Course Introduction and Overview

- Introduction to course structure and learning objectives

Understanding Backend Development

- Fundamentals of backend communications.
- Basics of communication protocols: HTTP
- Why Golang? Current trends in backend languages.

Setting Up Your Development Environment

- Git setup and introduction.
- Golang installation and terminal setup.
- Setting up GOPATH and understanding the workspace.
- Overview of Golang IDEs and their interfaces.

Go Language Basics

- Packages and code organization
- Imports & Exports in Go
- Structure of a Go application
- Variable types.
- Variables with Initializers
- Zero values and Short-hand declarations.
- Type Conversion
- Numeric Constants
- Understanding functions in Golang.
- Functions with multiple results
- Functions with named valued results

Error Handling and Best Practices

- Error handling in Go
- Panic and Recover
- Custom errors in Go
- Best Practices for error management

Methods and Interfaces

- Methods with Structs and Pointers
- Interfaces in Go: Implementation
- Type assertions and type switches

Introduction to Concurrency

- Concurrency vs Parallelism
- Golang’s approach to concurrency: Overview of Goroutines and Channels

Working with Goroutines

- Creating and managing Goroutines
- Synchronizing Goroutines using WaitGroups
- Mutexes and their use in Go

Channels in Depth

- Types of Channels: Buffered
- Channel Synchronization
- Channel Directions
- Channel Select and Non Blocking channels
- Closing Channels

Practical Concurrency

- Building a worker pool using Goroutines and Channels
- Practical examples of concurrency in backend development

Introduction to RESTful Services

- Basics of REST API design
- HTTP methods and status codes
- Go's net/http package: Building a simple REST API

Building REST APIs with Go (Without Framework)

- Project setup and standard file architecture
- Connecting to the DB - PostgreSQL setup
- CRUD operations and connecting to a database using Go's database/sql package
- Implementing middleware for logging and security

Exploring Go Web Frameworks

- Overview of popular frameworks: Echo
- Rebuilding the CRUD API using the Fiber framework
- Middleware integration using Fiber

Testing

- Writing unit tests for Go APIs
- Benchmarking API performance
- Documenting APIs with Swagger

Backend Architecture Patterns

- Monolith vs Microservices Architecture
- Popular design patterns in backend systems
- Singleton Pattern: Explanation and implementation in Go
- Factory Pattern: Explanation and implementation in Go
- Observer Pattern: Explanation and implementation in Go
- Decorator Pattern: Explanation and implementation in Go
- Best practices for designing scalable backend systems

Security in Go

- Secure coding practices in Go
- JWT Tokens: Explanation and Implementation
- OAuth 2.0 Explained!
- OAuth 2.0 Simulated Implementation in Go
- Handling sensitive data

Working with Databases

- Using SQL databases with Go: GORM
- Working with NoSQL databases: Redis
- Optimizing database queries and connections
- Using SQL databases with Go: sqlx
- Working with NoSQL databases: MongoDB

Building Microservices with Go

- Service discovery
- API Gateways
- Distributed Tracing

Deployment and DevOps

- Containerizing Go applications with Docker
- CI/CD pipelines with Jenkins and GitHub Actions
- Deploying Go applications on cloud platforms

Performance Optimization

- Profiling Go applications
- Benchmarking and optimizing code

Introduction to GraphQL

- Differences between REST and GraphQL
- GraphQL basic concepts
- Building a GraphQL API with Go - Querying data
- Building a GraphQL API with Go - Mutating data

Final Capstone Project

- Design and develop a comprehensive backend system with Go
- Incorporate API development