DSCI 551 Project Final Report
Team members: Zirong Huang, Zhuoqi Cai
Group 102

Meet Your Next Game

App Link: https://huan1551-edfs-dsci-551-project-edfs-llju8u.streamlit.app/
Video Link: https://www.youtube.com/watch?v=RUKcWtlEf7M
Code Link:
https://drive.google.com/drive/folders/1zSxucP8MIDLcrsVtnY5zyDIHncGsYJP2?usp=share_link

Backgrounds

EDFS Implementation Platform:
We have used firebase realtime database as our EDFS implementation platform.

Datasets:
We decided to use the steam games dataset and steam games image dataset from Kaggle.
(https://www.kaggle.com/datasets/nikdavis/steam-store-games?select=steam_description_data.csv)
Steam Games dataset contains relative information about the game, such as the game name, release date, categories, genres, and more.
Steam Games:

	A	B	C	D	E	F	G	H	I	J	K	L
1	appid	name	release_date	developer	publisher	platforms	categories	genres	positive_ratings	negative_ratings	owners	price
2	10	Counter-Strike	11/1/2000	Valve	Valve	windows;l	Multi-play	Action	124534	3339	10000000-2000	7.19
3	20	Team Fortress Classic	4/1/1999	Valve	Valve	windows;l	Multi-play	Action	3318	633	5000000-10000	3.99
4	30	Day of Defeat	5/1/2003	Valve	Valve	windows;l	Multi-play	Action	3416	398	5000000-10000	3.99
5	40	Deathmatch Classic	6/1/2001	Valve	Valve	windows;l	Multi-play	Action	1273	267	5000000-10000	3.99
6	50	Half-Life: Opposing For	11/1/1999	Gearbox S	Valve	windows;l	Single-pla	Action	5250	288	5000000-10000	3.99
7	60	Ricochet	11/1/2000	Valve	Valve	windows;l	Multi-play	Action	2758	684	5000000-10000	3.99
8	70	Half-Life	11/8/1998	Valve	Valve	windows;l	Single-pla	Action	27755	1100	5000000-10000	7.19
9	80	Counter-Strike: Conditi	3/1/2004	Valve	Valve	windows;l	Single-pla	Action	12120	1439	10000000-2000	7.19
10	130	Half-Life: Blue Shift	6/1/2001	Gearbox S	Valve	windows;l	Single-pla	Action	3822	420	5000000-10000	3.99

The steam games images dataset contains the url that direct to the header images for those games in the previous dataset
Steam Games images :

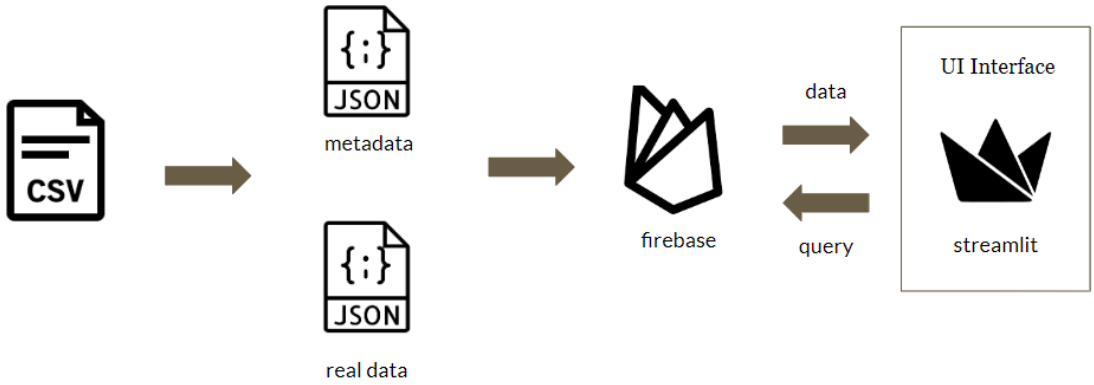
	A	B	C	D	E	F	G	H	I
1	steam_ap	header_image							
2	10	https://steamcdn-a.akamaihd.net/steam/apps/10/header.jpg?t=1528733245							
3	20	https://steamcdn-a.akamaihd.net/steam/apps/20/header.jpg?t=1528732825							
4	30	https://steamcdn-a.akamaihd.net/steam/apps/30/header.jpg?t=1512413490							
5	40	https://steamcdn-a.akamaihd.net/steam/apps/40/header.jpg?t=1528733362							
6	50	https://steamcdn-a.akamaihd.net/steam/apps/50/header.jpg?t=1542245736							
7	60	https://steamcdn-a.akamaihd.net/steam/apps/60/header.jpg?t=1528733092							
8	70	https://steamcdn-a.akamaihd.net/steam/apps/70/header.jpg?t=1530045175							
9	80	https://steamcdn-a.akamaihd.net/steam/apps/80/header.jpg?t=1512411962							
10	130	https://steamcdn-a.akamaihd.net/steam/apps/130/header.jpg?t=1542159361							
11	220	https://steamcdn-a.akamaihd.net/steam/apps/220/header.jpg?t=1541802014							

