Department of Electrical and Computer Engineering

Modern Distributed Systems

Assignment#1

Dr. Abdalkarim Awad

Student Name : Ali Hammoudeh.

Student Number : 1215263.

Socket programming is a mechanism for communication between computers using the Internet Protocol (IP). Sockets are endpoints of a bidirectional communication channel that allow sending and receiving data between applications running on different machines. It enables applications to communicate with each other, regardless of the underlying operating systems, network protocols, and programming languages used.

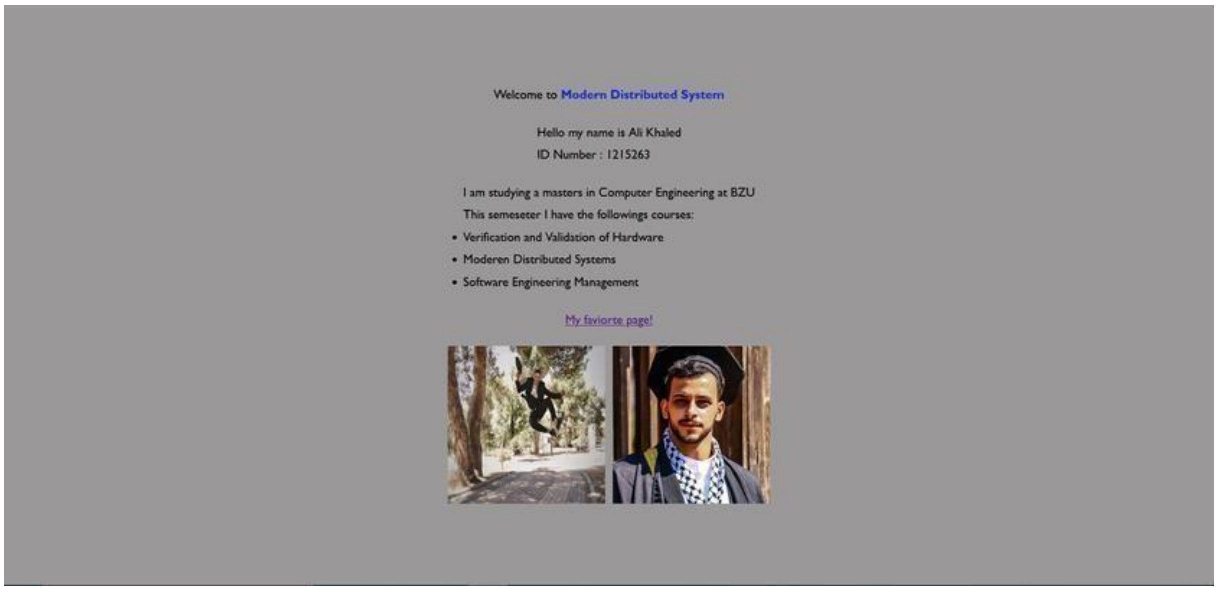
In this project (Assignment) I implements a simple HTTP web server that listens on port 6677 using golang and this document discuss the assignment requirement with the existing code.

```
import (
"fmt"
"net/http"
"os"
)
```

The code imports the "fmt", "net/http" and "os" packages from the Go Standard Library.

"fmt" is used for printing messages and formatting strings, "net/http" is a package used for building HTTP servers and clients, and "os" provides a platform-independent interface to operating system functionality, such as reading and writing files.

If the request is for the root path during code execution, serve the index.html file, which appears as shown in Image 1.



Welcome to Modern Distributed System

Hello my name is Ali Khaled

ID Number : 1215263

I am studying a masters in Computer Engineering at BZU

This semeseter I have the followings courses:

- Verification and Validation of Hardware
- Moderen Distributed Systems
- Software Engineering Management

My faviorte page!

