

LAPORAN PRAKTIKUM
PEMROGRAMAN BERBASIS WEB
LATIHAN 07

Untuk Memenuhi Latihan Praktikum Pemrograman Berbasis Web



Dosen: Adi Wahyu Pribadi, S.Si., M.Kom.

Disusun oleh:

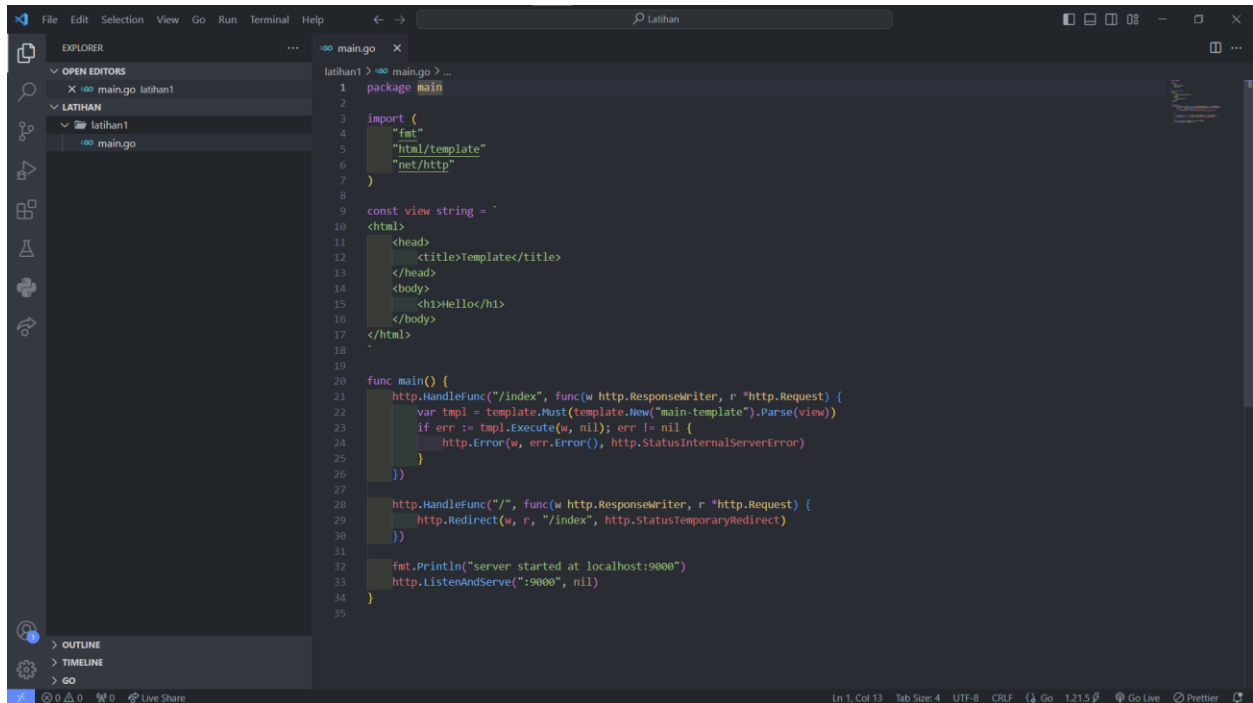
Nama : Muhammad Abduh Harry Malhotra
NIM : 4522210133
Semester : 4
Kelas : A

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PANCASILA
JAKARTA SELATAN
2023/2024

Link Github: <https://github.com/hotraa/PBW/>

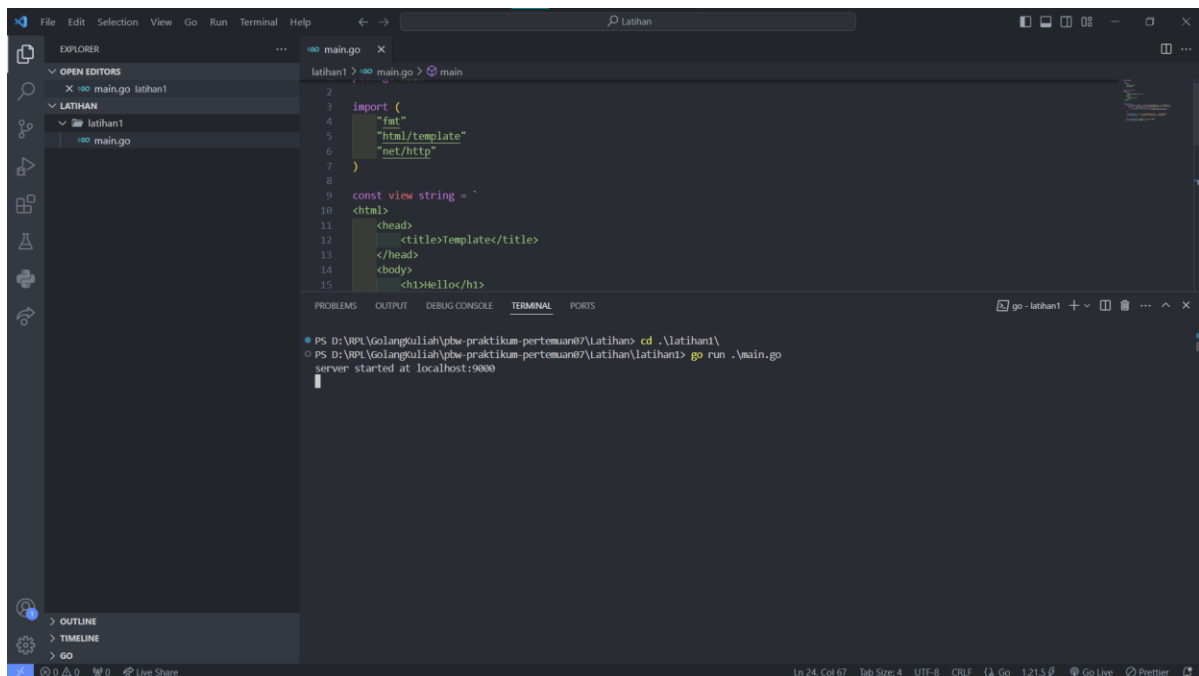
Screenshot Latihan 1 (Template: Render HTML String)

Screenshot Source Code

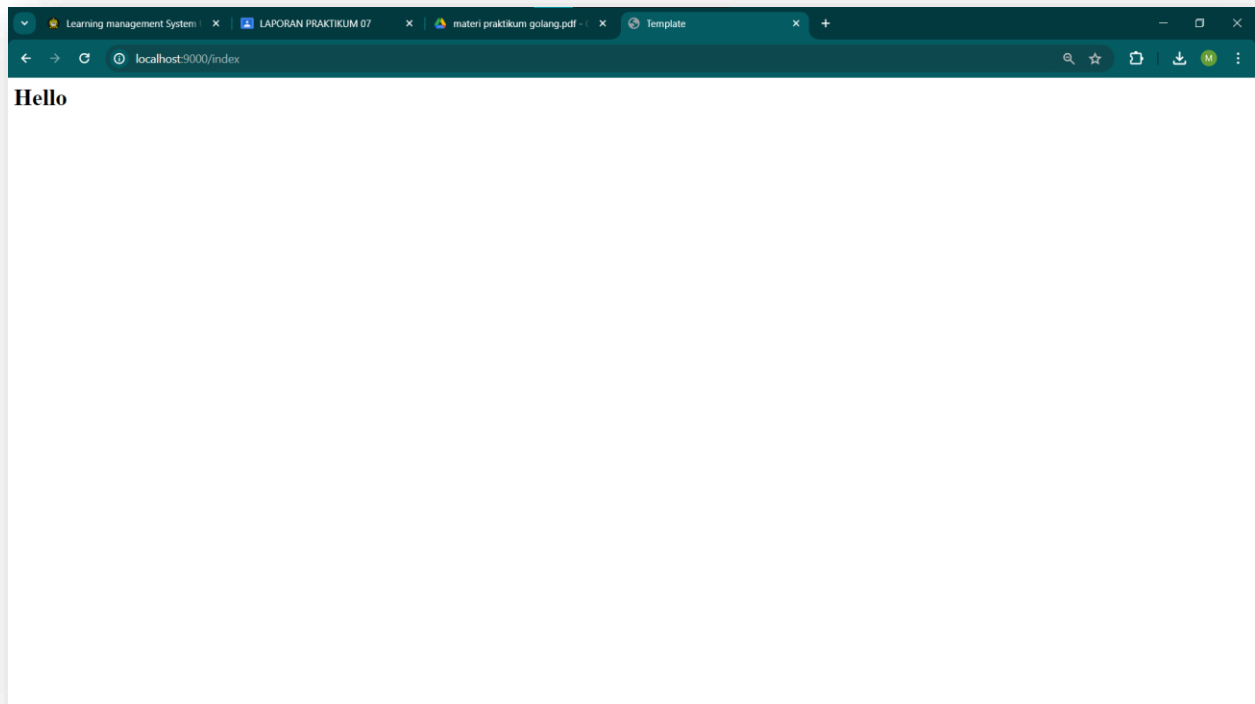


```
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "net/http"
7 )
8
9 const view string = `
10 <html>
11     <head>
12         <title>Template</title>
13     </head>
14     <body>
15         <h1>Hello</h1>
16     </body>
17 </html>
18 `
19
20 func main() {
21     http.HandleFunc("/index", func(w http.ResponseWriter, r *http.Request) {
22         var tmpl = template.Must(template.New("main-template").Parse(view))
23         if err := tmpl.Execute(w, nil); err != nil {
24             http.Error(w, err.Error(), http.StatusInternalServerError)
25         }
26     })
27
28     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
29         http.Redirect(w, r, "/index", http.StatusTemporaryRedirect)
30     })
31
32     fmt.Println("server started at localhost:9000")
33     http.ListenAndServe(":9000", nil)
34 }
35
```

Screenshot Output



```
PS D:\RPL\Golangkulia\pbw-praktikum-pertemuan7\latihan> cd .\latihan1\
PS D:\RPL\Golangkulia\pbw-praktikum-pertemuan7\latihan\latihan1> go run .\main.go
server started at localhost:9000
```



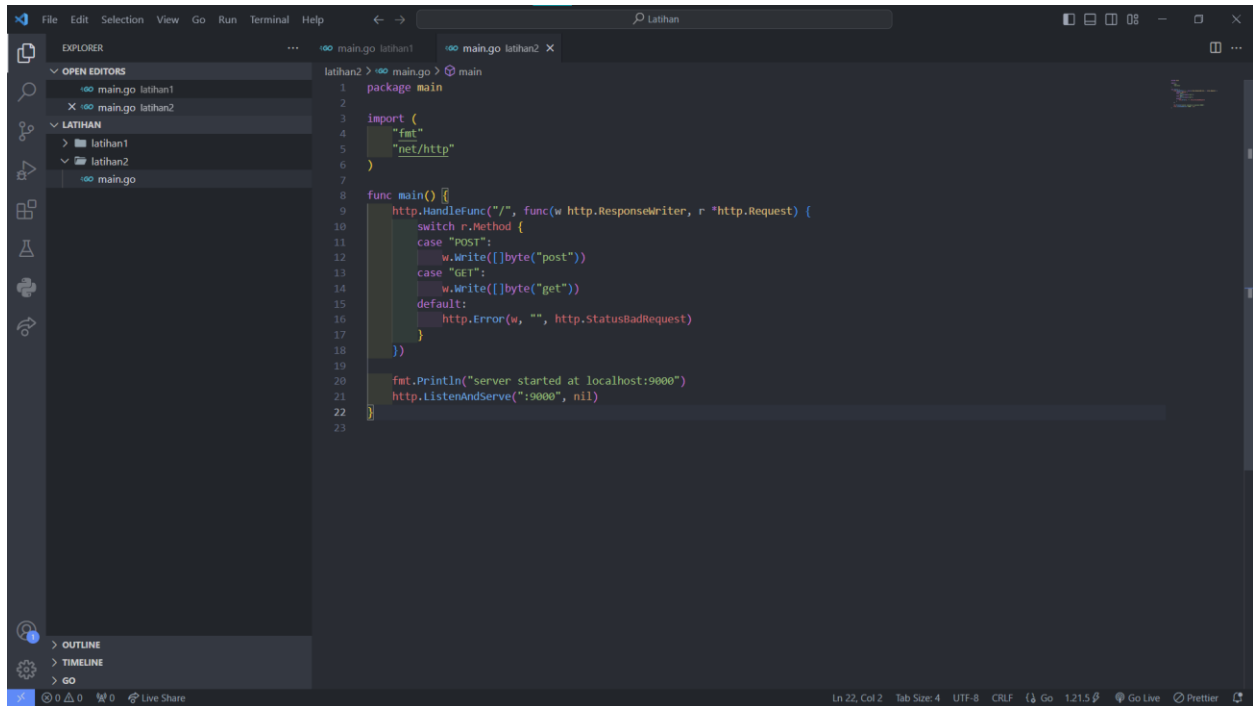
Soal dan Jawab Latihan 1

Jelaskan HTML dan Parsing Method!

- HTML adalah singkatan dari Hypertext Markup Language, yang digunakan untuk menentukan struktur dan konten suatu halaman web, yang termasuk teks, gambar, link, dan elemen lainnya. Dengan HTML, kita dapat membuat tampilan yang terorganisir dan terformat dalam browser.
- Method parsing dalam Go, misal pada `template.Template` adalah proses mengambil template (baik dari string maupun file eksternal) dan mengubahnya menjadi representasi yang dapat digunakan untuk menghasilkan output, seperti halaman HTML. Misal dalam latihan 1, kita menggunakan `template.Must(template.New("main template")).Parse(view)` yang memungkinkan template HTML yang tersimpan dalam variabel `view` diparse menjadi representasi, yang dapat digunakan oleh `tmpl` untuk menghasilkan output HTML yang diinginkan. Dengan demikian, kita bisa menggunakan HTML string langsung sebagai output di web dengan bantuan metode parsing tersebut.

Screenshot Latihan 2 (HTTP Method: POST & GET)

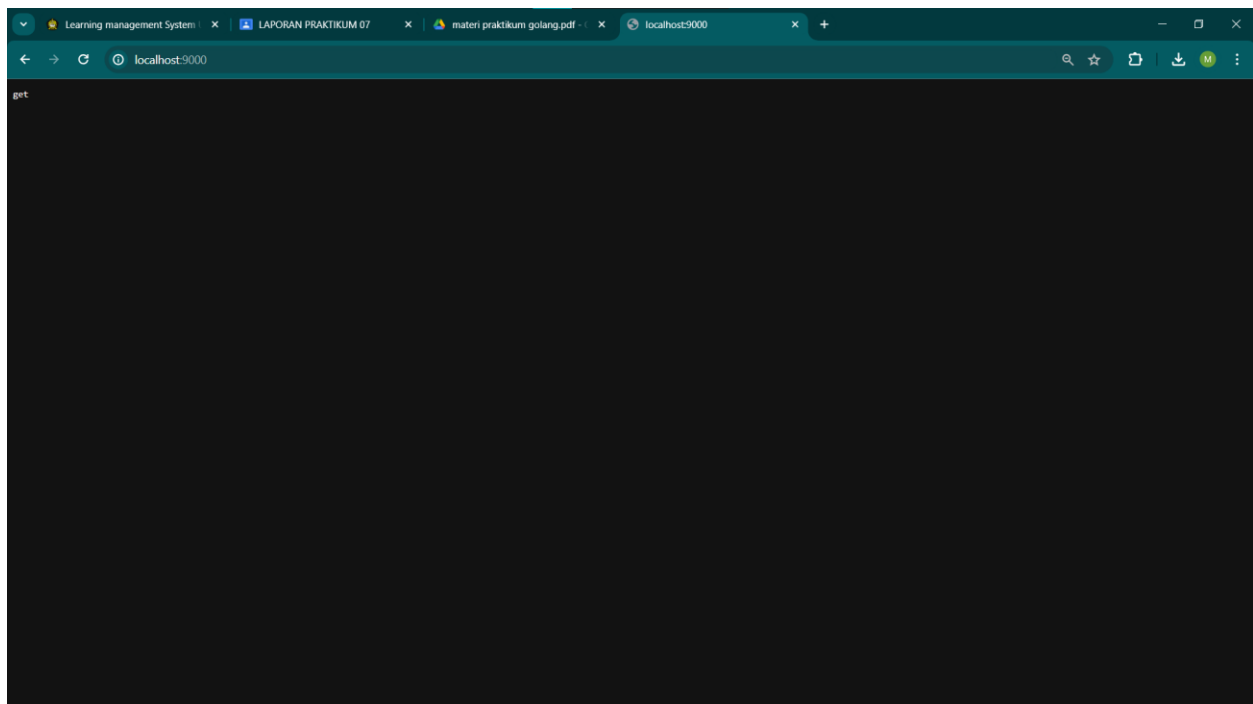
Screenshot Source Code

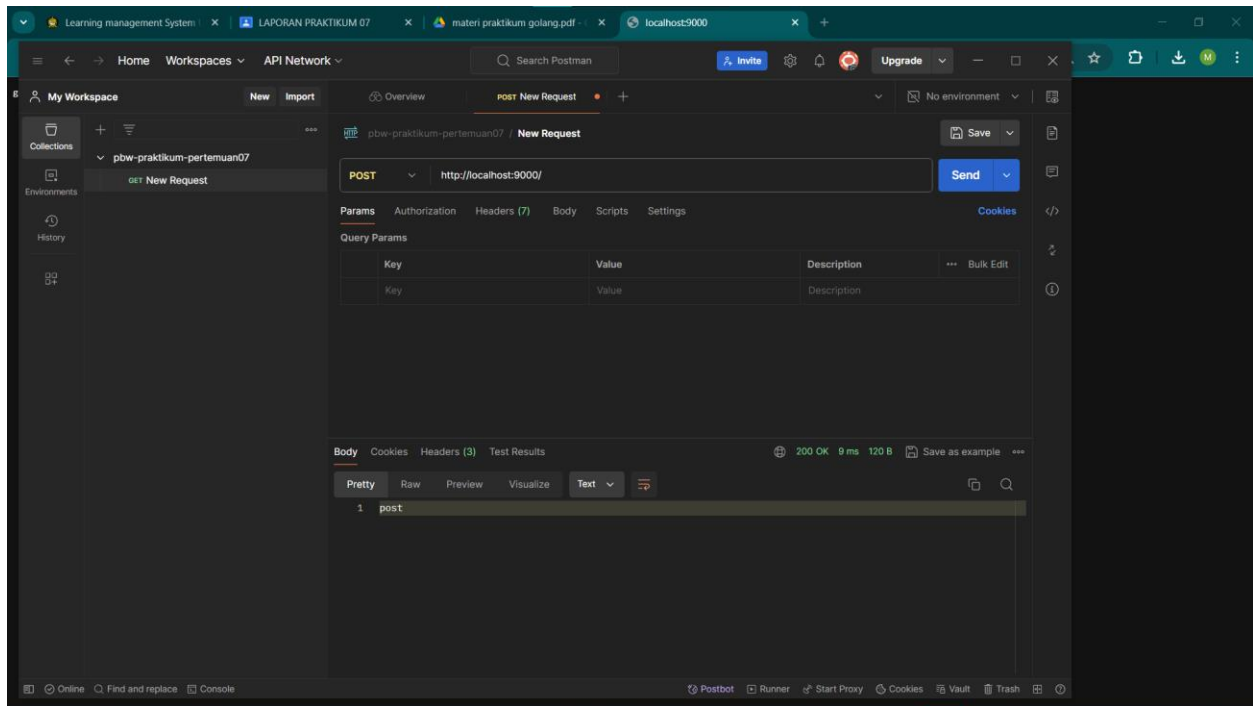
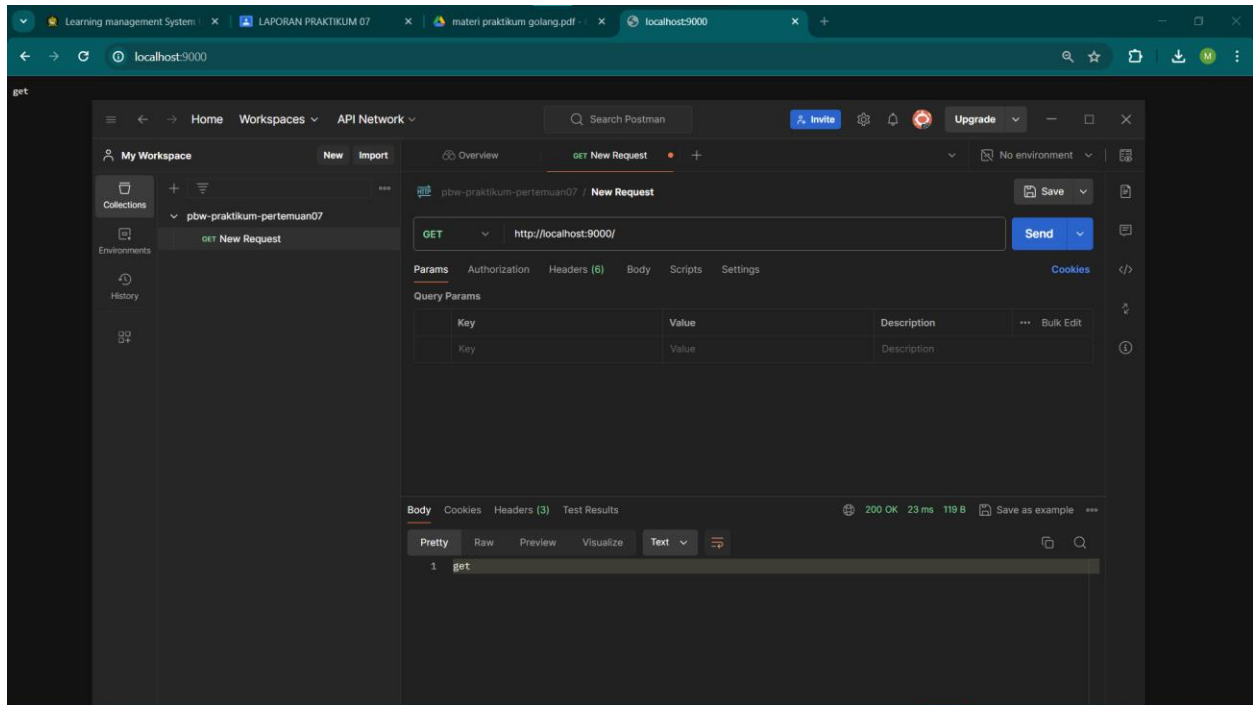


