Project Proposal

Table of Contents

Team Members

Sr No.	Name
1	Kayvan Shah

Project Topic

Firebase 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 similar to that in Firebase*

Planned Implementation

Milestones

- Finalizing the Tech Stack:
  - Implement a RESTFul API backend using FastAPI & Python
  - Using MongoDB Atlas as NoSQL Database for storing JSON data
  - Deploy the API/microservice using a FREE hosting platform
- Design API that fits well with the querying style of MongoDB, mimicking Firebase Realtime DB.
  - Indexing data
  - Generate unique IDs for document

-
    - Collection mapping
    - Query for nested JSON
- Data modeling
- Implement the directory structure for the project
- Implement a template endpoint to mimic Firebase Realtime DB Restful API functionality
    - GET
        - Filters:
            - orderBy
            - limitToLast
            - limitToFirst
            - equalTo
            - startAt
            - endAt
    - POST
        - Time-based unique ID generation
    - PUT
        - No unique ID generation when keys are present
    - PATCH
    - DELETE
- Test the API with curl commands on the localhost
- Deploy using a FREE hosting platform OR Docker
- Test the deployment
- Documentations

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