# 1. Title & Objective

**Title:** Building a Simple REST API with Go (Golang) – A Beginner's Toolkit

**Chosen Technology:** Go (Golang)

Why I Chose It:
I chose Go because it is a fast, statically typed language used heavily in modern cloud infrastructure, backend development, and scalable services. Learning Go will enhance my understanding of concurrent programming and efficient application architecture, moving beyond traditional scripting languages.
End Goal:
To successfully build and run a minimalist Go application that exposes a single RESTful endpoint (/status) and returns a simple JSON object (e.g., {"status": "API is running"}).

# 2. Quick Summary of the Technology

Go, often referred to as Golang, is an open-source programming language developed by Google. It is designed for simplicity, efficiency, and excellent support for concurrency.

- **What is it?** A compiled, concurrent, and garbage-collected programming language.
- **Where is it used?** Primarily for building fast command-line tools, efficient backend web servers, microservices, and network programming (e.g., Docker and Kubernetes are written in Go).
- **One Real-World Example:** Writing a high-performance, low-latency API service that must handle thousands of simultaneous requests.

# 6. AI Prompt Journal (Initial Plan)

This journal tracks the prompts used to learn the technology and build the minimal working example.

| Prompt Used | AI's Response Summary | Evaluation/Helpfulness |
|---|---|---|
| **Prompt 1 (Setup):** "Give me a step-by-step guide to download, install, and configure the GOPATH environment variable for Go on a Windows system, assuming I am using VS Code." | The AI provided commands to download the MSI installer, set the GOROOT and GOPATH environment variables, and verify the installation via go version. | **High:** Essential for initial environment setup and avoiding common path errors. |

| | | |
|---|---|---|
| **Prompt 2 (Code Generation):** "Write a minimal Go program that creates a web server on port 8080 and defines one GET endpoint, /status, that returns the JSON response: {\"status\": \"ok\"}. Include comments explaining how to start the server." | The AI provided the full main.go file using the net/http package and a handler function that uses json.Marshal for the response. | **High:** Provided the core runnable code, illustrating Go's standard library approach to HTTP routing. |
| **Prompt 3 (Refinement/Running):** "What is the specific terminal command I use to run the single Go file named main.go, and how do I test the endpoint from the terminal using cURL?" | The AI specified the go run main.go command and provided the cURL command: curl http://localhost:8080/status . | **High:** Confirmed the execution method and provided the necessary testing command. |