Net-Centric Lab 3

Student: Nguyen Duc Toan

ID: ITCSIU21112

Client.go

```go
package main

import (
    "fmt"
    "io"
    "net"
    "os"
    "strings"
)

const (
    HOST = "localhost"
    PORT = "8080"
    TYPE = "tcp"
)

func main() {
    tcpServer, err := net.ResolveTCPAddr(TYPE, HOST+":"+PORT)

    if err != nil {
        println("ResolveTCPAddr failed:", err.Error())
        os.Exit(1)
    }

    conn, err := net.DialTCP(TYPE, nil, tcpServer)
    if err != nil {
        println("Dial failed:", err.Error())
        os.Exit(1)
    }

    defer conn.Close() // Ensure connection is closed when the main function returns

    // Login or register
    fmt.Println("1. Login")
    fmt.Println("2. Register")
    fmt.Println("3. Exit")
    fmt.Print("Enter your choice: ")
    var choice int
    fmt.Scanln(&choice)

    switch choice {
    case 1:
        username, password := login()

        // Send the login data to the server
        _, err = conn.Write([]byte("Login" + "|" + username + "|" + password))
        if err != nil {
            println("Write data failed:", err.Error())
            os.Exit(1)
        }
```

```
52          received := make([]byte, 1024)
53          _, err = conn.Read(received)
54          if err != nil {
55              println("Read data failed:", err.Error())
56              os.Exit(1)
57          }
58          fmt.Println(string(received))
59
60          if strings.Contains(string(received), "successful") {
61              fmt.Println("Hello client from server")
62              fmt.Println("1. Play Game")
63              fmt.Println("2. Download File")
64              fmt.Println("3. Exit")
65
66              fmt.Print("Enter your choice: ")
67              var choice int
68              fmt.Scanln(&choice)
69
70              switch choice {
71              case 1:
72                  _, err = conn.Write([]byte("Game"))
73                  fmt.Println("Welcome to Guessing Game!")
74                  playGame(err, conn)
75
76              case 2:
77                  _, err = conn.Write([]byte("Download"))
78                  fmt.Print("Enter the name of the file to download: ")
79                  var fileName string
80                  fmt.Scanln(&fileName)
81
82                  // Send download request
83                  _, err = conn.Write([]byte(fileName))
84                  if err != nil {
85                      fmt.Println("Error sending file request:", err)
86                      return
87                  }
88
89                  // Receive the file data
90                  requestFile(conn, fileName)
91              case 3:
92                  _, err = conn.Write([]byte("Exit"))
93                  exit()
94              default:
95                  println("Invalid choice")
96                  os.Exit(0)
97              }
98          }
```

```
    case 2:
        username, password, fullname, email, address := register()

        // Send the register data to the server
        _, err = conn.Write([]byte("Register" + "|" + username + "|" + password + "|" + fullname + "|" + email + "|" + address))
        if err != nil {
            println("Write data failed:", err.Error())
            os.Exit(1)
        }
    case 3:
        exit()
    default:
        println("Invalid choice")
        return
    }

    conn.Close()

}

func login() (username, password string) {
    // Login to server
    fmt.Print("Enter username: ")
    // var username string
    fmt.Scanln(&username)

    fmt.Print("Enter password: ")
    // var password string
    fmt.Scanln(&password)

    return username, password
}

func register() (username, password, fullname, email, address string) {
    // Register to server
    fmt.Print("Enter username: ")
    fmt.Scanln(&username)

    fmt.Print("Enter password: ")
    fmt.Scanln(&password)

    fmt.Print("Enter fullname: ")
    fmt.Scanln(&fullname)

    fmt.Print("Enter email: ")
    fmt.Scanln(&email)

    fmt.Print("Enter address: ")
    fmt.Scanln(&address)
```

```
149
150         return username, password, fullname, email, address
151     }
152
153     func exit() {
154         return
155     }
156
157     func playGame(err error, conn *net.TCPConn) {
158         received := make([]byte, 1024)
159         _, err = conn.Read(received)
160         for {
161             fmt.Printf("Guess a number between 1 and 100 (press 0 to exit): ")
162             var guessNumber int
163             fmt.Scanln(&guessNumber)
164
165             if guessNumber == 0 {
166                 _, err = conn.Write([]byte("Exit"))
167                 return
168             }
169
170             _, err = conn.Write([]byte(fmt.Sprintf("%d", guessNumber)))
171             if err != nil {
172                 println("Write data failed:", err.Error())
173                 os.Exit(1)
174             }
175
176             // Read the response from the server
177             received := make([]byte, 1024)
178             _, err = conn.Read(received)
179             if err != nil {
180                 println("Read data failed:", err.Error())
181                 os.Exit(1)
182             }
183
184             response := strings.TrimSpace(string(received[:]))
185             fmt.Printf("Received response: '%s'\n", response)
```

```
186
187         if strings.Contains(response, "_Congratulations!") {
188
189             fmt.Println("Do you want to play again? (y/n)")
190             var playAgain string
191             fmt.Scanln(&playAgain)
192
193             if strings.ToLower(playAgain) == "n" {
194                 _, err = conn.Write([]byte("Exit")) // Send "Exit" to the server
195                 break
196             } else if strings.ToLower(playAgain) == "y" {
197                 // If the user says yes, send "Again" to the server
198                 _, err = conn.Write([]byte("Again"))
199                 _, err = conn.Read(received) // Wait for server's response to start the game again
200                 fmt.Println("Starting a new game...")
201                 conn.Read(received)
202             }
203         }
204     }
205 }
206
```

```go
func requestFile(conn net.Conn, fileName string) {
    // Create or open the file for writing (only if the file exists)
    outFile, err := os.Create("downloaded_" + fileName)
    if err != nil {
        fmt.Println("Error creating file:", err)
        return
    }
    defer outFile.Close()

    // Buffer to receive the file data
    buffer := make([]byte, 1024)

    // Read the response from the server
    for {
        n, err := conn.Read(buffer)
        // Handle server responses
        serverResponse := string(buffer[:n])
        // fmt.Print("Server response: ", serverResponse)
        if err != nil {
            if err == io.EOF {
                fmt.Println("File download completed.")
                break
            }
            fmt.Println("Error reading from server:", err)
            return
        }

        // Ignore notification messages from the server
        if serverResponse == "File download starting...\n" || serverResponse == "\nFile download complete\n" {
            continue
        }
        if serverResponse == "Error: File not found\n" {
            fmt.Println("Server response: ", serverResponse)
            return // Exit if the file does not exist
        }

        // Write file data to the local file if it's not an error message
        outFile.Write(buffer[:n])
    }
}

