Developing and Testing a Go-Based Web Server

Overview:
This project involves designing and testing a Go-based web server that serves static files. The server includes comprehensive unit tests to ensure functionality and reliability.

Objectives:

- Create a functional web server in Go.
- Test various server functionalities using HTTP test.
- Provide clear documentation for developers and users.

Technologies Used:

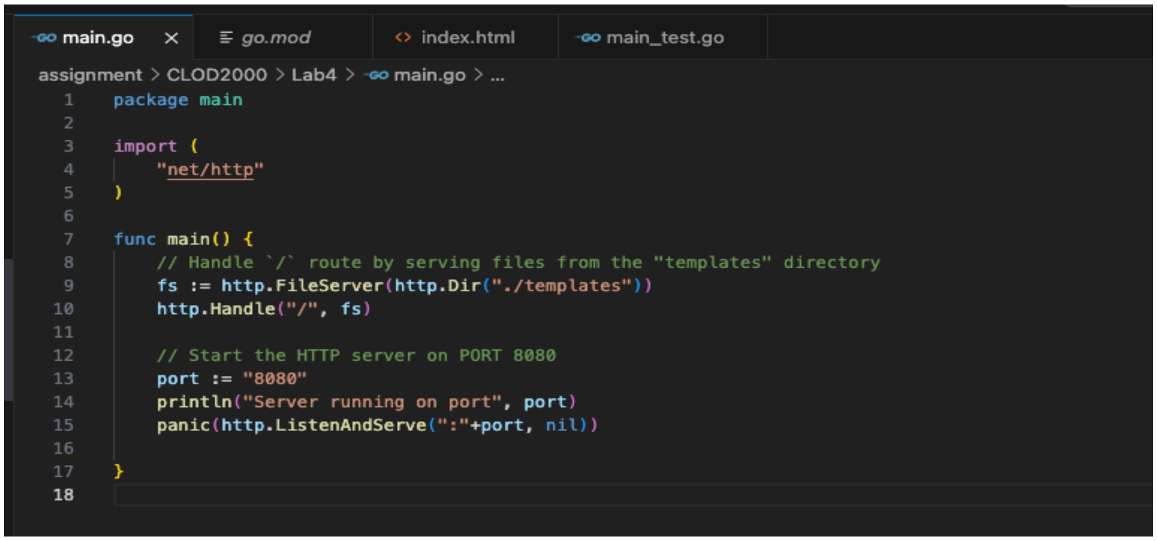
- Programming Language: Go
- Testing Framework: Go testing package
- Version Control: Git
- Editor: Visual Studio Code

Project Design and Implementation

Description:
The Go web server is designed to serve static files from the templates/ directory. It uses Go's HTTP. Fileserver to handle HTTP requests.

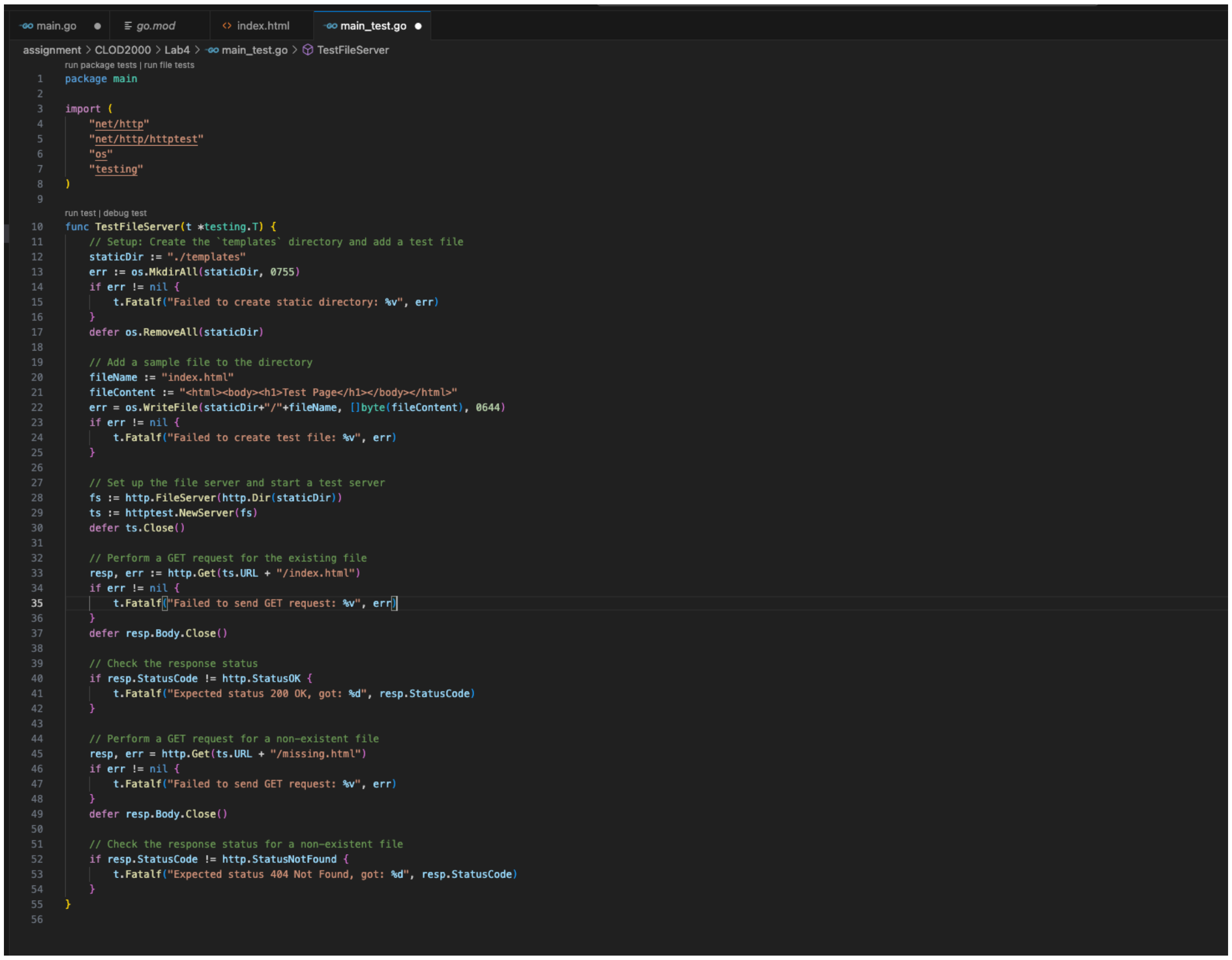
File Structure:

