

Firemongo

Firebase Emulator: Final Project Report



April 28, 2023

University of southern california

Kayvan Shah – kpshah@usc.edu – 1106-6506-85

Firemongo

Firebase real-time database emulator

Final project report

Table of Contents

[Abstract 2](#_Toc133263684)

[Team Members 2](#_Toc133263685)

[Project Topic 2](#_Toc133263686)

[Requirements 2](#_Toc133263687)

[Project Plan 3](#_Toc133263688)

[Timeline 3](#_Toc133263689)

[Milestones 3](#_Toc133263690)

[Task Level Progress 4](#_Toc133263691)

[Version 1 4](#_Toc133263692)

[Challenges & Outcomes 4](#_Toc133263693)

[Timeline Catchup & Mitigation 4](#_Toc133263694)

[Implementation 5](#_Toc133263695)

[Tech Stack 5](#_Toc133263696)

[Design 5](#_Toc133263697)

[App Layout And File Structure 5](#_Toc133263698)

[API 5](#_Toc133263699)

[Data Model 5](#_Toc133263700)

[Results 6](#_Toc133263701)

[GitHub Repository 6](#_Toc133263702)

[Landing Page 6](#_Toc133263703)

[Version 1 7](#_Toc133263704)

[Mongo Atlas Cluster 7](#_Toc133263705)

[Command Line Interface 7](#_Toc133263706)

[Swagger Documentation 8](#_Toc133263707)

[References 10](#_Toc133263708)

# Abstract

This project is a RESTful API designed to store and retrieve data using MongoDB as the database. The API is built using the FastAPI framework and deployed using Docker. Data is stored using MongoDB Atlas, a cloud-based database service. The API supports CRUD operations and can handle concurrent requests. The project is designed to be scalable, secure, and efficient emulating the key functionalities of Firebase Real-time Database. It is deployed on Okteto Cloud, a cloud-native application development platform, which provides a development environment that is isolated and can be easily shared with a team.

# Team Members

|  |  |
| --- | --- |
| Sr No. | Name |
| 1 | Kayvan Shah |

# Project Topic

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

# 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*

# Project Plan

## 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   * V1 * V2 | COMPLETED  COMPLETED  COMPLETED |
| REPOSITORY DIRECTORY STRUCTURE | COMPLETED |
| ENDPOINTS   * V1 * V2 | COMPLETED  COMPLETED  COMPLETED |
| TEST CURL COMMANDS | COMPLETED |
| LANDING PAGE | COMPLETED |
| DEPLOYMENT | COMPLETED |
| DOCUMENTATIONs | COMPLETED |

## Task Level Progress

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

|  |  |
| --- | --- |
| NAME | STATUS |
| ENDPOINTS version 1   1. post 2. put 3. patch 4. delete 5. get | DEPRECATED  COMPLETED  COMPLETED  COMPLETED  COMPLETED  BLOCKED |
| ENDPOINTS version 2   1. post 2. put 3. patch 4. delete 5. get | COMPLETED  COMPLETED  COMPLETED  COMPLETED  COMPLETED  COMPLETED |
| documentation   1. DOCstrings 2. API DOCS | COMPLETED  COMPLETED  COMPLETED |
| deployment   1. DOCKER 2. hosting | COMPLETED  COMPLETED  COMPLETED |
| testing   1. CURL 2. deployment | COMPLETED  COMPLETED  COMPLETED |

## Version 1

### 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.

# Implementation

## Tech Stack

|  |  |
| --- | --- |
| Tech Name | Description |
| Python | * Popular programming language with a large community and extensive library support * Provides many built-in data structures and data manipulation capabilities. * Supports both object-oriented and functional programming paradigms. * Good for scripting, automation, and building web applications |
| FastAPI | * Supports asynchronous programming, which can improve performance for I/O-bound tasks. * Built on top of the Starlette framework, which provides many useful features such as automatic request validation, dependency injection, and support for web sockets. * Provides automatic API documentation using the OpenAPI specification. * Easy to use and can be deployed easily on many platforms. |
| HTML5, CSS | * Easy to build static webpages for prototype projects |
| MongoDB Atlas | * A cloud-hosted version of the popular NoSQL document-oriented database MongoDB * Provides automatic scaling, backups, and monitoring. * Has a flexible schema-less data model that can handle complex data structures. * Provides a powerful query language and indexing system for fast data retrieval. * Offers many integrations with other cloud services and platforms |
| Okteto Cloud | * A Kubernetes-based development platform for cloud-native applications * Provides a fully managed Kubernetes cluster and development environment. * Allows developers to build, test, and deploy their applications in the cloud with ease. * Provides automatic scaling, load balancing, and high availability. * Supports many popular programming languages and frameworks |
| Docker | * A containerization platform that allows developers to package their applications into lightweight, portable containers. * Provides an isolated environment for running applications, which makes it easy to deploy and manage them across different environments. * Provides many useful features such as versioning, networking, and security. * Can be used to build, ship, and run applications anywhere, from local machines to the cloud. |

## Design

### App Layout And File Structure

│ .env – Environment variables for the project

│ dev-requirements.txt – dev dependencies

│ Dockerfile – Build docker image

│ okteto.yml – Deployment configuration for Okteto Cloud

│ README.md – Setup Documentation

│ requirements.txt – dependencies

├───app – Root folder for backend application

│ │ main.py – entrypoint of the server

│ ├───api – routes definition

│ │ ├───v1

│ │ │ ├───endpoints

│ │ ├───v2

│ │ │ ├───endpoints

│ ├───core – app settings

│ ├───crud – CRUD utils

│ ├───db – database configuration

│ ├───schemas – validation schemas

├───assets – static files like CSS and images

│ └───css

├───docs – some documents

├───scripts – Shell scripts

├───templates – HTML Templates

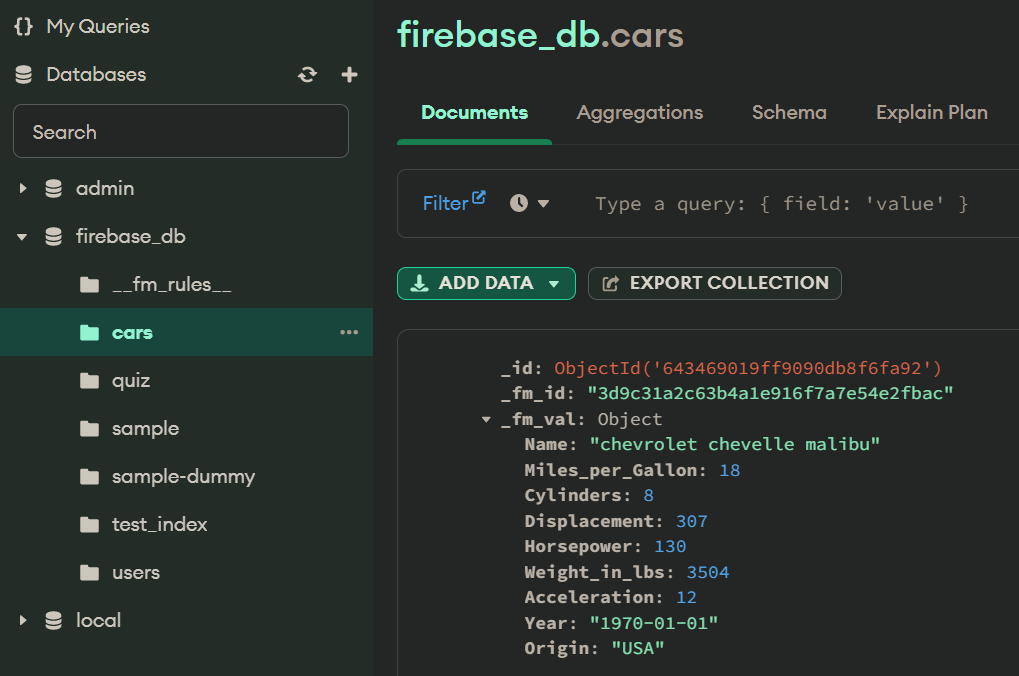
│ └───includes – generic templates

└───test – testing scripts

### Data Model

Graphical user interface, text, application

Description automatically generated



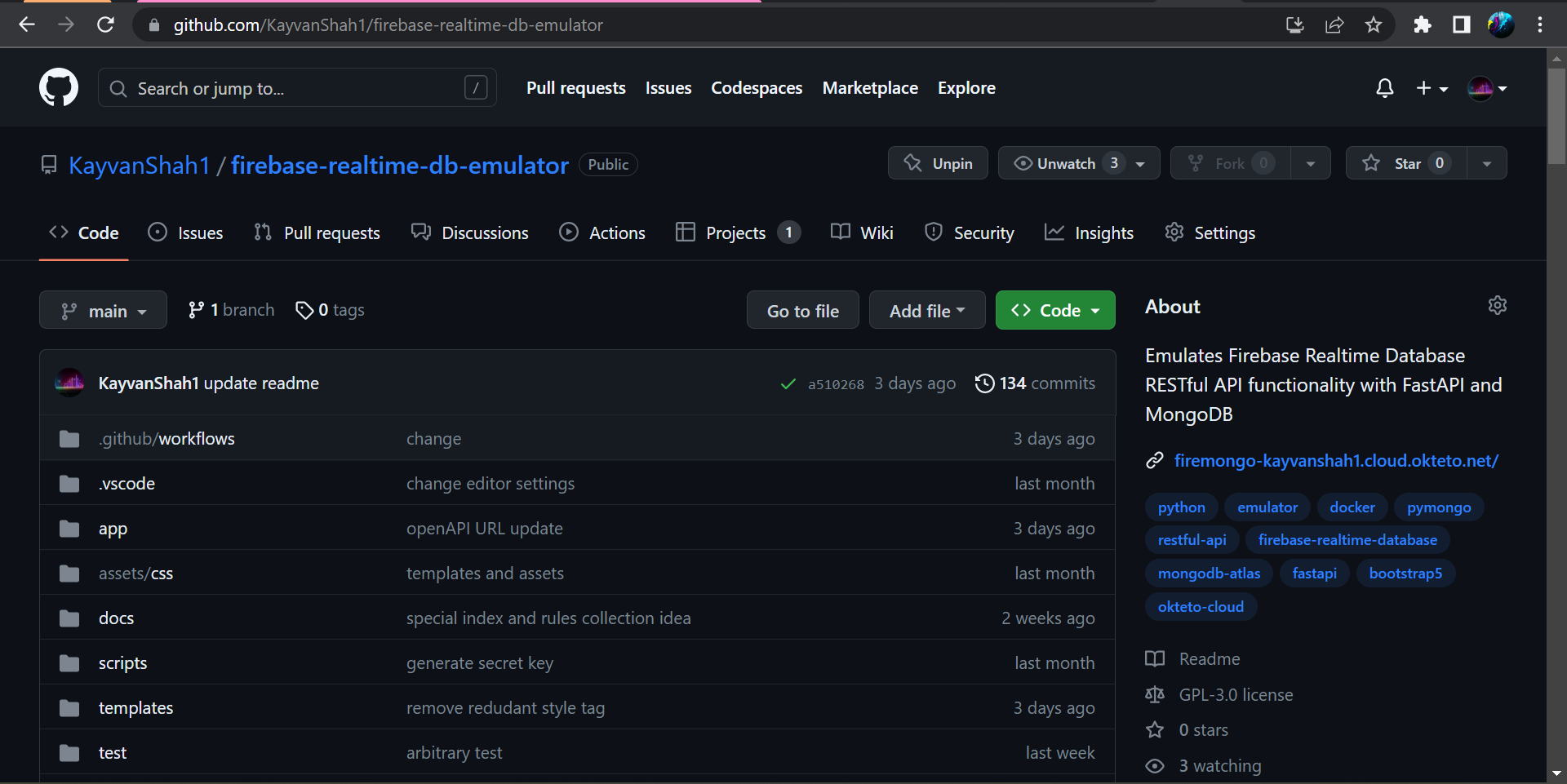
### API Design

# Results

## GitHub Repository

**Link GitHub Repository**: <https://github.com/KayvanShah1/firebase-realtime-db-emulator>

This is a public repository with documentation and details about cloning, setting up the development environment, installing dependencies and deploying it locally, on Docker or on a free hosting platform like Okteto Cloud.