```

Server.go

```go
package main

import (
    "encoding/base64"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "math/rand"
    "net"
    "os"
    "strconv"
    "strings"
)

const (
    HOST = "localhost"
    PORT = "8080"
    TYPE = "tcp"
)

type User struct {
    Username string   `json:"username"`
    Password string   `json:"password"`
    Fullname string   `json:"fullname"`
    Email    []string `json:"email"`
    Address  []string `json:"address"`
}

var randomKey = rand.Intn(1000) + 1
var key = strconv.Itoa(randomKey)

func main() {
    listen, err := net.Listen(TYPE, HOST+":"+PORT)
    if err != nil {
        log.Fatal(err)
        os.Exit(1)
    }
    // close listener
    defer listen.Close()
    for {
        conn, err := listen.Accept()
        if err != nil {
            log.Fatal(err)
            os.Exit(1)
        }

        go handleRequest(conn)
    }
}

```

```go
func handleRequest(conn net.Conn) {
    defer conn.Close()

    // incoming request
    buffer := make([]byte, 1024)
    _, err := conn.Read(buffer)
    if err != nil {
        log.Fatal(err)
    }

    clientMsg := string(buffer)

    if clientMsg[:5] == "Login" {
        username, password := handleLogin(clientMsg)

        if authenticateUser(username, password) {
            fmt.Println(key + "_Hello Server from client")
            _, err = conn.Write([]byte(key + "_Authentication successful\n"))
            conn.Read(buffer) // Read the next message from the client
            clientMsg := strings.TrimSpace(string(buffer))
            if strings.HasPrefix(clientMsg, "Game") {
                startGuessingGame(err, conn)
            } else if strings.HasPrefix(clientMsg, "Download") {
                startFileTransfer(conn)
            } else if strings.HasPrefix(clientMsg, "Exit") {
                return
            }
        } else {
            fmt.Println(key + "_Login failed")
            _, err = conn.Write([]byte(key + "_Login failed"))
        }

    } else if strings.HasPrefix(clientMsg, "Register") {
        handleRegister(clientMsg)
        conn.Write([]byte("_Register successful"))
    }
}

func encryptPassword(password string) string {
    return base64.StdEncoding.EncodeToString([]byte(password))
}

func decryptPassword(encryptedPassword string) string {
    decoded, _ := base64.StdEncoding.DecodeString(encryptedPassword)
    return string(decoded)
}

