Midterm Progress Report

Table of Contents

[Team Members 1](#_Toc130770021)

[Project Topic 1](#_Toc130770022)

[Given Requirements 1](#_Toc130770023)

[Implementation 2](#_Toc130770024)

[Timeline 2](#_Toc130770025)

[Milestones 2](#_Toc130770026)

[Task Level Progress 3](#_Toc130770027)

[Challenges & Outcomes 3](#_Toc130770028)

[Timeline Catchup & Mitigation 3](#_Toc130770029)

[Output 4](#_Toc130770030)

# 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   * V1 * V2 | IN PROGRESS  COMPLETED  IN PROGRESS |
| REPOSITORY DIRECTORY STRUCTURE | COMPLETED |
| ENDPOINTS   * V1 * V2 | IN PROGRESS  COMPLETED  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   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 | IN PROGESS  IN PROGESS  IN PROGESS  IN PROGESS  IN PROGESS  TODO |
| documentation   1. DOCstrings 2. API DOCS | IN PROGRESS  IN PROGRESS  TODO |
| deployment   1. DOCKER 2. hosting | TODO |
| testing   1. CURL 2. deployment | TODO |

## 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

A screenshot of a computer screen

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

A screenshot of a computer

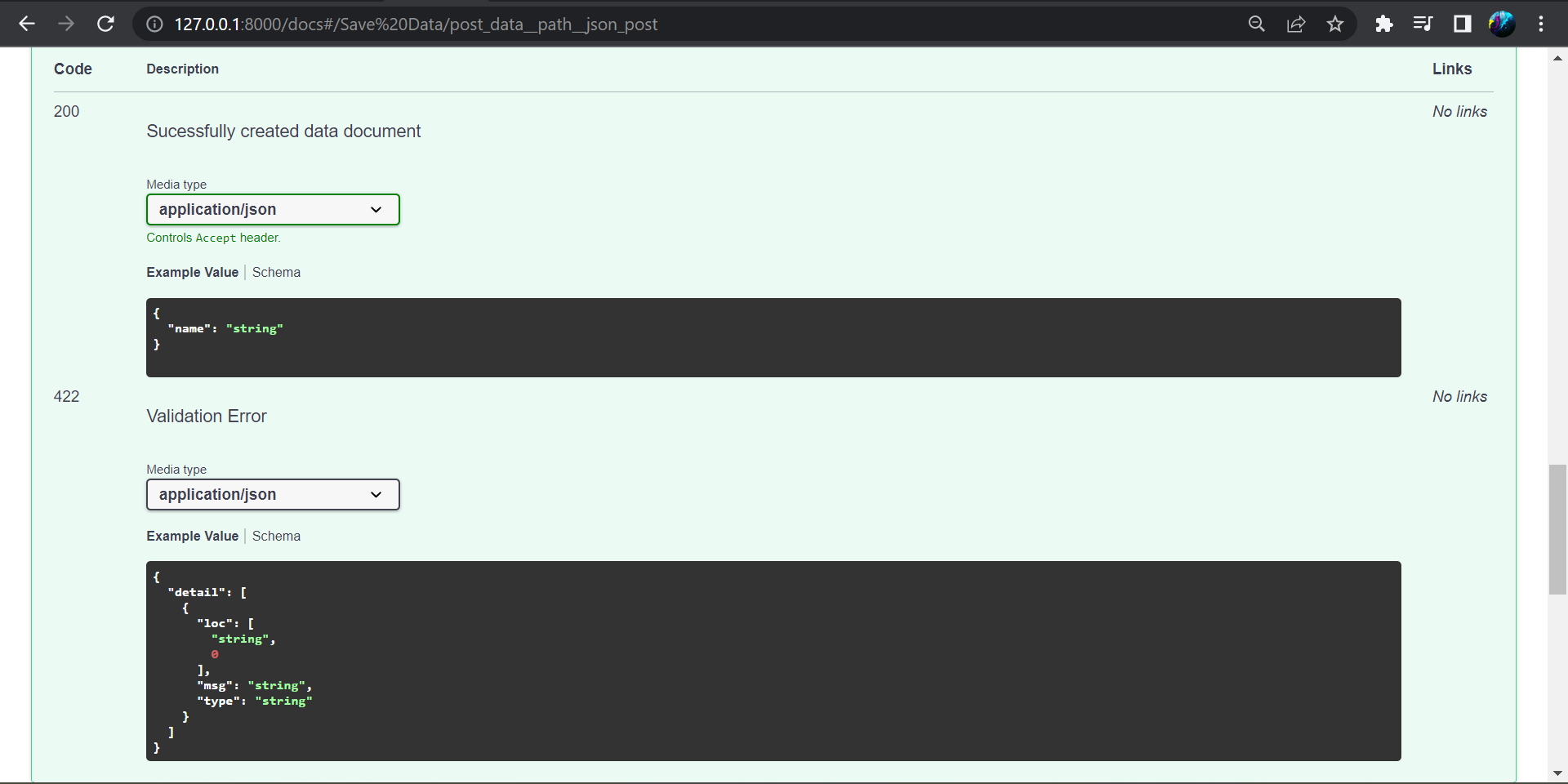
Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

A screenshot of a computer

Description automatically generated



Text

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence