiving data

1. Multiple submission from different users
2. Text/file exceed 150KB, submission denied
3. Duplicate keyword in text submission, query discarded
4. Submission from text area and file upload from different users

### Reorganizing data and write to output file

1. upload folder, buffer.json are empty but queries.json is not
2. upload folder, queries.json are empty but buffer.json is not
3. buffer.json, queries.json are empty but upload folder is not
4. upload folder is empty, queries.json and buffer.json are not
5. queries.json is empty, upload folder and buffer.json are not
6. buffer.json is empty, queries.json and upload folder are not

# Future enhancement

**Since GitHub doesn’t store empty folder, there are two empty.txt in /upload and / output. Please delete them before use.**

Detailed comments have been written inside the code, please check.

In query.ejs, top container and left-container are not complete. However, I copied all the code from Chengcheng. So it should be an easy fix.

In script.js line 11, the function should be replaced.

In index.js, line 16 to 22 need to be changed. In passportConfig.js, line 7 to line 13 need to be changed. Those codes are used to connect to the database.

Feel free to contact me at haoxuan@usf.edu.

There are always styling improvement and code optimizations.