Both datasets contain 27000+ rows

UI Interface:
We used Streamlit as our UI

EDFS Implementation

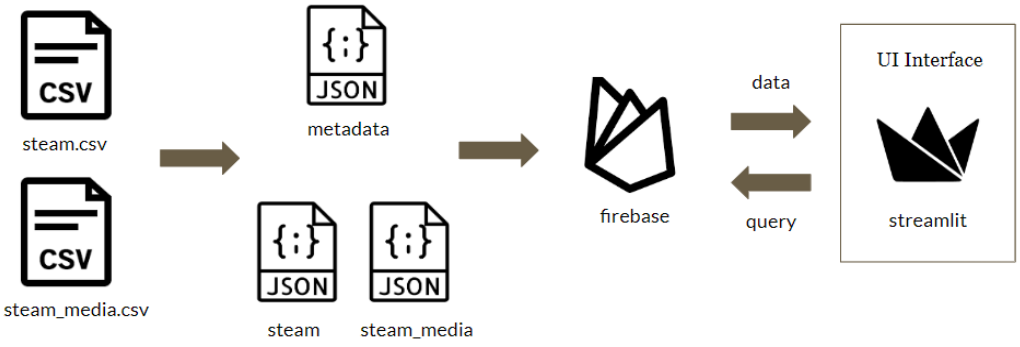
1. *mkdir /user/john*
   First use string slicing to get the directory '/user/john' since streamlit read the whole command as one string. Then set ref = '/user/john', then ref.set(''). Firebase does not allow set empty (e.g. set()), the only way to create an empty directory is set('').
2. *ls /user*
   Use string slicing to get /user directory. Then set ref = '/user' and ref.get(). Then the interface will show all content under the /user directory. To avoid showing extra content such as contents under /user/john. A try except block is implemented. If the dictionary returned from firebase has those extra content, then just show the keys of the dictionary, which are the content that we want.
3. *cat /user/john/cars.csv*
   Use string slicing to get /user/john/cars.csv directory. Then set ref = '/user' and ref.get(). This will display the partition locations of cars.csv
4. *rm /user/john*
   Use string slicing to get /user/john directory. Then set ref = '/user/john' and ref.delete()
5. *put(cars.csv, /user/john, k=5)*
   First, the user need to upload cars.csv to the interface
   Second, the interface allows user to chose the data to be sorted by ascending order or descending order respect to a numerical variable in the dataset
   Last, the user enters the command, then cars.csv will be separated into 5 partitions and stored in under the directory /cars_csv. If the data cannot be evenly divided, then the last partition will take rows that are left from previous partitions. For example, suppose there sre 251 rows, p1-p4 will have 50 rows each, and p5 will have 51 rows.
   The location of those partitions are stored under the directory /user/john/cars_csv



6. getPartitionLocations(cars.csv, /user/john)
   Use string slicing to get /user/john/cars_csv directory. Then set the reference and get the partition locations stored under the /user/john/cars_csv
7. readPartition(cars.csv, /user/john, 3)
   Use string slicing to get /user/john/cars_csv directory. Then set the reference to get the partition location for p3 stored under the /user/john/cars_csv. Then set reference to the location stored in p3. Finally use that reference to get the actual p3 cars.csv data

Game Research Aanalytics

We have two csv dataset for analytics. The steam.csv contain all of the information of the game, and the steam_media.csv contains the url link for each game in steam.csv. We provide choice of categories, platforms, and gernes to present the top 15 positive rating recommended games for the users.



STEP 1: transfer the input data to database

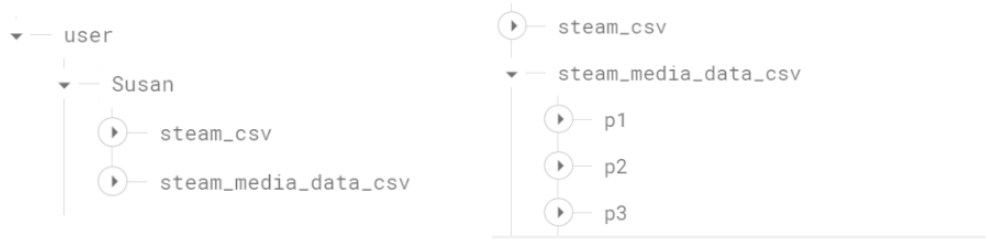
We store the two dataset to the firebase database. The dataset is sorted descending on positive_rating and partitioned every 2000 records.

