Midterm Progress Report

Table of Contents

Team Members

Sr No.	Name
1	Kayvan Shah

Project Topic

Title	**FireMongo**
Name	**Firebase Emulator**
About	**Firebase Realtime Database RESTful API Emulation**

Given Requirements

Requirements on your prototype system (database server):

- RESTful API which supports functions in Firebase RESTful API, which include: PUT, GET, POST, PATCH, DELETE, and filtering functions: orderBy="$key"/"$value"/"name", limitToFirst/Last, equalTo, startAt/endAt.
- Store JSON data in another database
- It should have a proper index created in the database to support orderBy. For example, for orderBy="name" on users.json, it should create an index on the name.
- A command-line interface that allows users to query/update the content of the database using the curl command (similar to that in Firebase), for example:
    - curl -X GET 'http://localhost:5000/users.json?orderBy="name"&limitToFirst=5'
    - curl -X PUT 'http://localhost:5000/users/200.json' -d '{"name": "john", "age": 25}'
- **Note**: *the command should return data/response in JSON format like that in Firebase*

Implementation

Timeline

Week	Dates	Tasks
Week 1	Feb 13-Feb 19	• Finalizing the tech stack • Going through the tutorials • Design API • Creating a Git Repo & project's directory structure • Sample data
Week 2	Feb 20-Feb 26	• Data modeling • PUT request function • POST request function
Week 3	Feb 27-Mar 5	• GET request function and filters
Week 4	Mar 6-Mar 12	• PATCH request function • DELETE request function
Week 5	Mar 13-Mar 19	• Deployment on a free site hosting platform OR using Docker • Test using "curl"
Week 6	Mar 20-Mar 26	Midterm Progress Report TESTING + BUG FIXES Documentation – Docstrings, Readme & Setup
Week 7	Mar 27-Apr 2	TESTING Video Documentation
Week 8	Apr 3-Apr 9	Final Report
Week 9	Apr 10-Apr 16	BUFFER TIME
Week 10	Apr 17-Apr 23	BUFFER TIME

Milestones

NAME	STATUS
FINALIZING THE TECH STACK	COMPLETED
API DESIGN	COMPLETED
DATA MODELING	IN PROGRESS
- **V1**	COMPLETED
- **V2**	IN PROGRESS
REPOSITORY DIRECTORY STRUCTURE	COMPLETED
ENDPOINTS	IN PROGRESS
- **V1**	COMPLETED
- **V2**	IN PROGRESS
TEST CURL COMMANDS	TODO
LANDING PAGE	COMPLETED
DEPLOYMENT	TODO
DOCUMENTATIONS	TODO

Task Level Progress

Some milestones are tasks by themselves, so they are not repeated below.

NAME	STATUS
ENDPOINTS VERSION 1	DEPRECATED
1. POST	COMPLETED
2. PUT	COMPLETED
3. PATCH	COMPLETED
4. DELETE	COMPLETED
5. GET	BLOCKED
ENDPOINTS VERSION 2	IN PROGESS
1. POST	IN PROGESS
2. PUT	IN PROGESS
3. PATCH	IN PROGESS
4. DELETE	IN PROGESS
5. GET	TODO
DOCUMENTATION	IN PROGRESS
1. DOCSTRINGS	IN PROGRESS
2. API DOCS	TODO
DEPLOYMENT	TODO
1. DOCKER	
2. HOSTING	
TESTING	TODO
1. CURL	
2. DEPLOYMENT	

Challenges & Outcomes

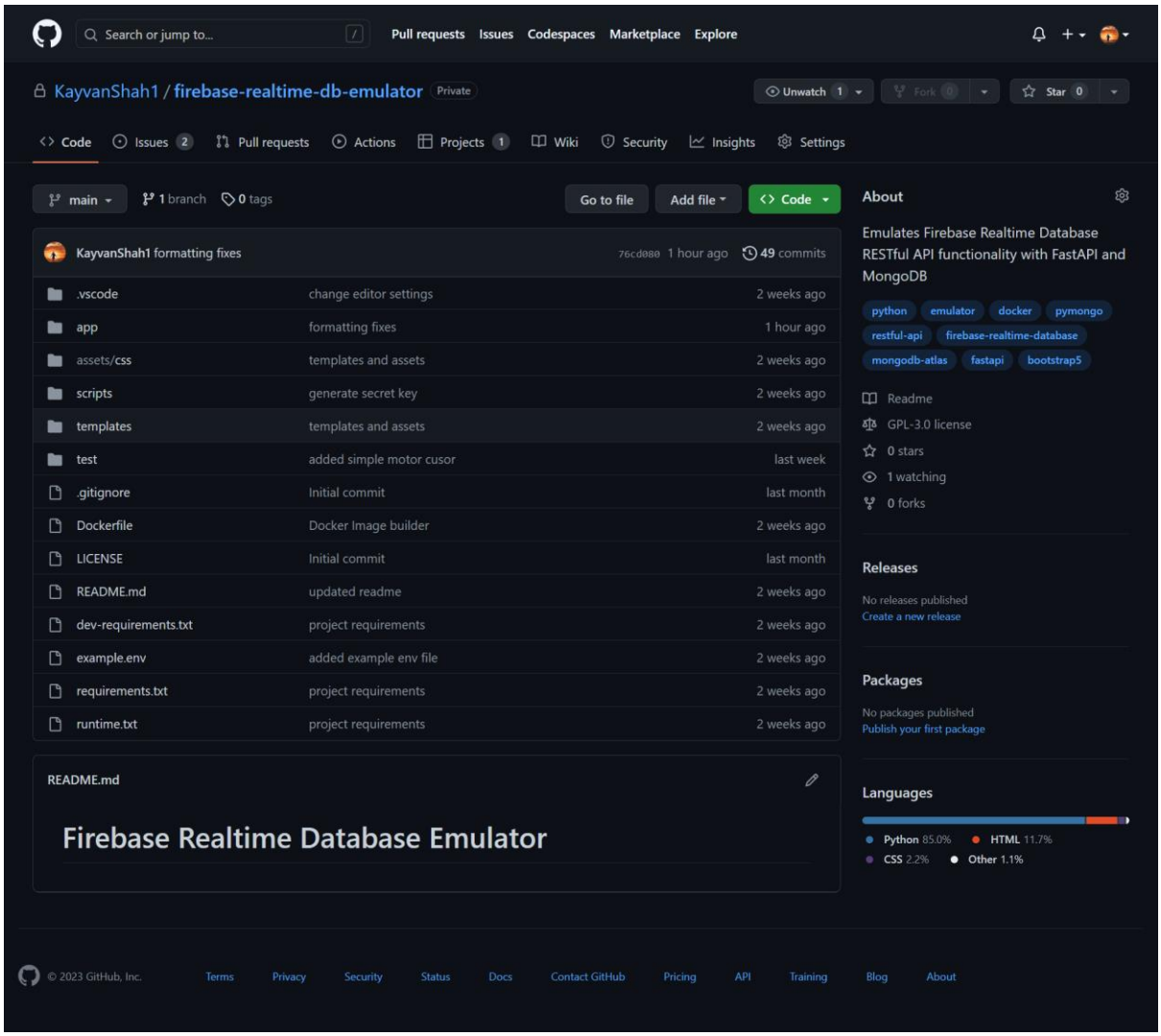
- Data model used in version 1 of endpoints didn't turn out to be feasible when retrieving data from the client end.
    - Followed a nested document structure, where every document in a collection had its schema.
    - Used a single collection for housing all the incoming data.
    - Create, Update & Delete operations were simplified using this data model.
    - Read operation turned out to be complicated, which involved writing complex queries on the database server side and writing complex filter logic to get the desired results.
    - The retrieval approach failed for basic filters and hence deprecated it.

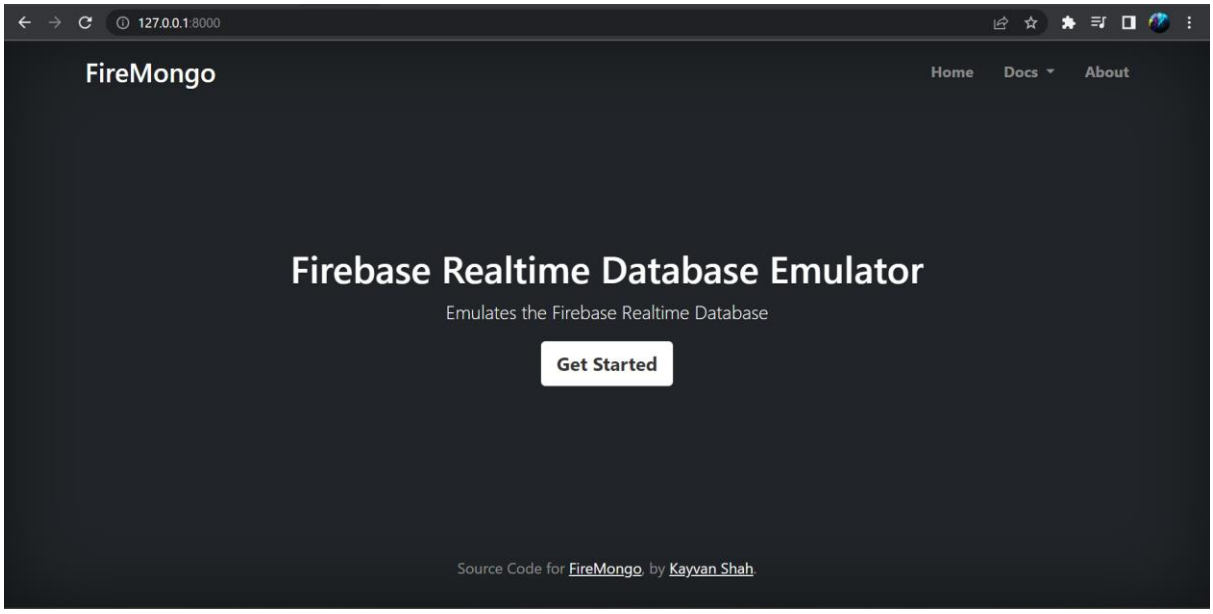
Timeline Catchup & Mitigation

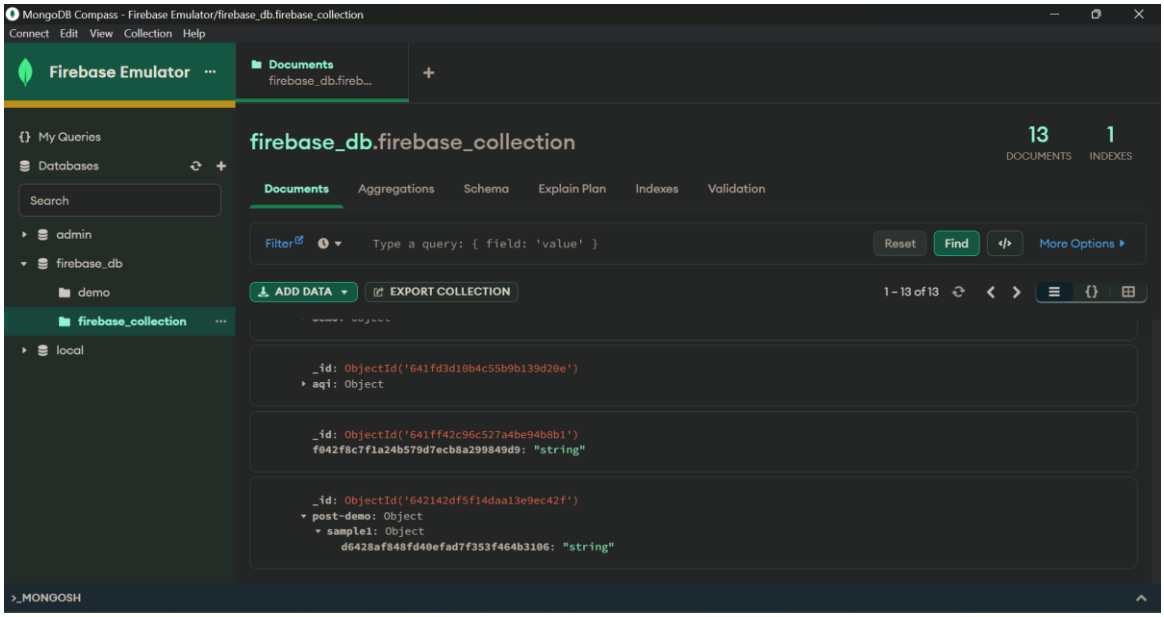
- Unexpected challenges pushed some important & secondary tasks to the upcoming week nearing the deadline and stressing the workload. Hopefully, a buffer time estimate becomes helpful here.
- Implement the ideas for a new data model such that indexing and querying data is easier by utilizing the prowess of the multiple Mongo Collections housing documents following similar JSON schema.

