

LATIHAN PERTEMUAN 8
PRAKTIKUM PEMOGRAMAN BERBASIS WEB
Untuk Memenuhi Praktikum Pemograman Berbasis Web



Oleh:

Nama : Siti Ghumaisa
NPM : 4522210131
Kelas : A
Semester : 4 (Genap)

Dosen :

ADI WAHYU PRIBADI ,S.SI.,M.KOM
S1-Teknik Informatika
Fakultas Teknik Universitas Pancasila
2023/2024

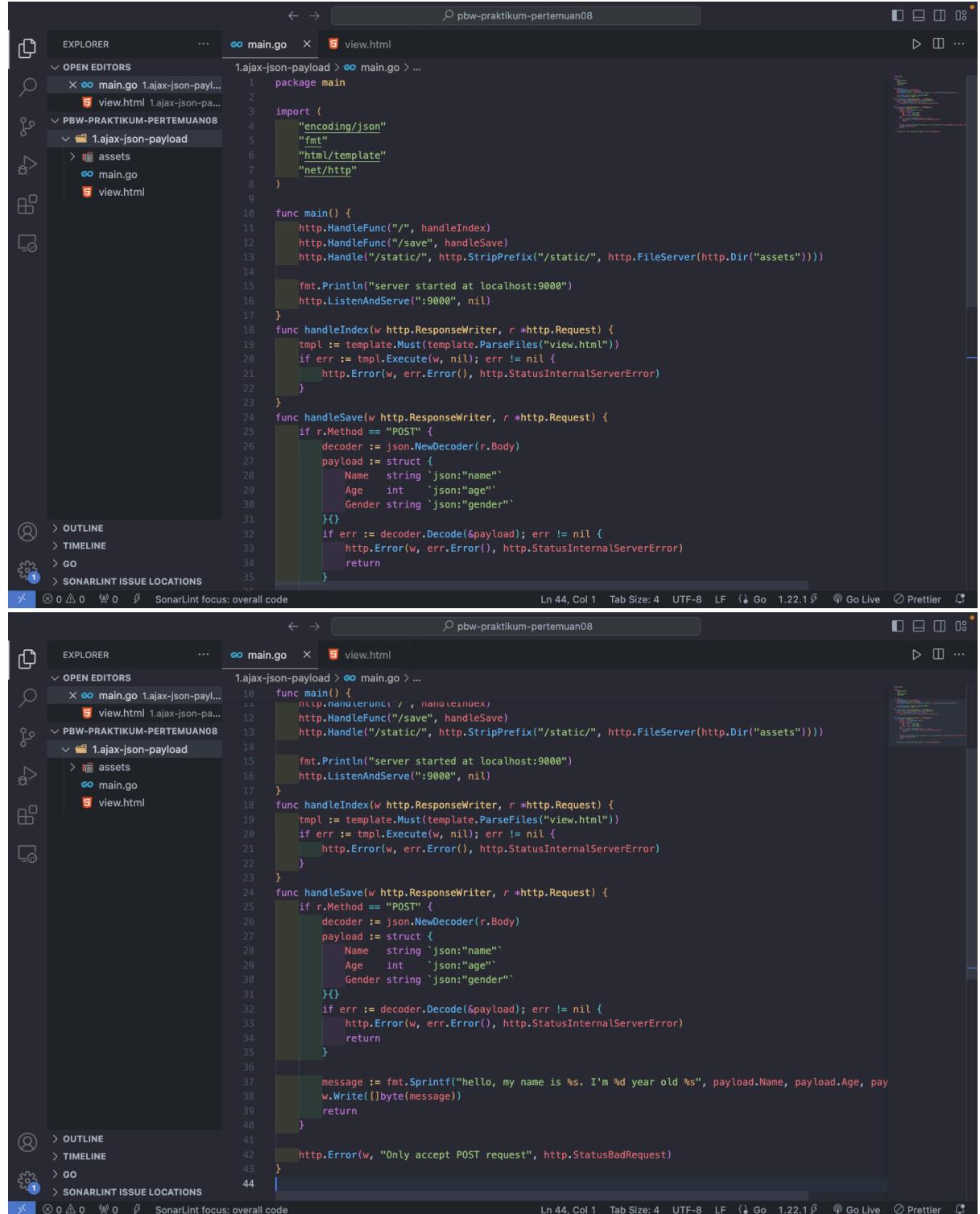
Link git-hub <https://github.com/Ghumaisa/PBW>

MATERI PRAKTIKUM :

1. AJAX JSON Payload

Screenshot Source Code

main.go



The screenshot shows the Visual Studio Code interface with the main.go file open. The code implements a simple web server using the Go standard library. It handles GET requests for the root path, POST requests for saving data, and static files from the assets directory. The code uses the json package for decoding POST payloads and the template package for rendering HTML templates.

```
package main

import (
    "encoding/json"
    "fmt"
    "html/template"
    "net/http"
)

func main() {
    http.HandleFunc("/", handleIndex)
    http.HandleFunc("/save", handleSave)
    http.Handle("/static/", http.StripPrefix("/static/", http.FileServer(http.Dir("assets"))))
}

func handleIndex(w http.ResponseWriter, r *http.Request) {
    tmpl := template.Must(template.ParseFiles("view.html"))
    if err := tmpl.Execute(w, nil); err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
    }
}

func handleSave(w http.ResponseWriter, r *http.Request) {
    if r.Method == "POST" {
        decoder := json.NewDecoder(r.Body)
        payload := struct {
            Name string `json:"name"`
            Age  int    `json:"age"`
            Gender string `json:"gender"`
        }()
        if err := decoder.Decode(&payload); err != nil {
            http.Error(w, err.Error(), http.StatusInternalServerError)
            return
        }

        message := fmt.Sprintf("hello, my name is %s. I'm %d year old %s", payload.Name, payload.Age, pay
        w.Write([]byte(message))
        return
    }

    http.Error(w, "Only accept POST request", http.StatusBadRequest)
}
```

view.html

The screenshot shows the VS Code interface with the file `view.html` open in the center editor. The code implements an AJAX POST request to save user form data. It includes jQuery code to handle the submission event, serialize the form data into JSON, and make an asynchronous POST request to the endpoint `/save`. The response is handled to either update the message area or show an alert.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>JSON Payload</title>
    <script src="static/jquery-3.3.1.min.js"></script>
    <script>
      $(function () {
        $("#user-form").on("submit", function (e) {
          e.preventDefault();

          var $self = $(this);
          var payload = JSON.stringify({
            name: $('[name="name"]').val(),
            age: parseInt($('[name="age"]').val(), 10),
            gender: $('[name="gender"]').val(),
          });

          $.ajax({
            url: $self.attr("action"),
            type: $self.attr("method"),
            data: payload,
            contentType: "application/json",
            success: function (res) {
              $(".message").text(res);
            },
            error: function (xhr) {
              alert("ERROR: " + xhr.responseText);
            },
          });
        });
      });
    </script>
  </head>
  <body>
    <p class="message"></p>
  </body>
</html>
```

The screenshot shows the VS Code interface with the file `view.html` open in the center editor. The code defines a form with three input fields: Name, Age, and Gender. The Name and Age fields are required and have placeholder text. The Gender field is a dropdown menu with options for Male and Female, and a placeholder text "Select one". The form has a method of "post" and an action of "/save".

```
<html lang="en">
  <body>
    <p class="message"></p>
    <form id="user-form" method="post" action="/save">
      <table border="1">
        <tr>
          <td><label for="name">Name :</label></td>
          <td><input required type="text" id="name" name="name" placeholder="Type name here" /></td>
        </tr>
        <tr>
          <td><label for="age">Age :</label></td>
          <td><input required type="number" id="age" name="age" placeholder="Set age" /></td>
        </tr>
        <tr>
          <td><label for="gender">Gender :</label></td>
          <td>
            <select id="gender" name="gender" required style="width: 100%">
              <option value="">Select one</option>
              <option value="male">Male</option>
              <option value="female">Female</option>
            </select>
          </td>
        </tr>
        <tr>
          <td colspan="2" style="text-align: right">
            <button type="submit">Save</button>
          </td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

The screenshot shows the VS Code interface with the file `view.html` open in the editor. The code is an HTML form for user input:

```
<html lang="en">
  <body>
    <form id="user-form" method="post" action="/save">
      <tr>
        <td><label for="name">Name :</label></td>
        <td><input required type="text" id="name" name="name" placeholder="Set name" /></td>
      </tr>
      <tr>
        <td><label for="age">Age :</label></td>
        <td><input required type="number" id="age" name="age" placeholder="Set age" /></td>
      </tr>
      <tr>
        <td><label for="gender">Gender :</label></td>
        <td><select id="gender" name="gender" required style="width: 100%">
          <option value="">Select one</option>
          <option value="male">Male</option>
          <option value="female">Female</option>
        </select></td>
      </tr>
      <tr>
        <td colspan="2" style="text-align: right"><button type="submit">Save</button></td>
      </tr>
    </form>
  </body>
</html>
```

The code editor has syntax highlighting and a vertical ruler. The status bar at the bottom shows: html > head > script Ln 17, Col 1 Spaces: 4 UTF-8 LF HTML Go Live Prettier.

Output

The screenshot shows the VS Code interface with the terminal tab active, displaying the output of the application:

```
pbw-praktikum-pertemuan0
1.ajax-json-payload git:(master) ✘ go run main.go
server started at localhost:9000
```

The terminal also shows a list of recent sessions:

- go ghumaisa
- zsh 1.ajax-json-p...
- go 1.ajax-json-p...

The status bar at the bottom shows: html > head > script Ln 17, Col 1 Spaces: 4 UTF-8 LF HTML Go Live Prettier.

JSON Payload

localhost:9000

Name : Age : Gender :

Select one

Save

Finish update

Memasukkan “Nama”, “Age”, dan “Gender”

JSON Payload

localhost:9000

Name : Age : Gender :

Female

Save

Finish update

Setelah di klik save

A screenshot of a web browser window titled "JSON Payload". The address bar shows "localhost:9000". The page content includes a text area with the message "hello, my name is Siti Ghumaisa. I'm 20 year old female". Below this is a form with fields: "Name : Age : Gender : ". A "Save" button is also present.

A screenshot of a web browser window titled "JSON Payload" with the developer tools Network tab open. The address bar shows "localhost:9000". The page content is identical to the first screenshot. The developer tools interface shows network requests. One request, labeled "save", is selected in the list. The "Payload" tab of the Network panel displays the JSON data sent to the server:

```
▼ Request Payload view source
  ▼ {name: "Siti Ghumaisa", age: 20, gender: "female"}
    age: 20
    gender: "female"
    name: "Siti Ghumaisa"
```

The developer tools also show other files loaded: "localhost", "jquery-3.3.1.min.js", and "favicon.ico". The bottom of the screen shows the Chrome 124 update highlights, including "Scroll-driven animations support" and "New Autofill panel".

2. AJAX JSON Response

Screenshot Source Code

The screenshot shows the VS Code interface with the following details:

- Editor Area:** Displays the Go file `main.go` containing code for handling JSON responses.
- Explorer Bar:** Shows the project structure with files like `1.ajax-json-payload`, `2.ajax-json-response`, and `3.ajax-multi-upload`.
- Bottom Status Bar:** Shows SonarLint focus: overall code, Ln 36, Col 1, Tab Size: 4, UTF-8, LF, 1.22.1, Go Live, Prettier, and a file icon.

```
package main

import (
    "encoding/json"
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/", ActionIndex)
    fmt.Println("server started at localhost:9000")
    http.ListenAndServe(":9000", nil)
}

func ActionIndex(w http.ResponseWriter, r *http.Request) {
    data := []struct {
        Name string
        Age int
    }{
        {"Richard Grayson", 24},
        {"Jason Todd", 23},
        {"Tim Drake", 22},
        {"Damian Wayne", 21},
    }

    jsonInBytes, err := json.Marshal(data)
    if err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }

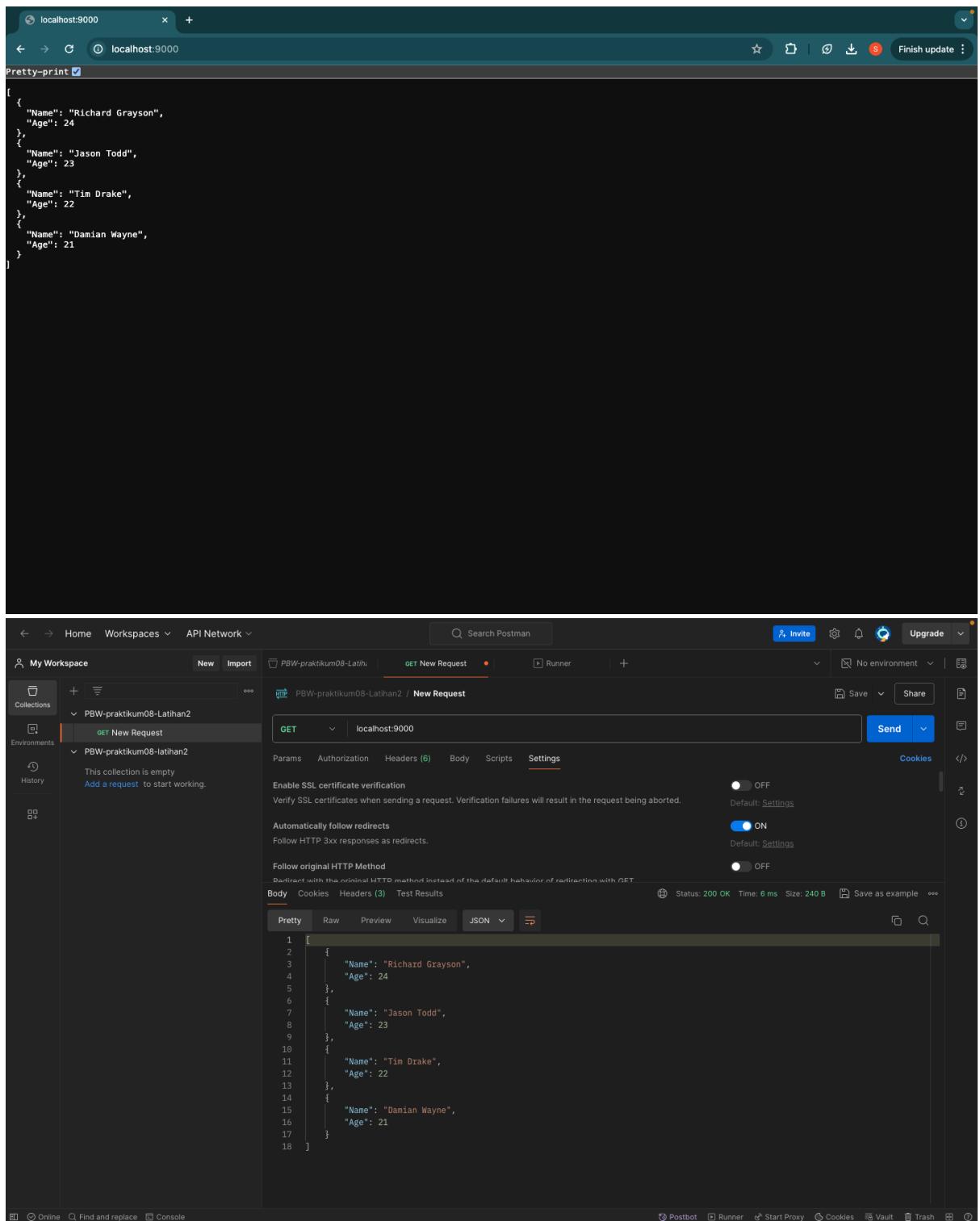
    w.Header().Set("Content-Type", "application/json")
    w.Write(jsonInBytes)
}
```

Output

The screenshot shows the VS Code interface with the following details:

- Editor Area:** Displays the same `main.go` file as the previous screenshot.
- Terminal:** Shows the command `go run main.go` being run, resulting in the output: "server started at localhost:9000".
- Bottom Status Bar:** Shows PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, zsh, go 2.ajax-json-re...

```
2.ajax-json-response git:(master) ✘ go run main.go
server started at localhost:9000
```



The screenshot shows a browser window at localhost:9000 displaying a JSON array of four objects. Each object has a 'Name' and an 'Age'. Below this, the Postman application interface is visible, showing a 'New Request' dialog with a GET method to localhost:9000. The request body contains the same JSON data as the browser. The Postman interface includes sections for Params, Authorization, Headers, Body, Scripts, and Settings, along with SSL certificate verification and redirect options.

```
[{"Name": "Richard Grayson", "Age": 24}, {"Name": "Jason Todd", "Age": 23}, {"Name": "Tim Drake", "Age": 22}, {"Name": "Damian Wayne", "Age": 21}]
```

3. AJAX Multiple File Upload

Screenshot Source Code

Main.go

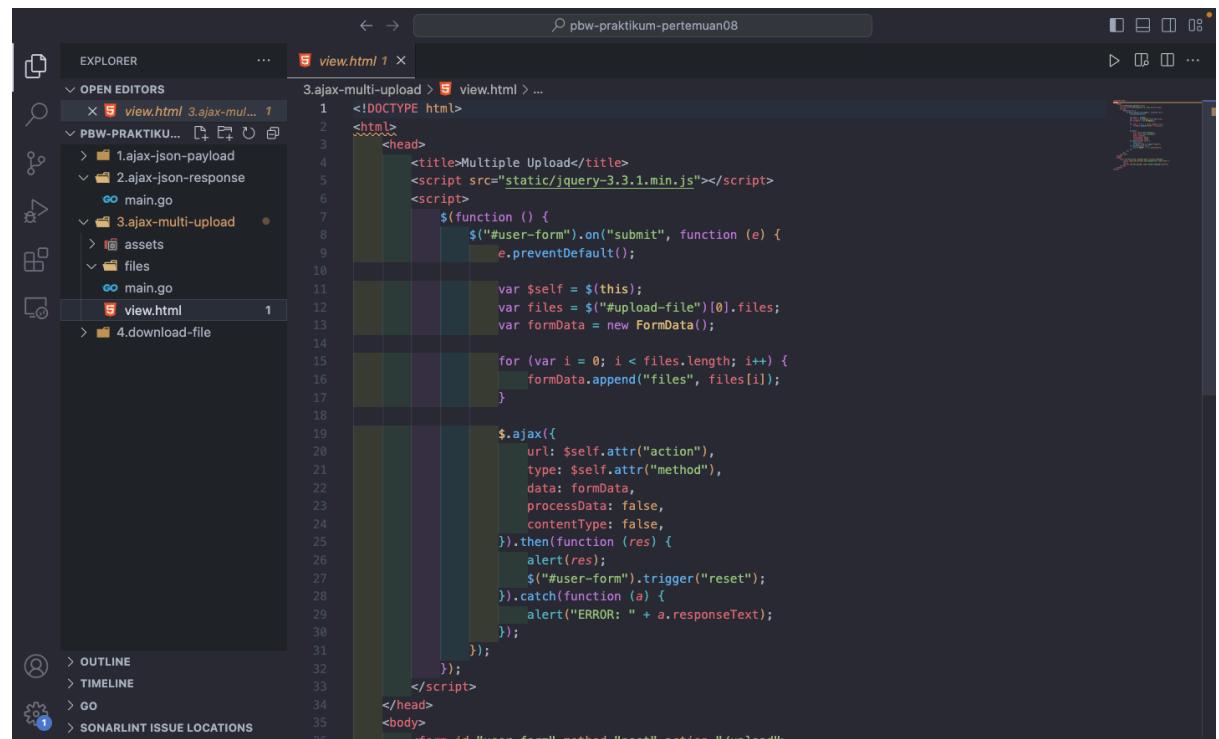
The screenshot shows the VS Code interface with the main.go file open in the editor. The code implements a simple HTTP server using the net/http package. It defines two handlers: handleIndex for static files and handleUpload for handling multipart POST requests. The handleUpload function reads the uploaded files and saves them to a 'files' directory.

```
1 package main
2
3 import "fmt"
4 import "net/http"
5 import "html/template"
6 import "path/filepath"
7 import "io"
8 import "os"
9
10 func main() {
11     http.HandleFunc("/", handleIndex)
12     http.HandleFunc("/upload", handleUpload)
13     http.Handle("/static/", http.StripPrefix("/static/", http.FileServer(http.Dir("assets"))))
14
15     fmt.Println("server started at localhost:9000")
16     http.ListenAndServe(":9000", nil)
17 }
18
19 func handleIndex(w http.ResponseWriter, r *http.Request) {
20     tmpl := template.Must(template.ParseFiles("view.html"))
21     if err := tmpl.Execute(w, nil); err != nil {
22         http.Error(w, err.Error(), http.StatusInternalServerError)
23     }
24 }
25
26 func handleUpload(w http.ResponseWriter, r *http.Request) {
27     if r.Method != "POST" {
28         http.Error(w, "Only accept POST request", http.StatusBadRequest)
29         return
30     }
31
32     basePath, _ := os.Getwd()
33
34     reader, err := r.MultipartReader()
35     if err != nil {
36         http.Error(w, err.Error(), http.StatusInternalServerError)
37         return
38     }
39
40     for {
41         part, err := reader.NextPart()
42         if err == io.EOF {
43             break
44         }
45
46         fileLocation := filepath.Join(basePath, "files", part.FileName())
47         dst, err := os.Create(fileLocation)
48         if dst != nil {
49             defer dst.Close()
50         }
51         if err != nil {
52             http.Error(w, err.Error(), http.StatusInternalServerError)
53             return
54         }
55
56         if _, err := io.Copy(dst, part); err != nil {
57             http.Error(w, err.Error(), http.StatusInternalServerError)
58             return
59         }
60     }
61
62     w.Write([]byte('all files uploaded'))
63 }
```

The screenshot shows the continuation of the main.go file from the previous screenshot. The handleUpload function continues by writing a success message to the response writer.

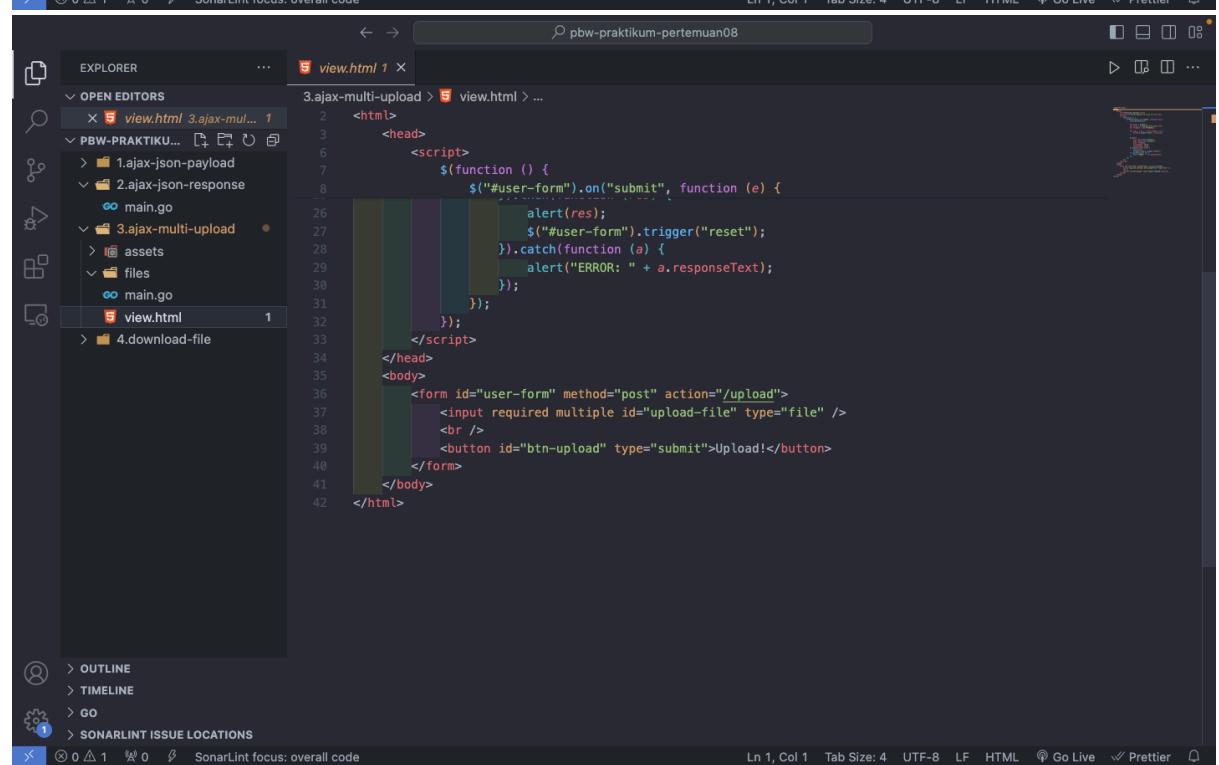
```
26 func handleUpload(w http.ResponseWriter, r *http.Request) {
27     basePath, _ := os.Getwd()
28
29     reader, err := r.MultipartReader()
30     if err != nil {
31         http.Error(w, err.Error(), http.StatusInternalServerError)
32         return
33     }
34
35     for {
36         part, err := reader.NextPart()
37         if err == io.EOF {
38             break
39         }
40
41         fileLocation := filepath.Join(basePath, "files", part.FileName())
42         dst, err := os.Create(fileLocation)
43         if dst != nil {
44             defer dst.Close()
45         }
46         if err != nil {
47             http.Error(w, err.Error(), http.StatusInternalServerError)
48             return
49         }
50
51         if _, err := io.Copy(dst, part); err != nil {
52             http.Error(w, err.Error(), http.StatusInternalServerError)
53             return
54         }
55     }
56
57     w.Write([]byte('all files uploaded'))
58 }
```

View.html



The screenshot shows the VS Code interface with the file `view.html` open in the editor. The code implements a multi-file upload feature using jQuery. It includes a script that listens for form submissions, prevents the default action, and then performs an AJAX POST request to the server. The response is handled with an alert. The code also includes a simple HTML form with a file input and a submit button.

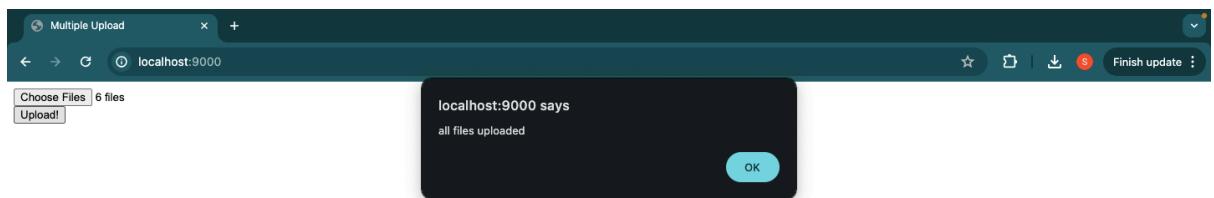
```
<!DOCTYPE html>
<html>
<head>
    <title>Multiple Upload</title>
    <script src="static/jquery-3.3.1.min.js"></script>
    <script>
        $(function () {
            $("#user-form").on("submit", function (e) {
                e.preventDefault();
                var $self = $(this);
                var files = $("#upload-file")[0].files;
                var formData = new FormData();
                for (var i = 0; i < files.length; i++) {
                    formData.append("files", files[i]);
                }
                $.ajax({
                    url: $self.attr("action"),
                    type: $self.attr("method"),
                    data: formData,
                    processData: false,
                    contentType: false,
                }).then(function (res) {
                    alert(res);
                    $("#user-form").trigger("reset");
                }).catch(function (a) {
                    alert("ERROR: " + a.responseText);
                });
            });
        });
    </script>
</head>
<body>
    <form id="user-form" method="post" action="/upload">
        <input required multiple id="upload-file" type="file" />
        <br />
        <button id="btn-upload" type="submit">Upload!</button>
    </form>
</body>
</html>
```



This screenshot shows the same VS Code interface, but the code in the editor has been partially removed or commented out. The removed code includes the entire script block and the entire form structure, leaving only the basic HTML shell.

```
<html>
<head>
    <script>
        $(function () {
            $("#user-form").on("submit", function (e) {
                e.preventDefault();
                var $self = $(this);
                var files = $("#upload-file")[0].files;
                var formData = new FormData();
                for (var i = 0; i < files.length; i++) {
                    formData.append("files", files[i]);
                }
                $.ajax({
                    url: $self.attr("action"),
                    type: $self.attr("method"),
                    data: formData,
                    processData: false,
                    contentType: false,
                }).then(function (res) {
                    alert(res);
                    $("#user-form").trigger("reset");
                }).catch(function (a) {
                    alert("ERROR: " + a.responseText);
                });
            });
        });
    </script>
</head>
<body>
    <form id="user-form" method="post" action="/upload">
        <input required multiple id="upload-file" type="file" />
        <br />
        <button id="btn-upload" type="submit">Upload!</button>
    </form>
</body>
</html>
```

Output



4. Download File

Screenshot Source Code

The image shows two screenshots of a code editor interface, likely Visual Studio Code, displaying a Go application. The application is designed to handle file download requests via HTTP. It includes functions for listing files in a directory and downloading specific files.

Top Screenshot (Line 1 to Line 35):

```
1 package main
2
3 import (
4     "encoding/json"
5     "fmt"
6     "html/template"
7     "io"
8     "net/http"
9     "os"
10    "path/filepath"
11 )
12
13 type M map[string]interface{}
14
15 func main() {
16     http.HandleFunc("/", handleIndex)
17     http.HandleFunc("/list-files", handleListFiles)
18     http.HandleFunc("/download", handleDownload)
19
20     fmt.Println("server started at localhost:9000")
21     http.ListenAndServe(":9000", nil)
22 }
23
24 func handleIndex(w http.ResponseWriter, r *http.Request) {
25     tmpl := template.Must(template.ParseFiles("view.html"))
26     if err := tmpl.Execute(w, nil); err != nil {
27         http.Error(w, err.Error(), http.StatusInternalServerError)
28     }
29 }
30
31 func handleListFiles(w http.ResponseWriter, r *http.Request) {
32     files := []M{}
33     basePath, _ := os.Getwd()
34     filesLocation := filepath.Join(basePath, "files")
35 }
```

Bottom Screenshot (Line 31 to Line 78):

```
31 func handleListFiles(w http.ResponseWriter, r *http.Request) {
32     basePath, err := filepath.Abs(filepath.Clean(r.Referer()))
33     if err != nil {
34         return err
35     }
36
37     if info.IsDir() {
38         return nil
39     }
40
41     files = append(files, M{"filename": info.Name(), "path": path})
42
43     return nil
44 }
45
46 if err != nil {
47     http.Error(w, err.Error(), http.StatusInternalServerError)
48     return
49 }
50
51 res, err := json.Marshal(files)
52 if err != nil {
53     http.Error(w, err.Error(), http.StatusInternalServerError)
54     return
55 }
56
57 w.Header().Set("Content-Type", "application/json")
58 w.Write(res)
59
60
61
62
63
64 func handleDownload(w http.ResponseWriter, r *http.Request) {
65     if err := r.ParseForm(); err != nil {
66         http.Error(w, err.Error(), http.StatusInternalServerError)
67         return
68     }
69
70     path := r.FormValue("path")
```

The screenshot shows the main.go file in a code editor. The code implements two functions: handleListFiles and handleDownload. handleListFiles takes an http.ResponseWriter and http.Request, reads files from memory, and writes them in JSON format. handleDownload takes an http.ResponseWriter and http.Request, reads a file from disk, and returns it as an attachment. Both functions handle errors and set appropriate headers.

```
func handleListFiles(w http.ResponseWriter, r *http.Request) {
    res, err := json.Marshal(files)
    if err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }
    w.Header().Set("Content-Type", "application/json")
    w.Write(res)
}

func handleDownload(w http.ResponseWriter, r *http.Request) {
    if err := r.ParseForm(); err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }
    path := r.FormValue("path")
    f, err := os.Open(path)
    if f != nil {
        defer f.Close()
    }
    if err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }
    contentDisposition := fmt.Sprintf("attachment; filename=%s", f.Name())
    w.Header().Set("Content-Disposition", contentDisposition)

    if _, err := io.Copy(w, f); err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }
}
```

View.html

The screenshot shows the view.html file in a code editor. It contains JavaScript code that handles file download requests. It uses XMLHttpRequest to get a list of files from the server, then creates a list of download links for each file. Each link has a href pointing to /download?path= and an innerText containing the file name.

```
<!DOCTYPE html>
<html>
<head>
    <title>Download file</title>
    <script>
        function Yo() {
            var self = this;
            var $ul = document.getElementById("list-files");

            var renderData = function (res) {
                res.forEach(function (each) {
                    var $li = document.createElement("li");
                    var $a = document.createElement("a");

                    $li.innerText = "download ";
                    $li.appendChild($a);
                    $ul.appendChild($li);

                    $a.href = "/download?path=" + encodeURI(each.path);
                    $a.innerText = each.filename;
                    $a.target = "_blank";
                });
            };

            var getAllListFiles = function () {
                var xhr = new XMLHttpRequest();
                xhr.open("GET", "/List-files");
                xhr.onreadystatechange = function () {
                    if (xhr.readyState == 4 && xhr.status == 200) {
                        var json = JSON.parse(xhr.responseText);
                        renderData(json);
                    }
                };
                xhr.send();
            };
        }
    </script>
</head>
<body>
    <ul id="list-files">
    </ul>
</body>

```

```
4.download-file > view.html 1
<html>
<head>
<script>
function Yo() {
    var getAllListFiles = function () {
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "/list-files");
        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
                var json = JSON.parse(xhr.responseText);
                renderData(json);
            }
        };
        xhr.send();
    };

    self.init = function () {
        getAllListFiles();
    };

    window.onload = function () {
        new Yo().init();
    };
}
</script>
</head>
<body>
<ul id="list-files"></ul>
</body>
</html>
```

Ln 51, Col 1 Spaces: 4 UTF-8 LF HTML ⚡ Go Live ✅ Prettier

Output

Recent Download History

- _Users_ghumaisa_Docu ah_Semester_4_Prak_Pbw_go_pbw-praktikum-pertemuan08_4.download-file_files_Screen Shot 2024-05-17 at 16.04.26.png 109 KB • Done
- _Users_ghumaisa_Documents_Kuli ah_Semester_4_Prak_PBW_go_pbw-praktikum-pertemuan08_4.download-file_files_Screen Shot 2024-05-17 at 16.02.52.png 172 KB • Done
- 4.download-file.zip 879 KB • 23 minutes ago
- Latihan08_4522210133_Muhammad Abdur Harry Malhotra.pdf 2.2 MB • 42 minutes ago

Full Download History

5. HTTP Cookie

Screenshot Source Code

The screenshot shows a code editor interface with two tabs open:

- main.go 5.http-cookie 1**: This tab contains the main function and ActionIndex handler. It imports fmt, net/http, and time packages, along with gubrak from github.com/novalagung/gubrak/v2. The main function starts an HTTP server at port 9000. The ActionIndex handler handles GET requests and sets a cookie named "CookieData".
- view.html 1**: This tab contains an HTML file with a single line of code: `<h1>Hello World</h1>`.

The code editor has a dark theme with syntax highlighting for Go. The bottom status bar shows the current line (Ln 52), column (Col 1), tab size (4), and other settings.

```
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6     "time"
7
8     gubrak "github.com/novalagung/gubrak/v2"
9 )
10
11 type M map[string]interface{}
12
13 var cookieName = "CookieData"
14
15 func main() {
16     http.HandleFunc("/", ActionIndex)
17     http.HandleFunc("/delete", ActionDelete)
18
19     fmt.Println("server started at localhost:9000")
20     http.ListenAndServe(":9000", nil)
21 }
22
23 func ActionIndex(w http.ResponseWriter, r *http.Request) {
24     cookieName := "CookieData"
25
26     c := &http.Cookie{}
27
28     if storedCookie, _ := r.Cookie(cookieName); storedCookie != nil {
29         c = storedCookie
30     }
31
32     if c.Value == "" {
33         c = &http.Cookie{}
34         c.Name = cookieName
35         c.Value = gubrak.RandomString(32)
36     }
37
38     w.Write([]byte(c.Value))
39 }
40
41 func ActionDelete(w http.ResponseWriter, r *http.Request) {
42     c := &http.Cookie{}
43     c.Name = cookieName
44     c.Expires = time.Unix(0, 0)
45     c.MaxAge = -1
46     http.SetCookie(w, c)
47
48     http.Redirect(w, r, "/", http.StatusTemporaryRedirect)
49 }
50
51 }
```

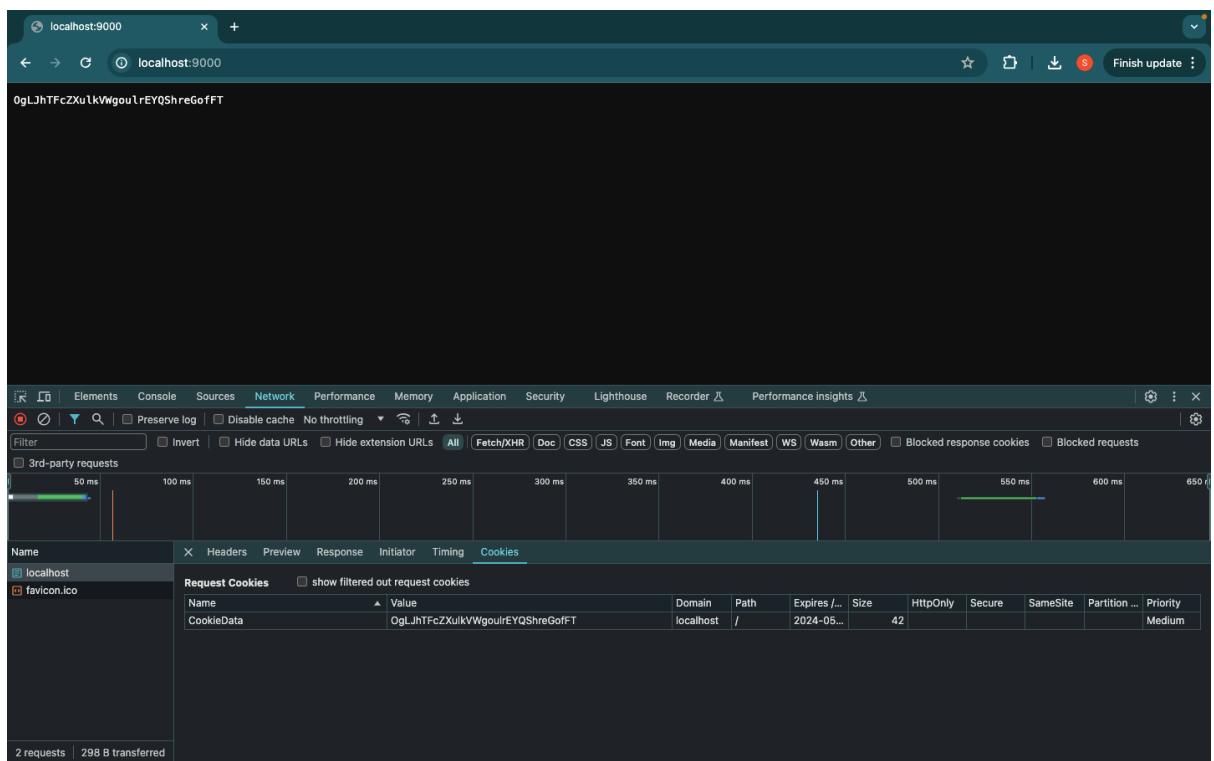
The second part of the screenshot shows the same code editor with the ActionIndex function expanded, highlighting the code that sets the cookie value to a random string generated by the gubrak library.

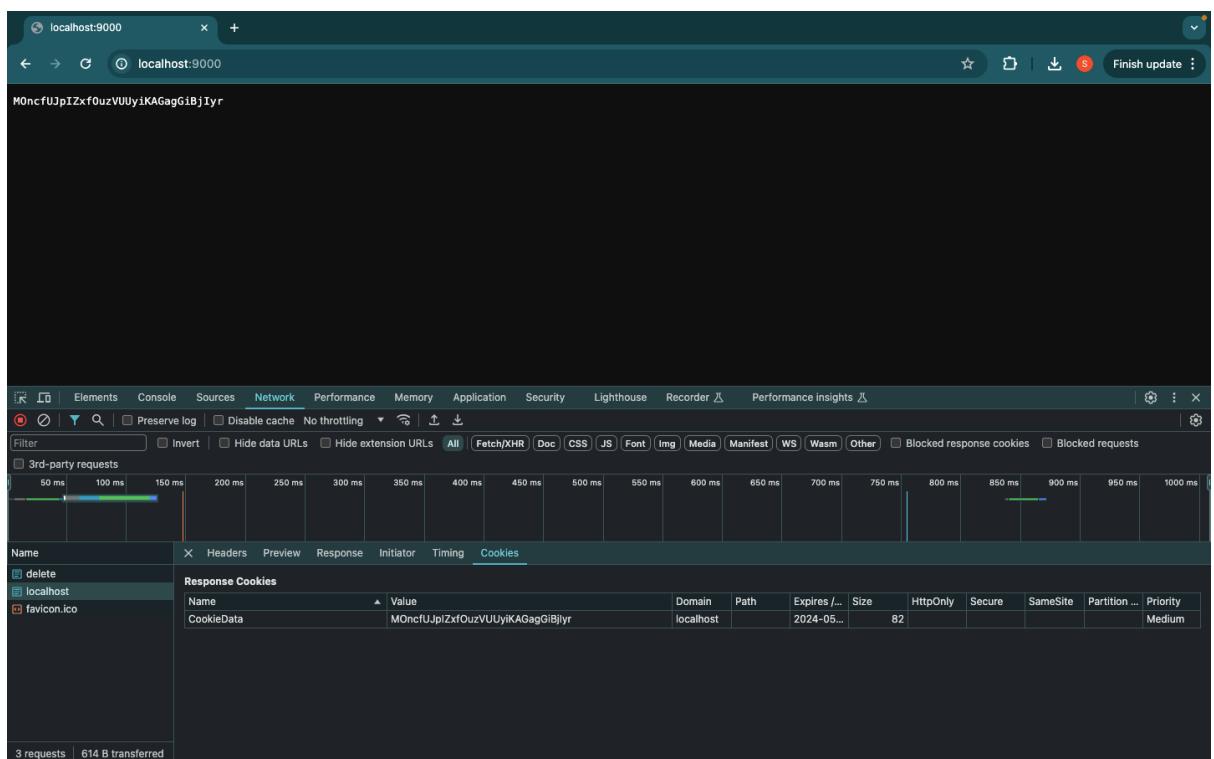
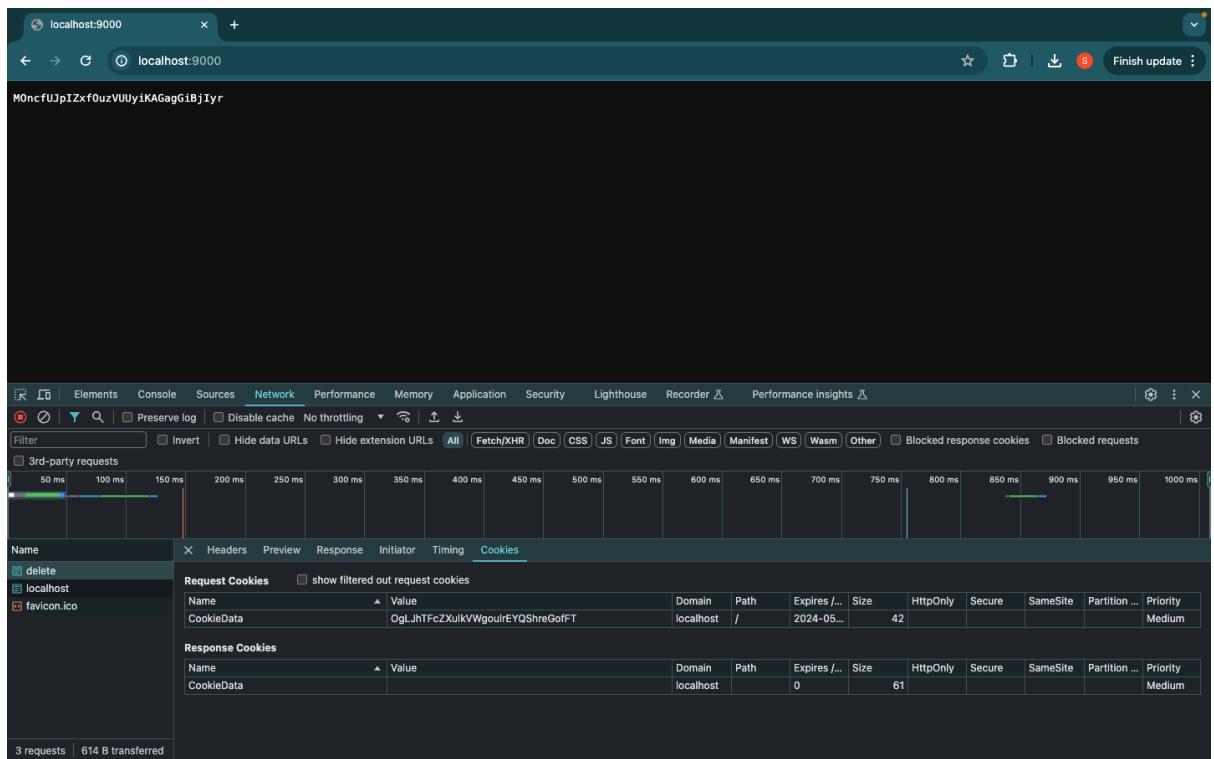
```
func ActionIndex(w http.ResponseWriter, r *http.Request) {
    if c.Value == "" {
        c = &http.Cookie{}
        c.Name = cookieName
        c.Value = gubrak.RandomString(32)
        c.Expires = time.Now().Add(5 * time.Minute)
        http.SetCookie(w, c)
    }
    w.Write([]byte(c.Value))
}
```

Output

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** Shows files like `main.go`, `view.html`, and `5.http-cookie`.
- OPEN EDITORS:** Shows `main.go` and `5.http-cookie`.
- CODE EDITOR:** Displays Go code for cookie management, specifically `ActionIndex` and `ActionDelete` functions.
- TERMINAL:** Shows command-line output related to module dependencies and compilation.
- SIDE BAR:** Includes sections for `OUTLINE`, `TIMELINE`, `GO`, and `SONARLINT ISSUE LOCATIONS`.
- STATUS BAR:** Shows file path (`pbw-praktikum-pertemuan08`), line (Ln 52), column (Col 1), tab size (Tab Size: 4), encoding (UTF-8), and other status information.





6. Pertanyaan

a. Apa itu AJAX?

- AJAX (Asynchronous JavaScript and XML) adalah teknik pengembangan web yang memungkinkan halaman web untuk berkomunikasi dengan server di belakang layar tanpa harus me-reload seluruh halaman. Dengan menggunakan AJAX, data dapat diambil atau dikirim ke server secara asinkron, sehingga pengguna dapat terus

berinteraksi dengan halaman web tanpa adanya gangguan atau penundaan yang disebabkan oleh muatan ulang halaman secara keseluruhan. Contohnya, ketika kita mengisi formulir online dan ada bagian dari formulir yang langsung memvalidasi data yang Anda masukkan (seperti email) tanpa perlu memuat ulang halaman, itu biasanya menggunakan AJAX.

b. Apa itu JSON? Apa kegunaannya?

- **JSON** (JavaScript Object Notation) : format pertukaran data yang ringan dan mudah dibaca serta ditulis oleh manusia, dan mudah diurai serta dihasilkan oleh mesin. JSON digunakan untuk menyimpan dan mentransfer data, terutama antara server dan aplikasi web, sebagai alternatif yang lebih sederhana dibandingkan dengan XML. Kegunaannya,
 1. **Pertukaran Data:** JSON sering digunakan dalam komunikasi antara klien dan server. Misalnya, saat aplikasi web mengambil data dari server menggunakan AJAX, data sering dikirim dalam format JSON.
 2. **Penyimpanan Data:** JSON juga dapat digunakan untuk menyimpan data dalam file atau database. Banyak database NoSQL, seperti MongoDB, menggunakan JSON atau format serupa untuk menyimpan data.
 3. **Konfigurasi:** Beberapa aplikasi menggunakan file konfigurasi dalam format JSON karena mudah dibaca dan dimodifikasi.

c. Apa itu ParseMultipartForm dan MultipartReader ? Jelaskan perbedaannya!

- **ParseMultipartForm** : fungsi atau metode yang digunakan untuk mem-parsing (menganalisis) data form multipart yang dikirimkan oleh klien (biasanya melalui form HTML dengan metode POST yang berisi file dan data teks). Fungsi ini akan mengekstrak semua bagian dari form (baik file maupun field teks) dan menyimpannya dalam struktur data yang mudah diakses oleh aplikasi. Misalnya, di bahasa pemrograman seperti Go, ParseMultipartForm adalah metode dari objek Request yang mengurai data form multipart dan menempatkannya dalam memori atau di file sementara.
- **MultipartReader** : objek atau kelas yang menyediakan cara membaca data form multipart secara bertahap (streaming) daripada menguraikannya sekaligus. Ini sangat berguna ketika bekerja dengan file yang sangat besar atau ketika ingin memproses bagian form satu per satu untuk menghemat memori. MultipartReader memungkinkan aplikasi untuk membaca bagian dari form satu per satu, yang bisa sangat efisien dalam hal penggunaan memori dan performa.

d. Apa itu JavaScript? Jelaskan kegunaannya dalam frontend !

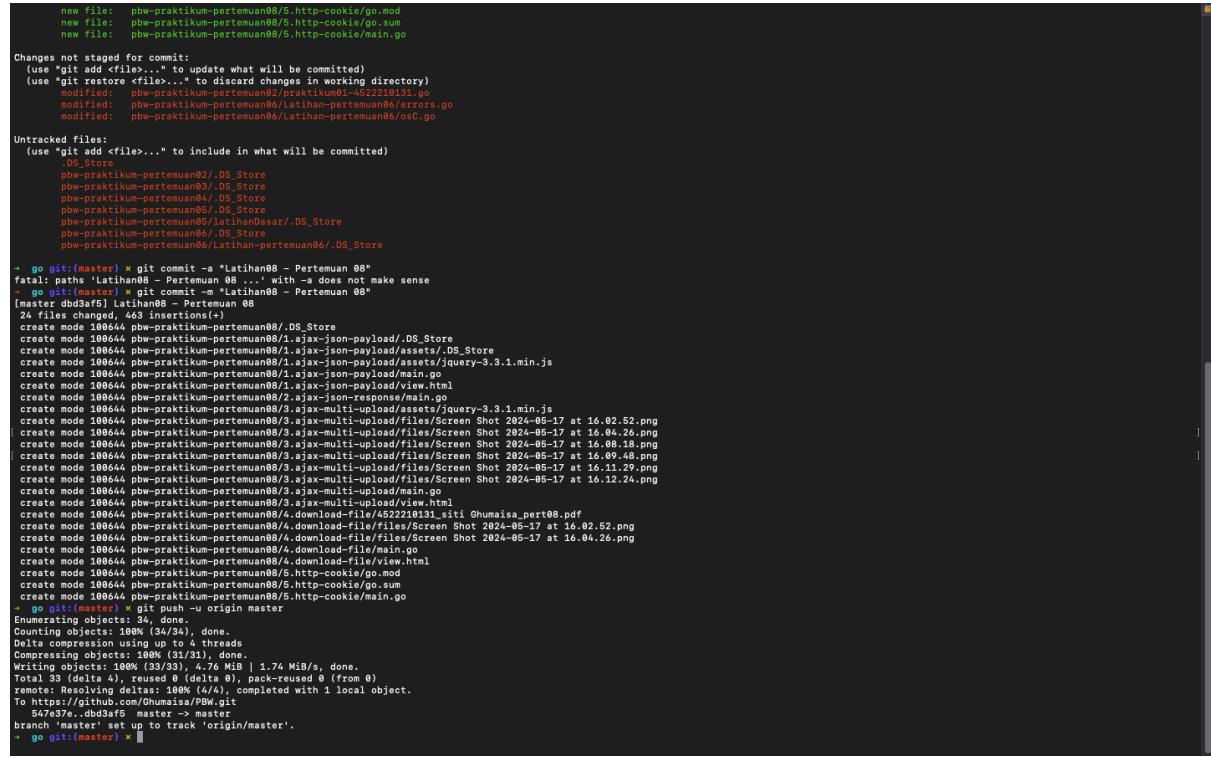
- JavaScript adalah bahasa pemrograman yang digunakan untuk membuat halaman web interaktif dan dinamis. Dalam frontend, JavaScript memungkinkan manipulasi konten HTML dan CSS secara langsung, membuat animasi, validasi form, pembuatan menu dropdown, dan galeri gambar interaktif. Selain itu, JavaScript dapat mengambil data dari server tanpa perlu memuat ulang halaman menggunakan AJAX, serta mendukung pengembangan aplikasi web kompleks dengan framework

seperti React, Angular, dan Vue.js. JavaScript membuat halaman web lebih responsif dan user-friendly.

e. Jelaskan apa itu EventHandler!

- EventHandler adalah fungsi atau kelas dalam pemrograman web yang bertanggung jawab untuk menangani peristiwa-peristiwa yang terjadi di halaman web, seperti klik tombol, pengisian formulir, pergerakan mouse, atau menekan tombol keyboard. Dengan menambahkan EventHandler ke elemen HTML tertentu, pengembang dapat menentukan tindakan apa yang harus dilakukan ketika suatu peristiwa terjadi, memungkinkan pengalaman pengguna menjadi lebih interaktif dan responsif.

Screenshot push Git-hub



```
new file: pbw-praktikum-pertemuan08/5.http-cookie/go.mod
new file: pbw-praktikum-pertemuan08/5.http-cookie/go.sum
new file: pbw-praktikum-pertemuan08/5.http-cookie/main.go

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
           modified: pbw-praktikum-pertemuan08/praktikum01-4522210131.go
           modified: pbw-praktikum-pertemuan08/Latihan-pertemuan08/errors.go
           modified: pbw-praktikum-pertemuan08/Latihan-pertemuan08/osC.go

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .DS_Store
pbw-praktikum-pertemuan08/.DS_Store
pbw-praktikum-pertemuan08//.DS_Store
pbw-praktikum-pertemuan08//.DS_Store
pbw-praktikum-pertemuan08//.DS_Store
pbw-praktikum-pertemuan08/LatihanDasar/.DS_Store
pbw-praktikum-pertemuan08/.DS_Store
pbw-praktikum-pertemuan08/Latihan-pertemuan08/.DS_Store

+ on nitz:[master] ✘ git commit -m "Latihan08 - Pertemuan 08"
fatal: path 'Latihan08 - Pertemuan 08 ...' with -a does not make sense
+ go git:(master) ✘ git commit -m "Latihan08 - Pertemuan 08"
[master ddd3af5] Latihan08 - Pertemuan 08
 24 files changed, 463 insertions(+)
create mode 100644 pbw-praktikum-pertemuan08/1.ajax-json-payload/.DS_Store
create mode 100644 pbw-praktikum-pertemuan08/1.ajax-json-payload/assets/.DS_Store
create mode 100644 pbw-praktikum-pertemuan08/1.ajax-json-payload/assets/jquery-3.3.1.min.js
create mode 100644 pbw-praktikum-pertemuan08/1.ajax-json-payload/main.go
create mode 100644 pbw-praktikum-pertemuan08/1.ajax-json-payload/view.html
create mode 100644 pbw-praktikum-pertemuan08/2.ajax-multi-upload/.DS_Store
create mode 100644 pbw-praktikum-pertemuan08/3.ajax-multi-upload/assets/jquery-3.3.1.min.js
create mode 100644 pbw-praktikum-pertemuan08/3.ajax-multi-upload/files/Screen Shot 2024-05-17 at 16.02.52.png
create mode 100644 pbw-praktikum-pertemuan08/3.ajax-multi-upload/files/Screen Shot 2024-05-17 at 16.04.26.png
create mode 100644 pbw-praktikum-pertemuan08/3.ajax-multi-upload/files/Screen Shot 2024-05-17 at 16.08.18.png
create mode 100644 pbw-praktikum-pertemuan08/3.ajax-multi-upload/files/Screen Shot 2024-05-17 at 16.09.48.png
create mode 100644 pbw-praktikum-pertemuan08/3.ajax-multi-upload/files/Screen Shot 2024-05-17 at 16.11.29.png
create mode 100644 pbw-praktikum-pertemuan08/3.ajax-multi-upload/files/Screen Shot 2024-05-17 at 16.12.24.png
create mode 100644 pbw-praktikum-pertemuan08/3.ajax-multi-upload/main.go
create mode 100644 pbw-praktikum-pertemuan08/3.ajax-multi-upload/view.html
create mode 100644 pbw-praktikum-pertemuan08/4.download-file/.DS_Store
create mode 100644 pbw-praktikum-pertemuan08/4.download-file/4522210131_siti_Ghumaisa_pert08.pdf
create mode 100644 pbw-praktikum-pertemuan08/4.download-file/files/Screen Shot 2024-05-17 at 16.02.52.png
create mode 100644 pbw-praktikum-pertemuan08/4.download-file/praktikum01-4522210131.go
create mode 100644 pbw-praktikum-pertemuan08/4.download-file/view.html
create mode 100644 pbw-praktikum-pertemuan08/5.http-cookie/.DS_Store
create mode 100644 pbw-praktikum-pertemuan08/5.http-cookie/go.mod
create mode 100644 pbw-praktikum-pertemuan08/5.http-cookie/go.sum
create mode 100644 pbw-praktikum-pertemuan08/5.http-cookie/main.go
+ git push -u origin master
Enumerating objects: 34, done.
Counting objects: 100% (34/34), done.
Delta compression using up to 4 threads
Compressing objects: 100% (31/31), done.
Writing objects: 100% (33/33) 14.76 MiB | 1.74 MiB/s, done.
Total 33 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To https://github.com/ohumaisa/PBW.git
  547e37e..dd3af5  master -> master
branch 'master' set up to track 'origin/master'.
+ go git:(master) ✘
```