```

```go
func loadUsers(filename string) []User {
    jsonFile, err := os.Open(filename)

    // Check if the file exists
    if err != nil {
        log.Fatal(err)
    }

    data, _ := io.ReadAll(jsonFile)
    var users []User
    json.Unmarshal(data, &users)
    return users
}

func authenticateUser(username, password string) bool {
    users := loadUsers("User.json")
    encryptedPassword := encryptPassword(password)
    for _, user := range users {
        if user.Username == username && user.Password == encryptedPassword {
            return true
        }
    }
    return false
}

func handleLogin(clientMsg string) (username, password string) {
    data := strings.Split(clientMsg, "|")
    username = strings.TrimSpace(data[1])
    password = strings.Trim(data[2], "\x00")

    return username, password
}

func saveUser(user User, filename string) {
    users := loadUsers(filename)
    users = append(users, user)
    data, _ := json.Marshal(users)
    _ = os.WriteFile(filename, data, 0644)
}
```

```go
func handleRegister(clientMsg string) {
    data := strings.Split(clientMsg, "|")
    username := strings.TrimSpace(data[1])
    password := strings.Trim(data[2], "\x00")
    fullname := strings.TrimSpace(data[3])
    email := strings.Split(strings.TrimSpace(data[4]), ",")
    address := strings.Split(strings.Trim(data[5], "\x00"), ",")

    user := User{
        Username: username,
        Password: encryptPassword(password),
        Fullname: fullname,
        Email:    email,
        Address:  address,
    }

    saveUser(user, "User.json")
}
```

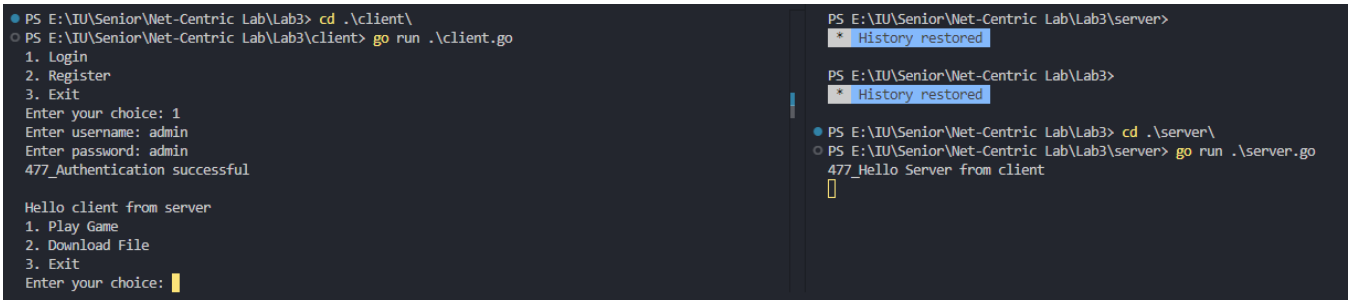
```go
func startGuessingGame(err error, conn net.Conn) {
    for {
        // Send the message to the client, to start the game
        conn.Write([]byte("Start"))

        // Generate random number
        randomNumber := rand.Intn(100) + 1
        target := randomNumber
        fmt.Println("Target number:", target)
        for {
            received := make([]byte, 1024)
            _, err = conn.Read(received)
            if err != nil {
                log.Fatal(err)
            }

            clientMsg := strings.TrimSpace(string(received[:]))

            if strings.Contains(clientMsg, "Exit") {
                break
            }

            if strings.Contains(clientMsg, "Again") {
                break
            }

            // Convert received data to integer
            // guessNumber, err := strconv.Atoi(string(received))
            guessNumber := 0
            fmt.Sscanf(string(received), "%d", &guessNumber)
            fmt.Println("Guess number:", guessNumber)

            if guessNumber < target {
                conn.Write([]byte(key + "_Your guess is too low. Try again."))

            } else if guessNumber > target {
                conn.Write([]byte(key + "_Your guess is too high. Try again."))

            } else {
                conn.Write([]byte(key + "_Congratulations! You guessed the number."))
                break
            }
        }
    }
}
```

```go
func startFileTransfer(conn net.Conn) {
    // Read the requested file name from the client
    buffer := make([]byte, 1024)
    n, err := conn.Read(buffer)
    if err != nil {
        fmt.Println(key+"_Error reading from client:", err)
        return
    }
    fileName := string(buffer[:n])
    fmt.Println("Client requested file:", fileName)

    // Check if the file exists on the server
    if _, err := os.Stat(fileName); os.IsNotExist(err) {
        // File does not exist, notify the client
        conn.Write([]byte("Error: File not found\n"))
        fmt.Println(key+"_File not found:", fileName)
        return
    }

    // File exists, start the file transfer
    file, err := os.Open(fileName)
    if err != nil {
        fmt.Println("Error opening file:", err)
        conn.Write([]byte("Error: Unable to open file\n"))
        return
    }
    defer file.Close()

    // Notify client that file is starting to download
    conn.Write([]byte("File download starting...\n"))

    // Send file data in chunks
    buffer = make([]byte, 1024)
    for {
        n, err := file.Read(buffer)
        if err != nil && err != io.EOF {
            fmt.Println("Error reading file:", err)
            break
        }
        if n == 0 {
            break
        }
        conn.Write(buffer[:n]) // Send the context to the client
    }

    // Notify client the file transfer is complete
    conn.Write([]byte("\nFile download complete\n"))
    fmt.Println("File download completed for:", fileName)
}
```