## Landing Page

Graphical user interface, text, application

Description automatically generated

## Version 1

### Mongo Atlas Cluster

A screenshot of a computer

Description automatically generated with medium confidence

### Command Line Interface

Text

Description automatically generated

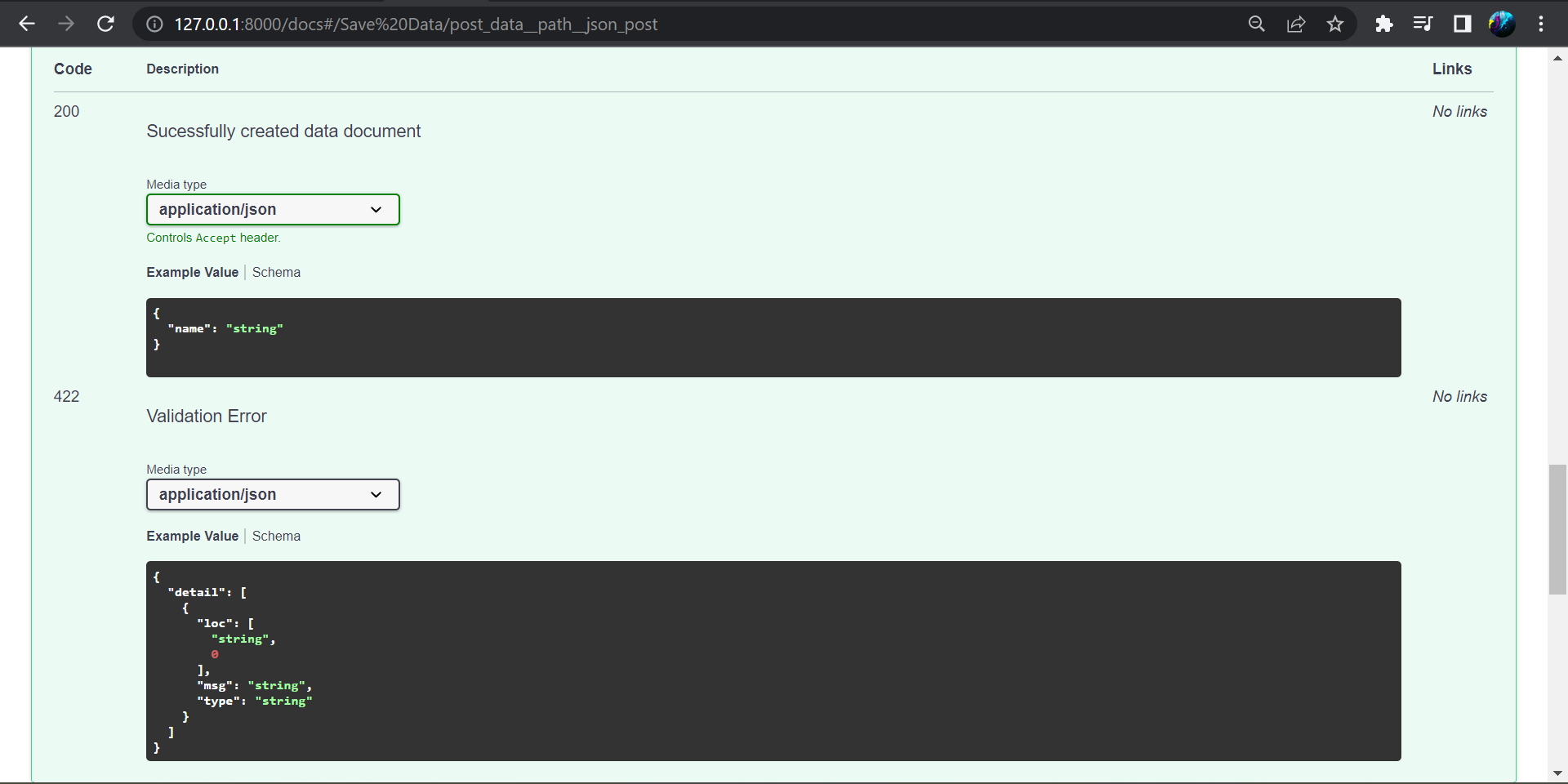
## Graphical user interface Description automatically generated with medium confidenceSwagger Documentation

Graphical user interface, application

Description automatically generated

A screenshot of a computer

Description automatically generated



# References

1. [Firebase. (n.d.). Use the Firebase Realtime Database REST API](https://firebase.google.com/docs/database/rest/start)
2. [The MongoDB documentation](https://docs.mongodb.com/)
3. [MongoDB Atlas. (2021). Cloud-hosted MongoDB](https://www.mongodb.com/cloud/atlas)
4. [Okteto. (2021). Okteto Cloud Documentation. Okteto Cloud.](https://okteto.com/docs/home)
5. [Sebastian Ramirez et al. FastAPI. 2020. [Online].](https://fastapi.tiangolo.com/)
6. [Deta. (n.d.). Deta Space Documentation](https://docs.deta.sh/docs/space/about)
7. [Docker. (2021). Docker Documentation](https://docs.docker.com/)
8. [Shah, K. (2021). KayvanShah1/blogAPI. GitHub.](https://github.com/KayvanShah1/blogAPI)