Image 1 : Index.html

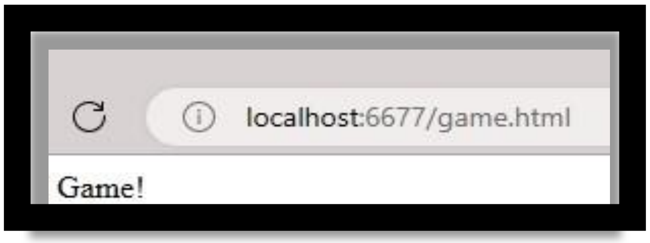
Image 1 displays the index.html file, a HTML webpage that encompasses the necessary page for the assignment.

```
// If the request is for a path ending with .html, serve the corresponding file
if r.URL.Path[len(r.URL.Path)-5:] == ".html" {
        http.ServeFile(w, r, r.URL.Path[1:])
        return
}

// If the request is for a path ending with .css, serve the corresponding file
if r.URL.Path[len(r.URL.Path)-4:] == ".css" {
        http.ServeFile(w, r, r.URL.Path[1:])
        return
}

// If the request is for a path ending with .js, serve the corresponding file
if r.URL.Path[len(r.URL.Path)-3:] == ".js" {
        http.ServeFile(w, r, r.URL.Path[1:])
        return
}

// If the request is for a path ending with .png, serve the corresponding file
if r.URL.Path[len(r.URL.Path)-4:] == ".png" {
        http.ServeFile(w, r, r.URL.Path[1:])
        return
}
// If the request is for a path ending with .jpg, serve the corresponding file
if r.URL.Path[len(r.URL.Path)-4:] == ".jpg" {
        http.ServeFile(w, r, r.URL.Path[1:])
        return
}
```

The code checks the extension of the file requested by the client in a HTTP request and serves the corresponding file if it has a recognized extension (".html", ".css", ".js", ".png", or

".jpg"). If the requested file has an unrecognized extension, the code will not serve any file. The serving of the file is done by the http.ServeFile function, which takes the HTTP response writer, the HTTP request and the file path as its arguments. The file path is constructed by taking the request's URL path and removing the first character (the '/' character) from it.

Image 2 depicts the control of my favorite page, which is simply an empty file with the text 'GAME!' included.

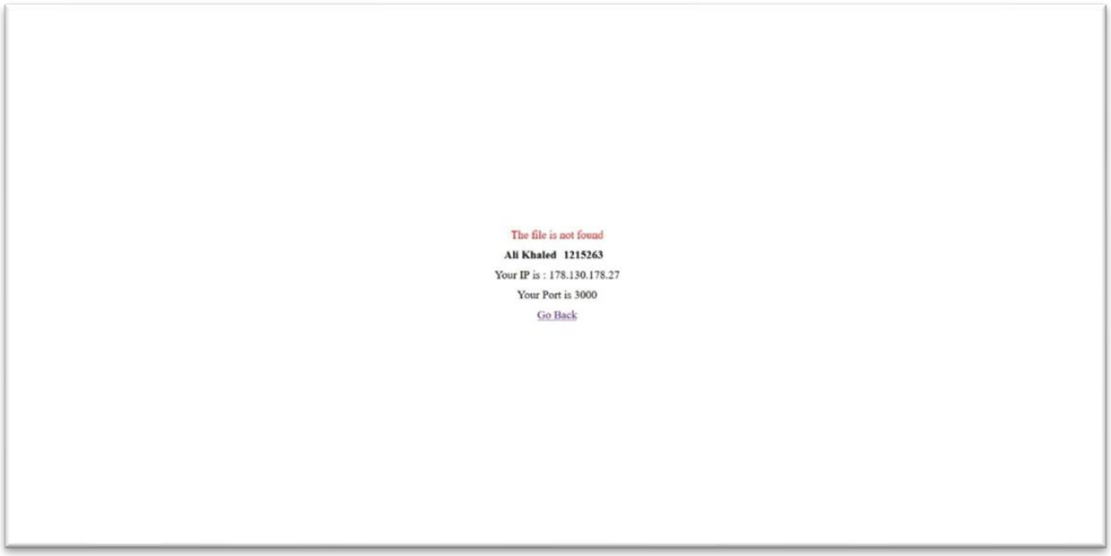


localhost:6677/game.html

Game!

Image 2:my favorit pag.

Image 3 shows a simple folder related to the 404 error, displaying an error message, my name, registration number, and the IP and PORT of the device accessing the page.