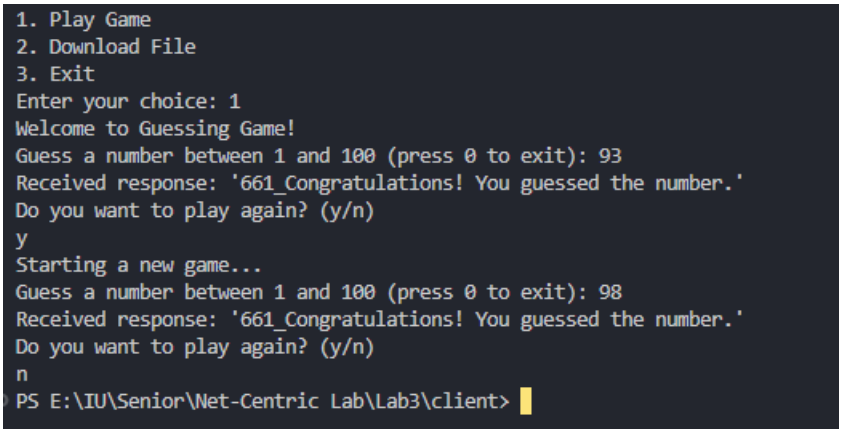
User.json

```json
[
    {
        "username": "admin",
        "password": "YWRtaW4=",
        "fullname": "Administrator",
        "email": [
            "admin@email.com",
            "ad@email.com"
        ],
        "address": [
            "HCM"
        ]
    }
]
```

Result:



Guessing Game:

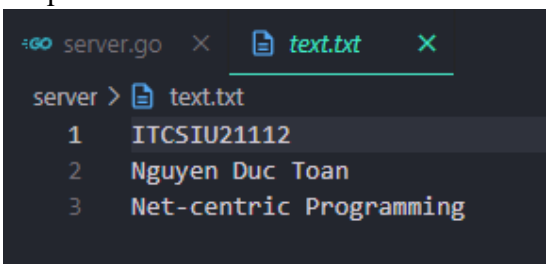


File download:

```
2. Register
3. Exit
Enter your choice: 1
Enter username: admin
Enter password: admin
200_Authentication successful

Hello client from server
1. Play Game
2. Download File
3. Exit
Enter your choice: 2
Enter the name of the file to download: text.txt
File download completed.
PS E:\IU\Senior\Net-Centric Lab\Lab3\client>
```

```
PS E:\IU\Senior\Net-Centric Lab\Lab3\server> go run .\server.go
200_Hello Server from client
Client requested file: text.txt
File download completed for: text.txt
```

- Request File:



- Downloaded File: