CS Dept. student job hunting tracking system-Final

1. Team Member

   Haorui Chen (4454226030), Zihao Zhang (4798698599), Zehao Li (3717211170)

2. Links:

   code: https://drive.google.com/drive/folders/13_iN6Yc0GCCntqfi_aAb6--JqfMm__WI?usp=sharing

   demo video:

   https://youtu.be/FXU5IREF7ZA

3. Tech selection

   Frontend: React.js; Backend: Node.js; Database: MySQL+Firebase

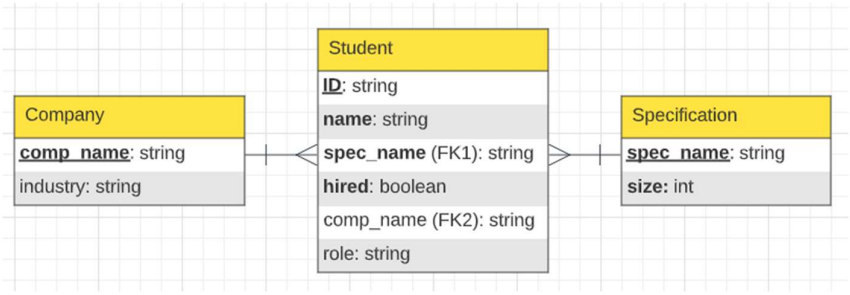
4. Design & Architecture Choices

   The frontend is designed as a single page app(SPA), i.e. all functionalities are provided in a single page.

   The backend can be basically divided into 4 parts: index.js serving as entrances for all requests from frontend; route.js route hdfs commands in task 1 to corresponding functions; cmd.js is for implementing all commands; analysis.js is for handling search & analysis functions in task 2.

   Regarding multi-database issue, our design is to CRUD data from/to each database separately for the convenience of presentation, i.e. each member can display his work without interfering with data in another database.

5. DB Conceptual design



This DB model is succinct in tracking recording student job hunting status while avoiding gathering redundant information/violating students' privacy.

6. Web App

   a. DB selection

   Simply a group of mutex buttons placed at the top of the page. The selected button indicates current DB you are reading from. As mentioned in the previous part, this is only useful with ls/cat command and seatch & analysis functionalities. Neither of the DBs will be selected upon first entering the page.

   b. file system navigation

File System Navigator:

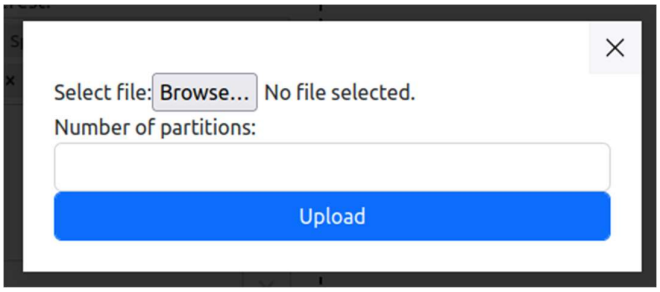
i) Go Back

The button implements 'cd' command with parameter being '..', indicating the parent directory. Upon clicking, The content of iii) & iv) will be updated to be the path of parent directory and list of folders and files in it.

Although when we do 'cd ..' in the root directory we will not go elsewhere, this may appear counter intuitive to general users (we should be able to go back further when being at root dir after all), so this button will be hidden when current directory is '/'.

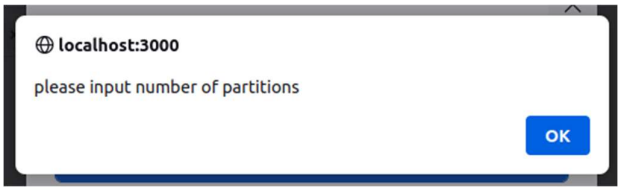
ii) Upload File

The button implements 'put' command.



Upon clicking, a popup window will appear, enabling user to select file and partitions. When user click 'Upload' with no file selected, an alert will popup:

And when no desired paartition number is specified, the user get the alert below:



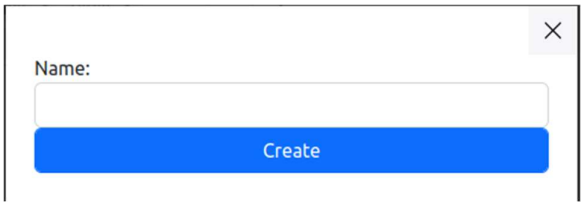
Such alert will also appear when user try to submit without anything in the input box for Create Folder, go to child directory, show file content and remove file. No screenshot will be given for brevity.

When such popup window appears, the rest of the page will get darkened to get users' focus on the popup. Click the dark area or the 'close' button will both enable us to exit the popup without submitting the form. This also applies to the pop for 'Create Folder' button.

After submitting the file and partition number, an 'ls' command will then be sent to the backend as content of current folder has just been updated.

iii) Create Folder

This button implements 'mkdir' command. The popup window looks like below:



After submitting the name of the new folder, an 'ls' command will then be sent to the backend as content of current folder has just been updated.

iv) Current dir

This shows the current folder the user is in. When first entering the system, the user will be placed at root dir '/' by default.

v) Content

This shows the result of 'ls' command. Note that 'ls' command does not have a dedicated button to trigger, it will be executed on every update of current folder indicated in iii) 'Current dir' instead.

vi) 'Which folder do you want to go?'

This implements 'cd' command with parameter being one of the child directories (if any) of current one, listed in iv). Upon submitting, The content of iii) & iv) will be updated to be the path of target directory and list of folders and files in it.

vii) 'Which file do you want to open?'

This implements 'cat' command with parameter being one of the files (if any) of current directory, listed in iv). We can see that the right half of the window shows 'File Content:', the following content will be content of the target file here.

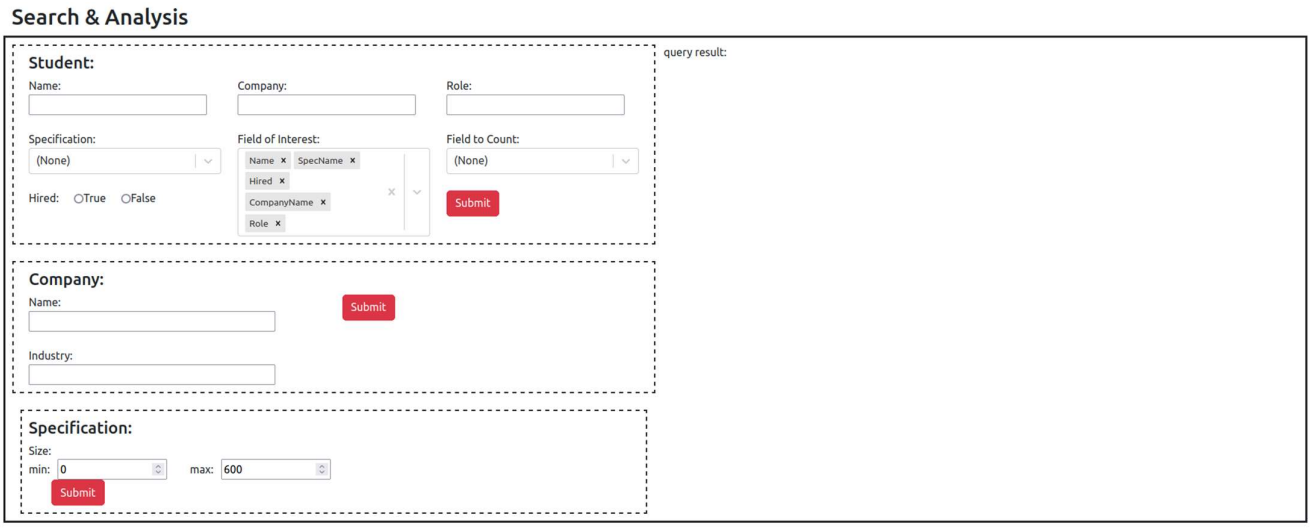
viii) 'Which file do you want to delete?'

This implements 'rm command' with parameter being one of the files (if any) of current directory, listed in iv). Upon submitting, The content of iv) will be updated to be the path of target directory and list of folders and files in it.

ix) 'Close File'

This is for wiping out content displayed at the right half of the box, it can be clicked anytime and will not interfere with ongoing traversing/querying.

c. Search & Analysis



We provide search and analysis functionalities on 2 entities: Student and Company.

For Student entity, we support search based on any of its attribute or their combinations. 'Field of INterest' indicates the field to project in final result, resembling columns to 'SELECT' in SQL; 'Field to Count' is for analysis functionality,

For Company entity, we support search based on any of its attribute or their combinations; For Specification entity, we support search based on range of size.

Query result will be shown in the 'query result:' part on the right half, indicating input and output for each step of MapReduce.

7. MySQL implementation

We used MySQL to emulate the storage process of HDFS and do Map-Reduce searching and analyzing based on Node.js. We have namenode(named "meta" in MySQL) table to record the fundamental information of both the folders and files, such as file(folder) name, inumber, creation time and file type.

```
1 •  SELECT * FROM playground.meta;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

inumber	filename	createtime	parent	typ
0	home	Sun Oct 30 2022 02:38:50 GMT-0700 (北美太...	NULL	d
1	f	Sun Oct 30 2022 03:00:50 GMT-0700 (北美太...	0	d
469	pp	Mon Nov 28 2022 22:32:13 GMT-0800 (北美太...	0	d
4314	student.json	Tue Nov 22 2022 19:46:51 GMT-0800 (北美太...	1	fs
4765	company.json	Mon Nov 28 2022 22:35:54 GMT-0800 (北美太...	1	fc
5101	teste	Sat Nov 26 2022 21:56:39 GMT-0800 (北美太...	0	d
6846	no_folder_name_1	Sun Nov 27 2022 17:34:27 GMT-0800 (北美太...	5101	fp
7168	folderf	Mon Nov 28 2022 22:01:05 GMT-0800 (北美太...	0	d

And, we have partition table(named Table 'Part' in MySQL) to trace all the partitions that are used to record the addresses where we save each partition of the real data.