- main.go: Contains the server implementation.



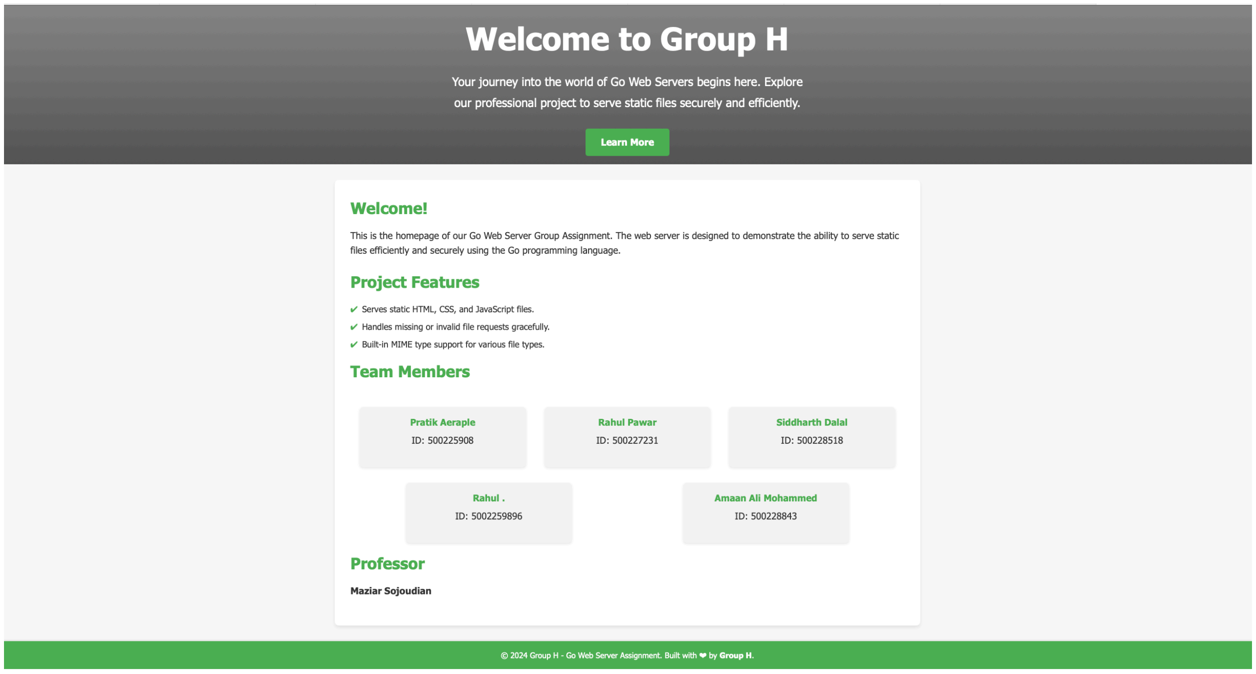
```go
package main

import (
	"net/http"
)

func main() {
	// Handle `/` route by serving files from the "templates" directory
	fs := http.FileServer(http.Dir("./templates"))
	http.Handle("/", fs)

	// Start the HTTP server on PORT 8080
	port := "8080"
	println("Server running on port", port)
	panic(http.ListenAndServe(":"+port, nil))

}
```

- main_test.go: Contains unit tests for the server.


