Technical Assignment - Software Engineering Intern

Overview

Please complete this assignment using GoLang(https://go.dev/) as the programming language. For simplicity, use a text file as the data persistence layer to store and retrieve data. This will allow you to focus on the core requirements without worrying about setting up a complex database.

Note:

- Assume the text file is in a format that can be easily parsed and processed by your GoLang program.
- You can use a simple JSON or CSV format for the text file, or design your own format as needed.

By using a text file as the data persistence layer, you can also demonstrate your ability to work with data storage and retrieval.

Assignment - Part 1

[Evaluates your knowledge on building a CRUD API in Go]

Build a REST API to perform CRUD operations on the following Book entity. The API should accept and return data in JSON format.

JavaScript
```javascript
{
  "bookId": "bb329a31-6b1e-4daa-87ee-71631aa05866",
  "authorId": "e0d91f68-a183-477d-8aa4-1f44ccc78a70",
  "publisherId": "2f7b19e9-b268-4440-a15b-bed8177ed607",
  "title": "The Great Gatsby",
  "publicationDate": "1925-04-10",
  "isbn": "9780743273565",
  "pages": 180,
  "genre": "Novel",
  "description": "Set in the 1920s, this classic novel explores
themes of wealth, love, and the American Dream.",
  "price": 15.99,
  "quantity": 5
}
```

The following API endpoints need to be implemented.

- `GET /books`: Return a list of all books
- `POST /books`: Create a new book
- `GET /books/{id}`: Return a single book by ID
- `PUT /books/{id}`: Update a single book by ID
- `DELETE /books/{id}`: Delete a single book by ID

Assignment- Part 2(Book Search)

Implement a new endpoint to perform a keyword search on books based on the `title` and `description` fields. The endpoint should accept a query parameter `q` containing the search keyword and return a list of books matching the provided keyword.

```
GET /books/search?q=<keyword>
```

The search mechanism should be case-insensitive and match keywords in both the `title` and `description` fields.

Optimize Search Performance

Optimize the performance of the search implementation using Go's concurrency primitives, such as:

- Goroutines: to parallelize the search process
- Channels: to communicate between goroutines and improve data exchange

Example Optimization Approach

1. Split the search process into smaller tasks, each handling a subset of books.
2. Create a goroutine for each task, allowing them to run concurrently.
3. Use channels to collect search results from each goroutine and merge them into a single response.

Bonus

Complete any or all of the following tasks to earn additional points:

1. Use Docker to containerize the application.
2. Choose an endpoint and write a unit test using Go's build-in 'testing' package.
3. Implement pagination on the `GET /books` endpoint. You may have to add query parameters for pagination (Ex: limit and offset).
4. Deploy the application on a Kubernetes cluster. You can use Minikube ((https://minikube.sigs.k8s.io/docs/)) or Kind (https://kind.sigs.k8s.io/) to run Kubernetes locally. Ensure all Kubernetes manifest files are included with the source code when submitting the assignment.

Submission:

Please submit your assignment by emailing us a .zip file containing your source code. Please note that the assignment should remain confidential and not be shared publicly (e.g, on GitHub).

Also, include a README file with instructions on how to run the code and any additional information you think would be helpful.