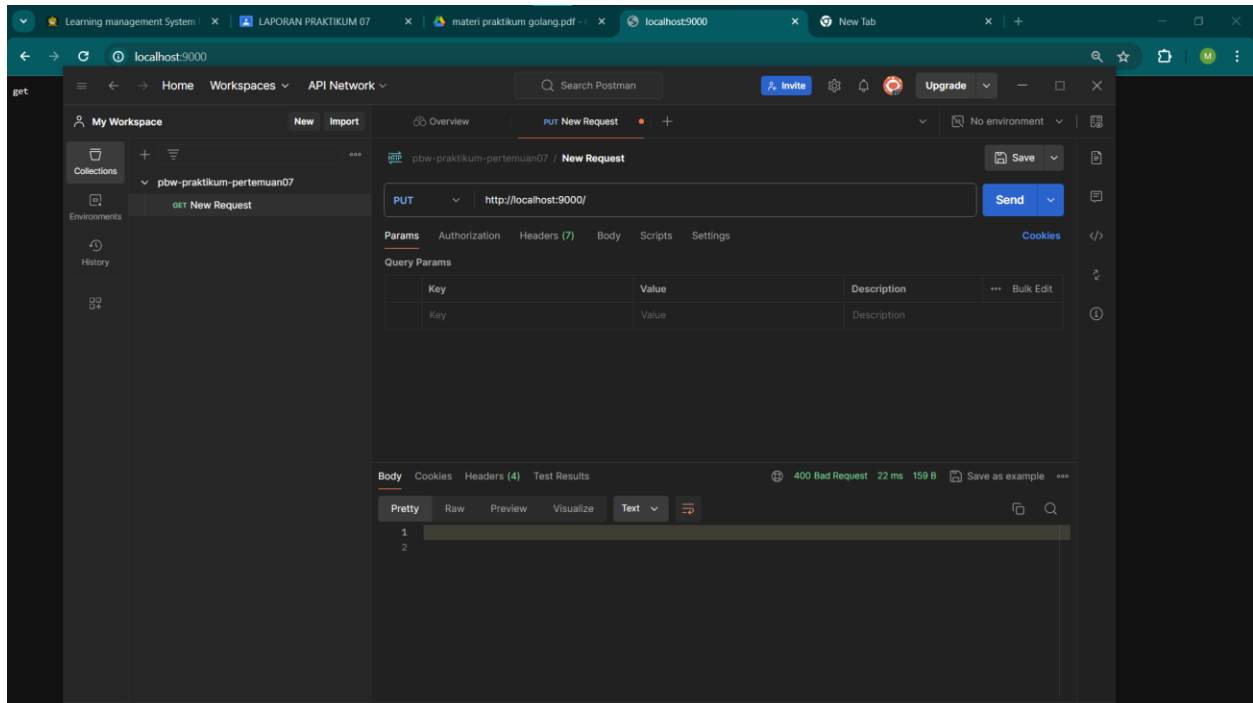
The screenshot shows the Visual Studio Code editor with a Go project. The Explorer sidebar on the left shows a folder named 'LATIHAN' containing 'latihan1' and 'latihan2'. The main editor window displays the source code for 'main.go' in the 'latihan2' folder. The code is a Go program that implements a simple web server using the 'net/http' package. It defines a 'main' function that calls 'http.HandleFunc' to register handlers for 'POST' and 'GET' methods. The 'POST' handler writes 'post' to the response, and the 'GET' handler writes 'get' to the response. The server is started on 'localhost:9000'.

```
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6 )
7
8 func main() {
9     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
10         switch r.Method {
11             case "POST":
12                 w.Write([]byte("post"))
13             case "GET":
14                 w.Write([]byte("get"))
15             default:
16                 http.Error(w, "", http.StatusBadRequest)
17         }
18     })
19
20     fmt.Println("server started at localhost:9000")
21     http.ListenAndServe(":9000", nil)
22 }
23
```

Screenshot Output







Soal dan Jawab Latihan 2

Jelaskan apa itu Postman dan kegunaannya!

- Postman adalah sebuah aplikasi yang digunakan untuk menguji, mengelola, dan mengembangkan layanan web (API). Dengan Postman, kita dapat dengan mudah membuat permintaan HTTP, mengirim data, mengelola permintaan, dan melihat respons dari server. Aplikasi ini juga memberikan kita environment yang ramah/friendly untuk melakukan pengujian dan debugging API dengan berbagai fitur seperti pengaturan header, mengirim permintaan dengan metode HTTP yang berbeda (GET, POST, PUT, DELETE, dll.), menyimpan dan mengelola koleksi permintaan, serta menyusun dan mengelola lingkungan pengujian yang berbeda (environments).

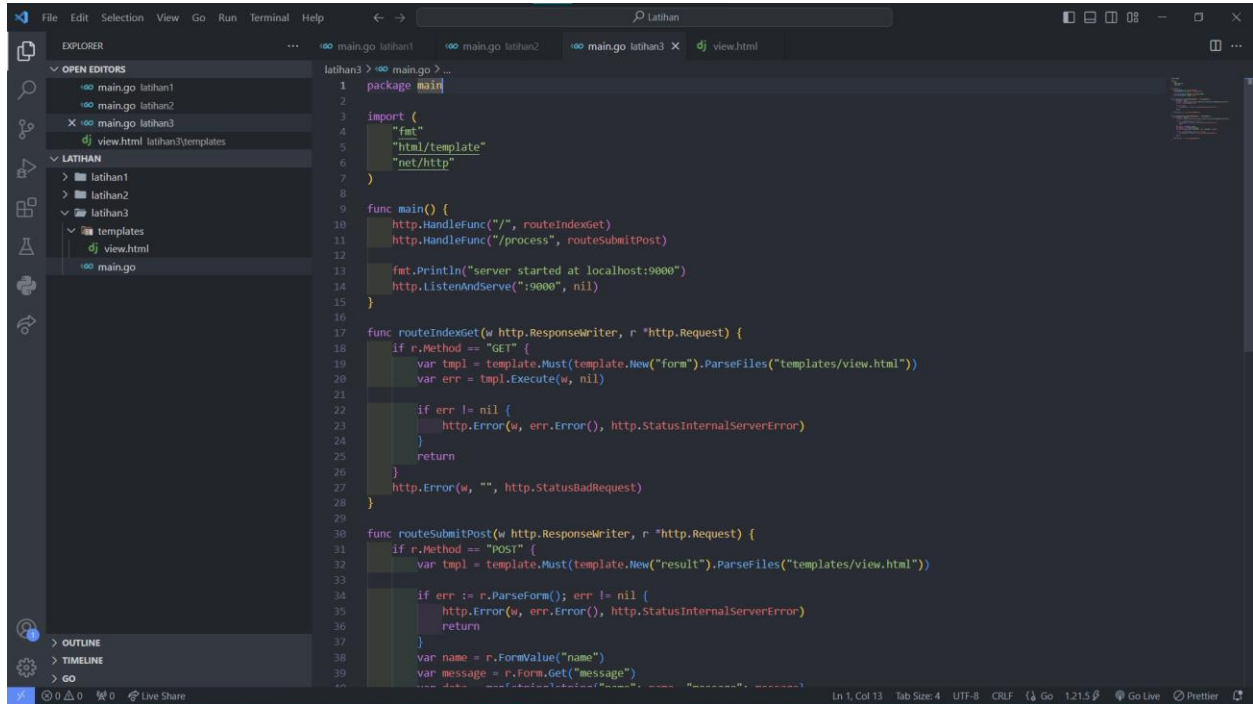
Setelah melakukan testing dengan method GET dan POST. Namun ketika menggunakan selain dua method diatas akan menghasilkan message 400 Bad Request.

- Karena disebabkan oleh pembatasan server yang hanya mengizinkan dua jenis metode, yakni "POST" dan "GET". Ketika server menerima permintaan dengan metode selain dua metode tersebut, server tidak dapat mengolah permintaan sesuai dengan logika yang telah diatur dalam kode. Sebagai tanggapan, server akan mengirimkan pesan kesalahan dengan

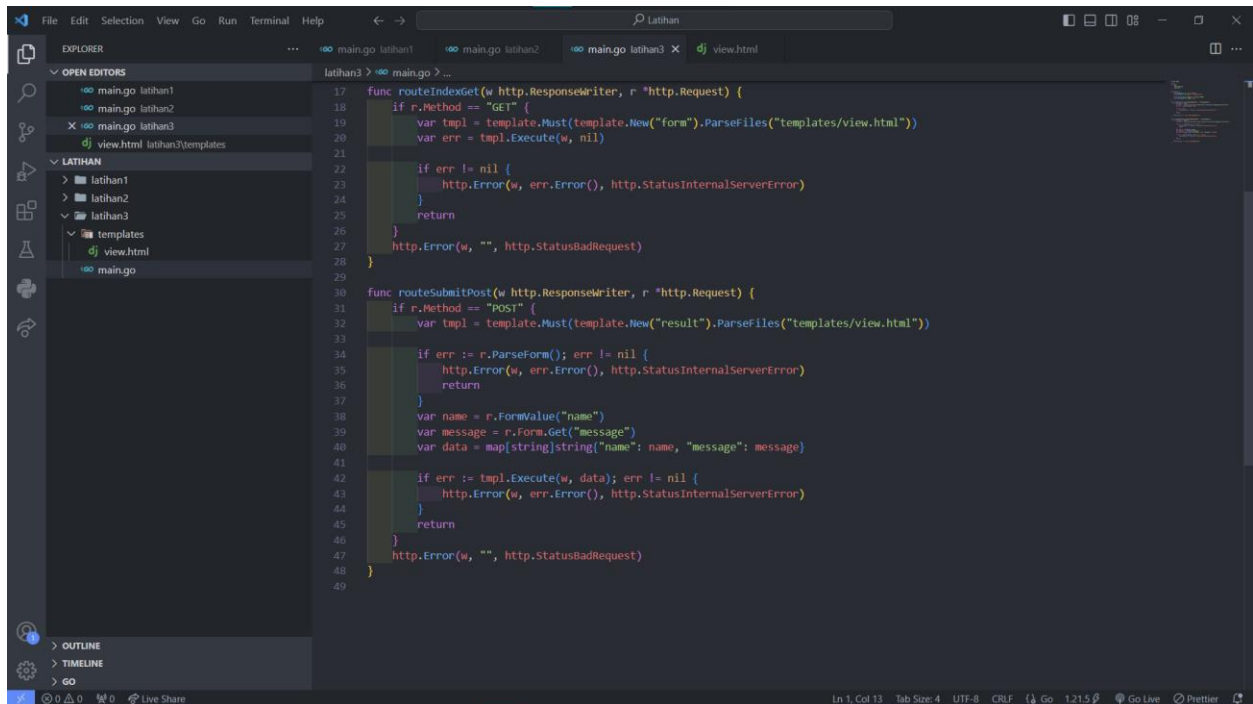
kode status 400 Bad Request, yang menunjukkan bahwa permintaan yang diberikan tidak dapat diproses karena kesalahan dari klien.

Screenshot Latihan 3 (Form Value)

Screenshot Source Code: [main.go](#)

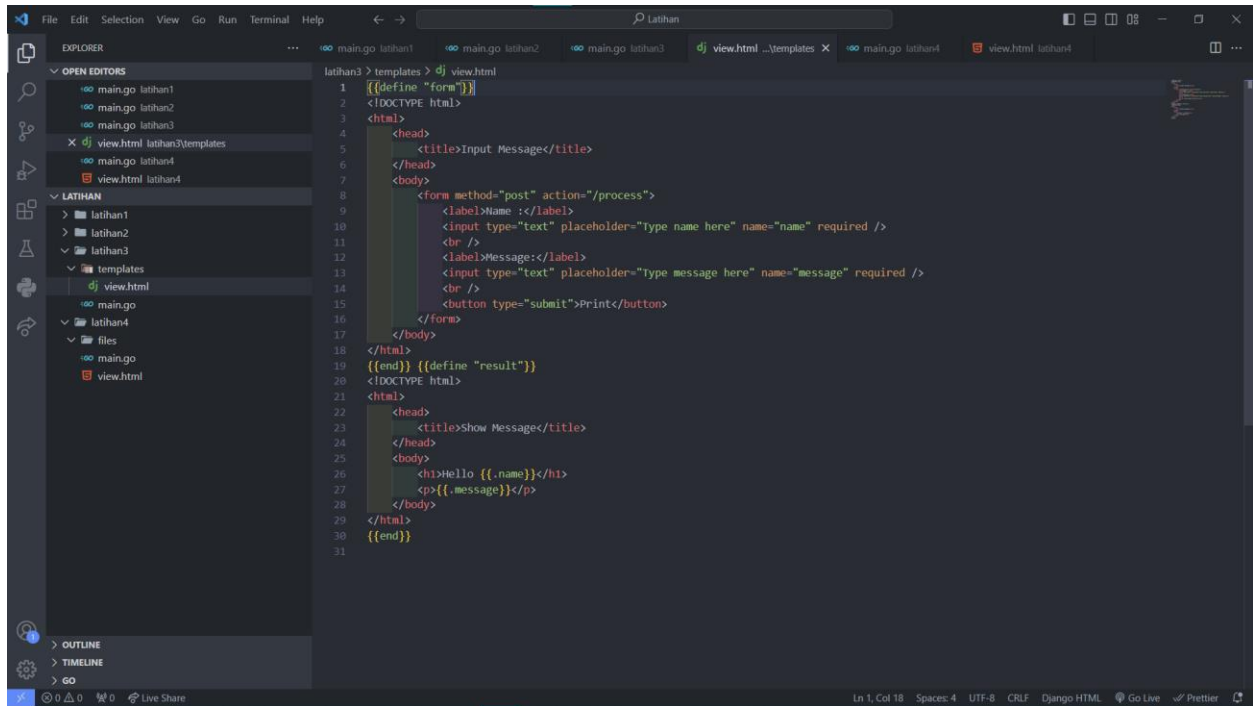


```
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "net/http"
7 )
8
9 func main() {
10     http.HandleFunc("/", routeIndexGet)
11     http.HandleFunc("/process", routeSubmitPost)
12     fmt.Println("server started at localhost:9000")
13     http.ListenAndServe(":9000", nil)
14 }
15
16 func routeIndexGet(w http.ResponseWriter, r *http.Request) {
17     if r.Method == "GET" {
18         var tpl = template.Must(template.New("form").ParseFiles("templates/view.html"))
19         var err = tpl.Execute(w, nil)
20
21         if err != nil {
22             http.Error(w, err.Error(), http.StatusInternalServerError)
23         }
24         return
25     }
26     http.Error(w, "", http.StatusBadRequest)
27 }
28
29 func routeSubmitPost(w http.ResponseWriter, r *http.Request) {
30     if r.Method == "POST" {
31         var tpl = template.Must(template.New("result").ParseFiles("templates/view.html"))
32
33         if err := r.ParseForm(); err != nil {
34             http.Error(w, err.Error(), http.StatusInternalServerError)
35             return
36         }
37         var name = r.FormValue("name")
38         var message = r.Form.Get("message")
39         var data = map[string]string{"name": name, "message": message}
40         if err := tpl.Execute(w, data); err != nil {
41             http.Error(w, err.Error(), http.StatusInternalServerError)
42         }
43         return
44     }
45     http.Error(w, "", http.StatusBadRequest)
46 }
```



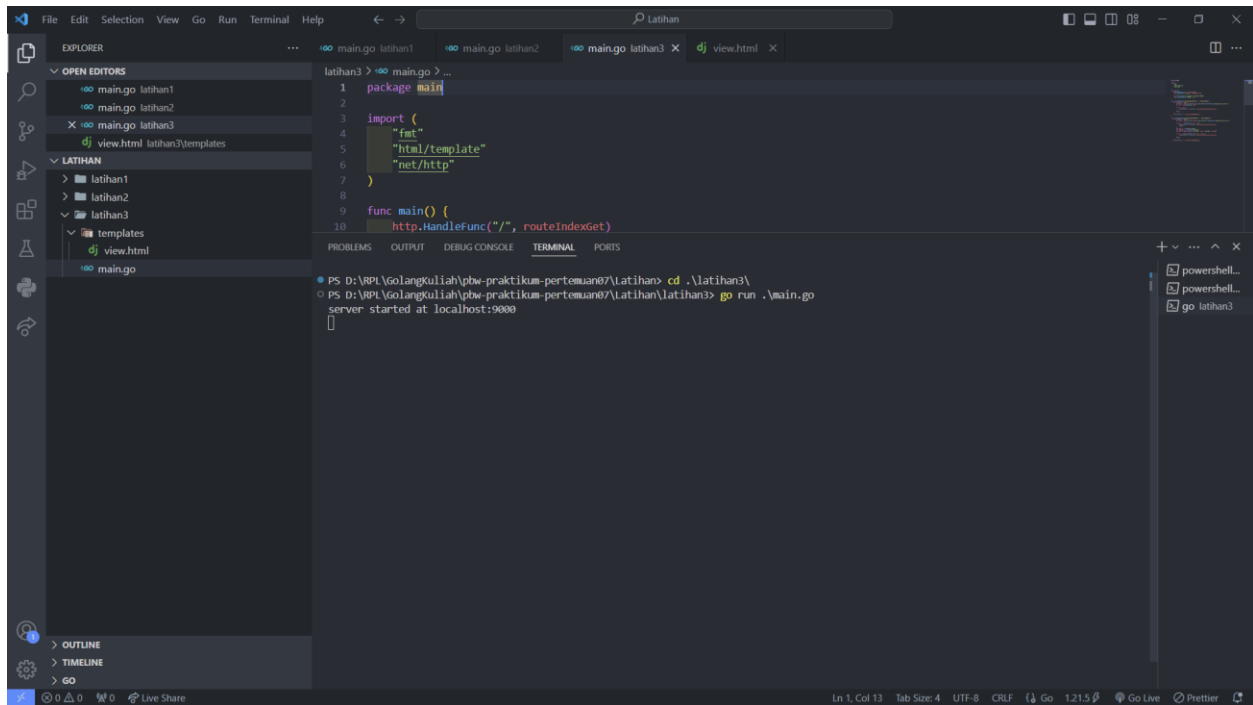
```
17 func routeIndexGet(w http.ResponseWriter, r *http.Request) {
18     if r.Method == "GET" {
19         var tpl = template.Must(template.New("form").ParseFiles("templates/view.html"))
20         var err = tpl.Execute(w, nil)
21
22         if err != nil {
23             http.Error(w, err.Error(), http.StatusInternalServerError)
24         }
25         return
26     }
27     http.Error(w, "", http.StatusBadRequest)
28 }
29
30 func routeSubmitPost(w http.ResponseWriter, r *http.Request) {
31     if r.Method == "POST" {
32         var tpl = template.Must(template.New("result").ParseFiles("templates/view.html"))
33
34         if err := r.ParseForm(); err != nil {
35             http.Error(w, err.Error(), http.StatusInternalServerError)
36             return
37         }
38         var name = r.FormValue("name")
39         var message = r.Form.Get("message")
40         var data = map[string]string{"name": name, "message": message}
41         if err := tpl.Execute(w, data); err != nil {
42             http.Error(w, err.Error(), http.StatusInternalServerError)
43         }
44         return
45     }
46     http.Error(w, "", http.StatusBadRequest)
47 }
48 }
```

Screenshot Source Code: [view.html](#)



```
1 {{define "form"}}
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <title>Input Message</title>
6   </head>
7   <body>
8     <form method="post" action="/process">
9       <label>Name :</label>
10      <input type="text" placeholder="Type name here" name="name" required />
11      <br />
12      <label>Message:</label>
13      <input type="text" placeholder="Type message here" name="message" required />
14      <br />
15      <button type="submit">Print</button>
16    </form>
17  </body>
18 </html>
19 {{end}} {{define "result"}}
20 <!DOCTYPE html>
21 <html>
22   <head>
23     <title>Show Message</title>
24   </head>
25   <body>
26     <h1>Hello {{.name}}</h1>
27     <p>{{.message}}</p>
28   </body>
29 </html>
30 {{end}}
```

Screenshot Output

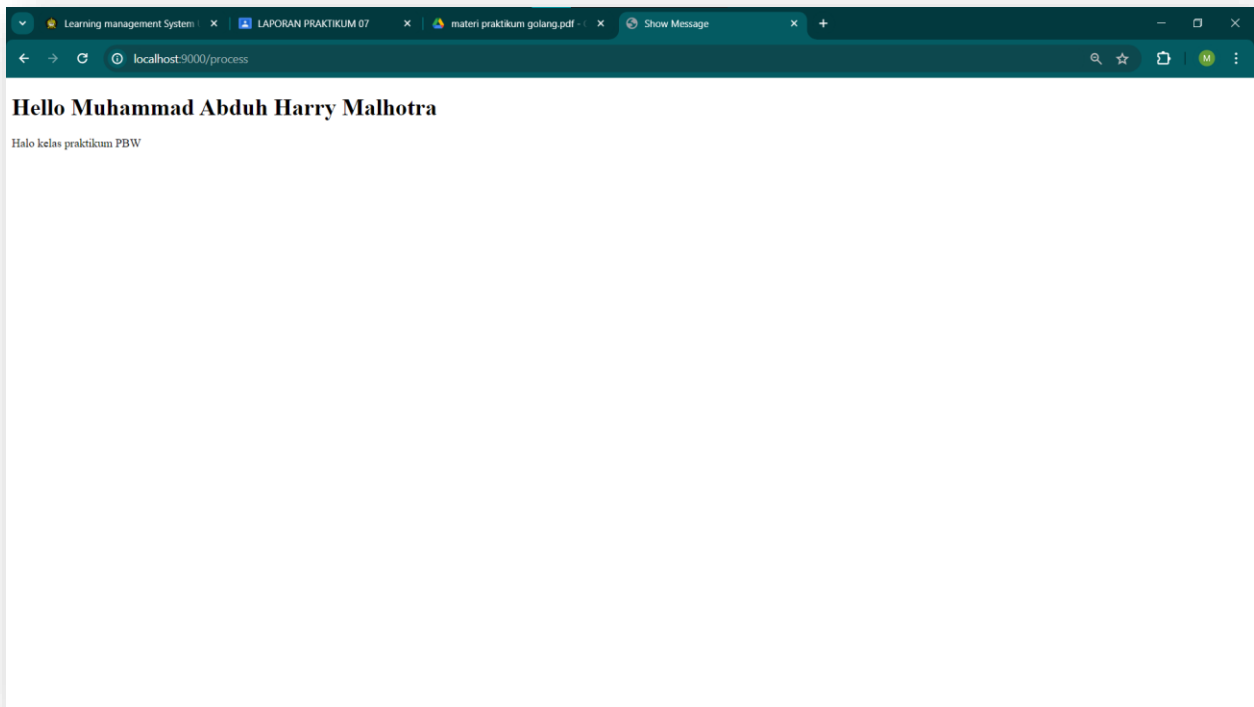
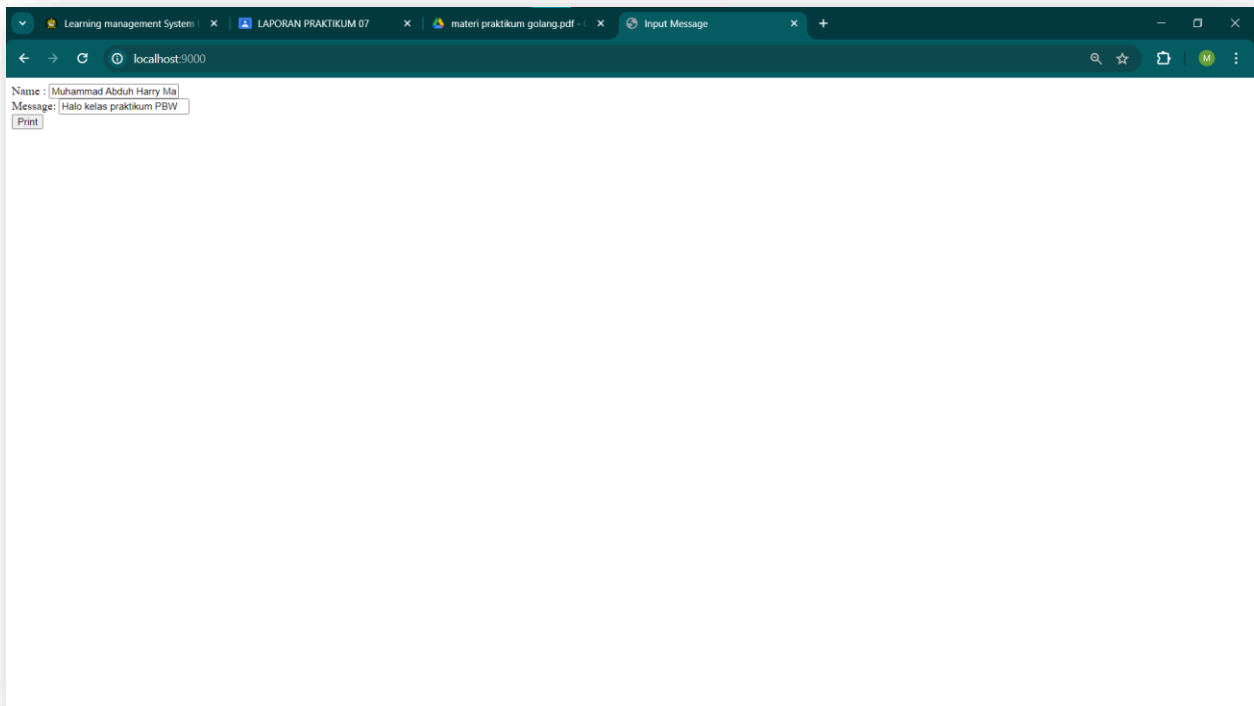


```
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "net/http"
7 )
8
9 func main() {
10    http.HandleFunc("/", routeIndexGet)
11 }
```

PS D:\RPL\GolangKuliah\pbw-praktikum-pertemuan07\latihan> cd .\latihan3\

PS D:\RPL\GolangKuliah\pbw-praktikum-pertemuan07\latihan3> go run .\main.go

server started at localhost:9000

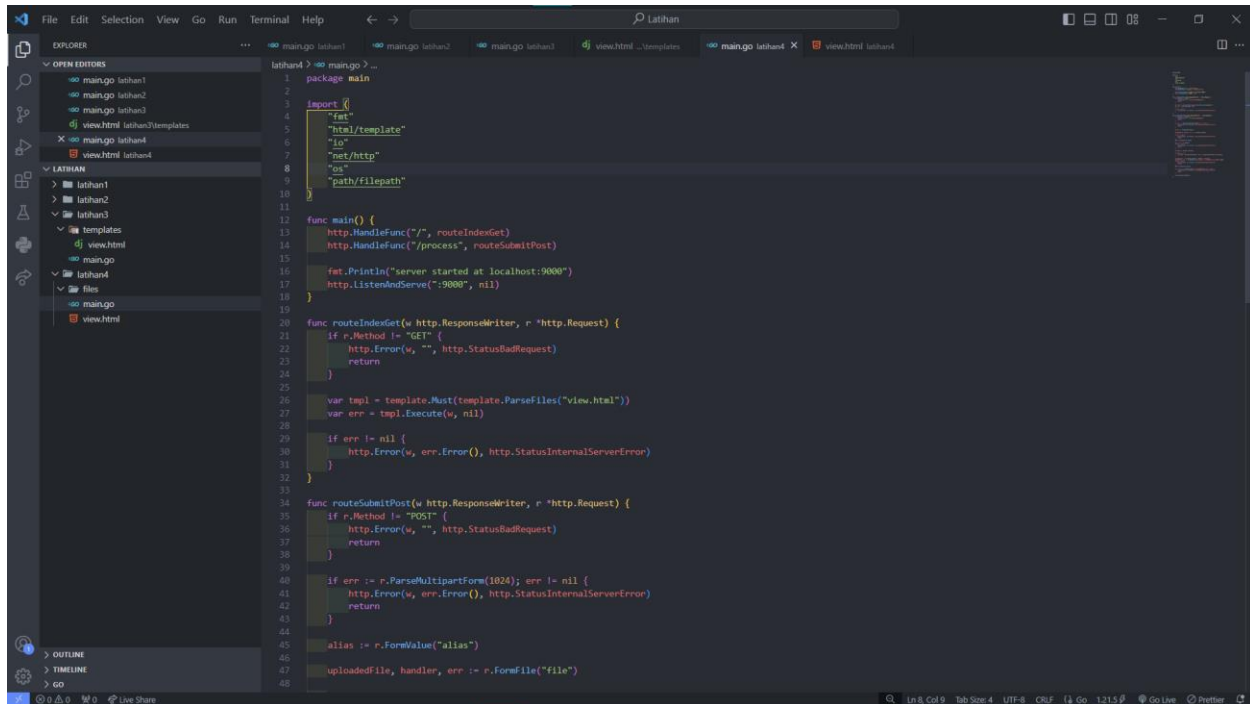


Soal dan Jawab Latihan 3

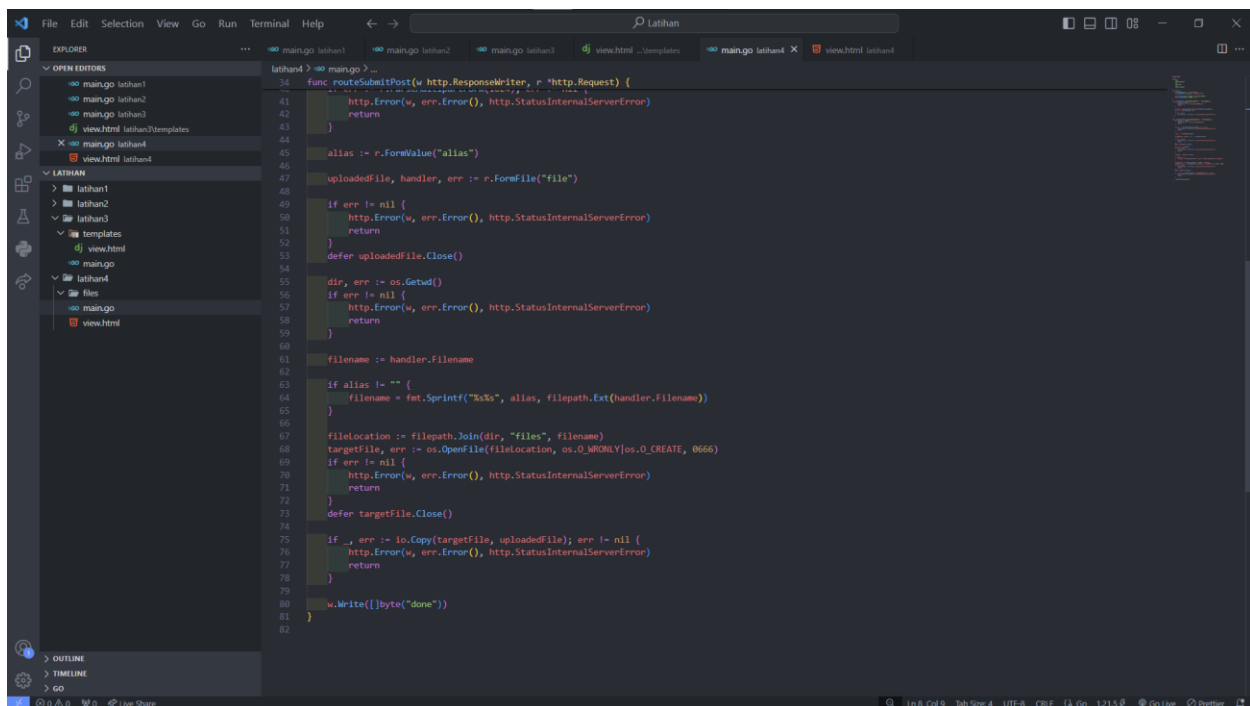
- Front end adalah bagian dari sebuah aplikasi atau situs web yang terlihat dan diakses langsung oleh user yang mencakup desain tampilan, user interface, interaksi langsung dengan user seperti button, form, dan animasi yang ditampilkan di browser atau perangkat pengguna lainnya. Front end juga bertanggung jawab untuk mengirimkan permintaan ke server dan menampilkan respons dari server kepada user.
- Back end adalah bagian dari aplikasi atau situs web yang berada di belakang layar dan tidak langsung diakses oleh user yang meliputi server, database, dan logika aplikasi yang mengelola dan memproses data, mengelola permintaan dari front end, dan menghasilkan respons yang dikirim kembali ke front end. Back end juga bertanggung jawab untuk mengatur keamanan dan kinerja sistem.
- API (Application Programming Interface) adalah kumpulan aturan dan protokol yang memungkinkan berbagai aplikasi atau komponen perangkat lunak berkomunikasi dan berinteraksi satu sama lain. API sering digunakan untuk mengintegrasikan sistem, sehingga memungkinkan aplikasi berbeda berinteraksi dan berbagi data. API juga memungkinkan untuk mengakses dan menggunakan fungsi atau layanan sistem tertentu, seperti pemanggilan fungsi, pengiriman data, atau interaksi lainnya.

Screenshot Latihan 4 (Form Upload File)

Screenshot Source Code: [main.go](#)

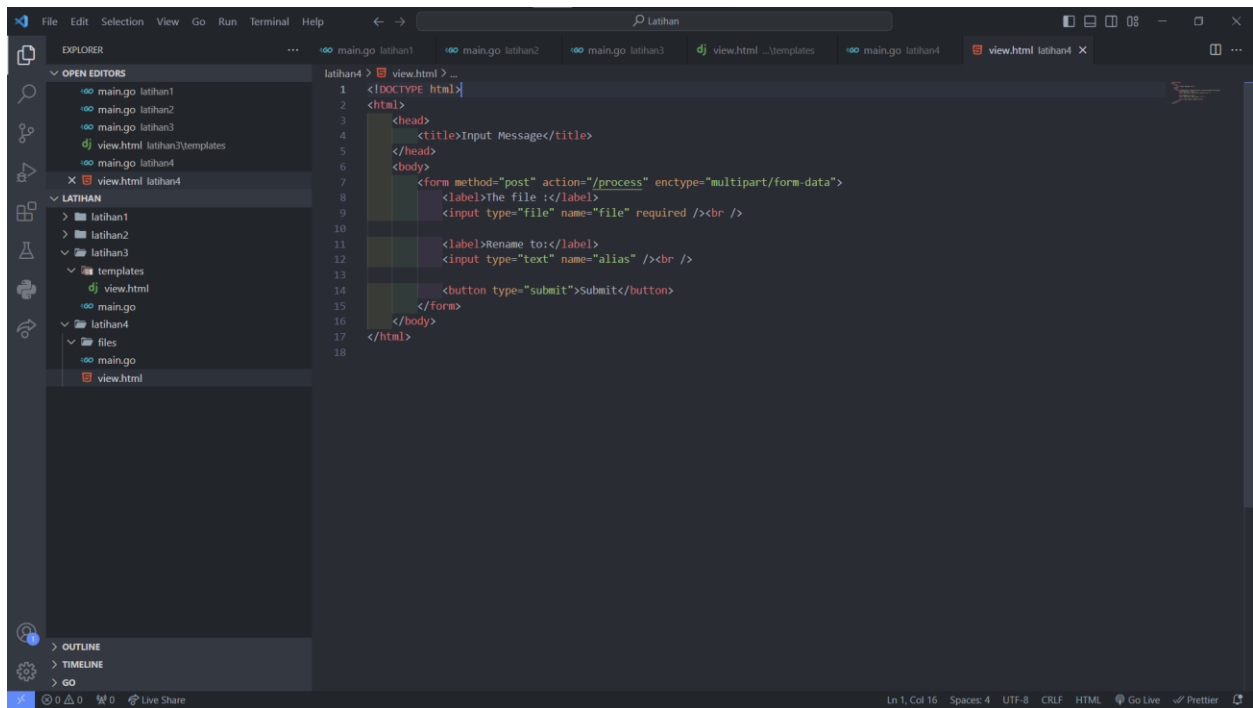


```
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "io"
7     "net/http"
8     "os"
9     "path/filepath"
10 )
11
12 func main() {
13     http.HandleFunc("/", routeIndexGet)
14     http.HandleFunc("/process", routeSubmitPost)
15     fmt.Println("server started at localhost:9000")
16     http.ListenAndServe(":9000", nil)
17 }
18
19
20 func routeIndexGet(w http.ResponseWriter, r *http.Request) {
21     if r.Method != "GET" {
22         http.Error(w, "", http.StatusBadRequest)
23         return
24     }
25
26     tmpl := template.Must(template.ParseFiles("view.html"))
27     var err = tmpl.Execute(w, nil)
28
29     if err != nil {
30         http.Error(w, err.Error(), http.StatusInternalServerError)
31     }
32 }
33
34 func routeSubmitPost(w http.ResponseWriter, r *http.Request) {
35     if r.Method != "POST" {
36         http.Error(w, "", http.StatusBadRequest)
37         return
38     }
39
40     if err := r.ParseMultipartForm(1024); err != nil {
41         http.Error(w, err.Error(), http.StatusInternalServerError)
42         return
43     }
44
45     alias := r.FormValue("alias")
46     uploadedFile, handler, err := r.FormFile("file")
47 }
```



```
48     defer uploadedFile.Close()
49
50     dir, err := os.Getwd()
51     if err != nil {
52         http.Error(w, err.Error(), http.StatusInternalServerError)
53         return
54     }
55
56     filename := handler.Filename
57
58     if alias != "" {
59         filename = fmt.Sprintf("%s%s", alias, filepath.Ext(handler.Filename))
60     }
61
62     fileLocation := filepath.Join(dir, "files", filename)
63     targetFile, err := os.OpenFile(fileLocation, os.O_WRONLY|os.O_CREATE, 0666)
64     if err != nil {
65         http.Error(w, err.Error(), http.StatusInternalServerError)
66         return
67     }
68     defer targetFile.Close()
69
70     if _, err := io.Copy(targetFile, uploadedFile); err != nil {
71         http.Error(w, err.Error(), http.StatusInternalServerError)
72         return
73     }
74
75     w.Write([]byte("done"))
76 }
```

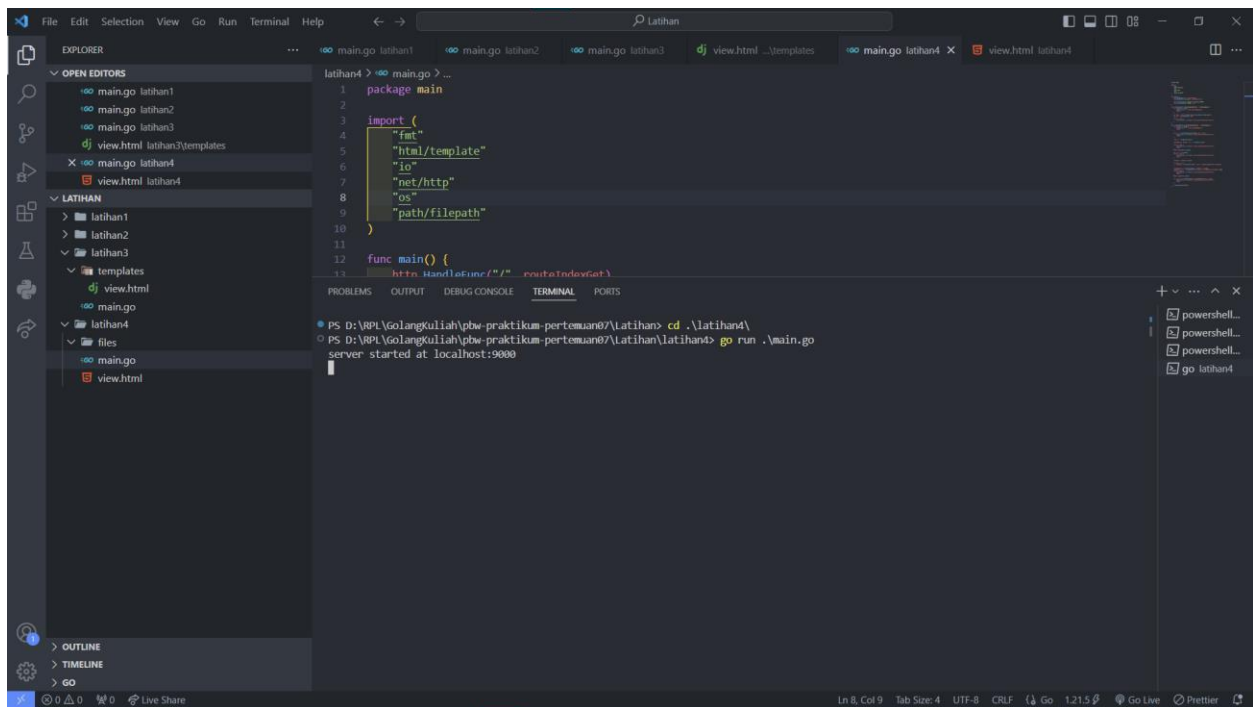
Screenshot Source Code: [view.html](#)



The screenshot shows the Visual Studio Code editor with the file explorer on the left and the editor window displaying the source code of `view.html`. The code is an HTML document with a form for file upload and renaming.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Input Message</title>
5   </head>
6   <body>
7     <form method="post" action="/process" enctype="multipart/form-data">
8       <label>The file :</label>
9       <input type="file" name="file" required /><br />
10
11       <label>Rename to:</label>
12       <input type="text" name="alias" /><br />
13
14       <button type="submit">Submit</button>
15     </form>
16   </body>
17 </html>
18
```

Screenshot Output

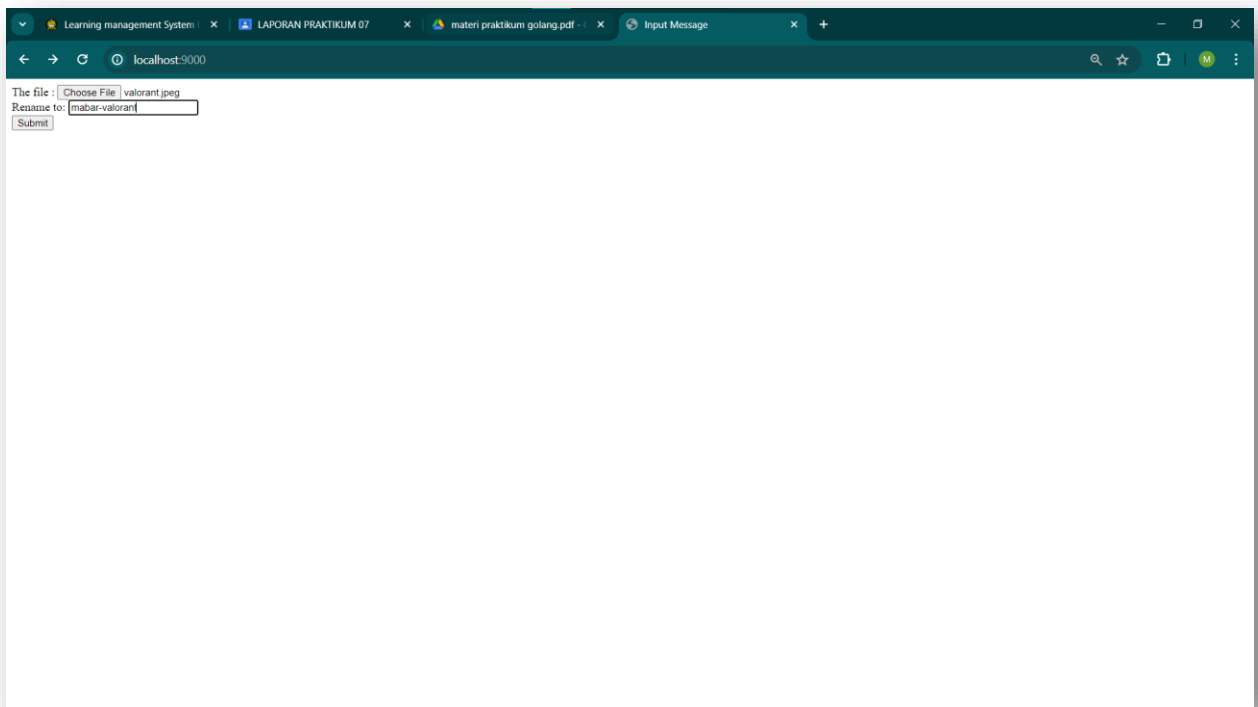
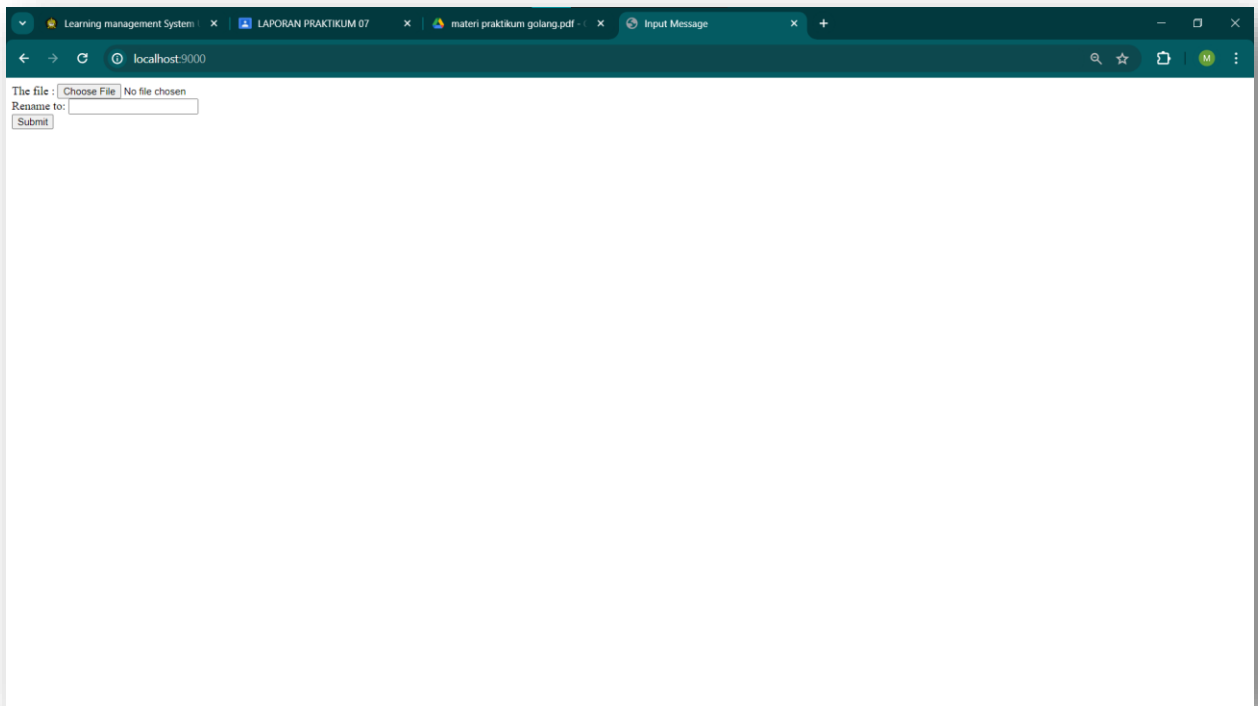


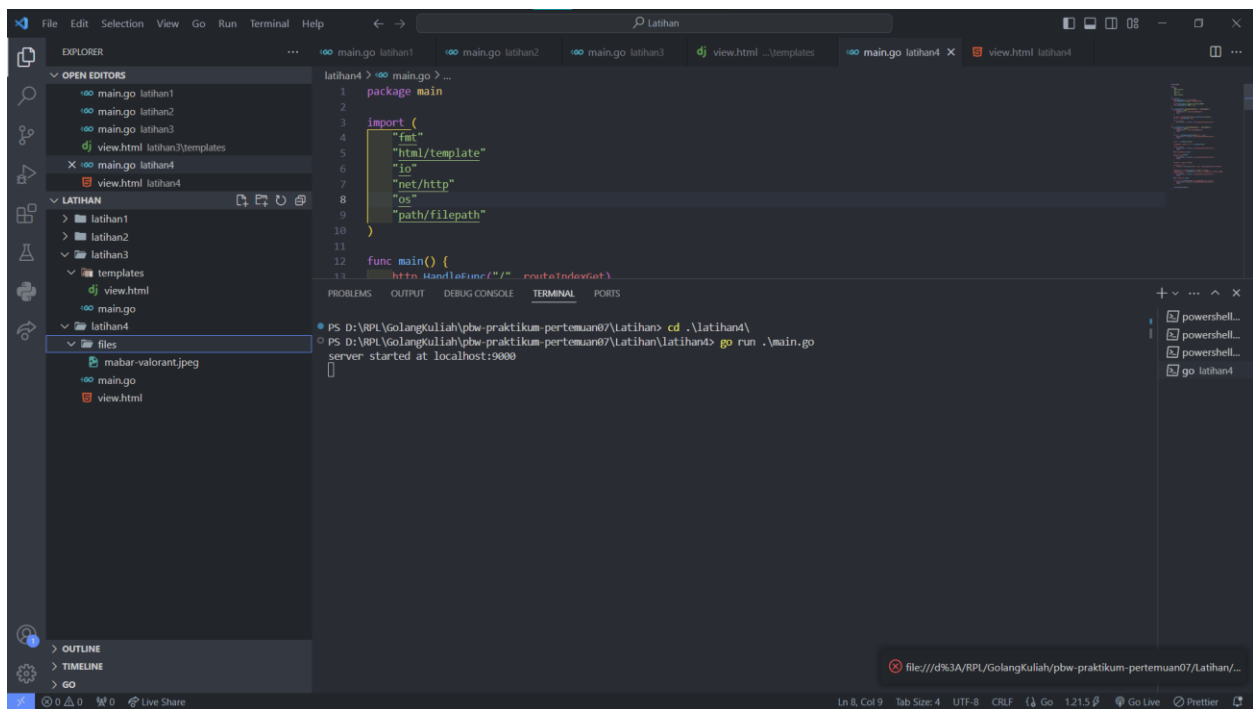
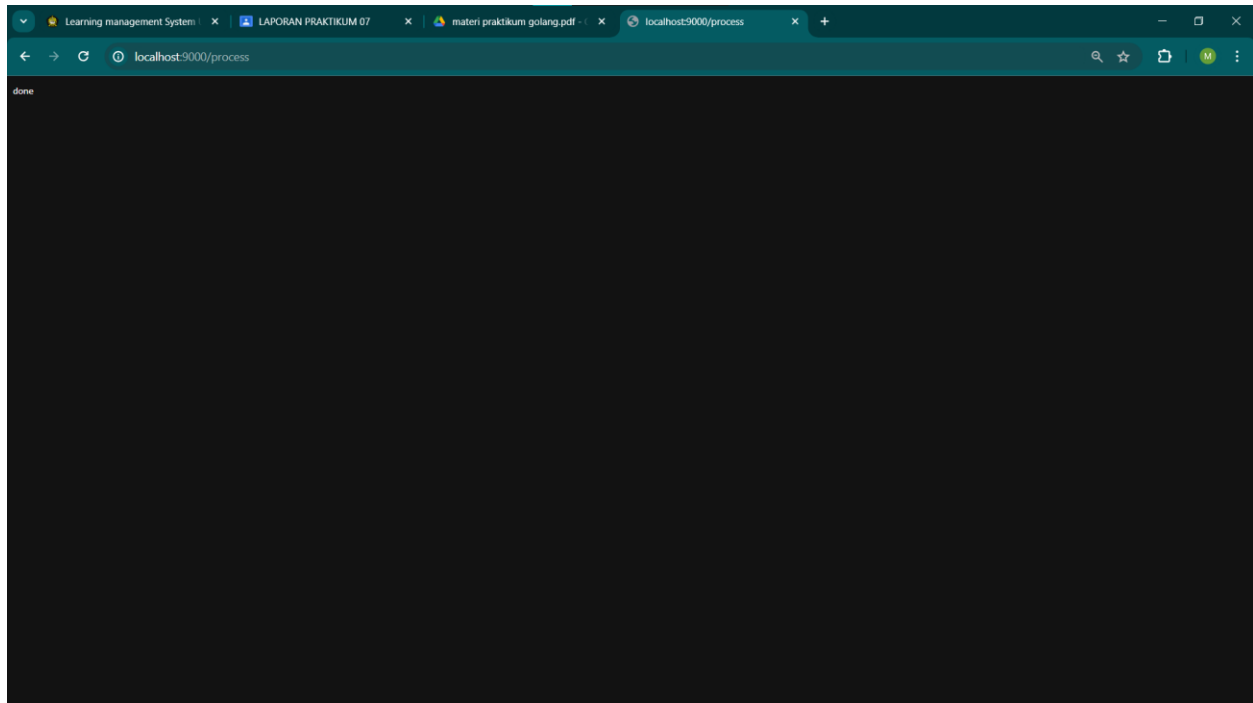
The screenshot shows the Visual Studio Code editor with the file explorer on the left and the editor window displaying the source code of `main.go`. The code is a Go program that serves the `view.html` file. The terminal at the bottom shows the command to run the program and the output indicating the server is started at `localhost:9000`.

```
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "io"
7     "net/http"
8     "os"
9     "path/filepath"
10 )
11
12 func main() {
13     http.HandleFunc("/?f=" + filepath.Join(".", "view.html"))
14 }
```

Terminal Output:

```
PS D:\VPL\golangkulia\pbw-praktikum-pertemuan0\Latihan> cd .\latihan4\
PS D:\VPL\golangkulia\pbw-praktikum-pertemuan0\Latihan\latihan4> go run .\main.go
server started at localhost:9000
```





Soal dan Jawab Latihan 4

Perlu diperhatikan, pada tag <form> perlu ditambahkan atribut **enctype= "multipart/form-data"**. Kenapa perlu ditambahkan?

- Atribut enctype="multipart/form-data" pada tag <form> diperlukan saat mengirimkan file melalui formulir HTML. Karena jenis data file tidak dapat dikodekan dalam format standar URL seperti data teks atau bilangan. Dengan menggunakan enctype="multipart/form-data", data file dapat dikirimkan dengan benar melalui permintaan HTTP POST tanpa kehilangan informasi atau mengalami masalah dalam pengkodean data file. Dengan demikian, enctype="multipart/form-data" memungkinkan form untuk mengirimkan file ke server dengan benar melalui permintaan POST.

Kesimpulan

Dari latihan 1-4 yang telah saya lakukan, dapat disimpulkan beberapa hal yang penting dalam pengembangan aplikasi web.

1. Pertama, HTML sebagai bahasa markup sangat penting dalam menentukan struktur dan konten halaman web, sedangkan parsing method dalam bahasa pemrograman seperti Go memungkinkan penggunaan template HTML untuk menghasilkan output yang diinginkan di web.
2. Kedua, aplikasi Postman memiliki peran penting dalam pengujian, manajemen, dan pengembangan layanan web (API). Dengan Postman, kita dapat dengan mudah menguji permintaan HTTP, mengelola respons server, serta menyimpan dan mengelola permintaan dalam berbagai koleksi untuk pengujian lebih lanjut.
3. Ketiga, pemahaman tentang front end, back end, dan API menjadi dasar penting dalam pengembangan aplikasi web. Front end bertanggung jawab atas tampilan yang terlihat dan interaksi langsung dengan user, sementara back end mengatur logika aplikasi, pengolahan data, dan menghasilkan respons yang dikirimkan kembali ke front end. API memungkinkan integrasi antara berbagai aplikasi dan layanan, sehingga memungkinkan interaksi dan pertukaran data yang lebih lancar.
4. Terakhir, atribut enctype="multipart/form-data" pada tag <form> sangat penting saat mengirimkan file melalui formulir HTML. Atribut ini memastikan bahwa data file dapat

dikirimkan dengan benar melalui permintaan HTTP POST, sehingga menghindari masalah dalam pengiriman file.

Upload ke Github

```
MINGW64/d/RPL/GolangKuliah
muhamRLAPTOP-CSP9DNVH MINGW64 /d/RPL/GolangKuliah (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  pbw-praktikum-pertemuan07/

nothing added to commit but untracked files present (use "git add" to track)
muhamRLAPTOP-CSP9DNVH MINGW64 /d/RPL/GolangKuliah (master)
$ git add pbw-praktikum-pertemuan07/
muhamRLAPTOP-CSP9DNVH MINGW64 /d/RPL/GolangKuliah (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   pbw-praktikum-pertemuan07/Latihan/latihan1/main.go
    new file:   pbw-praktikum-pertemuan07/Latihan/latihan2/main.go
    new file:   pbw-praktikum-pertemuan07/Latihan/latihan3/main.go
    new file:   pbw-praktikum-pertemuan07/Latihan/latihan3/templates/view.ht
ml
    new file:   pbw-praktikum-pertemuan07/Latihan/latihan4/files/mabar-valor
ant.jpeg
    new file:   pbw-praktikum-pertemuan07/Latihan/latihan4/main.go
    new file:   pbw-praktikum-pertemuan07/Latihan/latihan4/view.html

muhamRLAPTOP-CSP9DNVH MINGW64 /d/RPL/GolangKuliah (master)
$ git commit -m "Source Code Praktikum Latihan 7"
[master d2982c4] Source Code Praktikum Latihan 7
 7 files changed, 232 insertions(+)
 create mode 100644 pbw-praktikum-pertemuan07/Latihan/latihan1/main.go
 create mode 100644 pbw-praktikum-pertemuan07/Latihan/latihan2/main.go
 create mode 100644 pbw-praktikum-pertemuan07/Latihan/latihan3/main.go
 create mode 100644 pbw-praktikum-pertemuan07/Latihan/latihan3/templates/view.ht
ml
 create mode 100644 pbw-praktikum-pertemuan07/Latihan/latihan4/files/mabar-valor
ant.jpeg
 create mode 100644 pbw-praktikum-pertemuan07/Latihan/latihan4/main.go
 create mode 100644 pbw-praktikum-pertemuan07/Latihan/latihan4/view.html

muhamRLAPTOP-CSP9DNVH MINGW64 /d/RPL/GolangKuliah (master)
$ git push -u origin master
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 12 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (17/17), 15.91 KiB | 5.30 MiB/s, done.
Total 17 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/hotraa/PBW.git
 09f9f17..d2982c4  master -> master
branch 'master' set up to track 'origin/master'.

muhamRLAPTOP-CSP9DNVH MINGW64 /d/RPL/GolangKuliah (master)
$
```