Báo cáo kết quả nghiên cứu giai đoạn 1

1. Server

- Hàm main sẽ nhận URL tại cổng 8080 và sẽ thực hiện chạy hàm tương ứng, ở đây là hàm Action.

```
func main() {
    http.HandleFunc("/", Action)
    http.ListenAndServe(":8080", nil)
}
```

- Hàm Action sẽ đọc các file với key là "File" và lưu vào thư mục Uploads trên server, và trả kết quả lại cho phía client biết các thông tin về Tên file, kích thước, đường dẫn tại server.

```
func Action(w http.ResponseWriter, r *http.Request) {
    r.ParseMultipartForm(10 * 1024 * 1024)
    files := r.MultipartForm.File["File"]
    for _, a := range files {
        f, _ := a.Open()
        name := a.Filename[0 : len(a.Filename)-4]
        Type := a.Filename[len(a.Filename)-4 : len(a.Filename)]
        tempFile, _ := ioutil.TempFile("Uploads", name+"*"+Type)
        defer tempFile.Close()
        fileBytes, _ := ioutil.ReadAll(f)
        tempFile.Write(fileBytes)
        //------------------------------
        fmt.Fprintln(w, "\n----------------------")
        fmt.Fprintln(w, "Name file: ", a.Filename)
        fmt.Fprintln(w, "Size file: ", a.Size, "byte")
        fmt.Fprintln(w, "Path on Server: "+tempFile.Name())
        //------------------------------
    }
}
```

2. Client

- Hàm main sẽ chạy hàm input, sau đó sẽ chạy vô vòng lặp với điều kiện dừng là thời gian chạy chương trình sẽ bằng thời gian chúng ta nhập vô
- Sau đó sẽ chạy hàm Action sẽ được gọi liên tục sau mỗi Interval time

```
func main() {
    //--------------------------------------
    var Path, Type, ITime, FTime string
    input(&Path, &Type, &ITime, &FTime)
    //--------------------------------------
    t, _ := strconv.Atoi(ITime)
    FT, _ := strconv.Atoi(FTime)
    TimeStop := time.Now()
    for time.Since(TimeStop).Seconds() < float64(FT) {
        Action(Path, Type)
```

```
        time.Sleep(time.Duration(t) * time.Second)

    }
}
```

- Hàm input với các thông số đầu vào là
  - Path: Đường dẫn folder chứa các file cần gửi
  - Type: Kiểu gửi, song song hay tuần tự
  - ITime: Interval Time
  - FTime: Tổng thời gian chạy chương trình

```
func input(Path *string, Type *string, ITime *string, FTime *string) {
    cin := bufio.NewScanner(os.Stdin)
    fmt.Print("Path: ")
    cin.Scan()
    *Path = cin.Text()
    fmt.Print("Type: ")
    cin.Scan()
    *Type = cin.Text()
    fmt.Print("Interval Time: ")
    cin.Scan()
    *ITime = cin.Text()
    fmt.Print("Full time: ")
    cin.Scan()
    *FTime = cin.Text()
}
```

- Hàm Action sẽ thực hiện mở folder theo Path đã nhập, để đọc được các file cần gửi đi, sau đó sẽ gọi hàm MakeRequest (gửi tuần tự) hoặc gọi hàm MakeRequest2 (gửi song song)

```
func Action(Path string, Type string) {
    f, err := os.Open(Path)
    if err != nil {
        log.Fatal(err)
    }
    files, err := f.Readdir(-1)
    f.Close()
    if err != nil {
        log.Fatal(err)
    }
    //---------------------------------------
    if Type == "1" {
        for _, file := range files {
            MakeRequest(Path + "/" + file.Name())
        }
    } else {
        ch := make(chan bool)
        for _, file := range files {
            go MakeRequest2(ch, Path+"/"+file.Name())
        }
        for _, a := range files {
            if a == nil {
                fmt.Println(nil)
            }
            <-ch
        }
```

```
    }
}
```

- Hàm MakeRequest là hàm gửi file tới server, đầu tiên sẽ mở file cần gửi, và copy toàn bộ file cần gửi vào requestBody và gửi đi bằng phương thức Post, sau đó sẽ in ra thông tin trả về từ server.

```
func MakeRequest(PathFile string) {
    file, err := os.Open(PathFile)
    if err != nil {
        log.Fatalln(err)
    }

    defer file.Close()
    requestBody := &bytes.Buffer{}

    multiPartWriter := multipart.NewWriter(requestBody)

    fileWriter, err := multiPartWriter.CreateFormFile("File", filepath.Base(P
athFile))
    if err != nil {
        log.Fatalln(err)
    }

    _, err = io.Copy(fileWriter, file)
    if err != nil {
        log.Fatalln(err)
    }

    multiPartWriter.Close()

    req, err := http.NewRequest("POST", "http://localhost:8080", requestBody)
    if err != nil {
        log.Fatalln(err)
    }

    req.Header.Set("Content-Type", multiPartWriter.FormDataContentType())

    client := http.Client{}
    response, err := client.Do(req)
    if err != nil {
        log.Fatalln(err)
    }

    body, _ := ioutil.ReadAll(response.Body)
    fmt.Println(string(body))
}
```

- Hàm MakeRequest2 là hàm dùng để gửi song song, sử dụng chanel để routine chính không dừng lại khi các routine gửi đang còn chạy. Khi gọi hàm này sẽ sử dụng cú pháp "go MakeRequest2(ch, PathFile). Khi chạy hàm này sẽ gọi hàm MakeRequest để gửi file đi.

```
func MakeRequest2(ch chan bool, PathFile string) {
    MakeRequest(PathFile)
    ch <- true
}
```