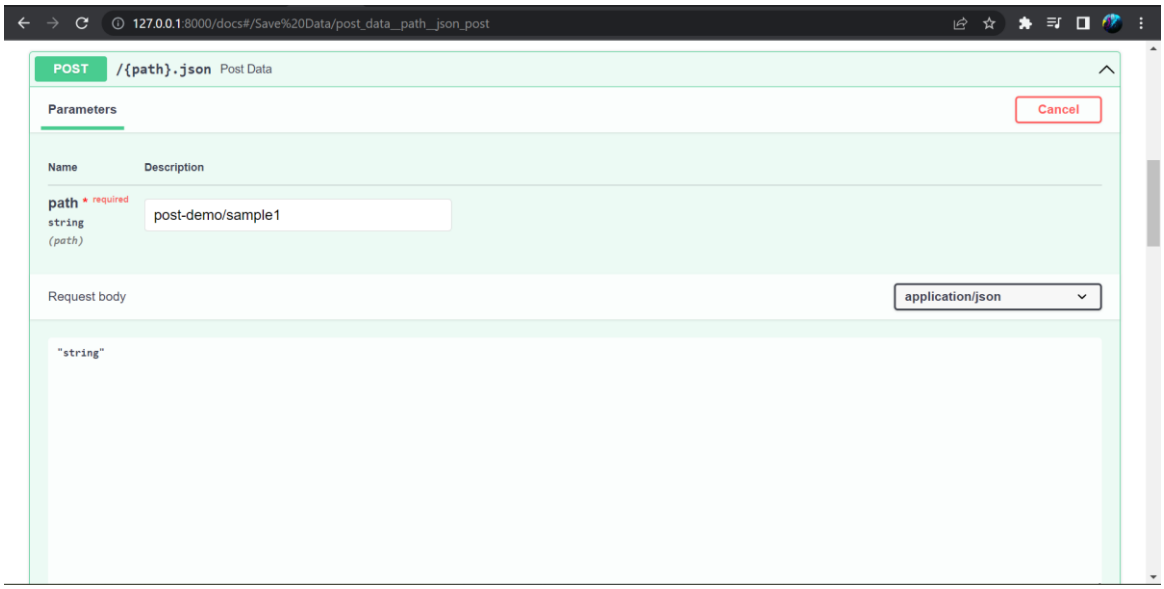
Output

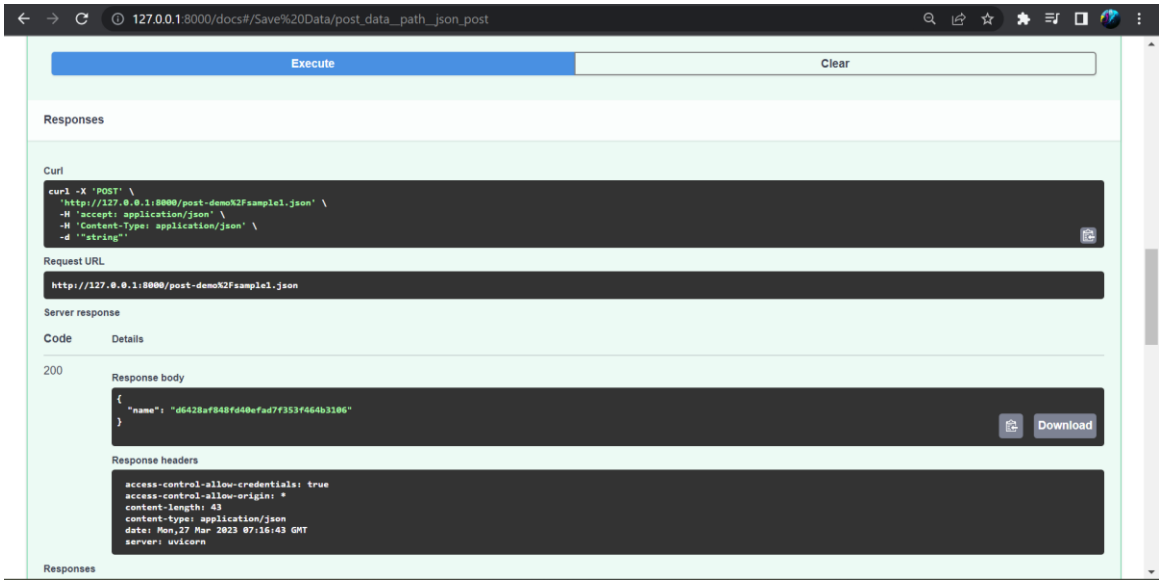
Link to Private GitHub Repository: https://github.com/KayvanShah1/firebase-realtime-db-emulator






MongoDB Compass - Firebase Emulator/firebase_db.firebase_collection
Connect Edit View Collection Help
Firebase Emulator
Documents
firebase_db.fireb...
My Queries
Databases
Search
admin
firebase_db
demo
firebase_collection
local
firebase_db.firebase_collection
13
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
Filter
Type a query: { field: 'value' }
Reset
Find
More Options
ADD DATA
EXPORT COLLECTION
1 - 13 of 13
_id: ObjectId('641fd3d10b4c55b9b139d20e')
aqi: Object
_id: ObjectId('641ff42c96c527a4be94b8b1')
f042f8c7f1a24b579d7ecb8a299849d9: "string"
_id: ObjectId('642142df5f14daa13e9ec42f')
post-demo: Object
sample1: Object
d6428af848fd40efad7f353f464b3106: "string"
>_MONGOSH


127.0.0.1:8000/docs#/Save%20Data/post_data__path__json_post
POST
/{path}.json
Post Data
Parameters
Cancel
Name
Description
path * required
string
(path)
post-demo/sample1
Request body
application/json
"string"


127.0.0.1:8000/docs#/Save%20Data/post_data__path__json_post
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://127.0.0.1:8000/post-demo%2Fsample1.json' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '"string"'
Request URL
http://127.0.0.1:8000/post-demo%2Fsample1.json
Server response
Code
Details
200
Response body
{
"name": "d6428af848fd40efad7f353f464b3106"
}
Download
Response headers
access-control-allow-credentials: true
access-control-allow-origin: *
content-length: 43
content-type: application/json
date: Mon,27 Mar 2023 07:16:43 GMT
server: uvicorn
Responses

127.0.0.1:8000/docs#/Save%20Data/post_data__path__json_post

Code	Description	Links
200	Sucessfully created data document Media type: application/json Controls Accept header. Example Value | Schema { "name": "string" }	No links
422	Validation Error Media type: application/json Example Value | Schema { "detail": [{ "loc": ["string", 0], "msg": "string", "type": "string" }] }	No links

MINGW64:/c/Users/shahk

```
shahk@K1-KINZL MINGW64 ~
$ curl -X 'PUT' \
  'http://127.0.0.1:8000/post-demo%2Fsample-6.json' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{"name": "John Doe", "age": 67, "scores": [1,2,3,4,5]}'
{"name":"John Doe","age":67,"scores":[1,2,3,4,5]}
shahk@K1-KINZL MINGW64 ~
$ curl -X 'POST' \
  'http://127.0.0.1:8000/post-demo%2Fsample-1.json' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '[1,2,3,4]'
{"name":"49b81a1ab3fa449fa69cac5f8d80ecf9"}
shahk@K1-KINZL MINGW64 ~
$ 7d82cadee19347a4bededc8cd363cda1
bash: 7d82cadee19347a4bededc8cd363cda1: command not found

shahk@K1-KINZL MINGW64 ~
$ curl -X 'DELETE' \
  'http://127.0.0.1:8000/post-demo/sample-18/7d82cadee19347a4bededc8cd363cda1.json' \
  -H 'accept: application/json'
null
```

FireMongo 0.1.0 OAS3

/api/v1/openapi.json

Firebase Realtime Database RestFul API Emulator

Retrieve Data

- GET `/.json` Query Data Root
- GET `/{path}.json` Query Data
- ~~GET `/api/v1/.json` Query Data Root~~
- ~~GET `/api/v1/{path}.json` Query Data~~

Save Data

- PUT `/.json` Put Data Root
- POST `/.json` Push Data Root
- DELETE `/.json` Delete Data Root
- PUT `/{path}.json` Put Data
- POST `/{path}.json` Post Data
- DELETE `/{path}.json` Delete Data
- PATCH `/{path}.json` Update Data
- ~~PUT `/api/v1/.json` Put Data Root~~
- ~~POST `/api/v1/.json` Push Data Root~~
- ~~DELETE `/api/v1/.json` Delete Data Root~~
- ~~PUT `/api/v1/{path}.json` Put Data~~
- ~~POST `/api/v1/{path}.json` Post Data~~
- ~~DELETE `/api/v1/{path}.json` Delete Data~~
- ~~PATCH `/api/v1/{path}.json` Update Data~~

Schemas

- HTTPValidationError
- PostDataResponse
- ValidationError