The file is not found

Ali Khaled 1215263

Your IP is : 178.130.178.27

Your Port is 3000

Go Back

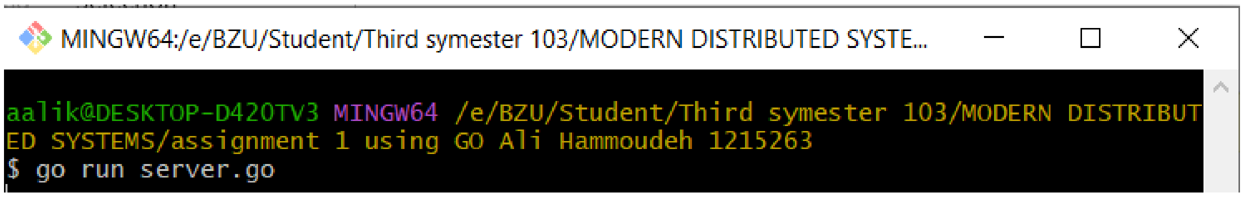
Image 3: 404 page.

The socket server is represented by the 'server.go' file, with the implementation detailed as follows:

```
package main

import (

"fmt"

"net/http"

"os"

)


func main() {

http.HandleFunc("/", func(w
http.ResponseWriter, r *http.Request) {

        // If the request is for the root path,
serve the index.html file

        if r.URL.Path == "/" {

                http.ServeFile(w, r,
"index.html")

                return

        }


        // If the request is for a path ending
with .html, serve the corresponding file

        if r.URL.Path[len(r.URL.Path)-5:] ==
".html" {

                http.ServeFile(w, r,
r.URL.Path[1:])

                return

        }


        // If the request is for a path ending
with .css, serve the corresponding file

        if r.URL.Path[len(r.URL.Path)-4:] ==
".css" {

                http.ServeFile(w, r,
r.URL.Path[1:])

                return

        }

        // If the request is for a path ending
with .js, serve the corresponding file

        if r.URL.Path[len(r.URL.Path)-3:] == ".js"
{

                http.ServeFile(w, r,
r.URL.Path[1:])

                return

        }


        // If the request is for a path ending
with .png, serve the corresponding file

        if r.URL.Path[len(r.URL.Path)-4:] ==
".png" {

                http.ServeFile(w, r,
r.URL.Path[1:])

                return

        }

        // If the request is for a path ending
with .jpg, serve the corresponding file

        if r.URL.Path[len(r.URL.Path)-4:] ==
".jpg" {

                http.ServeFile(w, r,
r.URL.Path[1:])

                return

        }


        // Check the requested path and
redirect to the appropriate website

        switch r.URL.Path {

        case "/gl":

                http.Redirect(w, r,
"https://www.google.com/",
http.StatusTemporaryRedirect)

                return
```

```
		case "/ghb":
			http.Redirect(w, r, "https://www.github.com/", http.StatusTemporaryRedirect)
			return
		case "/bzu":
			http.Redirect(w, r, "https://www.birzeit.edu/", http.StatusTemporaryRedirect)
			return
		}

		// Check if the file exists and serve it, or return an error page
		if _, err := os.Stat(r.URL.Path[1:]); err == nil {
			http.ServeFile(w, r, r.URL.Path[1:])
			return
		}

		// Set the response status to 404 Not Found
		w.WriteHeader(http.StatusNotFound)

		// Write the error page
		fmt.Fprintf(w, "<!DOCTYPE html>\n")
		fmt.Fprintf(w, "<html>\n")
		fmt.Fprintf(w, "<head>\n")
		fmt.Fprintf(w, "<title>Error</title>\n")
		fmt.Fprintf(w, "</head>\n")
		fmt.Fprintf(w, "<body>\n")
		fmt.Fprintf(w, "<p style='color: red'>The file is not found</p>\n")
		fmt.Fprintf(w, "<p><strong>Your names and IDs</strong></p>\n")
		fmt.Fprintf(w, "<p>The IP and port number of the client: %s</p>\n", r.RemoteAddr)
		fmt.Fprintf(w, "</body>\n")
		fmt.Fprintf(w, "</html>\n")

	// Otherwise, handle the request as before
	fmt.Fprintf(w, "Hello, you've requested: %s\n", r.URL.Path)
	})

	// Listen on port 6677
	http.ListenAndServe(":6677", nil)
	}
```

To run the server it is simple as showed in image 4.



```
MINGW64:/e/BZU/Student/Third symester 103/MODERN DISTRIBUTED SYSTE...
aalik@DESKTOP-D420TV3 MINGW64 /e/BZU/Student/Third symester 103/MODERN DISTRIBUT
ED SYSTEMS/assignment 1 using GO Ali Hammoudeh 1215263
$ go run server.go
```

Image 4: Go server running command.