venn

Hood Finder Backend Engineer Task

Overview:

At Venn, we are committed to transforming neighborhoods by fostering community connections, supporting local businesses, and encouraging active resident involvement. Your task is to build a simple HTTP-based API that will enable querying for neighborhoods based on various metrics.

Task Description:

1. **API Specification**:
   - Develop an API with the endpoint `/neighborhoods` to fetch a list of neighborhoods.
   - Implement request parameters for filtering, such as `ageRange=[minAge,maxAge]`, `maxDistance=[distance]km`, and `sortBy=[field,order]`.
   - The API should return data in a structured JSON format.

2. **Data Modeling**:
   - Design a data model to represent neighborhoods and their attributes. Consider how to structure relationships between different data entities.

3. **Database Integration**:
   - Choose and justify the use of either a SQL or NoSQL database.
   - Create a database schema and populate it with the provided mock data.

4. **Query Implementation**:
   - Implement queries to filter and sort neighborhood data based on criteria like age range, distance from the city center, and average income.
   - Use aggregate functions and optimize queries with indexes if applicable.

5. **Error Handling and Validation**:
   - Implement error handling for scenarios like invalid requests or server errors.
   - Validate request parameters to ensure they are within logical ranges and formats.

6. **Testing**:
   - Write unit and integration tests for the API endpoints and database queries.
   - Provide a brief explanation of your testing strategy.

7. **API Documentation**:
   - Document all API endpoints, including details on request parameters and example responses.
   - Optionally, use Swagger or a similar tool for documentation.

8. **Bonus Task - Geolocation Feature**:
   - Integrate a public location API (e.g., Google Maps API) to find all neighborhoods within a specified distance from a provided address.
   - Detail the implementation of distance calculations and filtering based on this feature.

9. **Performance Considerations**:
   - Describe potential optimizations for improving API performance, including database query optimization and caching strategies.

10. **Scalability and Deployment**:
    - Discuss how you would scale the application for a high number of users.
    - Optionally, outline a basic deployment plan or containerization strategy.

Guidelines:

- You may use any programming language/framework of your choice, though Typescript/Javascript or Go are preferred.
- Focus on aspects like code structure, performance, and API documentation.
- Submit your solution as an attached file via email or, preferably, as a GitHub repository.
- Include a Readme.md file with instructions on setting up and running your solution.
- Please complete and submit the task within 3 days.
- Feel free to email me if you have any questions or need clarification.