```
1 •  SELECT * FROM playground.parts;
```

Result Grid | Filter Rows: | Edit: | Expo

inumber	partno	tableno
1703	7526	7526
1703	8961	8961
2996	1614	1614
2996	6063	6063
2996	6664	6664
4314	3264	3264
4314	8333	8333
4314	8929	8929
4765	2112	2112
4765	4458	4458
4765	8905	8905

Besides, there are datanodes created and stored in MySQL, and each represents a partition of corresponding file.

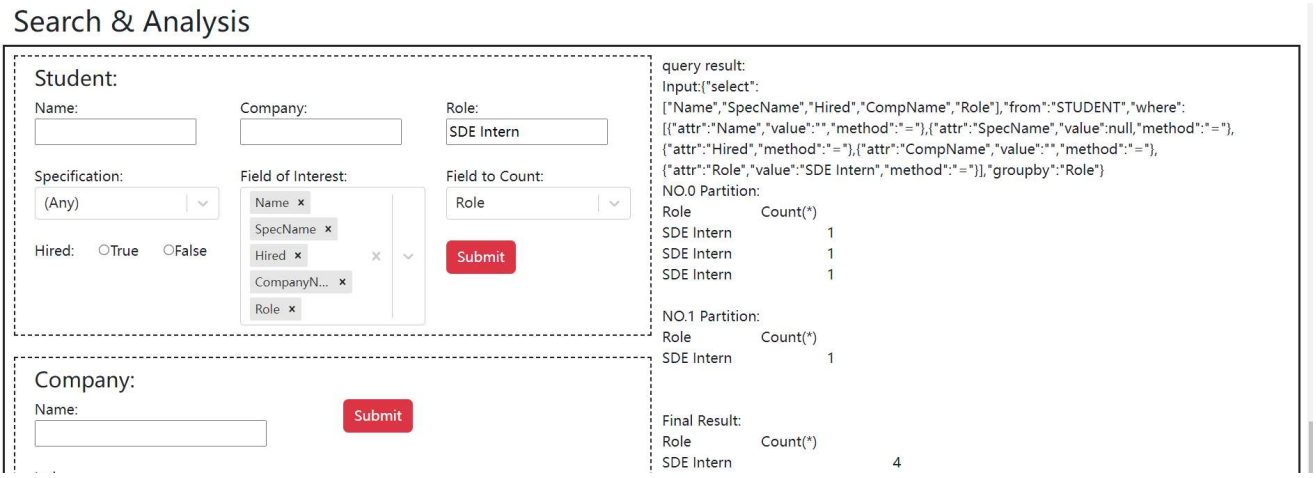
When inputting files, the meta table records the basic information of the file. Then, a certain number of partition numbers would be generated. For each partition number, we create a table for it. After that, we hash one of the attributes for each row of the data and store them separately in different tables.

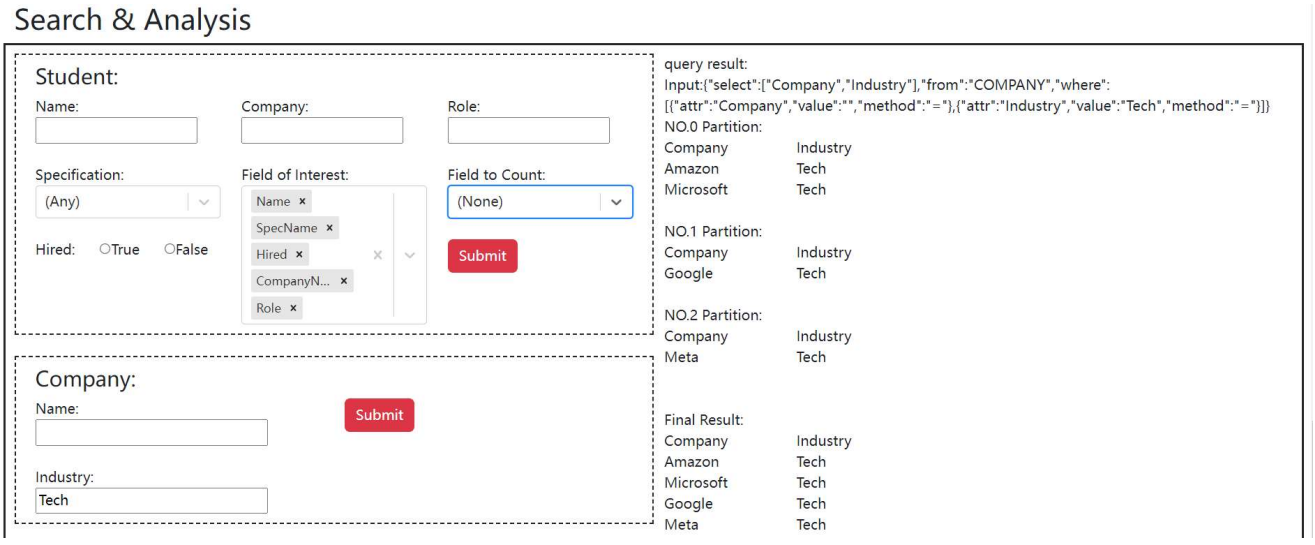
```
1 • SELECT * FROM playground.t8929;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	Name	SpecName	Hired	CompName	Role	pno
9282851045	Abigail	Data Science	1	Cisco	QA Engineer Intern	8929
1707796790	Bremen	Scientists & Engineers	1	Northrop Grumman	SDE Intern	8929
2784304057	Branny	Artificial Intelligence	1	Amazon	Project Manager Intern	8929
2789978083	Brigitte	Intelligent Robotics	1	Meta	MLE Intern	8929

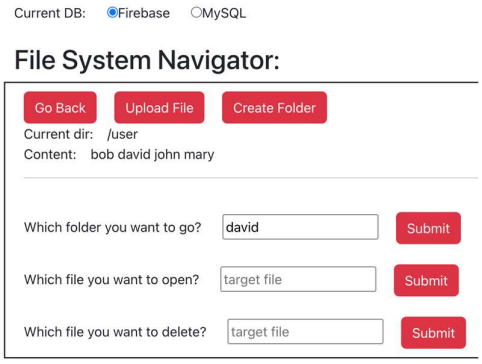
When doing searching and analysis, we firstly look into the meta table and partition table to get all the partitions(tables) numbers involved this operation. Then we go into each table, filtering the valid rows and turn them into a new map. And then, to get the final result, we keep merging the list until there's one list left.



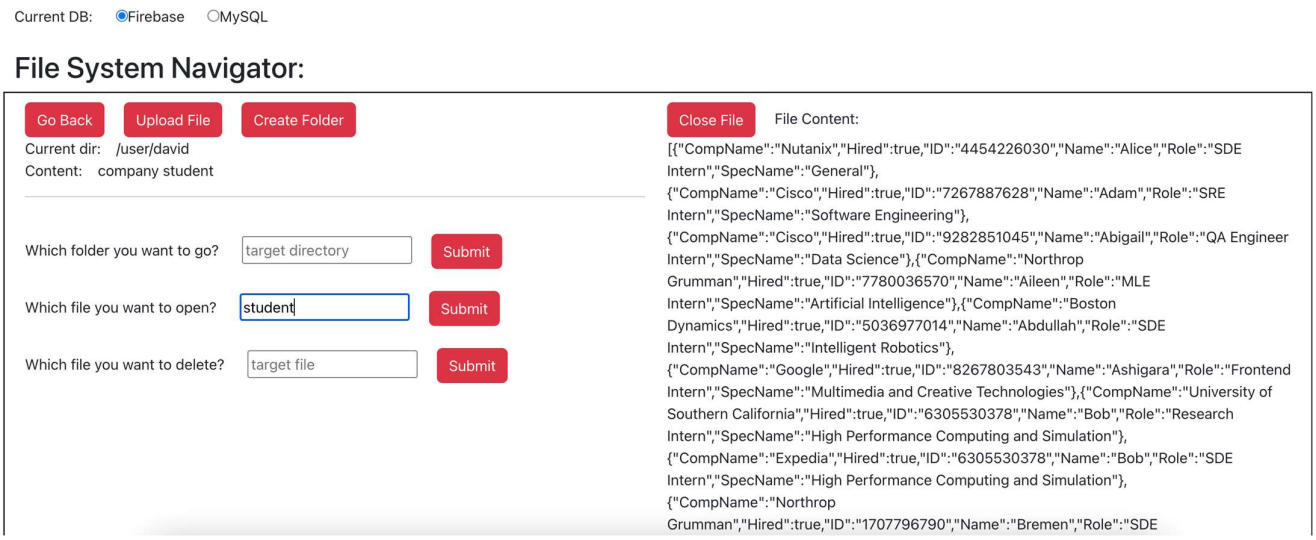


8. Firebase implementation

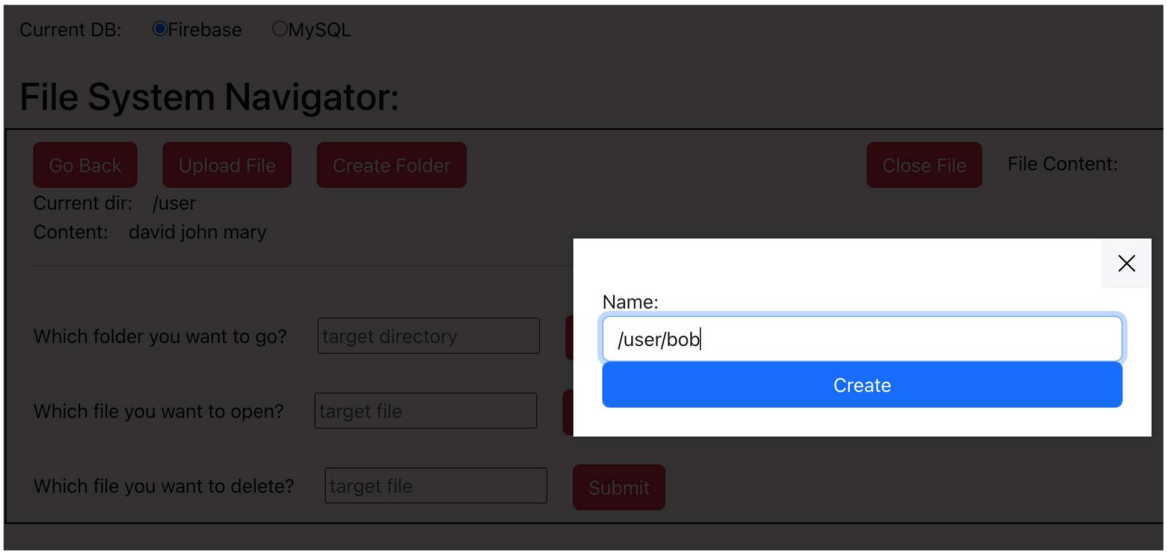
   a. **Easily enter one of the folders under the content to go into the folder**



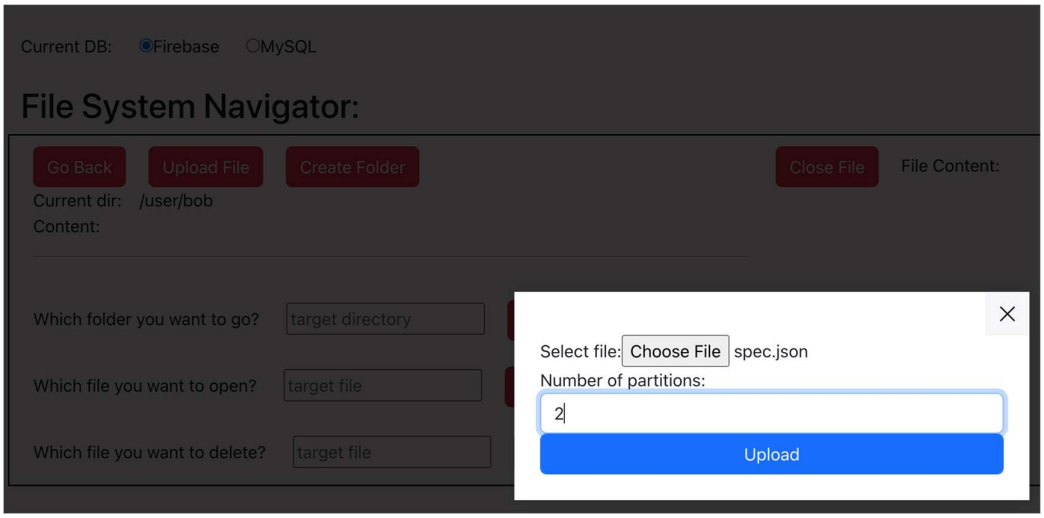
b. Enter the file name under the folder to cat



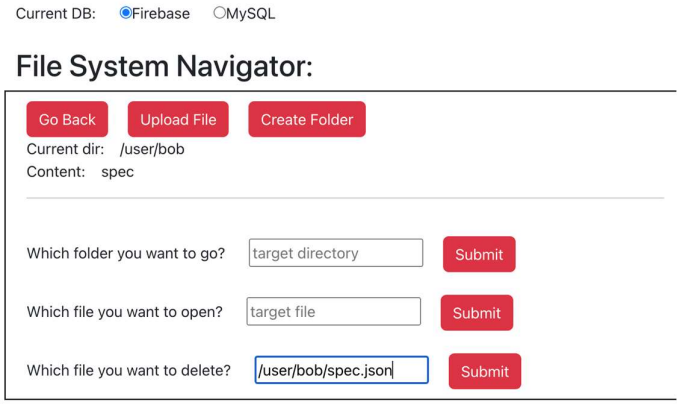
c. Need to include the parent directory when making new folder



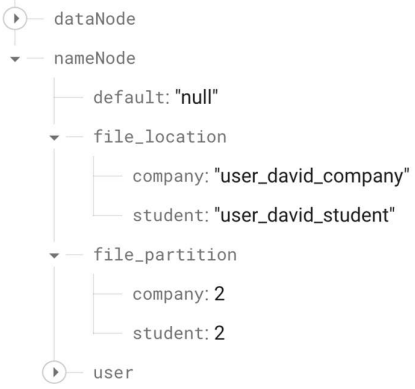
d. Upload the file under the Current directory



e. Need to include the parent directory and end with ".json" when deleting file



f. Overview of the Firebase Realtime Database



For Firebase HDFS implementation, it contains two folders which are dataNode and nameNode. In dataNode, it saves all the file data with different partitions. In nameNode, it saves all the directories as well as names of data sets. Besides that, under nameNode, we have two folders file_location and file_partition to record locations and the numbers of partitions for each file. We use file_location folder to check if the file exists when we need to read the file. It is also useful when we need to do getPartitionLocations command. When uploading file, it separates into several partitions, which depends on the number we input. And then put each partition into dataNode, record the file location in

file_location, record the number of partitions in file_partition. Therefore, when removing a file, we need to delete things from dataNode, file_location, and file_partition separately. When doing searching and analysis, we read each partition one by one to find the answer and then combine the answers from each partition to output the final answer.

9. Learning experience

Haorui Chen:

I am in charge of the whole frontend and architecting the whole project. Being one of the first 2 projects that I apply React.js since first learned it this summer, I greatly familiarized with this library through building this Web App. I got much more familiar with advanced topics like passing Components as props and useEffect and useState hook. This is also the first time I try with CSS and styling a webpage. Such frontend experience also helped me in my concurrent internship at CSSE Lab.

This is also the first time I try to architect a full-stack project from a top-down way. I paved the connection between frontend and backend endpoints and tested connection with MySQL and Firebase in JavaScript, thus enabling me to become more familiar with Node.js framework and full-stack architecture.

Zihao Zhang:

My job is to develop the MySQL emulation of HDFS and Map-Reduce using Node.js and JavaScript in this project. I think it's a precious opportunity for me to apply what we learned in class theocratically to real application and development. My teammates and I showed great passion and responsibilities in this wonderful journey. We are so glad to see finally we implemented this application from just some simple concepts. Besides, I have learned much knowledge in Node.js backend development. This was the first time I develop an app in Node.js. The main feature of it is its nonblocking synchronization, which takes me a lot of effort to get used to. As my knowledge of synchronized application development grew, I got to implement some relatively complicated algorithms with it like building hierarchical file system with parent-child relations. So proud that I finally made it. What a journey!

Zehao Li:

In this project, I aim to build a firebase emulated distributed file system and implement partition-based Map Reduce on data stored on the EDFS. It is a challenging for me because this is my first time working on a full-stack web development project and I have never used JavaScript before. To get myself familiarized with JavaScript and Web development, first of all, I leveraged available resources to learn independently. Some of the most helpful resources include tutorials videos on Youtube, as well as the official API document for JavaScript. Then the next thing is to build the application, which is where I can practice what I learned. During this time, I encountered several technical issues that were not covered in tutorials. I tried to search for these errors online and find some useful solutions. If I am still getting stuck, I just ask my teammates for help. In the end, I am able to build this application and have all functionalities work.

This was an amazing experience for me as I built my first application. It not only allows me to learn useful technical skills such as JavaScript and Firebase, but also teaches me how to learn by myself and tackle problems with research, both being crucial soft skills to be a successful engineer.