INDIAN INSTITUTE OF TECHNOLOGY, MADRAS

# Project Name

Product Search API

# Developer Name

Atharv Kaushik

Contents

[Project Name 1](#_cljzl1i2fggw)

[1. Project Overview 2](#_ewkh4uyf6jfg)

[2. Current Status 2](#_iucz1ne0w78q)

[3. Approach & Why 3](#_hl0olggo41o7)

[4. Performance Strategy 4](#_i6hsft3k3q06)

[5. Research & Learning Resources 4](#_4vtq5mfy5lei)

[6. Ai Help 5](#_64c77doxa0ic)

[7. Sign Off 6](#_jv2eqzb3zjim)

# Project Overview

Develop a backend API in Go that can efficiently perform search operations on 1 million product records.

The primary aim of this project is to architect and implement a high-performance, full-text search API utilizing the Go programming language. This API is designed to efficiently manage 1 million in-memory product entries, facilitate full-text searches through a REST endpoint, and guarantee quick response times, smooth shutdown processes.  
In additional scalability for future needs ( via Docker). This project simulates a real-world microservice that could be integrated into a broader e-commerce or catalog management ecosystem.

| **COMPONENT** | **PURPOSE** |
| --- | --- |
| Go (Golang) | Core language for API logic and concurrency management |
| Chi Router | Lightweight, idiomatic HTTP routing framework in Go |
| Bleve | Full-text indexing and search library tailored for Go |
| Postman | Tool for API testing and validation |
| Docker | Containerization tool for consistent local development and deployment |

# Current Status

* I have successfully implemented the product search API using Go and the Bleve library for full-text indexing and querying.
* The API supports the following features:
  + Generates 1 million product records in-memory.
  + Indexes the product data using Bleve for efficient full-text search.
  + Exposes a REST API endpoint /search?q=<query> that returns up to 50 matching products.
  + Utilizes the Chi router for HTTP routing and handling.
  + Implements graceful shutdown using Go's context.
* The API is fully functional and has been uploaded to a GitHub repository with a *Dockerfile* for containerization.

# Approach & Why?

This project transcends the construction of a basic API. It reflects real-world challenges where:

* Large datasets require real-time search capabilities.
* Backend systems must maintain responsiveness, fault tolerance, and production readiness.
* I choose (such as selecting Bleve, leveraging Go's goroutines, and utilizing Docker) influence scalability and maintainability.

Completing this task within a 24-hour timeframe not only showcases technical proficiency but also underscores design thinking, research capabilities, decision-making under constraints, and the agility to learn and implement new tools swiftly—all essential traits for an Developer.

| **Technical Decision** | **Reason** | **Challenges** | **Status** |
| --- | --- | --- | --- |
| Golang implementation | Its known for its performance, well suited for developing high speed servers | Limited prior experience, reviewing Go Documentation thoroughly. | Successfully implemented |
| Bleve Library | As a native Go library, it integrates seamlessly without the need for external services like Elasticsearch. | Limited prior experience with Bleve, required review of documentation and testing. | Successfully used |
| Chi Routing | Lightweight, idiomatic, and particularly well-suited for microservices. It provides middleware support and is compatible with Go's standard net/http package. | Ensuring seamless integration with Go's standard library and custom middleware. | Successfully integrated |
| Indexing Strategy | The index is stored in a file named products. bleve on disk and reused across application restarts. This approach eliminates the need to re-index 100000 products each time the application runs, significantly reducing startup time | Implementing logic to detect changes in product data and conditionally re-index as needed. | Successfully implemented |
| Api Endpoint Design | A single endpoint, GET /search?q=<term>, was implemented.  making it easy to test | Ensuring the endpoint handles edge cases and returns appropriate responses. | Successfully implemented |
| Graceful Shutdown | This method prevents abrupt shutdowns that could corrupt index files or leave network ports open. It allows ongoing HTTP requests to complete within a specified timeout. | Thoroughly testing the graceful shutdown mechanism to ensure a smooth transition. | Successfully implemented |
| Docker Integration | The application will be containerized using a multi-stage Dockerfile. It also mitigates issues related to Go version discrepancies or dependency conflicts. making it more reliable and efficient | Configuring the multi-stage Dockerfile to optimize image size and build times | Successfully Integrated |

# Performance Strategy

* **Indexing Efficiency**:
* One-Time Indexing
  + All 1M products are indexed once at application startup (or during the first run).
  + The index is persisted to disk (products.bleve) so subsequent runs reuse theexisting index, avoiding expensive reprocessing.
* Trade-Off:
* Slight delay (~10–20 seconds) on first startup to build the index.
* Saves significant time and memory on every subsequent run.

| **Optimization** | **Benefit** |
| --- | --- |
| Index persisted | Saves indexing time and CPU cycles |
| Result limit (50) | Reduces unnecessary computation & I/O |
| Efficient data struct | Keeps memory usage in check |
| Docker compatibility | Consistent performance in any env |
| Bleve indexing | Sub-second full-text search on 1M items |

# Research & Learning Resources

I Primarily Focused on official documentation, community tutorials, and trusted developer blog.

| **Topic** | **Purpose** | **Key Resources** |
| --- | --- | --- |
| Go Language & Modules | Understand Go syntax, modules, and core libraries | * [Go Modules Guide](https://go.dev/doc/tutorial/create-module) * [Go Language Specification](https://golang.org/ref/spec) * [Go Playground](https://play.golang.org/) |
| Full-Text Search with Bleve | Learn indexing and full-text querying with native Go libraries | * [Bleve Official Docs](http://blevesearch.com/) * [Bleve GitHub](https://github.com/blevesearch/bleve) * Community tutorials & code examples |
| HTTP Routing with Chi | Explore minimalist, idiomatic routing and middleware integration in Go | * [Chi GitHub](https://github.com/go-chi/chi) * [Chi Router Docs](https://pkg.go.dev/github.com/go-chi/chi) * Medium & Dev.to REST API tutorials |
| Graceful Shutdown & Context | Handle termination signals and timed shutdown using Go’s context.Context | * [DigitalOcean Guide](https://www.digitalocean.com/community/tutorials/how-to-use-contexts-in-go) * [Dev.to Guide on Shutdown](https://dev.to/mokiat/proper-http-shutdown-in-go-3fji) |

# Ai Help

Used ChatGPT,Copilot & Monica As a Learning Tool and to get accurate way to use Go Lang and other Tools.

| **Prompt** | **Purpose / Context** | **How It Helped** |
| --- | --- | --- |
| “How to implement full-text search in Go using Bleve?” | Needed to build the core indexing and search functionality using Bleve. | - Provided a working example using bleve.NewIndex(), bleve.NewMatchQuery(), and bleve.NewSearchRequest().  - Served as the foundation for implementing the /search handler. |
| “How to handle graceful shutdown in Go server?” | Wanted the server to shut down cleanly on CTRL+C or kill signals. | - Showed how to use signal.Notify, context.WithTimeout, and srv.Shutdown().  - Enabled implementation of non-blocking, clean server exit logic. |
| “What’s the most efficient way to simulate 1 million structs in Go?” | Needed to efficiently generate 1M product entries for indexing. | - Suggested use of make([]Product, 1000000) and a for loop.  - Included modular indexing for assigning category names cleanly. |
| “How to create a Dockerfile for a Go web server?” | Wanted to containerize the app for consistent builds and future deployments. | - Provided a multi-stage Dockerfile with go build in the builder stage.  - Helped reduce final size and simplify running the container. |

***Reflection on Ai Usage***:

| **Aspect** | **Notes** |
| --- | --- |
| Approach | Used AI as a guided learning tool, not a code generator. Each response was reviewed, adapted, and tested before use. |
| Value Added | Accelerated debugging, deepened understanding of Go idioms, and reduced implementation time. |
| Transparency | Prompts and outcomes are clearly documented to demonstrate ethical and responsible AI use. |

# Sign Off

| **Item** | **Status** |
| --- | --- |
| API Functional | ✅ Complete |
| 1M Records Indexed | ✅ Yes |
| Dockerized App | ✅ Yes |
| Decision Log | ✅ Attached |
| Graceful Shutdown | ✅ Implemented |
| Performance Tested | ✅ Verified |
| GitHub Repo | [product-search](https://github.com/ak2k4/product-search) (Link) |

This project may reflect my ability to **rapidly learn new tools**, write **backend code**, and document it in a **technical and approachable way**.  
  
  
Developed and Documented by:  
  
Atharv Kaushik  
+91-9712324571

[atharv.kaushik7@gmail.com](mailto:atharv.kaushik7@gmail.com)

[Github](http://www.github.com/ak2k4) | [linkedin](http://www.linkedin.com/in/ak2k4)