main.go
go.mod
index.html
main_test.go
assignment > CLOD2000 > Lab4 > main_test.go > TestFileServer
run package tests | run file tests
run test | debug test

```go
package main

import (
	"net/http"
	"net/http/httptest"
	"os"
	"testing"
)

func TestFileServer(t *testing.T) {
	// Setup: Create the `templates` directory and add a test file
	staticDir := "./templates"
	err := os.MkdirAll(staticDir, 0755)
	if err != nil {
		t.Fatalf("Failed to create static directory: %v", err)
	}
	defer os.RemoveAll(staticDir)

	// Add a sample file to the directory
	fileName := "index.html"
	fileContent := "<html><body><h1>Test Page</h1></body></html>"
	err = os.WriteFile(staticDir+"/"+fileName, []byte(fileContent), 0644)
	if err != nil {
		t.Fatalf("Failed to create test file: %v", err)
	}

	// Set up the file server and start a test server
	fs := http.FileServer(http.Dir(staticDir))
	ts := httptest.NewServer(fs)
	defer ts.Close()

	// Perform a GET request for the existing file
	resp, err := http.Get(ts.URL + "/index.html")
	if err != nil {
		t.Fatalf("Failed to send GET request: %v", err)
	}
	defer resp.Body.Close()

	// Check the response status
	if resp.StatusCode != http.StatusOK {
		t.Fatalf("Expected status 200 OK, got: %d", resp.StatusCode)
	}

	// Perform a GET request for a non-existent file
	resp, err = http.Get(ts.URL + "/missing.html")
	if err != nil {
		t.Fatalf("Failed to send GET request: %v", err)
	}
	defer resp.Body.Close()

	// Check the response status for a non-existent file
	if resp.StatusCode != http.StatusNotFound {
		t.Fatalf("Expected status 404 Not Found, got: %d", resp.StatusCode)
	}
}

```

- templates/: Directory containing static HTML files like index.html.



Welcome to Group H

Your journey into the world of Go Web Servers begins here. Explore our professional project to serve static files securely and efficiently.

Welcome!

This is the homepage of our Go Web Server Group Assignment. The web server is designed to demonstrate the ability to serve static files efficiently and securely using the Go programming language.

Project Features

✔ Serves static HTML, CSS, and JavaScript files.
✔ Handles missing or invalid file requests gracefully.
✔ Built-in MIME type support for various file types.

Team Members

Pratik Aeraple
ID: 500225908

Rahul Pawar
ID: 500227231

Siddharth Dalal
ID: 500228518

Rahul .
ID: 5002259896

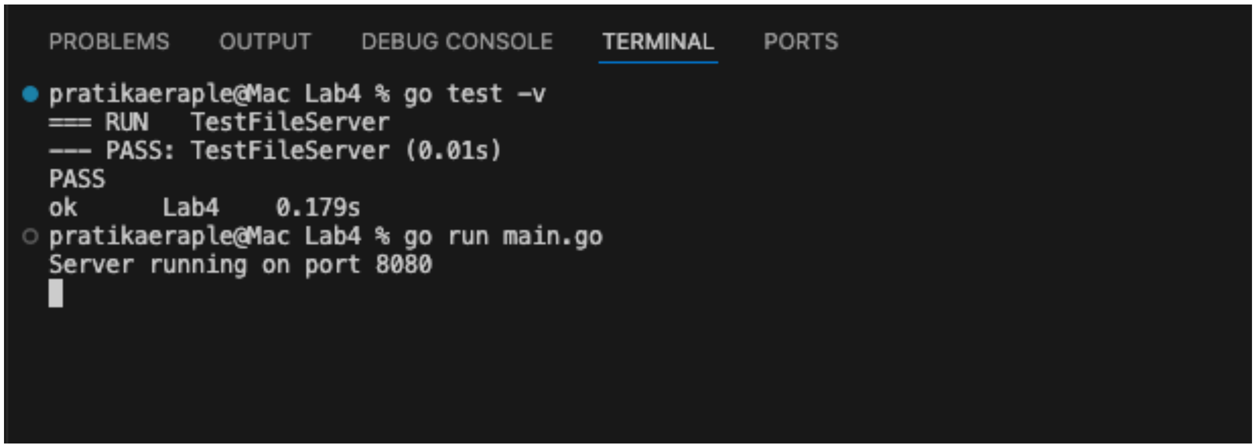
Amaan Ali Mohammed
ID: 500228843

Professor

Maziar Sojoudian

© 2024 Group H - Go Web Server Assignment. Built with ❤ by Group H.

- Testing: Contains the test output.



```
pratikaeraple@Mac Lab4 % go test -v
=== RUN   TestFileServer
--- PASS: TestFileServer (0.01s)
PASS
ok      Lab4    0.179s
pratikaeraple@Mac Lab4 % go run main.go
Server running on port 8080
```

- Github: https://github.com/PratikAeraple/WINP2000-Lab4