Via the EDFS system we created above, we would create a name file under user once user upload file to the database, and would generate a metadata file under the user contain all of the locations of the real partition data files.

Load function:

First, we would transfer the csv to dataframe, clean the data format, and sort the dataset by “positive_rating” columns. To separate for the partition, we use defaultdictionary, with index of the row as key and the whole game information using dictionary as value. We partition every 2000 records. At the same time, we would using the firebase database function to create and refer to the targeted data file, where we store the file location of the real data partition as a metadata file. Finally, we woud have our real data file and metadata store in json format successfully on firebase database for the future analytics.

Below is the example of game project data storages.



STEP 2: Check requests in partition datasets

The platform would let user to choose their preference on genres, categories, and platform. Once the user click his choices, it would generate a request that send to our partition check function.

mapPartition function:

As the dataset is partitioned based on the descending positive rating with 2000 records per set, the mapPartition function would search from p1 to the end based on the requests. For each partition, it would store whole the steam game that satisfied at least one of the requests to a reference dictionary, and union the appid lists get from category, gerne, and platform to generate a final appid list that satisfies the requests.

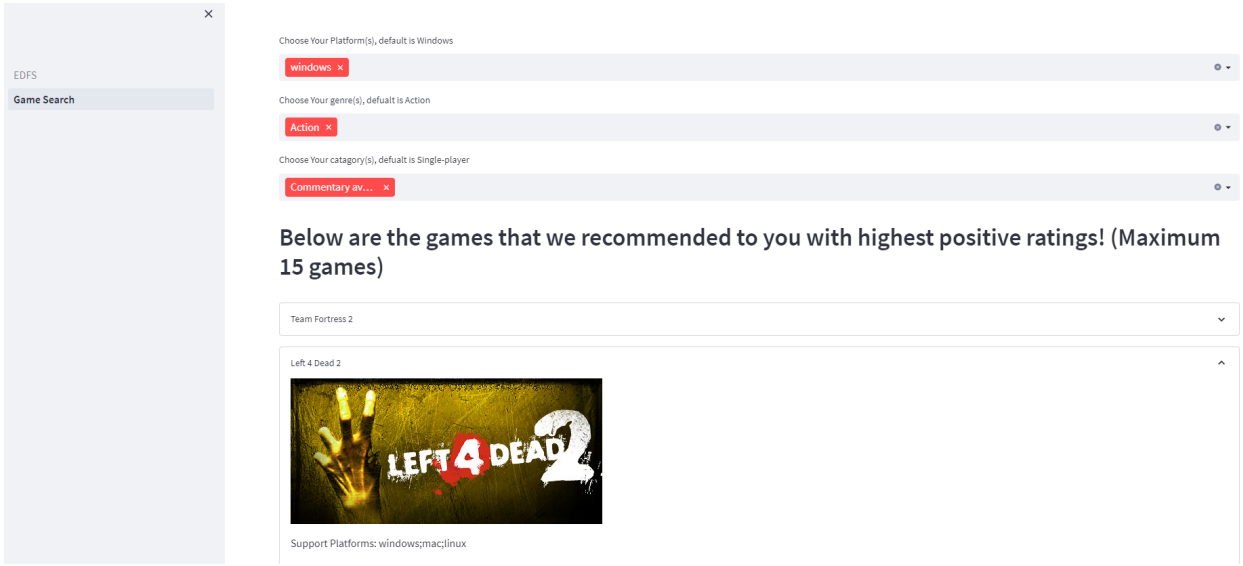
reduce function:

After searching one partition dataset, we would use the reduce function to check whether we reach to 15 records. If not, we would pass through to the next

partition and do the search again. If we have get 15 or larger records, we would generate a final result dictionary that contains only first 15th game results, with appid as the key and whole game information stored in dictionary format as value. We then call an additional function to match the media link for the satisfied game and add it into the final result.

STEP 3: send back to user

After we get the final result, we would send it back to the UI interface show as below. If there is no game satisfied, then present “No Games Found, Try other combinations”.



Learning Experience

Zirong Huang: I have learned how to use Streamlit from scratch, from creating a local app to deploy the app in streamlit cloud. I also got a deeper understanding of database management and how the system works by creating the EDFS interface.

Zhuoqi Cai: I have learned how to how to design a system and website from the very beginning. I also learned how to develop a mapreduce like program using python and firebase database, and had deeper understanding on the python data structure and the application of them.