



Modul Praktikum

Pemrograman Web

S1 Teknologi Informasi

Published by school of computing

Our official instagram



@informaticslab_telu

LEMBAR PENGESAHAN

Saya yang bertanda tangan di bawah ini:

Nama : MONTERICO ADRIAN, S.T., M.T.

NIK : 20870024

Koordinator Mata Kuliah : Pemrograman Web

Prodi : S1 Teknologi Informasi

Menerangkan dengan sesungguhnya bahwa modul ini digunakan untuk pelaksanaan praktikum di Semester Ganjil Tahun Ajaran 2025/2026 di Laboratorium Informatika Fakultas Informatika Universitas Telkom.

Bandung, 30 Agustus 2025

Mengesahkan,

Koordinator MK Pemrograman Web



MONTERICO ADRIAN, S.T., M.T.

Mengetahui,

Kepala Program Studi S1 Teknologi Informasi



Dr. AJI GAUTAMA PUTRADA, S.T., M.T.

PERATURAN PRAKTIKUM LABORATORIUM INFORMATIKA 2025/2026

1. Praktikum diampu oleh dosen kelas dan dibantu oleh asisten laboratorium dan asisten praktikum.
2. Praktikum dilaksanakan di Gedung TULT (Telkom University Landmark Tower) lantai 6 dan 7 sesuai jadwal yang ditentukan.
3. Praktikan wajib membawa modul praktikum, kartu praktikum, dan alat tulis.
4. Praktikan wajib mengecek kehadiran di igracias dan sheet yang dibagikan asisten.
5. Durasi kegiatan praktikum S-1 = 2 jam (100 menit).
6. Jumlah pertemuan praktikum:
 - 16 kali pertemuan
7. Praktikan wajib hadir minimal 75% dari seluruh pertemuan praktikum di lab.
8. Praktikan yang datang terlambat :
 - ≤ 5 menit : diperbolehkan mengikuti praktikum tanpa tambahan praktikum
 - ≥ 30 menit : tidak diperbolehkan mengikuti praktikum
9. Saat praktikum berlangsung, asisten praktikum dan praktikan:
 - Wajib menggunakan seragam sesuai aturan institusi.
 - Wajib mematikan/ mengkondisikan semua alat komunikasi.
 - Dilarang membuka aplikasi yang tidak berhubungan dengan praktikum yang berlangsung.
 - Dilarang mengubah pengaturan *software* maupun *hardware* komputer tanpa ijin.
 - Dilarang membawa makanan maupun minuman di ruang praktikum.
 - Dilarang memberikan jawaban ke praktikan lain.
 - Dilarang menyebarkan soal praktikum.
 - Dilarang membuang sampah di ruangan praktikum.
 - Wajib meletakkan alas kaki dengan rapi pada tempat yang telah disediakan.
10. Setiap praktikan dapat mengikuti praktikum susulan maksimal dua modul untuk satu mata kuliah praktikum.
 - Praktikan yang dapat mengikuti praktikum susulan hanyalah praktikan yang memenuhi syarat sesuai ketentuan institusi, yaitu: sakit (dibuktikan dengan surat keterangan medis), tugas dari institusi (dibuktikan dengan surat dinas atau dispensasi dari institusi), atau mendapat musibah atau keduakaan (menunjukkan surat keterangan dari orangtua/wali mahasiswa.)
 - Persyaratan untuk praktikum susulan diserahkan sesegera mungkin kepada asisten laboratorium untuk keperluan administrasi.
 - Praktikan yang diijinkan menjadi peserta praktikum susulan ditetapkan oleh Lab Informatika dan tidak dapat diganggu gugat.
11. Ketidakhadiran pada kelas praktikum:
 - **Nilai Modul = 0**
12. Meminta, mendapatkan, dan menyebarluaskan soal dan atau kunci jawaban praktikum:
 - **Penyebar soal dan kunci jawaban: Pengajuan sanksi kepada Komisi Disiplin Fakultas**
 - **Penerima soal dan kunci jawaban: Nilai '0' pada (seluruh assessment) praktikum**
13. Lupa menghapus file praktikum:
 - **Pengurangan nilai modul 20%**
14. Memicu kegaduhan, sehingga membuat situasi tidak kondusif (jalan-jalan, mengganggu teman, mengobrol, dll), asisten praktikum diwajibkan menegur sebanyak 3x
 - **Pengurangan nilai modul 50%**

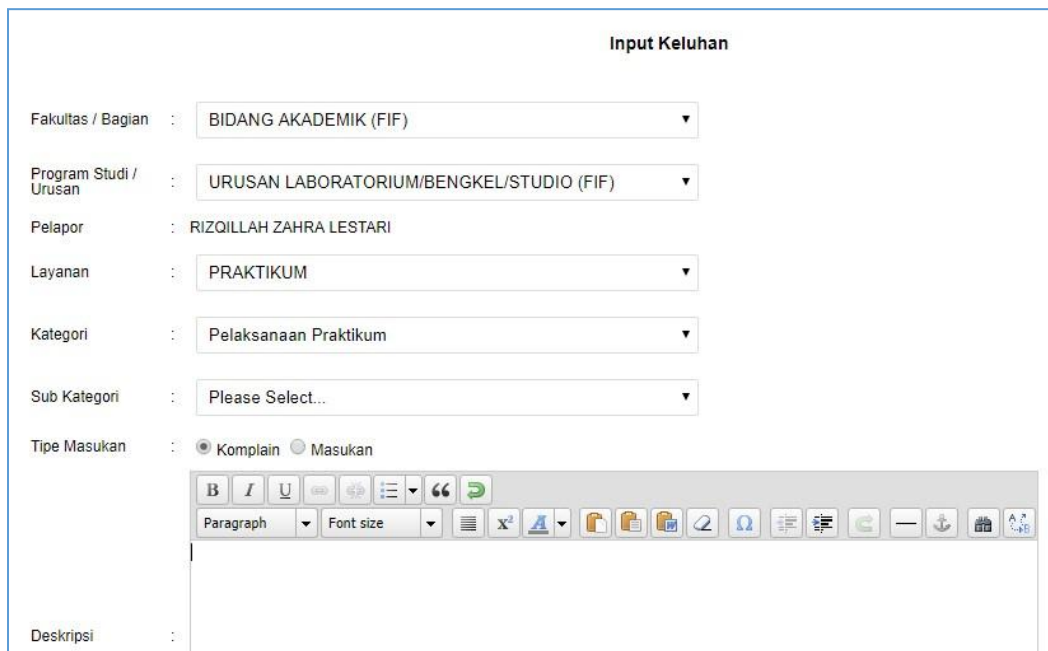
15. Menyalahgunakan fitur lms
 - **Siap menerima sanksi lebih lanjut dari IFLAB**

Tata cara Komplain Praktikum IFLab Melalui IGracias

1. Login IGracias
2. Pilih Menu **Masukan dan Komplain**, pilih **Input Tiket**



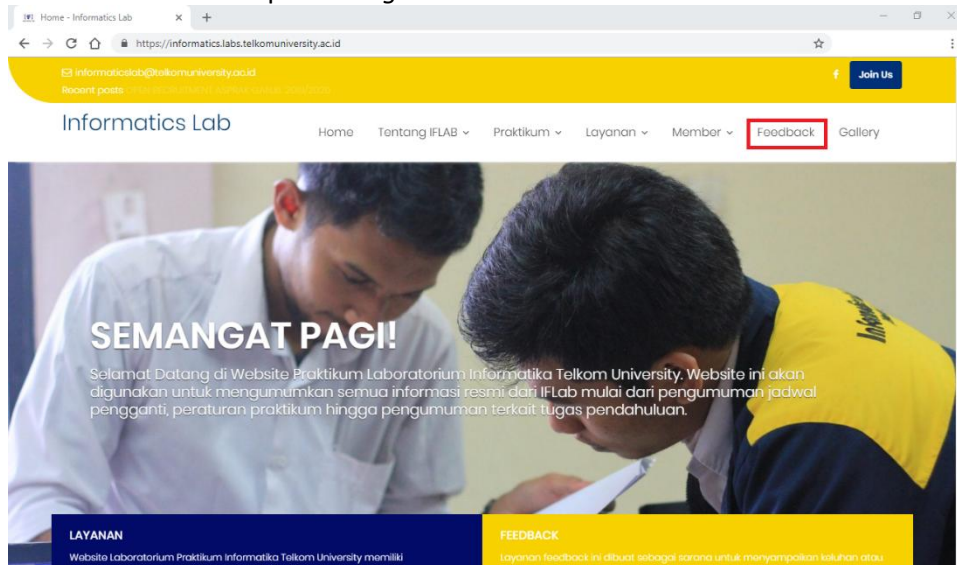
3. Pilih Fakultas/Bagian: **Bidang Akademik (FIF)**
4. Pilih Program Studi/Urusan: **Urusan Laboratorium/Bengkel/Studio (FIF)**
5. Pilih Layanan: **Praktikum**
6. Pilih Kategori: **Pelaksanaan Praktikum**, lalu pilih **Sub Kategori**.
7. Isi **Deskripsi** sesuai komplain yang ingin disampaikan.

A screenshot of a web form titled 'Input Keluhan'. The form contains several dropdown menus and a text input field. The fields are: 'Fakultas / Bagian' (set to 'BIDANG AKADEMIK (FIF)'), 'Program Studi / Urusan' (set to 'URUSAN LABORATORIUM/BENGGEL/STUDIO (FIF)'), 'Pelapor' (set to 'RIZQILLAH ZAHRA LESTARI'), 'Layanan' (set to 'PRAKTIKUM'), 'Kategori' (set to 'Pelaksanaan Praktikum'), 'Sub Kategori' (set to 'Please Select...'), and 'Tipe Masukan' (with radio buttons for 'Komplain' and 'Masukan', where 'Komplain' is selected). Below these fields is a rich text editor with a toolbar containing icons for bold, italic, underline, bulleted list, numbered list, link, unlink, and other formatting options. The text area for the description is empty.

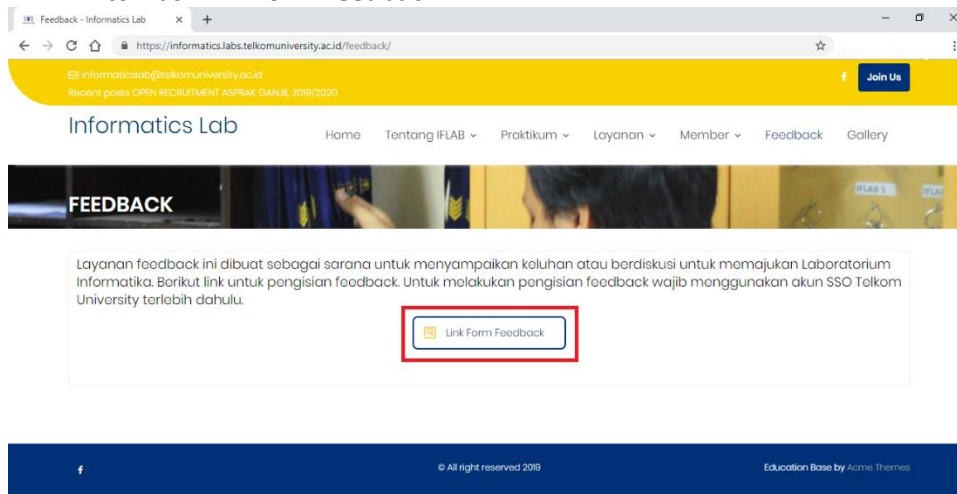
Lampirkan file jika perlu. Lalu klik Kirim.

Tata Cara Komplain Praktikum IFLAB Melalui Website

1. Buka website <https://informatics.labs.telkomuniversity.ac.id/> melalui browser.
2. Pilih menu **Feedback** pada *navigation bar* website.



3. Pilih tombol **Link Form Feedback**.



4. Lakukan *login* menggunakan akun **SSO Telkom University** untuk mengakses *form feedback*.
5. Isi *form* sesuai dengan *feedback* yang ingin diberikan.

DAFTAR ISI

LEMBAR PENGESAHAN	i
PERATURAN PRAKTIKUM LABORATORIUM INFORMATIKA 2025/2026	ii
Tata cara Komplain Praktikum IFLab Melalui IGracias	iii
Tata Cara Komplain Praktikum IFLAB Melalui <i>Website</i>	iv
DAFTAR ISI	v
MODUL 1. ATURAN PRAKTIKUM	viii
1.1. Aturan Praktikum.....	viii
MODUL 2. GIT	10
2.1. Pengenalan Git	10
2.2. Instalasi Git	10
2.3. Penggunaan Git	12
2.3.1. Membuat repositori baru	12
2.3.2. Menambahkan isi repositori.....	13
2.3.3. Membuat repositori online.....	14
2.3.4. Menyimpan hasil pekerjaan di repositori online	15
2.3.5. Clone repositori milik orang lain.....	15
MODUL 3. HTML	17
3.1 Pengenalan HTML.....	17
3.1.1. Tag HTML	17
3.1.2. Elemen HTML.....	17
3.1.3. Atribut HTML	18
3.2. Dasar Sintaks HTML	18
3.3. Heading.....	18
3.4. Hyperlink.....	19
3.5. Tabel	19
3.6. Image	21
3.7. Audio / Video Elemen.....	21
3.8. Form.....	22
3.8.1. Elemen Form.....	22
3.8.2. Atribut Elemen Form	23
MODUL 4. CSS – Cascading Style Sheet	26
4.1. Pengenalan CSS	26
4.1.1. Cara Menyisipkan CSS.....	26
4.1.2. Selector	27
4.2. Font Properties	27
4.3. List Properties.....	28
4.4. Alignment of Text	30
4.5. Colors.....	31
4.6. Span & Div	32
MODUL 5. BOOTSTRAP.....	33

5.1.	Pengenalan Bootstrap.....	33
5.1.1.	Pemasangan Bootstrap.....	33
5.2.	Bootstrap Container.....	34
5.3.	Bootstrap Grid.....	34
5.4.	Text Style.....	35
5.5.	Bootstrap Table, Image & Button	36
5.6.	Bootstrap Form	39
MODUL 6.	JAVASCRIPT & JQUERY	41
6.1.	Pengenalan Javascript	41
6.1.1.	Sejarah Singkat Javascript.....	41
6.1.2.	Prinsip Dasar Javascript	41
6.2	Sintaks Umum pada Javascript	41
6.2.1.	Tipe data dasar	41
6.2.2.	Variabel	42
6.2.3.	Array	42
6.2.4.	Pengendalian Struktur	43
6.3.	Object Orientation pada Javascript.....	43
6.3.1	Pembuatan Object pada Javascript	44
6.3.2.	Akses Nilai Property.....	44
6.3.3.	Prototype pada Javascript	45
6.4.	Function pada Javascript	45
6.4.1	Pembuatan Fungsi pada Javascript.....	45
6.4.2.	Pemanggilan Fungsi	46
6.5.	Pengenalan jQuery	47
6.6.	Kegunaan Lanjutan jQuery	48
6.6.1.	Efek hide/show	48
6.6.2.	Efek animasi.....	49
MODUL 7.	Coding on the Spot 1 (COTS 1)	51
7.1.	Program Learning Outcomes.....	51
7.2.	Course Learning Outcomes.....	51
7.3.	Komponen Penilaian.....	51
MODUL 8.	Responsi Dan Evaluasi Tugas Besar 1.....	52
8.1.	Program Learning Outcomes	52
8.2.	Course Learning Outcomes.....	52
8.3.	Komponen Penilaian.....	52
MODUL 9.	PHP.....	53
9.1.	Web Server dan Server Side Scripting	53
9.2.	Instalasi Apache, PHP dan MySQL dengan XAMPP	54
9.3.	Pengenalan PHP.....	57
9.4.	Variabel.....	58
9.5.	Konstanta.....	59
9.6.	Operator dalam PHP	59
9.7.	Struktur Kondisi	60

9.8. Perulangan (Looping).....	61
9.9. Function.....	62
9.10. Array.....	64
MODUL 10. AJAX.....	65
10.1. Apa Itu AJAX	65
10.2. Cara Kerja AJAX.....	65
10.3. Event Handling.....	65
MODUL 11. Laravel : Introduction.....	69
11.1 Framework & MVC	69
11.2 Pengenalan Laravel.....	70
11.3 Cara Kerja Laravel	71
11.3.1 Routing.....	72
11.3.2 View	72
11.3.3 Controller	73
MODUL 12. Laravel : Database 1.....	77
12.1 CRUD	77
12.1.1 Konfigurasi dan Skema	77
12.1.2 Model.....	79
12.1.3 Controller	79
12.1.4 View	81
12.1.5 Tampilan Halaman	83
12.2 Templating Halaman.....	84
12.3 Form Validation	85
MODUL 13. Laravel : Database 2.....	88
13.1 Session	88
13.2 Middleware.....	89
13.3 Model Relasi	90
MODUL 14. Coding on the Spot 2 (COTS 2).....	93
14.1. Program Learning Outcomes	93
14.2. Course Learning Outcomes.....	93
14.3. Komponen Penilaian.....	93
MODUL 15. Responsi Dan Evaluasi Tugas Besar 2.....	94
DAFTAR PUSTAKA	95

MODUL 1. ATURAN PRAKTIKUM

Tujuan Praktikum
1. Praktikan mampu memahami aturan praktikum

1.1. Aturan Praktikum

1. Praktikum diampu oleh dosen kelas dan dibantu oleh asisten laboratorium dan asisten praktikum.
2. Praktikum dilaksanakan pada MS.Teams dan GoogleMeet.
3. Praktikan wajib membawa modul praktikum dan alattulis.
4. Praktikan diabsen oleh asisten praktikum.
5. Durasi kegiatan praktikum S-1 = 1 SKS atau setara dengan 100 menit.
 - a. Terdapat Tugas Pendahuluan
 - b. 75 menit untuk penyampaian materi
 - c. 25 menit untuk pengerjaan jurnal dan tes akhir
6. Jumlah pertemuan praktikum:
 - 12 kali praktikum secara online (praktikum rutin)
 - 3 kali di luar lab (terkait Tugas Besar dan UAS)
 - 2 kali berupa presentasi Tugas Besar atau pelaksanaan UAS
7. Praktikan wajib hadir minimal 75% dari seluruh pertemuan praktikum di lab. Jika total kehadiran kurang dari 75% maka nilai UAS/ Tugas Besar = 0.
8. Praktikan yang datang terlambat :
 - <= 30 menit : diperbolehkan mengikuti praktikum tanpa tambahan waktu Tes Awal
 - > 30 menit : tidak diperbolehkan mengikuti praktikum
9. Saat praktikum berlangsung, asisten praktikum dan praktikan:
 - 9.1 Luring :
 - Wajib menggunakan seragam sesuai aturan institusi.
 - Wajib mematikan/ mengkondisikan semua alat komunikasi.
 - Dilarang membuka aplikasi yang tidak berhubungan dengan praktikum yang berlangsung.
 - Dilarang mengubah pengaturan software maupun hardware komputer tanpa ijin.
 - Dilarang membawa makanan maupunminumandiruang praktikum.
 - Dilarang memberikan jawaban ke praktikan lain.
 - Dilarang menyebarkan soal praktikum.
 - Dilarang membuang sampah di ruangan praktikum.
 - Wajib meletakkan alas kaki dengan rapi pada tempat yang telah disediakan.
 - 9.2 Daring :
 - Wajib menggunakan seragam sesuai aturan institusi.
 - Wajib menyalakan camera saat praktikum berlangsung.
 - Dilarang memberikan jawaban ke praktikan lain.
 - Dilarang menyebarkan soal praktikum.
10. Setiap praktikan dapat mengikuti praktikum susulan maksimal dua modul untuk satu mata kuliah praktikum, dengan ketentuan:
 - Praktikan yang dapat mengikuti praktikum susulan hanyalah praktikan yang memenuhi syarat sesuai ketentuan institusi, yaitu: sakit (dibuktikan dengan surat keterangan medis), tugas dari institusi (dibuktikan dengan surat dinas atau dispensasi dari institusi), atau mendapat musibah atau keduakaan (menunjukkan surat keterangan dari orangtua/walimahasiswa.)

- Persyaratan untuk praktikum susulan diserahkan sesegera mungkin kepada asisten laboratorium untuk keperluan administrasi.
- Praktikan yang diijinkan menjadi peserta praktikum susulan ditetapkan oleh Asman Lab dan Bengkel Informatika dan tidak dapat diganggu gugat. Pelanggaran terhadap peraturan praktikum akan ditindak secara tegas secara berjenjang di lingkup Kelas, Laboratorium, Fakultas, hingga Universitas.

MODUL 2. GIT

Tujuan Praktikum

1. Mahasiswa mampu memahami penggunaan Git.

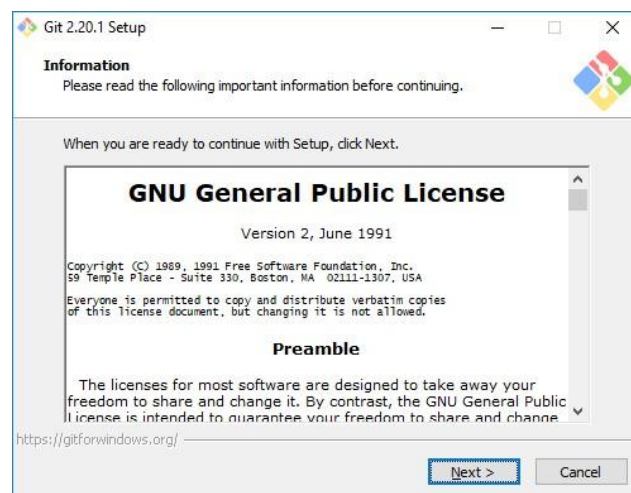
2.1. Pengenalan Git

Git adalah salah satu sistem pengontrol versi (*Version Control System*) pada proyek perangkat lunak yang diciptakan oleh Linus Torvalds. Pengontrol versi bertugas mencatat setiap perubahan pada *file* proyek yang dikerjakan oleh banyak orang maupun sendiri. Git dikenal juga dengan *distributed revision control* (VCS terdistribusi), artinya penyimpanan *database* Git tidak hanya berada dalam satu tempat saja.

2.2. Instalasi Git

Untuk melakukan instalasi git pada computer Anda, lakukan langkah berikut ini:

1. Buka link berikut ini untuk mengunduh Git. <https://git-scm.com/download/win>
2. Klik dua kali pada file yang sudah diunduh.
3. Maka akan muncul informasi lisensi Git, klik *next* untuk melanjutkan.



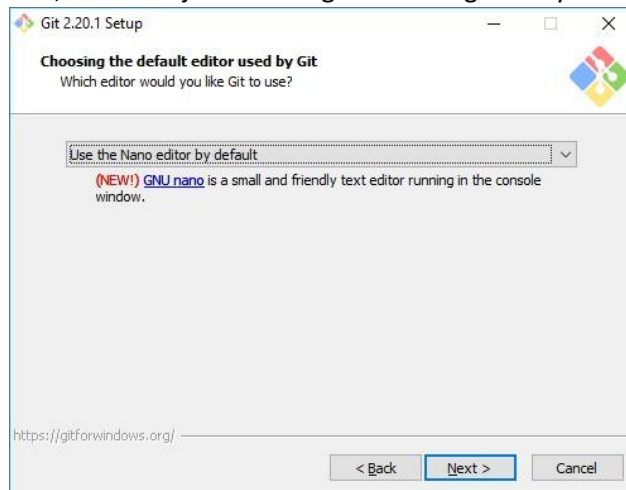
Gambar 2-1 Instalasi Git

4. Pada bagian ini, Anda dapat memilih komponen apa saja yang akan dipasang, jika sudah klik *next* untuk melanjutkan.



Gambar 2-2 Opsi instalasi pada git

5. Pilih editor yang akan digunakan secara *default* oleh Git. Gunakan Nano jika ingin editor yang lebih simpel untuk digunakan, atau Vim jika memang Anda menguasainya.



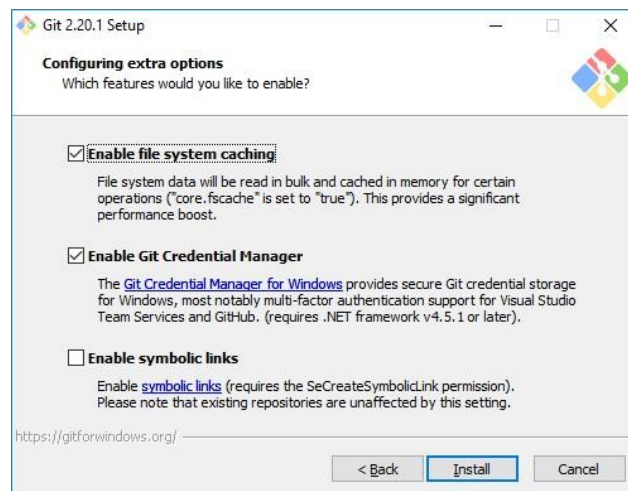
Gambar 2-3 Pilihan editor

6. Pilih opsi ketiga agar *Command Prompt* dapat mengenali Git dan beberapa perintah UNIX lainnya.



Gambar 2-4 Pilihan opsi environment

7. Untuk selanjutnya, gunakan opsi *default* sampai Anda berada pada tahap *install*. Lalu klik *install*.



Gambar 2-5 Fitur opsional pada Git

8. Pastikan Git sudah terinstall dengan melakukan perintah `git --version` pada *command prompt*.

```
C:\>git --version
git version 2.20.1.windows.1
```

Gambar 2-6 Mengecek versi Git

9. Lakukan konfigurasi awal dengan melakukan perintah

```
git config --global user.name "Nama Anda" git config --global user.email
email.anda@contoh.com
```

2.3. Penggunaan Git

2.3.1. Membuat repositori baru

Untuk membuat repositaru baru, gunakan perintah dibawah ini:

```
git init modul-1
```

Jika berhasil maka tampilannya akan seperti ini pada *command prompt* Anda.

```
D:\WEBPRO>git init modul-1  
Initialized empty Git repository in D:/WEBPRO/modul-1/.git/
```

Gambar 2-7 Inisialisasi repositori git

Perintah **git init** akan membuat sebuah direktori bernama **.git** di dalam proyek yang akan dikerjakan. Direktori ini digunakan Git sebagai *database* untuk menyimpan perubahan yang kita lakukan.

2.3.2. Menambahkan isi repositori

Untuk menambahkan suatu *file* ke dalam repositori, kita langsung dapat menambahkan *file* yang kita inginkan ke dalam *folder* proyek yang telah kita buat, contohnya dapat dilihat dalam dengan perintah yang ada pada Gambar 1-15.

```
D:\WEBPRO\modul-1>touch test.txt  
D:\WEBPRO\modul-1>echo "Halo Git" >> test.txt  
D:\WEBPRO\modul-1>cat test.txt  
"Halo Git"  
D:\WEBPRO\modul-1>ls  
test.txt
```

Gambar 2-8 Langkah membuat file

touch test.txt adalah perintah untuk membuat satu *file* baru yaitu test.txt, **echo** "Halo Git" >> test.txt adalah perintah untuk mengisi *file* test.txt dengan "Halo Git", lalu gunakan perintah **cat** test.txt untuk melihat apa isi yang ada pada *file* test.txt

WEBPRO > modul-1		Search modul-1	
Name	Date modified	Type	Size
test.txt	2018-12-30 4:29 PM	Text Document	1 KB

Gambar 2-9 Hasil file yang telah dibuat

Dapat kita lihat bahwa dengan mengeksekusi perintah **git status** kita dapat melihat *file* yang berubah yang ada pada project kita.

```
D:\WEBPRO\modul-1>git status  
On branch master  
  
No commits yet  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
  
test.txt  
  
nothing added to commit but untracked files present (use "git add" to track)
```

Gambar 2-10 Hasil perintah dari git status

Namun harus diingat bahwa *untracked files* menandakan bahwa *file* tersebut belum disimpan dalam catatan repository kita, untuk menyimpannya, kita bisa menggunakan perintah **git add test.txt**


```
D:\WEBPRO\modul-1>git add test.txt

D:\WEBPRO\modul-1>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   test.txt
```

Gambar 2-11 File telah ditambahkan

Dapat kita lihat bahwa sekarang file test.txt siap untuk disimpan, namun data belum benar-benar tersimpan sampai kita melakukan perintah `git commit -m "pesan commit"`. Setelah perintah `git commit`, maka git akan menyimpan semua perubahan yang ada, dan dapat dilihat dengan menggunakan perintah `git status`, maka akan jadi seperti ini.

```
D:\WEBPRO\modul-1>git commit -m "menambahkan test file"
[master (root-commit) 1570abc] menambahkan test file
1 file changed, 1 insertion(+)
create mode 100644 test.txt

D:\WEBPRO\modul-1>git status
On branch master
nothing to commit, working tree clean
```

Gambar 2-12 Setelah commit dan melihat status

2.3.3. Membuat repositori online

Pada tahap ini kita akan menggunakan tempat penyimpanan repositori *online* yang cukup populer, yaitu Github untuk menyimpan hasil pekerjaan kita. Pastikan kita sudah memiliki akun Github, dan sudah masuk kedalam akun kalian. Langkah yang harus kalian lakukan adalah sebagai berikut:

1. Buka link berikut ini. <https://github.com/new>
2. Isi detail dari *repository* yang akan kalian buat, lihatlah contoh acuan pada gambar 1-20.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: kamalhm / Repository name: Webpro

Great repository names are short and memorable. Need inspiration? How about literate-octo-spork.

Description (optional): Belajar membuat repository

☒ Public
Anyone can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

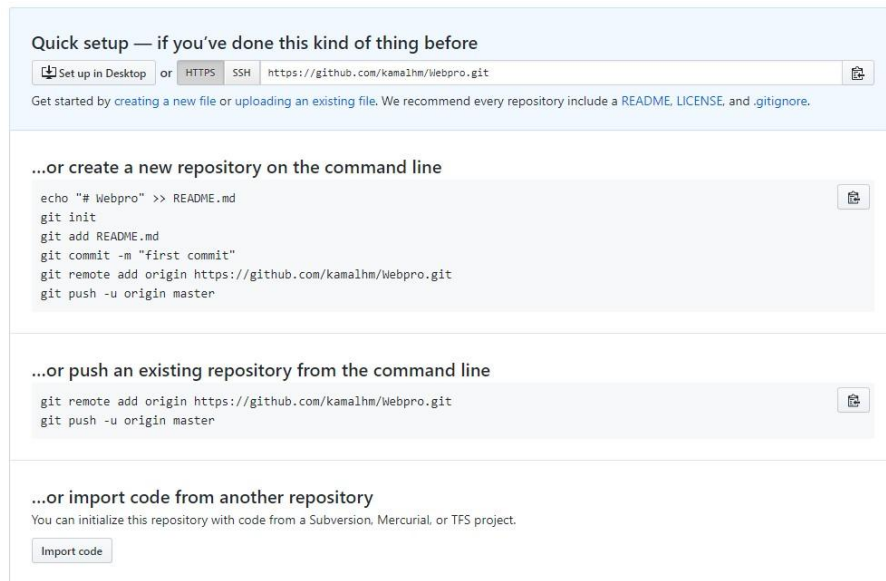
☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Create repository

Gambar 2-13 Tampilan membuat repositori pada Github

3. Lalu klik tombol "Create a repository".
4. Jika sudah maka akan muncul tampilan seperti gambar 2-13.

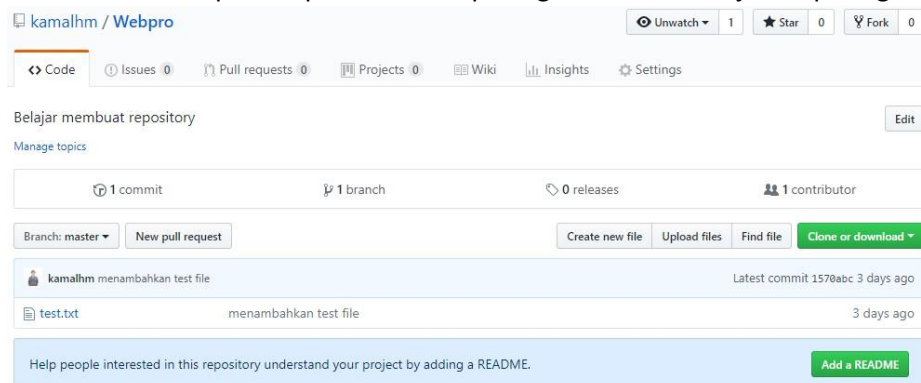


Gambar 2-14 Tampilan setelah repositori dibuat

2.3.4. Menyimpan hasil pekerjaan di repositori online.

Setelah membuat repositori pada Github, sekarang kita dapat menyimpan hasil pekerjaan yang telah kita buat kedalam Github. Untuk melakukan itu, lakukan langkah-langkah berikut:

1. Ketikkan perintah ini, sesuaikan dengan *username* dan repository Anda:
`git remote add origin https://github.com/usernameanda/namarepo.git`
Perintah ini akan menambahkan repositori *online* yang ada pada Github kedalam daftar repositori jarak jauh yang ada.
2. Untuk mengirimkan data yang ada di komputer kalian ke repositori jarak jauh, gunakan perintah ini:
`git push -u origin master`
3. Setelah selesai maka tampilan repositori Anda pada github akan menjadi seperti gambar 1-22

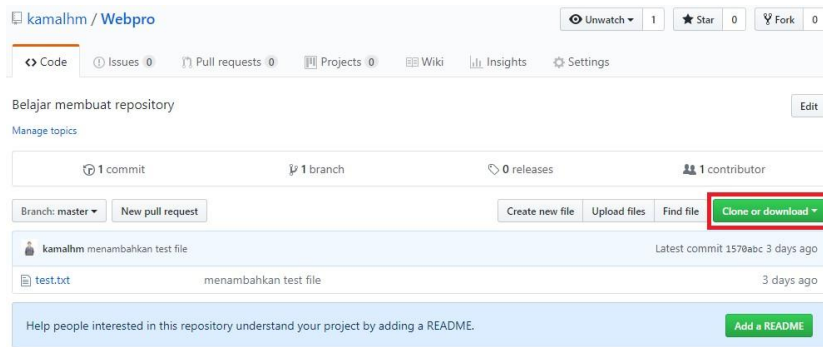


Gambar 2-15 Tampilan repositori setelah di-push pada Github

2.3.5. Clone repositori milik orang lain.

Untuk dapat bekerja sama dengan orang lain, kita dapat melakukan *cloning* repositori orang lain, berikut ini caranya:

1. Buka repositori yang akan di-*clone* pada Github, lalu klik tombol *clone*.



Gambar 2-16 Tombol clone terlihat pada kotak merah

2. *Copy text* yang muncul seperti dibawah ini, ini merupakan url dari repositori tujuan yang akan di *clone*.



Gambar 2-17 URL repositori target yang akan di-clone

3. Buka *command prompt* dan ketikkan perintah ini,
`git clone [url repositori tujuan]`

MODUL 3. HTML

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep dan implementasi HTML pada *web*.
2. Mahasiswa mampu memahami sintaks dan elemen-elemen HTML.

3.1 Pengenalan HTML

HTML atau *HyperText Markup Language* merupakan bahasa dasar yang digunakan untuk membangun sebuah *web* dimana HTML menangani elemen-elemen dasar pada pembangunan sebuah *website*. Struktur HTML paling dasar adalah sebagai berikut:



Gambar 3-1 Struktur dasar HTML

3.1.1. Tag HTML

Tag dalam HTML secara normal memiliki sepasang *tag* di mana *tag* pertama merupakan *tag* pembuka dan yang kedua merupakan *tag* penutup. Konten yang ingin ditampilkan pada laman *web* diletakkan di antara kedua *tag* tersebut.

`<nama_tag> letakkan konten di sini ... </nama_tag>`

Tag dalam HTML tidak semuanya berbentuk pasangan, ada beberapa *tag* yang hanya berdiri sendiri seperti *tag* “`
`” yang berguna untuk berpindah baris.

3.1.2. Elemen HTML

Elemen HTML merupakan *tag* HTML yang telah memiliki konten atau isi di antara kedua *tag* pembuka dan penutupnya. Elemen HTML dapat berupa teks atau juga dapat menyisipkan *tag* HTML lain pada elemen tersebut.

```
<!DOCTYPE html>
<html>

<head>
  <!-- Contoh elemen berisi tag lain -->
  <title>Page Title</title>
</head> asdff

<body>
  <h1>My First Heading</h1>
  <!-- Contoh Elemen berisi Teks -->
  <p>My first paragraph.</p>
</body>
```

```
</html>
```

3.1.3. Atribut HTML

Atribut HTML merupakan tambahan informasi dari sebuah *tag* HTML. Bentuk atribut untuk setiap *tag* HTML berbeda-beda sehingga kegunaan atribut juga berbeda seperti menambahkan informasi warna elemen, ukuran lebar, ukuran panjang dan lain-lain. Namun, mayoritas atribut yang sering muncul untuk setiap *tag* HTML adalah atribut “id” dan “class” karena kedua atribut ini berperan besar dalam pengembangan laman *web* dengan CSS dan JavaScript. Atribut HTML dideklarasikan di dalam *tag* pembuka pada setiap elemen HTML dengan format **nama_atribut=“value”**, setiap nilai atribut diapit oleh petik dua.

```
<a href="www.google.co.id" > Google.co.id </a>
<input type="button" id="btnSubmit" class="btnSubmit1" value="Kirim"/>
```

3.2. Dasar Sintaks HTML

```
<!DOCTYPE html>
<html>

<head>
  <title>Page Title</title>
</head>

<body>
  <h1>My First Heading</h1>
  <p>My first paragraph.</p>
</body>

</html>
```

Seperti yang sudah dijelaskan sebelumnya struktur dasar HTML antara lain berupa:

- Deklarasi <! DOCTYPE html> mendefinisikan dokumen menjadi HTML5
- Elemen <html> adalah elemen dasar dari halaman HTML
- Elemen <head> berisi informasi meta tentang dokumen
- Elemen <title> menentukan judul untuk dokumen
- Elemen <body> berisi konten halaman yang terlihat

3.3. Heading

Heading pada HTML merupakan *tag* yang berguna untuk menampilkan judul dari konten laman *web* yang dibangun. *Heading* dalam sebuah laman *web* berperan penting untuk aplikasi mesin pencarian karena sistem mesin pencarian bekerja dengan menggunakan *Heading* laman *web* kita sebagai *index* pencarian. Dalam HTML terdapat enam tingkatan *Heading* di mana semakin kecil nilai *heading* nya maka semakin penting dan semakin besar ukurannya pada laman *web*.

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
```

```
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

Gambar 3-2 Tingkatan heading

3.4. Hyperlink

Hyperlink dalam HTML memungkinkan halaman *web* berpindah laman atau bernavigasi menuju laman *web* yang lain. *Tag* yang digunakan adalah *tag* `<a>...`

```
<a href="www.google.co.id"> Visit Google </a>
```

Output :

[Visit Google](#)

Jika panah kursor kita arahkan pada teks tersebut, kursor akan berubah menjadi bentuk tangan.

Dalam *tag Hyperlink* pada HTML ada satu atribut yang harus digunakan agar konten yang ada di antara *tag hyperlink* berjalan dan dapat melakukan navigasi menuju laman *web* lain yaitu atribut **href**. Atribut ini bernilai *url* atau alamat dari laman *web* tujuan.

3.5. Tabel

Tabel pada HTML merupakan salah satu elemen penting khususnya digunakan untuk menampilkan data yang membutuhkan bentuk tabel. Tabel pada HTML didefinisikan dengan *tag* `<table></table>` dengan setiap pendefinisian baris menggunakan *tag* `<tr></tr>`, pendefinisian *heading* tabel menggunakan *tag* `<th></th>` dan pendefinisian kolom menggunakan *tag* `<td></td>`.

```
<table width="80%" height="50%" border="1">
  <tr>
    <th>Nama Lengkap</th>
    <th>Kota Kelahiran</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Budi</td>
    <td>Jakarta</td>
    <td>35</td>
  </tr>
  <tr>
    <td>Andi</td>
    <td>Semarang</td>
    <td>52</td>
  </tr>
```



```

<tr>
  <td>Rasyid</td>
  <td>Surabaya</td>
  <td>22</td>
</tr>
</table>

```

Nama Lengkap	Kota Kelahiran	Age
Budi	Jakarta	35
Andi	Semarang	52
Rasyid	Surabaya	22

Gambar 3-3 Tabel pada HTML

Dalam tabel HTML kita dapat melakukan operasi *Merge Cell* yang biasanya dapat dilakukan pada aplikasi perkantoran seperti Microsoft Word atau Excel dengan cara menambahkan atribut **colspan** dan **rowspan** pada *tag* pembuka kolom yaitu <td> nilai dari atribut tersebut berupa jumlah kolom atau baris yang akan digabungkan.

```

<table width="80%" height="50%" border="1">
  <tr>
    <th rowspan="2">Nama Lengkap</th>
    <th colspan="2">Gelar Pendidikan</th>
    <th rowspan="2">Age</th>
  </tr>
  <tr>
    <th>Sarjana </th>
    <th>Magister </th>
  </tr>
  <tr>
    <td>Budi</td>
    <td>S.Kom</td>
    <td>M.Sc</td>
    <td>35</td>
  </tr>
  <tr>
    <td>Andi</td>
    <td>S.SiKom</td>
    <td>M.T</td>
    <td>52</td>
  </tr>
</table>

```

Nama Lengkap	Gelar Pendidikan		Age
	Sarjana	Magister	
Budi	S.Kom	M.Sc	35
Andi	S.SiKom	M.T	52

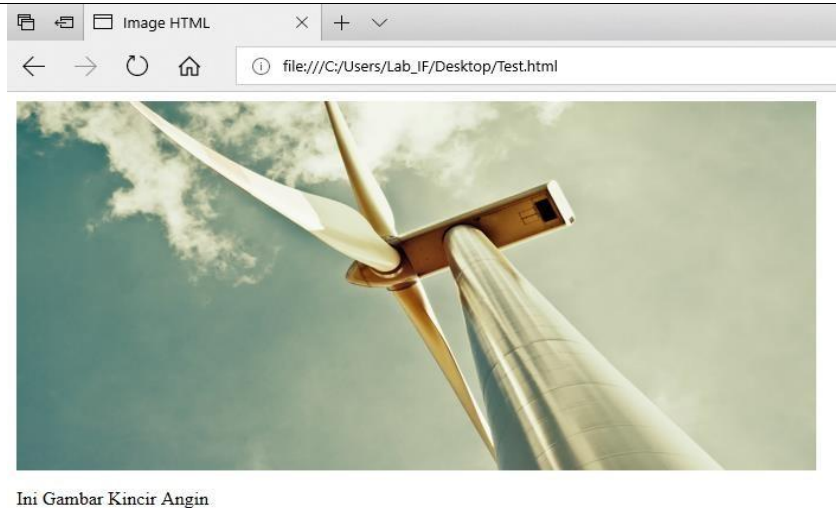
Gambar 3-4 Penggunaan colspan dan rowspan pada tabel HTML

3.6. Image

Menampilkan gambar pada halaman *web* merupakan sebuah improvisasi dalam pembuatan desain sebuah *web* yang dapat memperindah tampilan *website*. *Tag* HTML yang digunakan adalah `` *tag* ini tidak memiliki pasangan penutup maka dari itu diakhir *tag* pembuka ditambahkan garis miring seperti di atas. Terdapat satu atribut wajib yang harus ditambahkan seperti atribut **href** pada *tag Hyperlink* yaitu atribut **src** yang bernilai alamat direktori gambar disimpan.

```

<p>Ini Gambar Kincir Angin</p>
```



Gambar 3-5 Penggunaan image pada HTML

3.7. Audio / Video Elemen

Sebelum berkembangnya teknologi HTML5, untuk menyisipkan audio atau video, diperlukan sebuah *plugin* seperti *Flash Player* namun sekarang dengan HTML5 memiliki *tag* yang dapat menyisipkan audio atau video ke dalam laman *web*. Untuk audio menggunakan *tag* `<audio>` untuk *tag* pembuka dan `<source>` untuk memanggil *url* atau alamat direktori *file*. Sedangkan untuk video menggunakan *tag* `<video>`.

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg"> Your browser does not support the audio
  element.
</audio>

<video width="400" controls>
  <source src="mov_bbb.mp4" type="video/mp4">
  <source src="mov_bbb.ogg" type="video/ogg"> Your browser does not support HTML5
  video.
</video>

<p>
  Video courtesy of
  <a href="https://www.bigbuckbunny.org/" target="_blank">Big Buck Bunny</a>.
</p>
```



Video courtesy of [Big Buck Bunny](#).

Gambar 3-6 Penggunaan audio dan video pada HTML

3.8. Form

Form pada HTML digunakan sebagai wadah untuk menampung dan mengumpulkan data-data dari pengguna jika diperlukan untuk disimpan dalam sebuah *database*. *Tag* dasar untuk pemanggilan *form* adalah `<form> ... </form>` dan diantara *tag form* tersebut merupakan tempat mendefinisikan elemenelemen yang dibutuhkan *form* yang akan dibuat nantinya. Atribut utama dari *tag form* yaitu **action**, atribut ini bernilai tujuan data akan diolah dengan bahasa pemrograman *web* saat tombol “Submit” ditekan, selain itu atribut **method** yang hanya bernilai POST atau GET ini juga sangat dibutuhkan untuk pengolahan data dengan bahasa pemrograman *web*. Pembahasan lebih lanjut ada pada modul PHP.

3.8.1. Elemen Form

Elemen-elemen form pada html adalah sebagai berikut:

Tabel 3-1. Elemen form

Tag	Type	Fungsionalitas
<input/>	type = “text”	Menampilkan elemen untuk input data teks
	type = “password”	Menampilkan elemen untuk input data password
	type = “email”	Menampilkan elemen untuk input data email
	type = “radio”	Menampilkan elemen untuk pemilihan data berbentuk radio
	type = “checkbox”	Menampilkan elemen untuk pemilihan data berbentu <i>checkbox</i>
	type = “submit”	Menampilkan elemen tombol untuk pengolahan data <i>form</i>

<select> <option> ... </option> </select>	-	Menampilkan elemen untuk pemilihan data berbentuk <i>dropdown list</i>
<textarea> ... </textarea>	-	Menampilkan elemen untuk input data dalam bentuk paragraf panjang.
<button> ... </button>	type = "button"	

3.8.2. Atribut Elemen Form

Atribut-atribut elemen *form* yang sering digunakan antara lain sebagai berikut:

Tabel 3-2. Atribut elemen form

Atribut	Value	Fungsionalitas
id	Bebas	Menambahkan informasi ID dari elemen untuk kebutuhan CSS, JavaScript atau Bahasa Pemrograman Web yang lain.
name	Bebas	Menambahkan informasi Name dari elemen untuk kebutuhan JavaScript atau Bahasa Pemrograman Web yang lain.
class	Bebas	Menambahkan informasi Class dari elemen untuk kebutuhan CSS, JavaScript atau Bahasa Pemrograman Web yang lain.
placeholder	Bebas	Menampilkan informasi sementara tentang elemen sebelum memberikan inputan pada elemen.
value	Bebas	Memberikan nilai <i>default</i> pada elemen khususnya elemen type dan option
disabled	-	Membuat elemen menjadi tidak dapat dioperasikan
readonly	-	Membuat elemen type tidak dapat dirubah atau diberi inputan
checked	-	Membuat elemen <i>checkbox</i> atau radio button menjadi terpilih secara <i>default</i> .

selected	-	Membuat elemen option pada <i>tag select</i> menjadi terpilih secara <i>default</i> .
----------	---	---

```

<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Formulir Pendaftaran Praktikan</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Formulir Pendaftaran Praktikan</h1>
  <form action="#" method="POST">
    <div class="form-group">
      <label for="nama_id">Nama:</label>
      <input type="text" name="nama_input" id="nama_id" placeholder="Input Nama"
value="Praktikan" readonly>
    </div>
    <div class="form-group">
      <label for="uname_id">Username:</label>
      <input type="text" name="uname_input" id="uname_id" placeholder="Input
Username" value="Praktikum" disabled>
    </div>
    <div class="form-group">
      <label for="password_id">Password:</label>
      <input type="password" name="password_input" id="password_id"
placeholder="Input Password">
    </div>
    <div class="form-group">
      <label for="email_id">Email:</label>
      <input type="email" name="email_input" id="email_id" placeholder="Input
Email">
    </div>
    <fieldset>
      <legend>Jenis Kelamin</legend>
      <div>
        <input type="radio" name="jk_input" id="pria" value="Pria">
        <label for="pria">Pria</label>
      </div>
      <div>
        <input type="radio" name="jk_input" id="wanita" value="Wanita">
        <label for="wanita">Wanita</label>
      </div>
    </fieldset>
    <fieldset>
      <legend>Hobi</legend>
      <div>
        <input type="checkbox" name="hobi_input" id="renang" value="Renang">
        <label for="renang">Renang</label>
      </div>
      <div>
        <input type="checkbox" name="hobi_input" id="bersepeda"
value="Bersepeda">
        <label for="bersepeda">Bersepeda</label>
      </div>
      <div>
        <input type="checkbox" name="hobi_input" id="memancing"
value="Memancing">

```

```

        <label for="memancing">Memancing</label>
    </div>
</fieldset>
<div class="form-group">
    <label for="jp_id">Jenjang Pendidikan:</label>
    <select name="jp_input" id="jp_id">
        <option value="" selected>-----Pilih-----</option>
        <option value="D3">Tamat D3</option>
        <option value="S1">Tamat S1</option>
        <option value="S2">Tamat S2</option>
        <option value="S3">Tamat S3</option>
    </select>
</div>
<div class="form-group">
    <label for="kritik_saran">Kritik & Saran:</label>
    <textarea id="kritik_saran" name="kritik_saran" rows="5"></textarea>
</div>
<div class="form-actions">
    <button type="button" class="btn-cancel">Cancel</button>
    <button type="submit" class="btn-submit">Submit</button>
</div>
</form>
</body>
</html>

```

Formulir Pendaftaran Praktikan

Nama :
 Username :
 Password :
 Email :
 Jenis Kelamin : ☐ Pria ☒ Wanita
 Hobi : ☒ Renang
 ☐ Bersepeda
 ☒ Memancing
 Jenjang Pendidikan :
 Kritik & Saran :

Pilih

Tamat D3

Tamat S1

Tamat S2

Tamat S3

Gambar 3-7 Contoh form pendaftaran

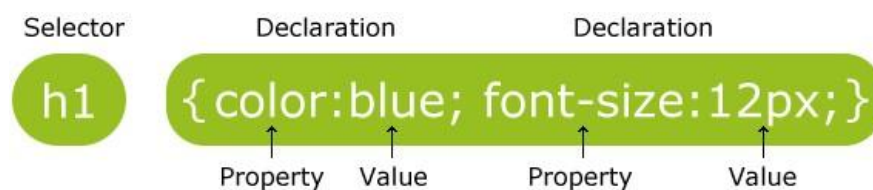
MODUL 4. CSS – Cascading Style Sheet

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep dan implementasi CSS pada *web*.
2. Mahasiswa mampu mengenali jenis-jenis dan penggunaan CSS.

4.1. Pengenalan CSS

Cascading Style Sheets (CSS) merupakan bahasa yang membantu memperindah tampilan dari laman *web* yang telah dibangun dengan HTML. CSS mendeskripsikan bagaimana bentuk tampilan elemen HTML seharusnya saat ditampilkan pada laman *browser*. Format penulisan CSS secara umum ditunjukkan pada gambar berikut.



Gambar 4-1. Penulisan CSS

Selector merupakan elemen HTML yang akan ditambahkan CSS kemudian diikuti dengan *declaration block* yang terdiri dari *property* elemen yang akan dirubah beserta *value* dari *property*-nya. Setiap deklarasi *selector* dapat merubah banyak nilai *property* sekaligus dengan dipisahkan dengan titik koma dan untuk semua *declaration block* dari satu *selector* berada di antara kurung kurawal.

4.1.1. Cara Menyisipkan CSS

Terdapat tiga cara untuk menyisipkan atau mendefinisikan CSS ke dalam HTML, antara lain:

a) *External Style Sheet*

Eksternal Style Sheet merupakan cara menyisipkan atau mendefinisikan CSS ke dalam HTML dengan memanggil file dengan ekstensi *.css* ke dalam *file* HTML. Pemanggilannya diletakkan di antara elemen `<head></head>` dengan menggunakan tag `<link/>`.

```
<head>

    <link rel="stylesheet" type="text/css" href="myStyleSheet.css">

</head>
```

b) *Internal Style Sheet*

Internal Style Sheet merupakan cara menyisipkan atau mendefinisikan CSS ke dalam HTML dengan menggunakan tag `<style> </style>` pada elemen `<head></head>`. Biasanya digunakan ketika satu laman membutuhkan *style* CSS yang berbeda dari yang telah dipanggil pada *Eksternal Style Sheet*.

```

<head>
  <style>
    body {
      background-color: blue;
    }

    h1 {
      color: maroon;
      margin-left: 40px;
    }
  </style>
</head>

```

c) *Inline Style*

Inline Style menyisipkan atau mendefinisikan CSS ke dalam HTML dengan menambahkan atribut **style** pada elemen yang ingin ditambahkan CSS. Biasanya digunakan hanya untuk satu elemen yang membutuhkan *style* CSS yang berbeda dari yang telah didefinisikan pada *Internal Style* atau *Eksternal Style*.

```

<h1 style="color:lightblue; font-size:30px;">Praktikum Web Programming</h1>

```

4.1.2. Selector

Selector pada CSS digunakan untuk menemukan elemen HTML untuk diberi CSS berdasarkan *selector* yang didefinisikan. Bentuk *selector* ada beberapa antara lain nama elemen HTML, atribut ID dan atribut Class.

```

/*Selector dengan Elemen
HTML*/
p {
  text-align: center;
  color: red;
}

/*Selector dengan Id Elemen
HTML*/
#para1 {
  text-align: center;
  color: red;
}

/*Selector dengan Class Elemen
HTML*/
p.center {
  text-align: center;
  color: red;
}

```

4.2. Font Properties

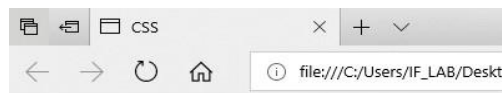
Sebuah laman *web* tentunya tidak lepas oleh penggunaan teks, oleh karena itu memiliki tampilan teks yang tepat sangat diperlukan agar sebuah *web* memiliki tampilan yang baik dan menarik. CSS dapat menangani kebutuhan tampilan teks dengan *font properties*.

Tabel 4-1. Tabel font properties

Font Properties	Keterangan
Font-family	Menentukan jenis font yang digunakan
Font-size	Mengatur ukuran font
Font-style	Mengatur style font (normal, italic, oblique)
Font-weight	Mengatur style font (normal atau bold)

Contoh penerapannya sebagai berikut:

```
p.example {
  font-family: Arial;
  font-size: 20px;
  color: light;
  font-style: italic;
  font-weight: bold;
}
```



Contoh penerapan font properties.

Gambar 4-2. Penerapan font properties

4.3. List Properties

Dalam HTML terdapat elemen yang berguna membuat sebuah *list* menggunakan simbol dan karakter. *Tag* yang digunakan adalah *tag* `` atau ``. *Tag* `` digunakan ketika akan menggunakan *list* dengan penanda berupa simbol atau bisa dikatakan *unordered list*, sedangkan *tag* `` digunakan ketika akan menggunakan *list* dengan penanda karakter yang memiliki urutan atau bisa dikatakan *ordered list*. Namun di dalam *tag* tersebut juga harus didefinisikan tag pendukung yaitu `` untuk mendefinisikan elemen-elemen *list* yang akan ditampilkan. Untuk setiap *tag ordered list* atau *unordered list* memiliki satu atribut untuk mendefinisikan tipe simbol atau karakter yang akan digunakan yaitu atribut **type**. Contoh penerapan dan tipe masing-masing *tag* sebagai berikut:

```
<h3>List of Property</h3>
<ol type="1">
  <li>Indoor
    <ul type="circle">
      <li>Sofa</li>
    </ul>
    <ul type="disc">
      <li>Tanaman Hias</li>
    </ul>
    <ul type="square">
      <li>Lampu Baca</li>
    </ul>
    <ul type="none">
      <li>Rak Buku</li>
    </ul>
  </li>
  <li>Outdoor
    <ol type="A">
```

```

    <li>Payung Pantai</li>
  </ol>
  <ol type="a">
    <li>Ayunan</li>
  </ol>
  <ol type="I">
    <li>Kursi Taman</li>
  </ol>
  <ol type="i">
    <li>Lampu Taman</li>
  </ol>
</li>
</ol>

```



List of Property

1. Indoor
 - Sofa
 - Tanaman Hias
 - Lampu Baca
 - Rak Buku
2. Outdoor
 - A. Payung Pantai
 - a. Ayunan
 - I. Kursi Taman
 - i. Lampu Taman

Gambar 4-3. Tampilan penggunaan *list properties*

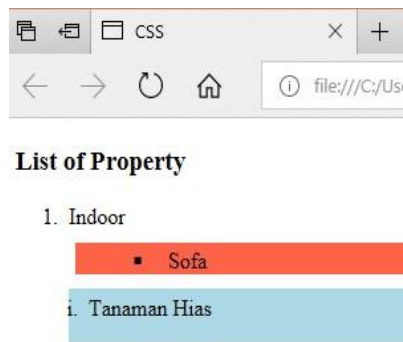
Dengan ditambahkan CSS pada elemen *list*, maka *list* yang ditampilkan dapat lebih menarik, berikut CSS properties untuk elemen *list*.

Tabel 4-2. CSS properties untuk elemen *list*

Lists Specified Properties	Keterangan
List-style-image	Membuat sebuah gambar menjadi penanda <i>list</i>
List-style-position	Mengatur posisi penanda list di dalam konten atau di luar konten
List-style-type	Mengatur jenis penanda <i>list</i>

Lists General Properties	Keterangan
Background-color	Mengatur warna latar belakang elemen <i>list</i>
Padding	Mengatur ruang jarak elemen konten dengan pembatas pada bagian dalam
Margin	Mengatur ruang jarak elemen konten dengan pembatas pada bagian luar

Contoh penerapannya sebagai berikut:



Gambar 4-4. *List properties* dengan tambahan CSS

```
ul.listsatu {  
  
    background-color: tomato;  
  
    margin: 10px 5px 10px 5px;  
  
    list-style-type: lower-alpha;  
  
    list-style-position: inside;  
}  
  
ol.listdua {  
  
    background-color: lightblue;  
  
    list-style-type: lower-roman;  
  
    padding: 5px 5px 15px 15px;  
  
    list-style-position: inside;  
}
```

4.4. Alignment of Text

Pengaturan *alignment* pada sebuah teks juga dapat ditangani oleh CSS dengan *properties* pada tabel 4-3.

Tabel 4-3. Alignment of text

Properties	Value	Keterangan
Text-align	Center	Membuat teks menjadi rata tengah
	Left	Membuat teks menjadi rata kiri
	Right	Membuat teks menjadi rata kanan
	Justify	Membuat paragraf menjadi rata kanan dan kiri

Contoh penerapannya pada gambar 2-5:



Gambar 4-5. Penerapan *alignment of text*

```
h1 {
    text-align: center;
}
h2 {
    text-align: left;
}
h3 {
    text-align: right;
}
```

4.5. Colors

Jika berbicara desain antar muka *web*, permasalahan tentang warna merupakan salah satu hal yang penting. Pada dasarnya Tag HTML dapat menangani pengaturan warna latar belakang atau teks menggunakan atribut dari HTML sendiri, namun CSS dapat menangani lebih baik dengan menawarkan pengaturan yang lebih lengkap.

Tabel 4-4. CSS Colors

Properties	Keterangan	Value	Color Names (Red, Green, Orange dll.)
Background-color	Mengatur warna latar belakang elemen HTML		RGB Value (R, G, B)
			Hex Value (#FFFF00)
			HSL Value (Hue, Saturation, Light)
Color	Mengatur warna teks elemen HTML		RGBA (dengan <i>Opacity</i>)
			HSLA (dengan <i>Opacity</i>)

Contoh penerapannya sebagai berikut :

```
body{
    background-color: HSL(20%, 40%, 70%);
    color: orange;
}

#teks{
    color: #2F3CDF;
}

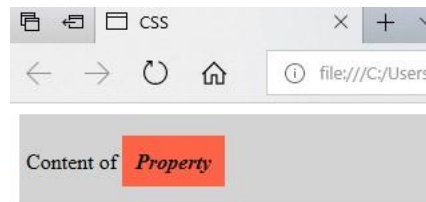
/*dengan opacity sebesar 0.5*/
input.text-field{
    background-color: RGBA(32, 55, 122, 0.5);
}
```


4.6. Span & Div

Span merupakan elemen HTML yang dapat menangani perubahan konten elemen pada satu baris. *Tag* yang digunakan adalah ``. Sedangkan *Div* merupakan elemen HTML yang digunakan untuk membuat *section* untuk beberapa elemen HTML di dalamnya. Tag yang digunakan yaitu `<div></div>`.

```
<div class="section1">
  <p> Content of <span class="mark"> Property </span> </p>
</div>

/*CSS Properties*/
.section1 {
  background-color: lightgrey;
  padding: 10px 5px 10px 5px;
}
.mark {
  background-color: tomato;
  font-style: italic;
  font-weight: bold;
  padding: 10px 10px 10px 10px;
}
```



Gambar 4-6. Penerapan Span & Div

MODUL 5. BOOTSTRAP

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep dan implementasi Bootstrap pada *web*.
2. Mahasiswa mengetahui jenis-jenis Bootstrap serta cara penggunaannya.

5.1. Pengenalan Bootstrap

Bootstrap merupakan sebuah *front-end framework* gratis untuk pengembangan antar muka *web* yang lebih cepat dan lebih mudah. Dikembangkan oleh Mark Otto dan Jacom Thornton di Twitter dan dirilis sebagai produk *open source* pada Agustus 2011 di GitHub. Bootstrap mencakup *template* desain berbasis HTML dan CSS untuk tipografi, *form*, *button*, navigasi, *modal*, *image carousells* dan masih banyak lagi, serta terdapat opsional *plugin* JavaScript. Selain itu, Bootstrap memiliki kemampuan untuk membuat desain responsif yang secara otomatis menyesuaikan diri agar terlihat baik di segala perangkat, mulai dari perangkat ponsel hingga *desktop pc*.

5.1.1. Pemasangan Bootstrap

Bootstrap merupakan produk yang mengusung konsep *open source* sehingga untuk pemasangannya dapat dilakukan dengan beberapa cara sebagai berikut:

- a. Unduh di <http://getbootstrap.com>, selanjutnya pasang pada *project web* kalian seperti memanggil *External Style Sheet* pada CSS.
- b. Memanggil Bootstrap CDN (*Content Delivery Network*), sehingga kita tidak perlu mengunduh dan memasangnya pada laman *website*, hanya memanggil *source* dari Bootstrap. Cara ini membutuhkan koneksi internet untuk menghasilkan perubahan tampilan CSS.

```
<!-- Pemanggilan Bootstrap dengan CDN -->

<!-- CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeRhO02iuK6FUUVM"
crossorigin="anonymous">

<!-- jQuery library -->

<script src="https://code.jquery.com/jquery-3.7.0.min.js" integrity="sha256-
2PmVv0kuTBOenSvLm6bvfbSSHrUJ+3A7x6P5Ebd07/g=" crossorigin="anonymous"></script>

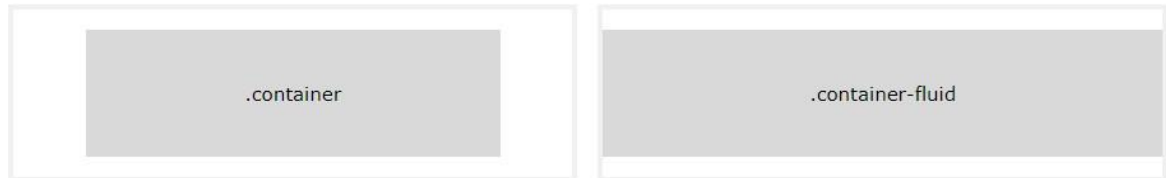
<!-- JavaScript -->

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"
integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bkz"
crossorigin="anonymous"></script>
```

5.2. Bootstrap Container

Bootstrap *container* adalah elemen paling dasar yang dibutuhkan dalam *layouting* menggunakan Bootstrap *Grid*. *Container* berbentuk class CSS yang sisipkan pada elemen HTML <div>. Pada gambar 5-1 terdapat dua *class container* pada Bootstrap yang dapat dipilih yaitu:

- Class **.container** menyediakan *container* yang *responsive* dengan lebar yang tetap.
- Class **.container-fluid** menyediakan *container* dengan lebar yang penuh mencakup seluruh area pandang.



Gambar 5-1. Class container

5.3. Bootstrap Grid

Sistem *grid* pada Bootstrap menggunakan rangkaian *container*, *rows* dan *column* untuk tata letak dan keselarasan elemen atau konten. Dibangun dengan *flexbox* dan sangat responsif terhadap perangkat yang digunakan untuk menampilkan laman *web*. Struktur dasar *grid* pada Bootstrap sebagai berikut.

```
<div class="row">
  <div class="col-*-#"></div>
  <div class="col-*-#"></div>
</div>
<div class="row">
  <div class="col-*-#"></div>
  <div class="col-*-#"></div>
  <div class="col-*-#"></div>
</div>
```

Pertama diawali dengan <div class="container">. Kemudian buat sebuah baris sebelum mendeklarasikan sebuah kolom dengan menggunakan <div class="row">. Terakhir buat elemen div dengan mendefinisikan class "col-*-#". Tanda * dan # mewakili jenis dan ukuran *column* yang akan digunakan, *value* yang dapat didefinisikan dapat dilihat pada tabel 1 :

Tabel 5-1. Sistem grid Bootstrap

	Extra Small	Small	Medium	Large	Extra Large
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# per columns	12				
Nestable	Yes				
Column Ordering	Yes				

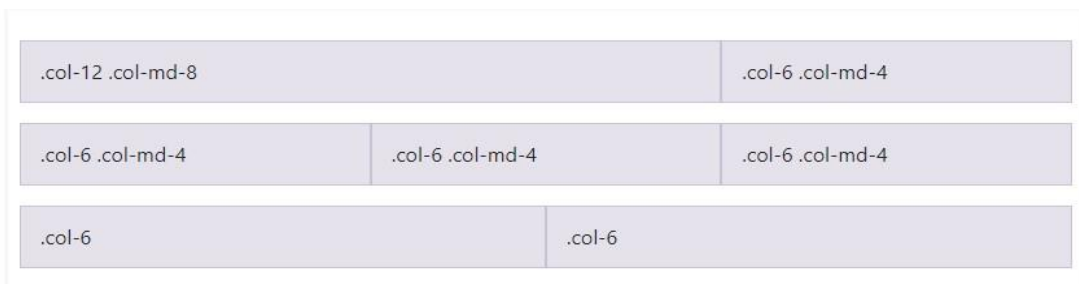
Contoh penerapannya sebagai berikut:

```
<div class="container">
  <div class="row">
    <div class="col-12 col-md-8">.col-12 .col-md-8</div>
```

```

    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>
<div class="row">
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>
<div class="row">
    <div class="col-6">.col-6</div>
    <div class="col-6">.col-6</div>
</div>
</div>

```



Gambar 5-2. Hasil penerapan Bootstrap *grid*

5.4. Text Style

Bootstrap menyediakan banyak *class* untuk mengatur *style* sebuah teks elemen HTML. Beberapa contohnya antara lain:

Tabel 6-2. Text style

Class	Keterangan
.text-left	Mengatur teks menjadi rata kiri dalam sebuah elemen.
.text-center	Mengatur teks menjadi rata tengah dalam sebuah elemen.
.text-right	Mengatur teks menjadi rata kanan dalam sebuah elemen.
.text-lowercase	Mengatur seluruh teks pada elemen menjadi huruf kecil
.text-uppercase	Mengatur seluruh teks pada elemen menjadi huruf besar
.text-capitalize	Menjadikan huruf pertama besar untuk setiap kata pad sebuah elemen.
.fw-bold	Mengatur ketebalan huruf menjadi <i>bold</i>

.fw-light	Mengatur ketebalah huruf menjadi <i>light</i>
.fw-normal	Mengatur ketebalan huruf menjadi normal
.fst-italic	Mengatur gaya teks menjadi miring
.h1 s.d .h6	Mengatur seluruh teks pada sebuah elemen sehingga memiliki tampilan selayaknya tag elemen H1 s.d H6 pada HTML.

5.5. Bootstrap Table, Image & Button

Bootstrap menyediakan *class* untuk pengaturan *style* elemen tabel, gambar dan tombol menjadi lebih menarik.

a. Bootstrap Table

Tabel pada Bootstrap dipanggil dengan *class* **.table** secara *default*, namun ada beberapa *class* tambahan yang dapat didefinisikan pada elemen tabel yang lain berikut dapat dilihat pada Tabel 5.3:

Tabel 5-3. Elemen tabel

Class	Keterangan
.table-dark	Membuat tampilan tabel memiliki latar belakang gelap
.thead-light	Membuat elemen <thead> pada tabel memiliki latar belakang cerah
.thead-dark	Membuat elemen <thead> pada tabel memiliki latar belakang gelap
.table-striped	Membuat tampilan tabel memiliki latar belakang setiap row yang berbeda
.table-bordered	Membuat tampilan tabel sederhana dengan border tipis
.table-hover	Membuat tampilan tabel yang akan berubah warna latar belakang row saat didekati kursor.
.table-sm	Membuat tampilan tabel sederhana yang dengan ukuran <i>padding</i> yang minim

Contoh penerapannya sebagai berikut:

```
<!--Tabel Hover Style -->
<table class="table table-hover">
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">Nama Lengkap</th>
      <th scope="col">Asal Kota</th>
      <th scope="col">Umur</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1</th>
```

```

        <td>Budi Rojadi</td>
        <td>Semarang</td>
        <td>35 th</td>
    </tr>
    <tr>
        <th scope="row">2</th>
        <td>Yulia Santi</td>
        <td>Bekasi</td>
        <td>32</td>
    </tr>
    <tr>
        <th scope="row">3</th>
        <td>Fahri Abdilah</td>
        <td>Medan</td>
        <td>38 th</td>
    </tr>
</tbody>
</table>

```

#	Nama Lengkap	Asal Kota	Umur
1	Budi Rojadi	Semarang	35 th
2	Yulia Santi	Bekasi	32
3	Fahri Abdilah	Medan	38 th

Gambar 5-3. Hasil penerapan Class.table-hover

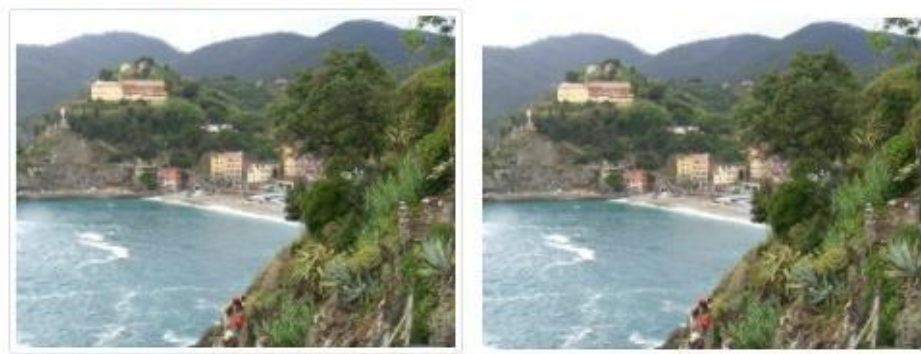
#	Nama Lengkap	Asal Kota	Umur
1	Budi Rojadi	Semarang	35 th
2	Yulia Santi	Bekasi	32

Gambar 5-4. Hasil penerapan Class.thead-dark

b) Bootstrap Image

Bootstrap dapat menangani desain gambar agar responsif pada setiap perangkat yang menampilkan laman *web*. Dengan menambahkan *class .img-fluid* pada elemen *tag * pada HTML maka gambar yang didefinisikan pada laman *web* akan memiliki ukuran yang responsif menyesuaikan ukuran layar perangkat. *Class* tersebut mengatur ukuran gambar dengan menyesuaikan ukuran dari *parent element* sebagai wadah atau *container* elemen gambar. Terdapat *class .thumbnail* yang berguna menjadikan gambar menjadi berukuran kecil dan sedikit memiliki border disekitarnya dapat dilihat pada gambar 5-5.

Thumbnail & Fluid



Gambar 5-5. Image class.thumbnail dan class.fluid

```
<div class="container">
  <h2>Thumbnail & Fluid</h2>
  
  
</div>
```

c) Bootstrap Button

Tampilan *button* pada elemen HTML dapat dirubah dengan menambahkan beberapa *class* untuk *button* oleh Bootstrap. Bootstrap membuat tampilan *button* menjadi lebih menarik dan memberikan *user experience* yang baik. *Class* yang digunakan secara *default* adalah **.btn** namun dengan disertai *class* lain seperti berikut untuk memberikan perubahan warna dan ukuran *button*:

Tabel 5-4. Class button

Class	Keterangan
.btn-primary	Membuat tampilan button dengan desain utama
.btn-secondary	Membuat tampilan button dengan ukuran medium (standar) dan desain "secondary"
.btn-danger	Membuat tampilan button dengan desain berwarna merah
.btn-success	Membuat tampilan button dengan desain berwarna hijau
.btn-warning	Membuat tampilan button dengan desain berwarna kuning
.btn-info	Membuat tampilan button dengan desain sebuah informasi
.btn-link	Membuat tampilan button dengan desain sebuah <i>hyperlink</i>

Class	Keterangan
.btn-sm	Membuat tampilan button berukuran <i>small</i>
.btn-lg	Membuat tampilan button berukuran besar

Contoh penerapan *class button*:

```
<div class="container">
  <h4>Button Styles</h4>
  <button type="button" class="btn btn-secondary">Secondary</button>
  <button type="button" class="btn btn-primary btn-lg">Primary</button>
  <button type="button" class="btn btn-success w-100">Success</button>
  <button type="button" class="btn btn-info btn-sm">Info</button>
  <button type="button" class="btn btn-warning">Warning</button>
  <button type="button" class="btn btn-danger">Danger</button>
  <button type="button" class="btn btn-link">Link</button>
</div>
```

Button Styles



Gambar 5-6. Hasil penerapan class button

5.6. Bootstrap Form

Bootstrap menyediakan perubahan elemen form pada HTML baik pada segi tata letak tampilan atau tampilan antarmuka elemen-elemen dalam form. Class `.form-control` digunakan untuk sebagian besar elemen input dalam tag `<form>` untuk memberikan styling yang konsisten. Ada beberapa cara untuk mengatur tata letak tampilan form di Bootstrap:

1. **Vertical Form (Default):** Ini merupakan tampilan default saat tag form tidak didefinisikan class khusus. Setiap elemen form akan ditampilkan secara vertikal.
2. **Inline Form:** Untuk membuat form inline di Bootstrap, Anda dapat menggunakan utility classes tertentu pada container form. Ini akan membuat elemen-elemen form berada dalam satu baris.
3. **Horizontal Form:** Untuk membuat form horizontal di Bootstrap, Anda dapat menggunakan sistem grid Bootstrap. Gunakan class `.row` pada container dan `.col-*` untuk mengatur lebar kolom label dan input.

```
<div class="container">
  <h3>Horizontal form</h3>
  <form action="/action_page.php">
    <div class="row mb-3">
      <label for="uname" class="col-sm-2 col-form-label">Username:</label>
      <div class="col-sm-10">
```



```

        <input type="text" class="form-control" id="uname" placeholder="Enter username"
name="uname">
    </div>
</div>
<div class="row mb-3">
    <label for="pwd" class="col-sm-2 col-form-label">Password:</label>
    <div class="col-sm-10">
        <input type="password" class="form-control" id="pwd" placeholder="Enter
password" name="pwd">
    </div>
</div>
<div class="row">
    <div class="col-sm-10 offset-sm-2">
        <button type="submit" class="btn btn-success">Submit</button>
    </div>
</div>
</form>
</div>

```

Horizontal form

Username:

Password:

Gambar 5-7. Contoh Bootstrap form

MODUL 6. JAVASCRIPT & JQUERY

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep dan implementasi Javascript pada *web*.
2. Mahasiswa mampu memahami sintaks, elemen, dan fungsi pada Javascript.
3. Mahasiswa mampu memahami konsep dan implementasi jQuery pada *web*.
4. Mahasiswa mampu memahami sintaks jQuery.

6.1. Pengenalan Javascript

6.1.1. Sejarah Singkat Javascript

Javascript, seperti namanya, merupakan bahasa pemrograman *scripting*. Dan seperti bahasa *scripting* lainnya, Javascript umumnya digunakan hanya untuk program yang tidak terlalu besar, biasanya hanya beberapa ratus baris. Javascript pada umumnya mengontrol program yang berbasis Java. Jadi memang pada dasarnya Javascript tidak dirancang untuk digunakan dalam aplikasi skala besar.

Meskipun dibuat dengan tujuan awal untuk mengendalikan program Java, komunitas Javascript menggunakan bahasa ini untuk tujuan lain, memanipulasi gambar dan isi dari dokumen HTML. Singkatnya, pada akhirnya Javascript digunakan untuk satu tujuan utama, “menghidupkan” dokumen HTML dengan mengubah konten statis menjadi dinamis dan interaktif. Bersamaan dengan perkembangan Internet dan dunia *web* yang pesat, Javascript akhirnya menjadi bahasa utama dan satu-satunya untuk membuat HTML menjadi interaktif di dalam browser.

6.1.2. Prinsip Dasar Javascript

Prinsip dasar yang terdapat pada bahasa pemrograman javascript adalah sebagai berikut.

1. Javascript mendukung paradigma pemrograman imperatif (Javascript dapat menjalankan perintah program baris demi baris, dengan masing-masing baris berisi satu atau lebih perintah), fungsional (struktur dan elemen-elemen dalam program sebagai fungsi matematis yang tidak memiliki keadaan (*state*) dan data yang dapat berubah (*mutable data*)), dan orientasi objek (segala sesuatu yang terlibat dalam program dapat disebut sebagai “objek”).
2. Javascript memiliki model pemrograman fungsional yang sangat ekspresif.
3. Pemrograman berorientasi objek (PBO) pada Javascript memiliki perbedaan dari PBO pada umumnya.
4. Program kompleks pada Javascript umumnya dipandang sebagai program-program kecil yang saling berinteraksi.

6.2 Sintaks Umum pada Javascript

6.2.1. Tipe data dasar

Seperti kebanyakan bahasa pemrograman lainnya, Javascript memiliki beberapa tipe data untuk dimanipulasi. Seluruh nilai yang ada dalam Javascript selalu memiliki tipe data. Tipe data yang dimiliki oleh Javascript adalah sebagai berikut:

- *Number* (bilangan)
- *String* (serangkaian karakter)
- *Boolean* (benar / salah)

- *Object*
- Function (fungsi)
- Array
- Date
- *RegExp (regular expression)*
- Null (tidak berlaku / kosong)
- Undefined (tidak didefinisikan)

Kebanyakan dari tipe data yang disebutkan di atas sama seperti tipe data sejenis pada bahasa pemrograman lainnya. Misalnya, sebuah boolean terdiri dari dua nilai saja, yaitu true dan false.

6.2.2. Variabel

Seperti pada bahasa pemrograman lainnya, variabel dalam Javascript merupakan sebuah tempat untuk menyimpan data sementara. Variabel dibuat dengan kata kunci var pada Javascript.

```
var a;           // a berisi undefined
var nama = "Budi"; // nama berisi "Budi"
```

Nilai yang ada di dalam variabel dapat diganti dengan mengisi nilai baru, dan bahkan dapat diganti tipe datanya juga.

```
nama = "Anton"; // nama sekarang berisi string "Anton"
nama = 1;       // nama sekarang berisi integer 1
```

Walaupun kemampuan untuk menggantikan tipe data ini sangat memudahkan kita dalam mengembangkan aplikasi, fitur ini harus digunakan dengan sangat hati-hati. Perubahan tipe data yang tidak diperkirakan dengan baik dapat menyebabkan berbagai kesalahan (*error*) pada program, misalnya jika kita mencoba mengakses method **charAt()** (fungsi yang mengembalikan nilai *char* pada indeks sebuah *string*) setelah mengubah tipe pada contoh di atas.

6.2.3. Array

Array merupakan sebuah tipe data yang digunakan untuk menampung banyak tipe data lainnya. Berbeda dengan tipe data *object*, *array* pada Javascript merupakan sebuah tipe khusus. Walaupun memiliki *method* dan properti, *array* bukanlah objek, melainkan sebuah tipe yang “mirip objek”. Pembuatan *array* dalam Javascript dilakukan dengan menggunakan kurung siku ([]):

```
var data = ["satu", 2, true];
```

Elemen *array* pada Javascript tidak harus memiliki tipe data yang sama seperti contoh diatas. Selain itu Javascript juga mendukung untuk membuat *array* di dalam *array* yang biasa dikenal dengan *array* dua dimensi seperti contoh dibawah ini.

```
var arr2 = ["satu", "dua", ["tiga", "empat"]];
```

Pengaksesan elemen dalam *array* dilakukan dengan menggunakan kurung siku. Nilai yang kita berikan dalam kurung siku adalah urutan elemen penulisan *array* (indeks), yang dimulai dari nilai 0. Jika indeks yang diakses tidak ada, maka kita akan mendapatkan nilai *undefined*.

```
data[2]; // mengembalikan true
arr2[0][1]; // mengembalikan "dua"
data[10]; // mengembalikan undefined
```

Sebagai sebuah objek khusus, *array* juga memiliki *method* dan properti. Beberapa *method* dan properti yang populer misalnya *length*, *pop()*, dan *push()*.

```
var data = ["a", "b", "c"];  
// data.length mengembalikan 3  
data.push("d"); // mengembalikan 4 data menjadi ["a", "b", "c", "d"]  
data.pop(); // mengembalikan "d", data menjadi ["a", "b", "c"]
```

6.2.4. Pengendalian Struktur

Javascript memiliki perintah-perintah pengendalian struktur (*control structure*) yang sama dengan bahasa dalam keluarga C. Perintah `if` dan `else` digunakan untuk percabangan, sementara perintah `for`, `for-in`, `while`, dan `do-while` digunakan untuk perulangan.

Percabangan pada Javascript bisa dikatakan sama persis dengan C atau Java:

```
var gelar;  
var pendidikan = "S2";  
if (pendidikan === "S1") {  
    gelar = "Sarjana";  
} else if (pendidikan === "S2") {  
    gelar = "Master";  
} else if (pendidikan === "S3") {  
    gelar = "Doktor";  
} else {  
    gelar = "Tidak Diketahui";  
}  
gelar; // gelar berisi "Master"
```

Satu hal yang perlu diperhatikan, tiga buah sama dengan (`===`) digunakan pada operasi perbandingan di Javascript. Javascript mendukung dua operator perbandingan sama dengan, yaitu `==` dan `===`. Perbedaan utamanya adalah `==` mengubah tipe data yang dicek menjadi nilai terdekat, sementara `===` memastikan tipe data dari dua nilai yang dibandingkan sama. Untuk mendapatkan nilai perbandingan paling akurat, selalu gunakan `===` ketika mengecek nilai.

Sama seperti `if`, perulangan `do`, `do-while`, dan `for` memiliki cara pemakaian yang dapat dikatakan sama persis dengan C atau Java:

```
while (true) {  
    // tak pernah berhenti  
} var input; do {  
    input = get_input();  
} while (inputIsValid(input)) for  
(var i = 0; i < 5; i++) {  
    // berulang sebanyak 5 kali }
```

6.3. Object Orientation pada Javascript

Javascript memiliki dua jenis tipe data utama, yaitu tipe data dasar dan objek. Tipe data dasar pada Javascript adalah angka (*numbers*), rentetan karakter (*strings*), boolean (*true* dan *false*), *null*, dan *undefined*. Nilai-nilai selain tipe data dasar secara otomatis dianggap sebagai objek. Objek dalam Javascript didefinisikan sebagai *mutable properties collection*, yang artinya adalah sekumpulan properti (ciri khas) yang dapat berubah nilainya. Karena nilai-nilai selain tipe data dasar merupakan objek, maka pada Javascript sebuah *Array* adalah objek. Fungsi adalah objek dan *Regular expression* juga merupakan objek.

6.3.1 Pembuatan Object pada Javascript

Notasi pembuatan objek pada Javascript sangat sederhana, yaitu sepasang kurung kurawal yang membungkus properti. Notasi pembuatan objek ini dikenal dengan nama *object literal*. *Object literal* dapat digunakan kapanpun pada ekspresi Javascript yang valid:

```
var objek_kosong = {};  
var mobil = {  
    "warna-badan": "merah",  
    "nomor-polisi": "BK1234AB"  
};
```

Nama properti dari sebuah objek harus berupa *string*, dan boleh berisi *string* kosong (""). Jika merupakan nama Javascript yang legal, kita tidak memerlukan petik ganda pada nama properti. Petik ganda seperti pada contoh ("warna-badan") hanya diperlukan untuk nama Javascript ilegal atau kata kunci seperti "if" atau "var". Misalnya, "nomor-polisi" memerlukan tanda petik, sementara nomor_polisi tidak. Contoh lain, variasi tidak memerlukan tanda petik, sementara "var" perlu.

Sebuah objek dapat menyimpan banyak properti, dan setiap properti dipisahkan dengan tanda koma (,). Jika ada banyak properti, nilai dari properti pada setiap objek boleh berbeda-beda:

```
var jadwal = {  
    platform: 34,  
    telah_berangkat: false,  
    tujuan: "Medan",  
    asal: "Jakarta"  
};
```

Karena dapat diisi dengan nilai apapun (termasuk objek), maka kita dapat membuat objek yang mengandung objek lain (*nested object*; objek bersarang) seperti berikut:

```
var jadwal = {    platform: 34,  
telah_berangkat: false,    asal:  
{        kode_kota: "MDN",  
nama_kota: "Medan",  
waktu: "2013-12-29 14:00"  
    },    tujuan: {  
kode_kota: "JKT",  
nama_kota: "Jakarta",  
waktu: "2013-12-29 17.30"  
    }  
};
```

6.3.2. Akses Nilai Property

Akses nilai properti dapat dilakukan dengan dua cara, yaitu.

1. Penggunaan kurung siku ([]) setelah nama objek. Kurung siku kemudian diisi dengan nama properti, yang harus berupa *string*. Cara ini biasanya digunakan untuk nama properti yang adalah nama ilegal atau kata kunci Javascript.
2. Penggunaan tanda titik (.) setelah nama objek diikuti dengan nama properti. Notasi ini merupakan notasi yang umum digunakan pada bahasa pemrograman lainnya. Notasi ini tidak dapat digunakan untuk nama ilegal atau kata kunci Javascript.

Contoh penggunaan kedua cara pemanggilan di atas adalah sebagai berikut:

```
mobil["warna-badan"] // Hasil: "merah"
jadwal.platform       // Hasil: 34
```

Sebagai bahasa dinamis, Javascript tidak akan melemparkan pesan kesalahan jika kita mengakses properti yang tidak ada dalam objek. Kita akan menerima nilai *undefined* jika mengakses properti yang tidak ada:

```
jadwal.nomor_kursi    // Hasil: undefined mobil
["jumlah-roda"]      // Hasil: undefined
```

Pengaksesan properti pada Javascript juga dapat digunakan secara dinamis untuk mengubah nilai dari properti tersebut. Perubahan nilai properti juga dapat dilakukan untuk properti yang bahkan tidak ada pada objek tersebut:

```
mobil["jumlah-roda"] = 4;
mobil.bahan_bakar = "Bensin";
```

6.3.3. Prototype pada Javascript

Pada Javascript yang mengimplementasikan PBO kita tidak lagi perlu menuliskan kelas, dan langsung melakukan penurunan terhadap objek. Misalkan kita memiliki objek mobil yang sederhana seperti berikut:

```
var mobil = { nama: "Mobil",
              jumlahBan: 4 };
```

Kita dapat langsung menurunkan objek tersebut dengan menggunakan fungsi `Object.create` seperti berikut:

```
var truk = Object.create(mobil);
// truk.nama === "Mobil"
// truk.jumlahBan === 4
```

6.4. Function pada Javascript

Sebuah fungsi membungkus satu atau banyak perintah. Setiap kali fungsi dipanggil, maka perintah-perintah yang ada di dalam fungsi tersebut dijalankan. Secara umum fungsi digunakan untuk penggunaan kembali kode (*code reuse*) dan penyimpanan informasi (*information hiding*). Implementasi fungsi kelas pertama juga memungkinkan penggunaan fungsi sebagai unit-unit yang dapat dikombinasikan, seperti layaknya sebuah lego. Dukungan terhadap pemrograman berorientasi objek juga berarti fungsi dapat digunakan untuk memberikan perilaku tertentu dari sebuah objek.

6.4.1 Pembuatan Fungsi pada Javascript

Sebuah fungsi pada Javascript dibuat dengan cara seperti berikut:

```
function tambah(a, b) {
    hasil = a + b;
    return hasil;
}
```

Cara penulisan fungsi seperti diatas dikenal dengan nama *function declaration*, atau deklarasi fungsi. Terdapat empat komponen yang membangun fungsi di atas, yaitu:

1. Kata kunci *function*, yang memberitahu Javascript bahwa akan dibuat sebuah fungsi.

2. Nama fungsi, dalam contoh di atas adalah tambah. Dengan memberikan sebuah fungsi nama maka pemanggilan fungsi dapat dirujuk dengan nama tersebut. Harus diingat bahwa nama fungsi bersifat opsional, yang berarti **fungsi pada Javascript tidak harus diberi nama**.
3. Daftar parameter fungsi, yaitu a, b pada contoh di atas. Daftar parameter ini selalu dikelilingi oleh tanda kurung (). Parameter boleh kosong, tetapi tanda kurung wajib tetap dituliskan. Parameter fungsi akan secara otomatis didefinisikan menjadi variabel yang hanya bisa dipakai di dalam fungsi. Variabel pada parameter ini diisi dengan nilai yang dikirimkan kepada fungsi secara otomatis.
4. Sekumpulan perintah yang ada di dalam kurung kurawal {}. Perintah-perintah ini dikenal dengan nama badan fungsi. Badan fungsi dieksekusi secara berurut ketika fungsi dijalankan.

Penulisan deklarasi fungsi (*function declaration*) seperti di atas merupakan cara penulisan fungsi yang umumnya kita gunakan pada bahasa pemrograman imperatif dan berorientasi objek. Tetapi selain deklarasi fungsi Javascript juga mendukung cara penulisan fungsi lain, yaitu dengan memanfaatkan ekspresi fungsi (*function expression*). Ekspresi fungsi merupakan cara pembuatan fungsi yang memperbolehkan menuliskan fungsi tanpa nama. Fungsi yang dibuat tanpa nama dikenal dengan sebutan fungsi anonim atau fungsi lambda. Berikut adalah cara membuat fungsi dengan ekspresi fungsi:

```
var tambah = function (a, b) {  
    hasil = a + b;  
    return hasil;  
};
```

Terdapat hanya sedikit perbedaan antara ekspresi fungsi dan deklarasi fungsi:

1. Penamaan fungsi. Pada deklarasi fungsi, nama fungsi langsung diberikan sesuai dengan sintaks yang disediakan Javascript. Penggunaan ekspresi fungsi pada dasarnya menyimpan sebuah fungsi anonim ke dalam variabel dan nama fungsi adalah nama variabel yang dibuat. Perlu diingat juga bahwa pada dasarnya ekspresi fungsi adalah fungsi anonim. Penyimpanan ke dalam variabel hanya diperlukan karena kita akan memanggil fungsi nantinya.
2. Ekspresi fungsi dapat dipandang sebagai sebuah ekspresi atau perintah standar bagi Javascript, sama seperti penulisan kode `var i = 0`. Deklarasi fungsi merupakan konstruksi khusus untuk membuat fungsi. Hal ini berarti pada akhir dari ekspresi fungsi kita harus menambahkan, sementara pada deklarasi fungsi hal tersebut tidak penting.

6.4.2. Pemanggilan Fungsi

Sebuah fungsi dapat dipanggil untuk menjalankan seluruh kode yang ada di dalam fungsi tersebut, sesuai dengan parameter yang kita berikan. Pemanggilan fungsi dilakukan dengan cara menuliskan nama fungsi tersebut, kemudian mengisi argumen yang ada di dalam tanda kurung.

Misalkan fungsi tambah yang kita buat pada bagian sebelumnya:

```
var tambah = function (a, b) {  
    hasil = a + b;  
    return hasil;  
};
```

dapat dipanggil seperti berikut:

```
tambah(3, 5);
```

Yang terjadi pada kode di atas adalah nilai *a* dan *b* masing-masing digantikan dengan 3 dan 5. Seperti yang dapat dilihat, hal ini berarti pengisian argumen pada saat pemanggilan fungsi harus berurut sesuai dengan deklarasi fungsi.

Sama seperti sebuah variabel, fungsi juga mengembalikan nilai ketika dipanggil. Dalam kasus di atas, `tambah(3, 5)` akan mengembalikan nilai 8. Nilai ini tentunya dapat disimpan ke dalam variabel baru, atau bahkan dikirimkan sebagai sebuah argumen ke fungsi lain lagi:

```
var simpan = tambah(3, 5); // simpan === 8
tambah(simpan, 2);         // mengembalikan 10
tambah(tambah(3, 5), 2)    // juga mengembalikan 10
tambah(tambah(2, 3), 4)    // mengembalikan 9
```

Fungsi akan mengembalikan nilai ketika kata kunci *return* ditemukan. Pengembalian nilai fungsi dapat dilakukan kapanpun, dan fungsi akan segera berhenti ketika kata kunci *return* ditemukan. Berikut adalah contoh kode yang memberikan gambaran tentang pengembalian nilai fungsi:

```
var naikkan = function (n) {
  var hasil = n + 10; return
  hasil;
  // kode di bawah tidak dijalankan lagi hasil
  = hasil * 100;
} naikkan(10); // mengembalikan
20 naikkan(25); // mengembalikan
35
```

Sebuah ekspresi dapat juga diberikan langsung kepada *keyword return*, dan ekspresi tersebut akan dijalankan sebelum nilai dikembalikan. Hal ini berarti fungsi `tambah` maupun `naikkan` yang sebelumnya bisa disederhanakan dengan tidak lagi menyimpan nilai di variabel `hasil` terlebih dahulu:

```
var naikkan = function (n) {
  return n + 10;
} var tambah = function (a,
b) {      return a + b;
} tambah(4, 4); // mengembalikan 8
naikkan(10);    // mengembalikan 20
tambah(naikkan(5), 7); // mengembalikan 22
```

6.5. Pengenalan jQuery

jQuery adalah sebuah library Javascript yang dibuat oleh John Resig pada tahun 2006. jQuery memungkinkan manipulasi dokumen HTML dilakukan hanya dalam beberapa baris *code*. Beberapa fitur utama yang terdapat pada jQuery adalah:

- **DOM manipulation** – jQuery memungkinkan untuk memodifikasi DOM (*Document Object Model*) menggunakan *source selector* yang disebut dengan *Sizzle*.
- **Event Handling** – jQuery dapat menangani sebuah aksi pada dokumen HTML seperti saat pengguna melakukan *click* pada sebuah objek.
- **Ajax Support** – jQuery dapat memfasilitasi pembuatan *website* menggunakan teknologi AJAX.
- **Animations** – pada jQuery terdapat *build-in* animasi yang dapat digunakan pada halaman *web*.
- **Lightweight** – ukuran *file* jQuery sangat ringan yaitu sekitar 19KB.

jQuery dapat dengan mudah digunakan pada sebuah situs *web* dengan berbagai cara.

1. Instalasi Lokal

- Kunjungi link <https://jquery.com/download/> untuk mengunduh *library* jQuery.
- Letakkan *library* yang sudah diunduh pada satu folder yang sama dengan *file* HTML dengan kode berikut.

```
<html>

<head>
  <title>The jQuery Example</title>
  <script type="text/javascript" src="jquery-3.7.0.min.js">
  </script>

  <script type="text/javascript">
    $(document).ready(function() {
      document.write("Hello, World!");
    });
  </script>
</head>

<body>
  <h1>Hello</h1>
</body>

</html>
```

Note: pastikan atribut `src` memiliki nilai yang sama dengan nama *file library* jQuery.

- Buka *file* HTML tersebut menggunakan *web browser* seperti Mozilla atau Chrome. Dan hasil yang didapatkan adalah sebuah teks “Hello World” seperti yang ditulis pada bagian `document.write()`.

2. Menggunakan CDN (*Content Delivery Network*)

- Buka *file* HTML tersebut menggunakan *web browser* seperti Mozilla atau Chrome. Dan hasil yang didapatkan adalah sebuah teks “Hello World” seperti yang ditulis pada bagian `document.write()`.

<https://code.jquery.com/jquery-3.7.1.min.js>

- Buka *file* HTML menggunakan *web browser* dan hasil yang ditampilkan akan sama dengan cara instalasi lokal.

6.6. Kegunaan Lanjutan jQuery

6.6.1. Efek hide/show

Contoh *code*:

```

<!DOCTYPE html>
<html>

<head>
  <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
  <script>
    $(document).ready(function() {
      $("#hide").click(function() {
        $("p").hide();
      });
      $("#show").click(function() {
        $("p").show();
      });
    });
  </script>
</head>

<body>

  <p>If you click on the "Hide" button, I will disappear.</p>
  <button id="hide">Hide</button>
  <button id="show">Show</button>

</body>

</html>

```

Penjelasan *code*:

- Pada baris 4-5, dilakukan pemanggilan *library* jQuery menggunakan CDN.
- Pada baris 6-15, fungsi jQuery *hide/show* diaplikasikan pada *tag* html <p>. Apabila *button* dengan id 'hide' diklik, maka semua konten pada *tag* <p> akan disembunyikan. Selain itu, apabila *button* dengan id 'show' diklik, maka semua konten pada *tag* <p> akan dimunculkan.

6.6.2. Efek animasi

Contoh *code*:

```

<!DOCTYPE html>
<html>

<head>
  <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
  <script>
    $(document).ready(function() {
      $("button").click(function() {
        $("div").animate({
          height: 'toggle'
        });
      });
    });
  </script>
</head>

<body>

  <p>Click the button multiple times to toggle the animation.</p>
  <button>Start Animation</button>

```

```
<p>By default, all HTML elements have a static position, and cannot be moved.  
To manipulate the position, remember to first set the CSS position property of  
the element to relative, fixed, or absolute!</p>  
<div style="background:#98bf21; height:100px; width:100px; position:absolute;">  
</div>  
  
</body>  
  
</html>
```

Penjelasan *code*:

- Pada baris 6-15, fungsi jQuery menganimasikan *"toggle"* pada *tag* `<div>` yang terdapat pada *body* HTML berdasarkan tinggi elemen.

MODUL 7. Coding on the Spot 1 (COTS 1)

Tujuan Praktikum

1. Praktikan mampu membangun arsitektur web, pemrograman web dasar, teknologi yang mendukung pemrograman web, dan pemrograman web lanjut dengan framework.

7.1. Program Learning Outcomes

Menguasai metode dan proses analisis, perencanaan, pengelolaan, serta evaluasi yang berkaitan.

7.2. Course Learning Outcomes

Mahasiswa mampu merancang aplikasi dari studi kasus.

7.3. Komponen Penilaian

Komponen penilaian dari MONEV 1 adalah sebagai berikut:

NO	Komponen Penilaian	Nilai				
		0 Point	10 Point	15 Point	20 Point	30 Point
1	Menggunakan Framework Bootstrap	Tidak ada	Sebagian	Ada	-	-
2	3 Halaman (Form , table, CRUD)	Tidak ada	Sebagian halaman	Setengah halaman	Sebagian besar dari halaman	Lengkap
3	Poin 2 menggunakan Framework CI	Tidak ada	Ada	-	Jika menggunakan NodeJS	-
4	JQuery Plugin	Tidak ada	Sebagian halaman	Sebagian besar dari halaman	Lengkap	-
5	Menggunakan data JSON untuk table dan datatable (jQuery)	Tidak menggunakan	Sebagian halaman	Lengkap	-	-
6	Dokumentasi	Tidak ada	Ada	-	-	-

MODUL 8. Responsi Dan Evaluasi Tugas Besar 1

Tujuan Praktikum

1. Praktikan mampu membangun arsitektur web, pemrograman web dasar, teknologi yang mendukung pemrograman web, dan pemrograman web lanjut dengan framework.

8.1. Program Learning Outcomes

Menguasai metode dan proses analisis, perencanaan, pengelolaan, serta evaluasi yang berkaitan.

8.2. Course Learning Outcomes

Mahasiswa mampu merancang aplikasi dari studi kasus.

8.3. Komponen Penilaian

Komponen penilaian dari MONEV 1 adalah sebagai berikut:

NO	Komponen Penilaian	Nilai				
		0 Point	10 Point	15 Point	20 Point	30 Point
1	3 Halaman (Form, Data, Table)	Tidak lengkap	Sebagian halaman	Setengah halaman	Sebagian besar dari halaman	Lengkap
2	Menggunakan JQUERY, AJAX, JSON	Tidak ada	Sebagian halaman	Setengah halaman	Sebagian besar dari halaman	Lengkap
3	JQuery Plugin	Tidak ada	Sebagian halaman	Sebagian besar dari halaman	Ada	-
4	Slide	Tidak ada	Bagus	-	-	-
5	Presentasi (Individu)	Tidak Hadir	Bagus	-	-	-

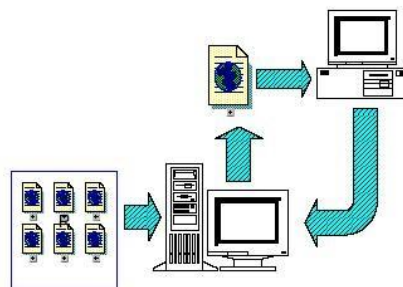
MODUL 9. PHP

Tujuan Praktikum

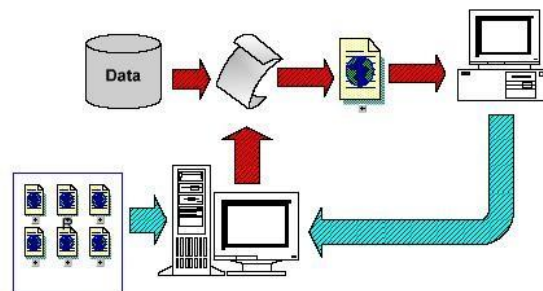
1. Mahasiswa mampu memahami konsep dan implementasi PHP pada *web*.
2. Mahasiswa mampu memahami sintaks, elemen, dan fungsi pada PHP.

9.1. Web Server dan Server Side Scripting

Web Server merupakan sebuah perangkat lunak dalam *server* yang berfungsi menerima permintaan (*request*) berupa halaman *web* melalui HTTP atau HTTPS dari *client* yang dikenal dengan *web browser* dan mengirimkan kembali (*response*) hasilnya dalam bentuk halaman-halaman *web* yang umumnya berbentuk dokumen HTML.



Gambar 9-1. Arsitektur web standar



Gambar 9-2. Arsitektur web dinamis

Beberapa *web server* yang banyak digunakan antara lain seperti berikut:

1. Apache Web Server (<https://httpd.apache.org/>)
2. Internet Information Service, IIS (<https://www.iis.net/>)
3. Xitami Web Server
4. Sun Java System Web Server

Server Side Scripting merupakan sebuah teknologi *scripting* atau pemrograman *web* dimana *script* (program) dikompilasi atau diterjemahkan di *server*. Dengan *server side scripting*, memungkinkan untuk menghasilkan halaman *web* yang dinamis.

Beberapa contoh *Server Side Scripting (Programming)* :

1. ASP (Active Server Page) dan ASP.NET
2. ColdFusion (<http://www.adobe.com/products/coldfusion-family.html>)
3. Java Server Pages (<http://www.oracle.com/technetwork/java/javasee/jsp/index.html>)

4. Perl (<https://www.perl.org/>)
5. Python (<https://www.python.org/>)
6. PHP (<http://www.php.net/>)

Keistimewaan PHP sebagai bahasa pemrograman berbasis *web* adalah :

1. Cepat
2. *Free*
3. Mudah dipelajari
4. *Multi-platform*
5. Dukungan *technical support*
6. Banyaknya komunitas PHP
7. Aman

9.2. Instalasi Apache, PHP dan MySQL dengan XAMPP

Proses instalasi Apache, PHP dan MySQL terkadang menjadi kendala tersendiri bagi yang ingin memulai belajar pemrograman *web*. Namun saat ini sudah tersedia banyak aplikasi paket yang mencakup ketiga aplikasi tersebut. Beberapa diantaranya adalah sebagai berikut:

1. XAMPP (versi windows) dan LAMPP (versi linux) yang dapat diunduh di <https://www.apachefriends.org/download.html>
2. WAMP Server
3. APPServ
4. PHPTriad

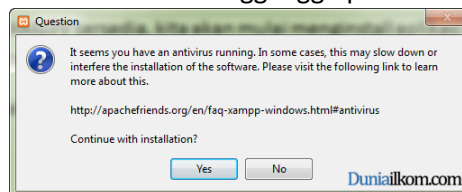
Pada modul ini, akan digunakan aplikasi paket XAMPP sebagai sarana pendukung dan pembelajaran pemrograman *web*.

a. Persiapan Instalasi

1. Pastikan komputer anda tidak terinstall *web server* lain seperti IIS atau PWS karena dapat menyebabkan bentrok dengan *web server* Apache yang akan dipasang. Namun jika anda tetap ingin mempertahankannya, setelah instalasi *web server* Apache selesai, anda dapat mengkonfigurasinya secara manual untuk mengganti nomor *port* yang akan digunakan oleh Apache.
2. *Download source* XAMPP versi 5.6.32 (yang akan digunakan di modul ini) pada <https://www.apachefriends.org/download.html> dan tersedia untuk sistem operasi Windows, Linux dan Mac.

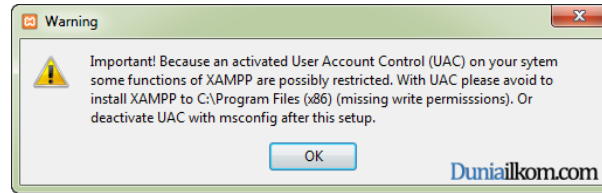
b. Proses Instalasi

1. Jalankan *file installer* XAMPP
2. Akan ditampilkan jendela instalasi XAMPP. Pilih **Yes**. Peringatan tersebut menunjukkan bahwa antivirus sedang berjalan dan tidak akan mengganggu proses instalasi.



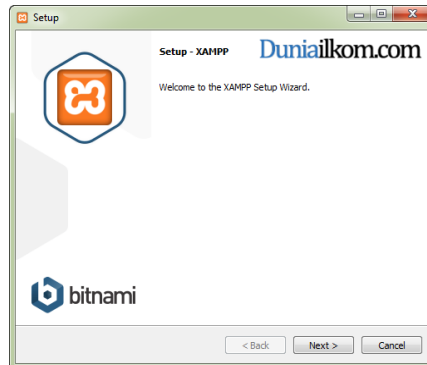
Gambar 9-3. Peringatan antivirus pada instalasi XAMPP

3. Setelah itu akan muncul jendela peringatan UAC (*User Account Control*). Klik **OK**.



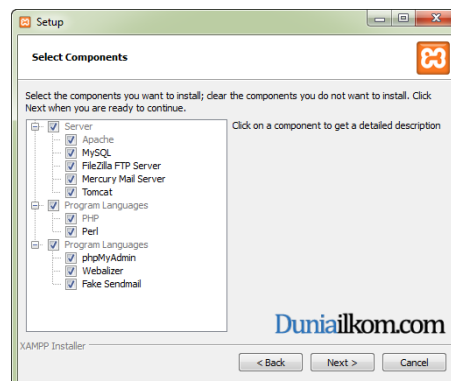
Gambar 9-4. Peringatan UAC pada instalasi XAMPP

4. Akan muncul jendela awal instalasi. Klik **Next** untuk melanjutkan instalasi.



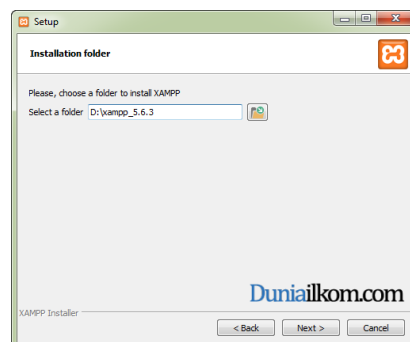
Gambar 9-5. Jendela awal instalasi XAMPP

5. Jendela berikutnya adalah "**Select Component**". Pada bagian ini kita bisa memilih aplikasi apa saja yang akan dipasang. Setelah selesai dipilih, klik **Next** untuk melanjutkan.



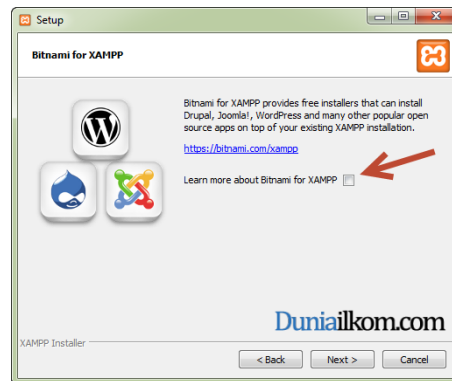
Gambar 9-6. Jendela pemilihan komponen pada XAMPP

6. Selanjutnya adalah memilih lokasi dimana XAMPP akan dipasang. Jika sudah ditentukan lokasinya, klik next **Next**.



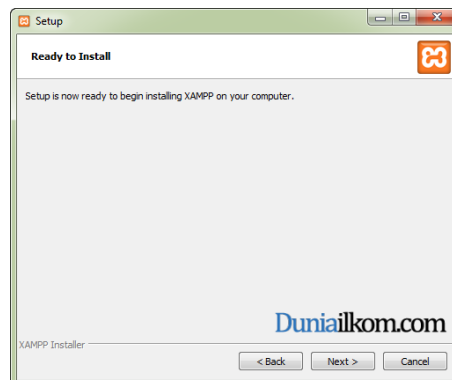
Gambar 9-7. Jendela pemilihan lokasi instalasi

7. Tampilan berikutnya adalah jendela **“Bitnami for XAMPP”**. Hilangkan centang pada bagian **“Learn more about Bitnami for XAMPP”**. Klik **Next**.



Gambar 9-8. Jendela instalasi Bitnami pada XAMPP

8. Jendela berikutnya adalah konfirmasi untuk mulai memasang XAMPP. Klik **Next** dan proses instalasi akan berlangsung.

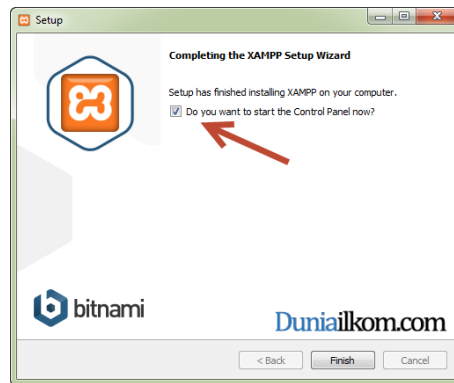


Gambar 9-9. Jendela konfirmasi instalasi



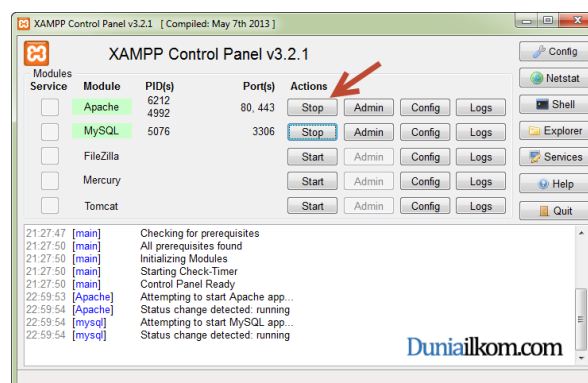
Gambar 9-10. Jendela proses instalasi

9. Jika jendela **“Completing the XAMPP Setup Wizard”** telah tampil, maka proses instalasi XAMPP telah selesai. Centang bagian **“Do you want to start the Control Panel now?”** untuk menjalankan XAMPP.



Gambar 9-11. Jendela instalasi XAMPP selesai

10. Akan muncul jendela Control Panel dari XAMPP. Pada bagian ini, kita bisa mengaktifkan *service* apa saja yang akan dijalankan dengan cara menekan tombol **Start** pada setiap *service* yang ada.



Gambar 9-12. Jendela ketika service pada XAMPP dijalankan

11. Untuk mengujinya, buka *web browser* dan ketikkan alamat **localhost** pada *address bar*, kemudian tekan **enter**. Akan tampil jendela XAMPP yang menunjukkan bahwa proses instalasi dan *service* Apache berjalan dengan baik.



Gambar 9-13. Tampilan ketika localhost dijalankan pada *browser*

9.3. Pengenalan PHP

Merupakan singkatan rekursif dari **PHP** : **H**ypertext **P**reprocessor. Pertama kali diciptakan oleh **Rasmus Lerdorf** pada tahun 1994. PHP sendiri harus ditulis diantara tag :

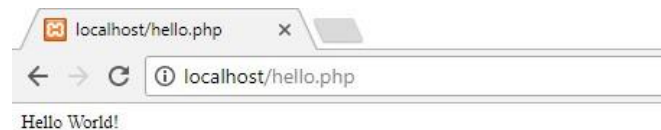
- `<? dan ?>`

- `<?php` dan `?>`
- `<script language="php">` dan `</script>`
- `<%` dan `%>`

Setiap satu *statement* (perintah) biasanya diakhiri dengan titik-koma (;). PHP juga **case sensitive** untuk nama *identifier* yang dibuat oleh *user* sedangkan *identifier* bawaan dari PHP **tidak case sensitive**. Contoh program yang ditulis dengan bahasa PHP

```
<?php echo "Hello World!"; ?>
```

Simpan *file* tersebut dengan nama **hello.php** pada direktori **htdocs** yang ada di folder XAMPP. Kemudian, jalankan pada *browser* dengan mengetikkan alamat <http://localhost/hello.php>. Hasilnya akan muncul di *web browser* seperti berikut:



Gambar 9-14. Tampilan ketika menjalankan file PHP pada browser

9.4. Variabel

Variabel digunakan untuk menyimpan sebuah *value* (nilai), data atau informasi. Nama variabel pada PHP diawali dengan tanda **\$**. Panjang dari suatu variabel tidak terbatas dan variabel tidak perlu dideklarasikan terlebih dahulu sebelumnya. Setelah tanda **\$**, dapat diawali dengan huruf atau *under-score* (_). Karakter berikutnya bisa terdiri dari huruf, angka dan atau karakter tertentu yang diperbolehkan (karakter ASCII dari 127 – 255).

Variabel pada PHP bersifat *case sensitive* artinya besar kecilnya suatu karakter berpengaruh pada variabel tersebut. Suatu karakter pada PHP tidak boleh mengandung spasi.

Berikut adalah contoh penggunaan variabel pada PHP:

```
<?php
    $nim = "1301165454";
    $nama = "Baharudin";

    echo "NIM : " . $nim;
    echo "Nama : " . $nama;
?>
```

Pada PHP, tipe data dari suatu variabel tidak didefinisikan langsung oleh *programmer*, akan tetapi secara otomatis akan ditentukan oleh interpreter PHP. Namun demikian, PHP mendukung 8 (delapan) buah tipe data primitif, yaitu:

1. *Boolean*
2. *Integer*
3. *Float*
4. *String*
5. *Array*
6. *Object*
7. *Resource*
8. *NULL*

9.5. Konstanta

Konstanta merupakan variabel konstan yang nilainya tidak berubah-ubah. Untuk mendefinisikan konstanta pada PHP, dapat menggunakan fungsi `define()` yang telah tersedia pada PHP. Berikut adalah contohnya :

```
<?php
define("NAMA" , "Baharuddin");
define("NIM" , "1301165454");
echo "Nama : " . NAMA;
echo "NIM : " . NIM;
?>
```

9.6. Operator dalam PHP

Ada beberapa jenis operator pada PHP, yaitu:

Tabel 8-1. Operator pada PHP

Jenis Operator	Operator	Contoh	Keterangan
Aritmatika	+	\$a + \$b	Pertambahan
	-	\$a - \$b	Pengurangan
	*	\$a * \$b	Perkalian
	/	\$a / \$b	Pembagian
	%	\$a % \$b	Modulus, sisa hasil bagi
Penugasan	=	\$a = 4	Variabel \$a akan diisi oleh 4
Bitwise	&	\$a & \$b	Bitwise AND
		\$a \$b	Bitwise OR
	^	\$a ^ \$b	Bitwise XOR
	~	~\$a	Bitwise NOT
	<<	\$a << \$b	Shift Left
	>>	\$a >> \$b	Shift Right
Perbandingan	==	\$a == \$b	Sama dengan
	===	\$a === \$b	Identik
	!=	\$a != \$b	Tidak sama dengan
	<>	\$a <> \$b	Tidak sama dengan
	!==	\$a !== \$b	Tidak identik
	<	\$a < \$b	Kurang dari
	>	\$a > \$b	Lebih dari
	<=	\$a <= \$b	Kurang dari sama dengan
	>=	\$a >= \$b	Lebih dari sama dengan
Logika	and	\$a and \$b	TRUE jika \$a dan \$b TRUE
	&&	\$a && \$b	TRUE jika \$a dan \$b TRUE

	or	\$a or \$b	TRUE jika \$a atau \$b TRUE
		\$a \$b	TRUE jika \$a atau \$b TRUE
	xor	\$a xor \$b	TRUE jika \$a atau \$b TRUE, tapi tidak keduanya
	!	!\$b	TRUE jika \$b FALSE
String	.	\$a . \$b	Penggabungan string \$a dan \$b

9.7. Struktur Kondisi

Struktur kondisi pada PHP sama halnya dengan bahasa pemrograman lainnya seperti Java. Berikut adalah contoh penulisan struktur kondisi *if-then* pada PHP:

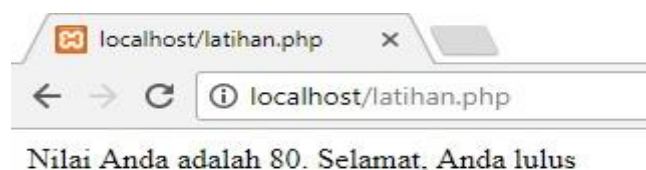
```
if (kondisi) {
    statement-jika-kondisi-TRUE;
} else {
    Statement-jika-kondisi-FALSE; }
```

Selain struktur kondisi *if-then*, terdapat pula struktur kondisi *switch-case* seperti berikut:

```
switch ($var) {
    case '1' : statement-1; break;
    case '2' : statement-2; break;
    . . . .
}
```

Berikut adalah contoh ketika *statement* kondisi *if-then* dijalankan :

```
$nilai = 80; if ($nilai > 50) {
    echo "Nilai Anda adalah " . $nilai . ". Selamat, Anda lulus";
} else {
    echo "Nilai Anda adalah " . $nilai . ". Maaf, Anda tidak lulus";
}
```



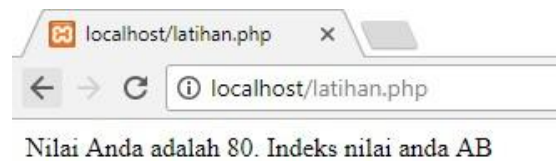
Gambar 9-15. Hasil penggunaan IF-THEN

Dan berikut ini adalah contoh ketika *statement* *switch-case* dijalankan:

```

$nilai = 80;
switch ($nilai) {
    case ($nilai > 50 && $nilai <= 60) : echo "Nilai Anda adalah " .
$nilai . ". Indeks nilai anda C"; break;
    case ($nilai > 60 && $nilai <= 70) : echo "Nilai Anda adalah "
.$nilai . ". Indeks nilai anda BC"; break;
    case ($nilai > 70 && $nilai <= 75) : echo "Nilai Anda adalah "
.$nilai . ". Indeks nilai anda B"; break;
    case ($nilai > 75 && $nilai <= 80) : echo "Nilai Anda adalah "
.$nilai . ". Indeks nilai anda AB"; break;
    case ($nilai > 80 && $nilai <= 100) : echo "Nilai Anda adalah "
.$nilai . ". Indeks nilai anda A"; break;
    default :
echo "Nilai Anda adalah " . $nilai . ". Maaf, Anda tidak lulus";
break; }

```



Gambar 9-16. Hasil penggunaan **switch-case**

9.8. Perulangan (Looping)

Banyak jenis perulangan yang terdapat pada PHP. Adapun beberapa diantaranya adalah :

1. Perulangan **for**

```

for (init_awal, kondisi, counter) {
    statement;
}

```

2. Perulangan **while**

```

init_awal; while (kondisi) {
    statement;
    counter;
}

```

3. Perulangan **do-while**

```

init_awal;
do {
    statement;    counter;
} while (kondisi);

```

4. Perulangan **foreach**

```

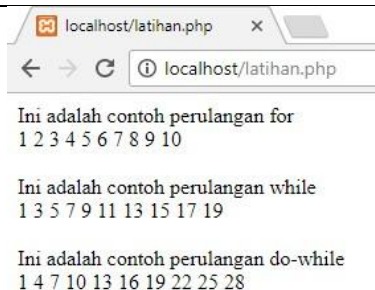
foreach (array_expression as $value) {

```

```
        statement;  
    }
```

Berikut adalah contoh penggunaan perulangan :

```
<?php  
echo "Ini adalah contoh perulangan for";  
echo "<br>";  
for ($i = 1; $i <= 10; $i++) {  
    echo $i . " ";  
}  
  
echo "<br>";  
echo "<br>";  
echo "Ini adalah contoh perulangan while";  
echo "<br>";  
$i = 1;  
while ($i <= 20) {  
    echo $i . " ";  
    $i += 2;  
}  
  
echo "<br>";  
echo "<br>";  
echo "Ini adalah contoh perulangan do-while";  
echo "<br>";  
$i = 1;  
do {  
    echo $i . " ";  
    $i += 3;  
} while ($i < 30);  
?>
```



Gambar 9-17. Hasil penggunaan perulangan

9.9. Function

Dalam merancang kode program, kadang kita sering membuat kode yang melakukan tugas yang sama secara berulang-ulang, seperti membaca tabel dari *database*, menampilkan penjumlahan, dan lainlain. Tugas yang sama ini akan lebih efektif jika dipisahkan dari program utama, dan dirancang menjadi sebuah **fungsi**. Fungsi dipanggil dengan menulis nama dari fungsi tersebut, dan diikuti dengan argumen (jika ada). Argumen ditulis di dalam tanda kurung, dan jika jumlah argumen lebih dari satu, maka diantaranya dipisahkan oleh karakter koma.

Bentuk umum pendefinisian fungsi pada PHP adalah sebagai berikut:

```
function nama_fungsi(parameter1, parameter2,  
.... , n) {
```

```
        statement;  
    }
```

Contoh fungsi pada PHP tanpa menggunakan parameter dan *return value*:

```
<?php  
function cetakGenap()  
{  
    for ($i = 1; $i <= 100; $i++) {  
        if ($i % 2 == 0) {  
            echo "$i ";  
        }  
    }  
}  
  
//pemanggilan fungsi  
cetakGenap();  
?>
```

Contoh fungsi pada PHP menggunakan parameter dan tanpa *return value*:

```
<?php  
function cetakGenap($awal, $akhir)  
{  
    for ($i = $awal; $i <= $akhir; $i++) {  
        if ($i % 2 == 0) {  
            echo "$i ";  
        }  
    }  
}  
  
//pemanggilan fungsi  
$a = 10;  
$b = 50;  
echo "Bilangan ganjil dari $a sampai $b adalah : <br>";  
cetakGenap($a, $b);  
?>
```

Contoh fungsi pada PHP dengan *return value*:

```
<?php function luasSegitiga($alas, $tinggi)  
{  
    return 0.5 * $alas * $tinggi;  
}  
  
//pemanggilan fungsi  
$a = 10;  
$t = 50;  
echo "Luas Segitiga dengan alas $a dan tinggi $t adalah : " . luasSegitiga($a,  
$t);  
?>
```


9.10. Array

Array merupakan tipe data terstruktur yang berguna untuk menyimpan sejumlah data yang bertipe sama. Bagian yang menyusun *array* disebut elemen *array*, yang masing-masing elemen dapat diakses tersendiri melalui *index array*. *Index array* dapat berupa bilangan *integer* atau *string*.

Untuk mendeklarasikan atau mendefinisikan sebuah *array* di PHP bisa menggunakan keyword `array()`. Jumlah elemen *array* tidak perlu disebutkan saat deklarasi. Sedangkan untuk menampilkan isi *array* pada elemen tertentu, cukup dengan menyebutkan nama *array* beserta *index array*-nya.

Berikut adalah cara mendeklarasikan suatu *array* di PHP :

```
<?php
$arrKendaraan = ["Mobil", "Pesawat", "Kereta Api", "Kapal Laut"];
echo $arrKendaraan[0] . "<br>"; //Mobil
echo $arrKendaraan[2] . "<br>"; //Kereta Api

$arrKota = [];
$arrKota[] = "Jakarta";
$arrKota[] = "Medan";
$arrKota[] = "Bandung";
$arrKota[] = "Malang";
$arrKota[] = "Sulawesi";

echo $arrKota[1] . "<br>"; //Medan
echo $arrKota[2] . "<br>"; //Bandung
echo $arrKota[4] . "<br>"; //Sulawesi
?>
```

Cara mendeklarasikan suatu *array* pada PHP bisa dengan *index string* atau yang dinamakan dengan *array* asosiatif. Berikut adalah contoh pendeklarasian *array* asosiatif :

```
<?php
$arrAlamat = [
    "Rona" => "Banjarmasin",
    "Dhiva" => "Bandung",
    "Ilham" => "Medan",
    "Oku" => "Hongkong",
];

echo $arrAlamat["Dhiva"] . "<br>"; //Bandung
echo $arrAlamat['Oku'] . "<br>"; //Hongkong

$arrNim = [];
$arrNim["Rona"] = "11011112";
$arrNim["Dhiva"] = "11011101";
$arrNim["Ilham"] = "11011309";
$arrNim["Oku"] = "11014765";
$arrNim["Fadhlan"] = "11011113";

echo $arrNim["Ilham"] . "<br>"; //11011309
echo $arrNim['Fadhlan'] . "<br>"; //11011113
?>
```

MODUL 10. AJAX

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep dan implementasi AJAX.
2. Mahasiswa mampu memahami sintaks AJAX.

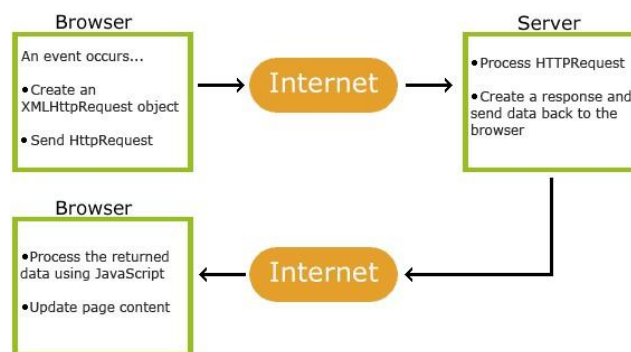
10.1. Apa Itu AJAX

AJAX (*Asynchronous JavaScript and XML*) suatu teknik pemrograman berbasis *web* untuk menciptakan aplikasi *web* interaktif. Tujuannya adalah untuk memindahkan sebagian besar interaksi pada komputer *user*, melakukan pertukaran data dengan *server* di belakang layar, sehingga halaman *web* tidak harus dibaca ulang secara keseluruhan setiap kali seorang pengguna melakukan perubahan. Hal ini akan meningkatkan interaktivitas, kecepatan, dan *usability*.

Secara umum, AJAX melibatkan dua hal yakni:

1. Objek XMLHttpRequest bawaan *browser* (untuk meminta data dari sebuah *web server*).
2. Javascript dan HTML DOM (untuk menampilkan data pada *web browser*).

10.2. Cara Kerja AJAX



Gambar 10-1. Cara kerja AJAX

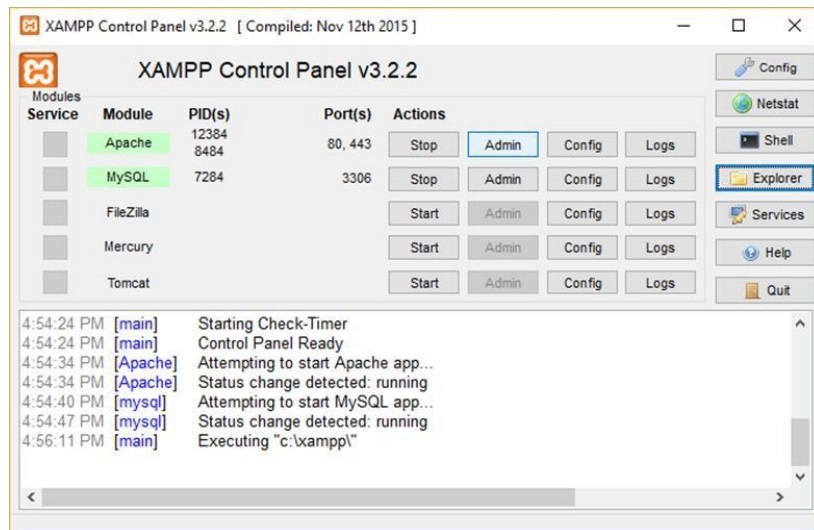
Dalam aplikasinya, AJAX melakukan hal-hal berikut:

1. Suatu *event* terjadi pada halaman *web* (seperti *page loaded* atau *button clicked*).
2. Sebuah objek XMLHttpRequest dibuat oleh Javascript
3. Objek XMLHttpRequest mengirimkan *request* kepada *web server*.
4. *Web server* mengelola *request*.
5. *Web server* mengirimkan *response* kepada *client*.
6. *Response* dibaca oleh Javascript.
7. Javascript melakukan perubahan pada halaman *web* menggunakan DOM.

10.3. Event Handling

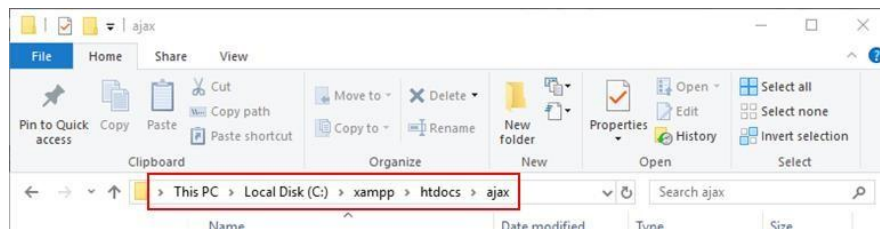
Pada contoh berikut, akan dilakukan perubahan halaman *web* menggunakan teknik AJAX. Berikut langkah-langkah yang perlu dilakukan:

1. Pastikan PHP *web server* sudah berjalan dengan baik. Pada modul ini digunakan Apache *web server* yang terdapat pada XAMPP v3.2.2.



Gambar 10-2. Tampilan control panel XAMPP

2. Akses folder `htdocs` pada *local server*, dan kemudian buat folder baru dengan nama seperti: *ajax*



Gambar 10-3. Folder penyimpanan file AJAX

3. Buat *file* `.txt` berikut yang berfungsi sebagai pengganti konten halaman *web*.

```
<h1>AJAX</h1>
<p>AJAX is not a programming language.</p>
<p>AJAX is a technique for accessing web servers from a web
page.</p>
<p>AJAX stands for Asynchronous JavaScript And XML.</p>
```

Simpan *file* sebagai `ajax_info.txt`.

4. Buat juga *file* HTML utama yang berisikan *code* sebagai berikut. Dan simpan sebagai `index.html`

```
<!DOCTYPE html>
<html>

<body>

  <h2>The XMLHttpRequest Object</h2>

  <p id="demo">Let AJAX change this text.</p>

  <button type="button" onclick="loadDoc()">Change Content</button>

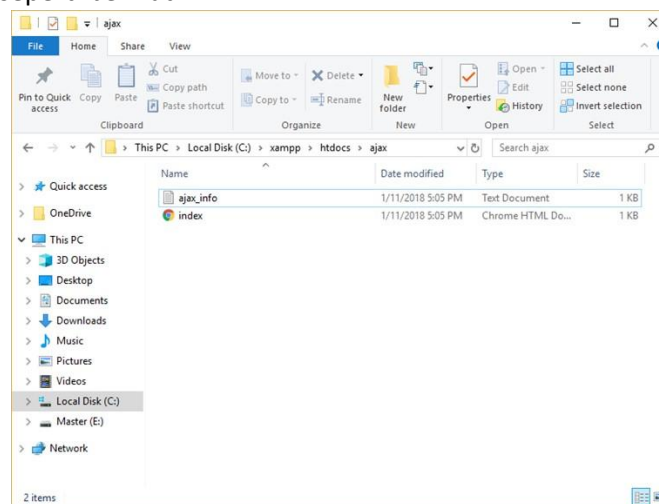
  <script>
    function loadDoc() {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
          document.getElementById("demo").innerHTML = this.responseText;
        }
      };
      xhttp.open("GET", "ajax_info.txt", true);
      xhttp.send();
    }
  </script>
</body>
</html>
```

```

    }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
}
</script>
</body>
</html>

```

- Tempatkan kedua *file* tersebut kedalam folder ajax yang telah dibuat pada langkah kedua sehingga posisi kedua *file* seperti berikut.



Gambar 10-4. File ajax_info pada folder penyimpanan file AJAX

- Ketika anda mengakses halaman *web* tersebut pada alamat <http://localhost/ajax/>, tampilan yang akan muncul adalah seperti gambar 5-7.



Gambar 10-5. Tampilan file AJAX ketika dieksekusi

- Namun jika anda melakukan *action* yaitu menekan *button Change Content*, maka konten pada halaman *web* akan menjadi seperti ini.



Gambar 10-6. Tampilan file AJAX-2

Berikut adalah penjelasannya:

1. Peran terbesar AJAX pada contoh diatas adalah pada *code* Javascript yang terdapat di *file* index.html.

```
<script>
function loadDoc() {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById("demo").innerHTML =
this.responseText;
        }
    };
    xmlhttp.open("GET", "ajax_info.txt", true);
    xmlhttp.send();
}
</script>
```

Code tersebut akan dieksekusi apabila *button Change Content* ditekan karena atribut `onClick()` yang terdapat pada *button* tersebut.

2. Pembuatan objek `XMLHttpRequest()` dilakukan pada *variable* `xhttp`.
3. Selanjutnya pada fungsi `onreadystatechange()` yang terdapat pada objek `XMLHttpRequest` dilakukan pengecekan apakah *request* dapat dilakukan.
4. Jika pengecekan pada langkah ke-3 berhasil, maka dilakukan perubahan isi konten dengan atribut `id="demo"` pada bagian *body* HTML menjadi *response* yang didapat pada baris selanjutnya.
5. Objek `XMLHttpRequest` mengeksekusi *method* `open()` untuk mengirim *request* kepada *web server*. Parameter yang terdapat pada *method* `open()` adalah sebagai berikut:

<code>open(method, url, async, user, psw)</code>	Specifies the request
	<i>method</i> : the request type GET or POST
	<i>url</i> : the file location
	<i>async</i> : true (asynchronous) or false (synchronous)
	<i>user</i> : optional user name
	<i>psw</i> : optional password

Gambar 10-7. Parameter pada salah satu fungsi AJAX

6. Sehingga yang dilakukan pada *code* secara *asynchronous* kemudian mengirimkannya ke *server*.

```
xhttp.open("GET","ajax_info.txt", true);
xhttp.send();
```

7. Jika *file* yang diminta terdapat pada *server*, maka konten akan berubah.

MODUL 11. Laravel : Introduction

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep dan implementasi MVC menggunakan *web framework* Laravel.

11.1 Framework & MVC

Framework atau dalam Bahasa Indonesia dapat diartikan sebagai “kerangka kerja” merupakan kumpulan dari fungsi-fungsi/prosedur-prosedur dan *class-class* untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang *programmer*, tanpa harus membuat fungsi atau *class* dari awal.

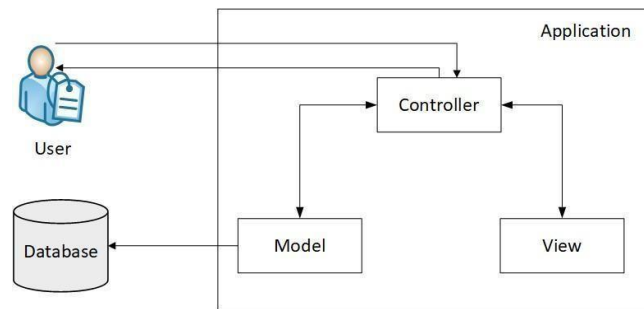
Alasan mengapa menggunakan *Framework*

- Mempercepat dan mempermudah pembangunan sebuah aplikasi *web*.
- Relatif memudahkan dalam proses *maintenance* karena sudah ada pola tertentu dalam sebuah *framework* (dengan syarat *programmer* mengikuti pola standar yang ada).
- Umumnya *framework* menyediakan fasilitas-fasilitas yang umum dipakai sehingga kita tidak perlu membangun dari awal (misalnya validasi, ORM, *pagination*, *multiple database*, *scaffolding*, pengaturan *session*, *error handling*, dll).
- Lebih bebas dalam pengembangan jika dibandingkan CMS.

Model-View-Controller (MVC) merupakan suatu konsep yang cukup populer dalam pembangunan aplikasi *web*. MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, *user interface*, dan bagian yang menjadi kontrol aplikasi. Terdapat 3 jenis komponen yang membangun suatu MVC *pattern* dalam suatu aplikasi yaitu:

- *View*, merupakan bagian yang menangani *presentation logic*. Pada suatu aplikasi *web* bagian ini biasanya berupa *file template* HTML, yang diatur oleh *controller*. *View* berfungsi untuk menerima dan merepresentasikan data kepada *user*. Bagian ini tidak memiliki akses langsung terhadap bagian *model*.
- *Model*, biasanya berhubungan langsung dengan *database* untuk memanipulasi data (*insert*, *update*, *delete*, *search*), menangani validasi dari bagian *controller*, namun tidak dapat berhubungan langsung dengan bagian *view*.
- *Controller*, merupakan bagian yang mengatur hubungan antara bagian *model* dan bagian *view*, *controller* berfungsi untuk menerima *request* dan data dari *user* kemudian menentukan apa yang akan diproses oleh aplikasi.

Singkat kata **Model** untuk mengatur alur *database*, **View** untuk menampilkan *web*, sedangkan **Controller** untuk mengatur alur kerja antara *Model* dan *View*. Jadi misalnya Anda akan membuat akun *e-mail*. Pertama anda akan melihat tampilan *sign-up/register*, itulah yang disebut dengan *View*. Kemudian Anda mengisi *form username*, *password*, dan lain-lain dan Anda klik tombol *Register*, maka disinilah *View* akan memanggil *Controller* dan *Controller* memanggil *Model*. Adapun tugas *Model* disini untuk mengecek apakah Anda sudah mengisi sesuai dengan kriteria dan akan dihubungkan dengan *database*. Kemudian *Model* akan mengembalikan ke *Controller* dan *Controller* akan mengembalikan ke *View*. Berikut adalah gambaran konsep MVC yang diterapkan pada *framework*.



Gambar 11.1 Cara kerja MVC

11.2 Pengenalan Laravel

Laravel adalah sebuah web application framework yang bersifat open-source yang digunakan untuk membangun aplikasi php dinamis. Laravel menjadi sebuah framework PHP dengan model MVC (Model, View, Controller) yang dapat mempercepat pengembang untuk membuat sebuah aplikasi web. Selain ringan dan cepat, Laravel juga memiliki dokumentasi yang lengkap disertai dengan contoh implementasi kodenya.

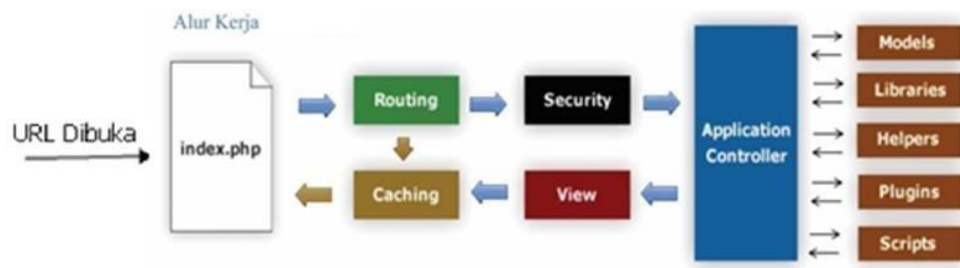
Laravel pertama kali dikembangkan pada tahun 2011 oleh Taylor Otwell. Saat ini Laravel sudah mencapai versi 9 yang dirilis pada tanggal 8 Februari 2022. Sebagai web framework populer yang menggunakan bahasa pemrograman PHP, Laravel mempunyai beberapa keunggulan yaitu:

1. *Free*, karena berada di bawah lisensi open source, kita dapat melakukan apa pun.
2. Menggunakan kaidah MVC, dengan menggunakan *Model-View-Controller*, kita dapat memisahkan bagian *logic* dan *presentation* dari aplikasi yang kita bangun.
3. Menghasilkan URL yang bersih. URL yang dihasilkan oleh Codeigniter bersih dan ramah terhadap *search engine*. Codeigniter menggunakan pendekatan *segment-based* dibandingkan dengan *query string* yang biasa digunakan oleh programmer yang tidak menggunakan *web framework*.
4. *Packs a Punch*, Laravel hadir dengan berbagai *library* yang akan membantu tugas-tugas di pengembangan web yang sudah umum dan sering dilakukan, seperti mengakses *database*, mengirim email, validasi data dari form, mengelola *session*, memanipulasi file, dan masih banyak lagi.
5. *Extensible*, kita dapat menambahkan *library* atau *helper* yang kita ciptakan sendiri ke dalam Laravel. Selain itu, kita dapat juga menambahkan fitur lewat *class extension*.
6. *Thoroughly Documented*, hampir semua fitur, *library*, dan *helper* yang ada di Laravel telah terdokumentasi dengan lengkap dan tersusun dengan baik. Dokumentasi cara penggunaannya dapat dilihat di <https://laravel.com/docs/9.x>.

Berikut adalah istilah yang sering ditemui di Laravel.

1. *Model*, *class* PHP yang dirancang untuk bekerja dengan informasi dari *database*.
2. *Controller*, inti aplikasi yang menentukan penanganan logic dari aplikasi web
3. *Route*, bagian yang menangani HTTP *request*.
4. *View*, halaman web seperti *header*, *footer*, *sidebar*, dan sebagainya yang ditanamkan di halaman web. *View* tidak pernah dipanggil secara langsung, tetapi harus dipanggil dari *controller*.
5. *Library*, *class* yang berisi fungsi-fungsi untuk penyelesaian kasus tertentu.

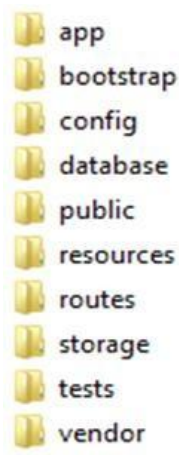
11.3 Cara Kerja Laravel



Gambar 11.2 Alur kerja Laravel

Ketika suatu URL dibuka, maka file `index.php` akan dibaca. Di dalam file ini, Laravel memeriksa apakah URL tersebut terdaftar pada Routing yang sudah dibuat. Jika ada pemetaannya dan memenuhi Security (jika diharuskan, autentikasi misalnya) maka Laravel akan memanggil Controller sesuai yang dipetakan oleh Routing. Di dalam Controller semua logika back-end dieksekusi, lalu dapat mengembalikan View (tampilan) ke browser. Tampilan ini disimpan dalam Caching agar pemrosesan setelahnya lebih cepat.

Sedangkan struktur folder Laravel dapat dilihat pada gambar di bawah ini:



Gambar 11.3 Struktur Folder Laravel

- **app**
Folder ini berisi Controller, Model, Middleware, dan Provider
- **bootstrap**
Folder ini berisi Cache untuk mempercepat pemrosesan
- **config**
Folder ini berisi semua file konfigurasi untuk aplikasi
- **database**
Folder ini berisi file-file migrasi dari/ke database
- **public**
Folder ini berisi file yang dapat diakses langsung, biasanya file asset (gambar, CSS, dan JS) atau file konten (file untuk di-download)

- resources
Folder ini berisi Routing untuk pemetaan URL ke aplikasi
- storage
Folder ini berisi hasil kompilasi View dan log dari aplikasi
- tests
Folder ini berisi file-file untuk unit testing
- vendor
Folder ini berisi file-file library yang dibutuhkan aplikasi

11.3.1 Routing

Pengaturan Routing berada pada file `routes/web.php`. Disini kita dapat mengatur pemetaan URL dengan aksi yang ingin kita lakukan. Method nya pun dapat diatur jika hanya untuk menerima method tertentu saja (post, get, atau yang lainnya). Contoh:

```
use App\Http\Controllers\SiteController;
use App\Http\Controllers\ProductController;

// Menampilkan halaman welcome.blade.php
Route::get('/', function () {
    return view('welcome');
});

// Alternatif menggunakan Route::view
Route::view('/', 'welcome');

// Memanggil fungsi dari Controller
Route::post('/auth', [SiteController::class, 'auth']);

// Menggunakan route resource untuk ProductController
Route::resource('products', ProductController::class);

// Jika Anda ingin mendefinisikan route secara individual
Route::get('/products', [ProductController::class, 'index']);
```

Routing pun bisa diatur dengan parameter, middleware, group, maupun dengan prefix.

11.3.2 View

Secara default file tampilan pada Laravel menggunakan template engine, yaitu Blade. Sehingga, setiap file tampilan yang kita buat harus mengikuti penamaan **namaview.blade.php** dan diletakkan pada folder **resources/views** agar dapat di-load oleh fungsi **view()**. Penggunaan Blade juga menyediakan banyak directive yang sangat berguna terutama untuk looping data dan templating halaman agar kita tidak menulis kode yang sama berulang kali (biasanya header, menu samping, dan footer memiliki tampilan yang sama untuk tiap halaman). Secara isi, tampilan pada Laravel tetap menggunakan HTML, CSS, dan Javascript. Hal yang harus diperhatikan yaitu jika ingin menggunakan file eksternal CSS / Javascript, maka harus menggunakan fungsi **{{ asset('path file di folder public') }}**, dan khusus untuk halaman yang berisi form dengan method selain GET maka harus menambahkan directive **@csrf** untuk

keamanan pengiriman form. Berikut adalah contoh tampilan halaman login yang menggunakan file eksternal jquery.js yang diletakkan di dalam folder **assets**:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
  <script src="{{ asset('js/jquery.min.js') }}"></script>
</head>
<body class="bg-light">
  <div class="container">
    <div class="row justify-content-center align-items-center min-vh-100">
      <div class="col-md-4">
        <div class="card">
          <div class="card-body">
            <h3 class="card-title text-center mb-4">Login</h3>
            <form method="POST" action="{{ route('auth') }}">
              @csrf
              <div class="mb-3">
                <label for="email" class="form-label">Email</label>
                <input type="email" name="email" id="email"
class="form-control" required>
              </div>
              <div class="mb-3">
                <label for="password" class="form-label">Password</label>
                <input type="password" name="password"
id="password" class="form-control" required>
              </div>
              <div class="d-grid">
                <button type="submit" class="btn btn-
primary">Login</button>
              </div>
            </form>
          </div>
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```

11.3.3 Controller

File Controller diletakkan pada folder **app/Http/Controllers**. Cara membuatnya bisa dengan cara pembuatan file manual dengan memperhatikan hal-hal berikut: meng-extend base Controller dari Laravel atau turunannya dan dituliskan namespace “App\Http\Controllers” agar dapat digunakan oleh file lain dan dipetakan dari Routing. Atau cara lain dengan perintah di command prompt / console / terminal. Perintahnya:

```
php artisan make:controller SiteController
```

Jika Controller akan dipakai untuk manajemen data dengan Database, maka perintah di atas dapat ditambahkan `--resource` sehingga fungsi-fungsi default yang akan digunakan untuk manajemen data di-generate secara otomatis oleh Laravel.

```
php artisan make:controller ProductController --resource
```

Kelebihan lain dari menggunakan resource ini adalah kita dapat menambah satu Routing saja untuk menangani semua aksi dari satu Controller. Di `web/routes.php` kita cukup menambahkan:

```
Route::resource('product', ProductController::class);
```

Routing akan secara otomatis menangani URL berikut ini:

Tabel 11.1 URL yang ditangani otomatis

Method	URL	Fungsi yang dieksekusi
GET	/product	index
GET	/product/create	create
POST	/product	store
GET	/product/{id}	show
GET	/product/{id}/edit	edit
PUT/PATCH	/product/{id}	update
DELETE	/product/{id}	destroy

Berikut adalah hasil generate dari `ProductController`:

```

<?php

namespace App\Http\Controllers;

use App\Models\Product;
use Illuminate\Http\Request;

class ProductController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        //
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        //
    }

    /**
     * Display the specified resource.
     */
    public function show(Product $product)
    {
        //
    }

    /**
     * Show the form for editing the specified resource.
     */
    public function edit(Product $product)
    {
        //
    }
}

```

```
/**
 * Update the specified resource in storage.
 */
public function update(Request $request, Product $product)
{
    //
}

/**
 * Remove the specified resource from storage.
 */
public function destroy(Product $product)
{
    //
}
}
```

MODUL 12. Laravel : Database 1

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep dan implementasi CodeIgniter pada pembuatan aplikasi berbasis *web*.
2. Mahasiswa mampu menerapkan CRUD menggunakan Laravel dan konsep MVC.
3. Mahasiswa mampu mengimplementasikan fitur-fitur yang sering digunakan pada Laravel.

12.1 CRUD

Pada modul ini kita akan membuat aplikasi yang dapat melakukan create, read, update, dan delete terhadap suatu data/tabel. Aplikasi yang akan kita bangun yaitu aplikasi e-commerce dimana kita akan fokus melakukan CRUD terhadap tabel produknya saja.

12.1.1 Konfigurasi dan Skema

Hal pertama yang harus dilakukan agar aplikasi dapat terhubung dengan database adalah mengatur konfigurasi koneksi. Secara detail, konfigurasi database berada pada file **config/database.php**. Tetapi untuk konfigurasi standar, kita dapat mengubahnya di file **.env** pada baris yang berisi **DB_CONNECTION** hingga **DB_PASSWORD**. Ubah nilainya dengan pengaturan yang kita inginkan.

```
...
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=ecommerce
DB_USERNAME=root
DB_PASSWORD=
DB_CHARSET=utf8mb4
DB_COLLATION=utf8mb4_unicode_
ci
...
```

Buat database bernama **ecommerce** di DBMS. Untuk membuat tabelnya, dapat dilakukan secara otomatis (di-generate) oleh Laravel dari command prompt / console / terminal dengan perintah. Perintah yang pertama yaitu membuat file migrasi sebagai skema dari tabel:

```
php artisan make:migration create_products_table
```

File migrasi **xxx_create_products_table.php** akan terbuat pada folder **database/migrations**. Edit file ini pada fungsi **Schema::create()** untuk mendefinisikan kolom yang kita inginkan.

```
Schema::create('products', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->integer('price');
    $table->timestamps();
});
```

Penjabaran dari kode ini:

- Fungsi **id()** yaitu kita mendefinisikan kolom **id** sebagai PK dengan tipe int dan auto-increment.
- Fungsi **string('name')** yaitu kita mendefinisikan kolom **name** dengan tipe varchar
- Fungsi **integer('price')** yaitu kita mendefinisikan kolom **price** dengan tipe int
- Fungsi **timestamps()** yaitu kita mendefinisikan kolom **created_at** yang akan terisi jika data

dibuat melalui ORM dan **updated_at** yang akan terisi jika data diubah melalui ORM

Untuk membuat tabel ke database dari skema ini, maka ketikkan perintah berikut pada command prompt / console / terminal:

```
php artisan migrate
```

12.1.2 Model

Laravel memberikan tiga cara untuk mengakses ataupun manipulasi data ke database: query langsung, query builder, dan ORM. Query langsung dan query builder menggunakan library Illuminate (**Illuminate\Support\Facades\DB**) sedangkan ORM menggunakan Eloquent, yang merupakan kelas extend dari Illuminate. File Model diletakkan pada folder **app/Models**. Untuk membuat Model dalam Laravel, dapat dilakukan dengan manual dengan menambahkan namespace "App\Models" dan meng-extend kelas base Model dari Eloquent (**Illuminate\Database\Eloquent\Model**) ataupun di-generate oleh Laravel dari command prompt / console / terminal dengan perintah:

```
php artisan make:model Product
```

Jika sebelumnya belum membuat Controller atau file migration-nya, perintah generate Model ini dapat ditambahkan parameter sehingga bisa sekaligus me-generate file Controller-nya (-c), file migration-nya (-m), ataupun keduanya langsung (-cm).

Eloquent memiliki beberapa konvensi / aturan yang harus diperhatikan yaitu:

- Nama sebuah Model "X" secara otomatis merepresentasikan tabel database bernama "xs". Contoh, Model Product akan merepresentasikan tabel **products**. Jika tabel yang direpresentasikan berbeda, maka harus ditambahkan atribut **protected \$table = 'nama_tabel'**; pada kelas Model.
- Kolom PK pada tabel bernama "id". Jika kolom PK bukan "id", maka harus ditambahkan atribut **protected \$primaryKey = 'nama_kolom_PK'**; pada kelas Model.
- Kolom PK pada tabel di-set auto-increment. Jika kolom PK tidak auto-increment, maka harus ditambahkan atribut **public \$incrementing = false**; pada kelas Model.
- Kolom PK pada tabel bertipe integer. Jika kolom PK bukan integer, maka harus ditambahkan atribut **protected \$keyType = 'string'**; pada kelas Model.
- Ada kolom "created_at" dan "updated_at" pada tabel. Jika tidak ada, maka harus ditambahkan atribut **public \$timestamps = false**; pada kelas Model.

Beberapa aturan lain juga dapat diatur seperti date format, koneksi DB yang berbeda, dan nilai default untuk atribut tertentu.

12.1.3 Controller

Setelah Model siap, kita tinggal memanggilnya pada Controller. Berikut kode lengkapnya dalam ProductController:

```
<?php

namespace App\Http\Controllers;

use App\Models\Product;
use Illuminate\Http\Request;

class ProductController extends Controller
{
    public function index()
```



```

{
    $products = Product::all();
    return view('products.index', ['products' => $products]);
}

public function create()
{
    return view('products.form', [
        'title' => 'Tambah',
        'product' => new Product(),
        'route' => route('products.store'),
        'method' => 'POST',
    ]);
}

public function store(Request $request)
{
    $validated = $request->validate([
        'name' => 'required|min:4',
        'price' => 'required|integer|min:1000000',
    ]);

    Product::create($validated);
    return redirect()->route('products.index')->with('success', 'Produk berhasil
ditambahkan');
}

public function show(Product $product)
{
    return view('products.show', compact('product'));
}

public function edit(Product $product)
{
    return view('products.form', [
        'title' => 'Edit',
        'product' => $product,
        'route' => route('products.update', $product),
        'method' => 'PUT',
    ]);
}

public function update(Request $request, Product $product)
{
    $validated = $request->validate([
        'name' => 'required|min:4',
        'price' => 'required|integer|min:1000000',
    ]);

    $product->update($validated);
    return redirect()->route('products.index')->with('success', 'Produk berhasil
diperbarui');
}

public function destroy(Product $product)
{
    $product->delete();
    return redirect()->route('products.index')->with('success', 'Produk berhasil
dihapus');
}
}

```

12.1.4 View

Untuk tampilannya, buat file dengan nama **index.blade.php** yang diletakkan di folder **resources/views/product** (folder product dibuat dahulu agar rapi).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Daftar Produk</title>
    <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="container">
    <div class="row justify-content-center mt-5">
        <div class="col-md-8">
            @if(session('success'))
                <div class="alert alert-success">{{ session('success') }}</div>
            @endif
            <div class="d-flex justify-content-between align-items-center mb-3">
                <span>{{ session('msg') }}</span>
                <a href="{{ route('products.create') }}" class="btn btn-
primary">Tambah</a>
            </div>
            <table class="table table-bordered table-striped">
                <thead>
                    <tr>
                        <th>Nama</th>
                        <th>Harga</th>
                        <th>Aksi</th>
                    </tr>
                </thead>
                <tbody>
                    @foreach($products as $product)
                        <tr>
                            <td>{{ $product->name }}</td>
                            <td>{{ $product->price }}</td>
                            <td>
                                <a href="{{ route('products.edit', $product->id) }}"
class="btn btn-sm btn-primary">Edit</a>
                                <form method="POST" action="{{ route('products.destroy',
$product->id) }}" style="display:inline" onsubmit="return confirm('Yakin hapus?')">
                                    @csrf
                                    @method('DELETE')
                                    <button class="btn btn-sm btn-danger">Hapus</button>
                                </form>
                            </td>
                        </tr>
                    @endforeach
                </tbody>
            </table>
        </div>
    </div>
    <script
src="https://cdn.jsdelivrivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></
script>
</body>
</html>
```

Dan untuk tampilan form tambah dan edit, dapat dibuat satu halaman saja karena hampir sama. Hanya perlu tambahan pengkondisian di beberapa tempat.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Form {{ $title }} Produk</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="container">
  <div class="row justify-content-center mt-5">
    <div class="col-md-6">
      <h4>Form {{ $title }} Produk</h4>
      <form class="border p-4" method="POST" action="{{ $route }}">
        @csrf
        @if($method === 'PUT')
          @method('PUT')
        @endif
        <div class="mb-3">
          <label for="name" class="form-label">Nama</label>
          <input type="text" name="name" id="name" class="form-control
@error('name') is-invalid @enderror" value="{{ old('name', $product->name) }}">
          @error('name')
            <div class="invalid-feedback">{{ $message }}</div>
          @enderror
        </div>
        <div class="mb-3">
          <label for="price" class="form-label">Harga</label>
          <input type="number" name="price" id="price" class="form-
control @error('price') is-invalid @enderror" value="{{ old('price', $product-
>price) }}">
          @error('price')
            <div class="invalid-feedback">{{ $message }}</div>
          @enderror
        </div>
        <div class="text-center">
          <button type="submit" class="btn btn-
success">Simpan</button>
        </div>
      </form>
    </div>
  </div>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.j
s"></script>
</body>
</html>
```

12.1.5 Tampilan Halaman

Jalankan aplikasi. Jika *kode* yang dibuat sesuai dengan semua gambar di atas pada modul ini, maka tampilan aplikasi *webnya* akan seperti gambar dibawah ini.

1. <http://localhost:8000/products>



Gambar 12.1 Tampilan halaman view

2. <http://localhost:8000/products/create>

Form Tambah Produk

Nama

Harga

Gambar 12.2 Tampilan halaman form tambah produk

3. <http://localhost:8000/products>

The screenshot shows the same web interface as before, but the table now contains one row of data. The 'Aksi' column has two buttons: 'Edit' (blue) and 'Hapus' (red). A blue 'Tambah' button is still in the top right corner.

Nama	Harga	Aksi
Laptop	10.000.000	<input type="button" value="Edit"/> <input type="button" value="Hapus"/>

Gambar 12.3 Tampilan halaman view setelah tambah data

4. [http://localhost:8000/products/\[id\]/edit](http://localhost:8000/products/[id]/edit)

Form Edit Produk

Nama

Harga

Gambar 12.4 Tampilan halaman form ubah data

12.2 Templating Halaman

Sebelumnya kita sudah menggunakan beberapa directive Blade. Directive ini juga dapat digunakan untuk templating halaman, yaitu bagian-bagian dari halaman yang sama / berulang untuk semua halaman dapat dibuat menjadi satu file Blade, dan semua halaman itu dapat me-load-nya tanpa harus ditulis ulang.

Langkah pertama dalam templating adalah membuat file template-nya, yang dimiliki oleh tiap halaman. Untuk contoh kasus pada modul ini, kita membuat file template dengan nama **template.blade.php** dengan isi:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>@yield('title')</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
  <script src="{{ asset('js/jquery.min.js') }}"></script>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"
"></script>
</head>
<body style="width:95%">
  @yield('content')
</body>
</html>
```

Directive **@yield** digunakan untuk menandakan bahwa di bagian itulah yang akan berbeda untuk tiap halamannya. Langkah selanjutnya adalah mengubah tampilan di tiap halaman seperti contoh **index.blade.php** di bawah ini:

```
@extends('template')

@section('title', 'Daftar Produk')

@section('content')
<body class="container">
  <div class="row justify-content-center mt-5">
    <div class="col-md-8">
      @if(session('success'))
        <div class="alert alert-success">{{ session('success') }}</div>
      @endif
      <div class="d-flex justify-content-between align-items-center mb-3">
        <span>{{ session('msg') }}</span>
        <a href="{{ route('products.create') }}" class="btn btn-
primary">Tambah</a>
      </div>
      <table class="table table-bordered table-striped">
        <thead>
```

```

        <tr>
            <th>Nama</th>
            <th>Harga</th>
            <th>Aksi</th>
        </tr>
    </thead>
    <tbody>
        @foreach($products as $product)
            <tr>
                <td>{{ $product->name }}</td>
                <td>{{ $product->price }}</td>
                <td>
                    <a href="{{ route('products.edit', $product->id) }}"
class="btn btn-sm btn-primary">Edit</a>
                    <form method="POST" action="{{ route('products.destroy',
$product->id) }}" style="display:inline" onsubmit="return confirm('Yakin
hapus?') ">
                        @csrf
                        @method('DELETE')
                        <button class="btn btn-sm btn-danger">Hapus</button>
                    </form>
                </td>
            </tr>
        @endforeach
    </tbody>
</table>
</div>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js
"></script>
</body>
@endsection

```

Directive **@extend** digunakan untuk menentukan file template mana yang digunakan oleh halaman ini. Sedangkan directive **@section** digunakan untuk mengisi halaman sesuai dengan nama **yield** yang di-define dalam file template. Directive **@section** dapat diisi dengan string ataupun tag HTML.

12.3 Form Validation

Laravel dapat ditambahkan form validation dari sisi server. Di controller, sebelum kita memanggil fungsi untuk menambahkan (fungsi **store**) / mengubah (fungsi **update**), kita dapat menambahkan kode:

```

...
public function store(Request $request)
{
    $validated = $request->validate([
        'name' => 'required|min:4',
        'price' => 'required|integer|min:1000000',
    ]);
    ...
public function update(Request $request)
{
    $validated = $request->validate([
        'name' => 'required|min:4',
        'price' => 'required|integer|min:1000000',
    ]);
    ...

```

Di kode validasi ini, kita mengatur nama tidak boleh kosong dan minimal 4 karakter, sedangkan harga tidak boleh kosong, harus integer, dan nilai minimal 1.000.000. Kemudian di halaman **form.blade.php** kita harus menambahkan directive **@error** seperti ini:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form {{ $title }} Produk</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="container">
    <div class="row justify-content-center mt-5">
        <div class="col-md-6">
            <h4>Form {{ $title }} Produk</h4>
            <form class="border p-4" method="POST" action="{{ $route }}">
                @csrf
                @if($method === 'PUT')
                    @method('PUT')
                @endif
                <div class="mb-3">
                    <label for="name" class="form-label">Nama</label>
                    <input type="text" name="name" id="name" class="form-control
@error('name') is-invalid @enderror" value="{{ old('name', $product->name) }}">
                    @error('name')
                        <div class="invalid-feedback">{{ $message }}</div>
                    @enderror
                </div>
                <div class="mb-3">
                    <label for="price" class="form-label">Harga</label>
                    <input type="number" name="price" id="price" class="form-
control @error('price') is-invalid @enderror" value="{{ old('price', $product-
>price) }}">
                    @error('price')

```

```

        <div class="invalid-feedback">{{ $message }}</div>
        @enderror
    </div>
    <div class="text-center">
        <button type="submit" class="btn btn-success">Simpan</button>
    </div>
</form>
</div>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"
></script>
</body>
</html>

```

Directive **@error** diletakkan di atribut class pada tag <input> untuk validasinya dan diletakkan di bawah tag <input> untuk menampilkan pesan validasinya. Kemudian agar data pada form di-repopulate, maka isikan atribut value pada tag <input> dengan **old('value-di-atribut-name')**.

MODUL 13. Laravel : Database 2

Tujuan Praktikum

4. Mahasiswa mampu memahami konsep dan implementasi CodeIgniter pada pembuatan aplikasi berbasis *web*.
5. Mahasiswa mampu menerapkan CRUD menggunakan Laravel dan konsep MVC.
6. Mahasiswa mampu mengimplementasikan fitur-fitur yang sering digunakan pada Laravel.

13.1 Session

Session memiliki fungsi menyimpan suatu variabel pada server dan variabel ini dapat diakses dan diubah dimanapun: Routing, Controller, ataupun halaman manapun. Session default pada Laravel dikelola sendiri oleh Laravel dan disimpan dalam bentuk file. Selain file, Session pada Laravel juga dapat diubah menjadi disimpan ke database atau menggunakan mekanisme lain.

Laravel memiliki dua jenis Session, yaitu Session biasa dan Session flash. Session biasa akan selalu ada selama belum dihapus, time-out, atau browser ditutup. Sedangkan Session flash adalah session yang hanya berlaku untuk satu request saja, setelah itu Laravel akan menghapusnya secara otomatis. Contoh Session flash sudah pernah kita gunakan melalui fungsi **with('x', y)** yang digunakan bersamaan dengan fungsi **redirect**. Untuk Session biasa, berikut contoh penggunaannya di Routing:

```
...
Route::get('/login', function () {
    if (session()->has('email')) return redirect('/product');
    return view('login');
});

Route::get('/logout', function () {
    session()->flush();
    return redirect('/login');
});
...
```

Fungsi **session()->has('x')** digunakan untuk memeriksa apakah ada Session bernama "x". Sedangkan **session()->flush()** digunakan untuk menghapus semua Session. Contoh lain, berikut penggunaan Session di SiteController:

```
...
public function auth(Request $req) {
    $u = User::where([
        ['email', $req->em],
        ['password', $req->pwd],
    ])->first();
    if (isset($u)) {
```

```

    session()->put('email', $u->email);
    session()->put('name', $u->name);
    return "<script>
    alert('Welcome, " . session('name') . "');
    location.href='/product';
    </script>";
}
return redirect('/login')->with('msg', 'Email / password salah');
}
...

```

Fungsi **session()->put('x', y)** digunakan untuk membuat Session bernama "x" dengan nilai y. Sedangkan **session('x')** digunakan untuk mengambil nilai dari Session "x".

13.2 Middleware

Middleware pada Laravel adalah suatu mekanisme untuk menyaring request yang masuk ke suatu Routing. Sebagai contoh, kita bisa membuat Middleware untuk memverifikasi apakah seorang user sudah terautentikasi atau memiliki hak akses untuk mengakses URL. Jika user belum terautentikasi atau tidak memiliki akses, maka Middleware dapat me-redirect user tersebut ke URL lain. Sebaliknya jika user terautentikasi atau memiliki hak akses, maka Middleware akan meneruskan request ke aksi yang dipetakan di Routing.

Middleware dapat melakukan hal lain selain untuk autentikasi, misalnya untuk menulis log atau error yang terjadi. Dan Middleware dapat kita atur mekanisme yang dilakukannya, yaitu bisa sebelum request diteruskan atau sesudah request diteruskan. Middleware yang kita buat harus berada di folder **app/Http/Middleware**.

Sekarang kita akan membuat autentikasi dengan menggunakan library **Auth**. Library ini sudah termasuk di dalam paket instalasi Laravel, sehingga kita tidak perlu menambahkannya lagi melalui composer. Dalam library ini sudah mencakup semua fitur untuk autentikasi, registrasi, dan middleware-nya. Dengan menggunakan **Auth**, autentikasi menjadi lebih simpel, aman, dan bisa menjadi alternatif pengganti yang lebih baik dari Session.

Langkah pertama adalah pengaturan pada Routing. Ganti Routing untuk login, logout, dan product:

```

...
Route::get('/login', function () {
    if (Auth::check()) return redirect('/product');
    return view('login');
})->name('login');

Route::get('/logout', function () {
    Auth::logout();
    return redirect('/login');
});

Route::resource('product', ProductController::class)->middleware('auth');

```

Fungsi **Auth::check()** digunakan untuk memeriksa apakah user telah terautentikasi. Routing login kita berikan nama alias karena kebutuhan dari Middleware **auth**. Sedangkan **Auth::logout()** digunakan untuk menghapus autentikasi. Untuk product, Middleware kita ganti dengan Middleware **auth** yang berkolaborasi dengan library Auth.

Lalu langkah kedua ubah isi dari fungsi **auth** di **SiteController**:

```

...
use Illuminate\Support\Facades\Auth;

class SiteController extends Controller
{
    public function auth(Request $req) {
        if (Auth::attempt(['email'=>$req->em, 'password'=>$req->pwd])) {
            //jika ada nilai lain selain data user yang ingin disimpan di session, baru
            gunakan session disini
            return redirect('/product');
        }
        return redirect('/login')->with('msg', 'Email / password salah');
    }
}
...

```

Fungsi **Auth::attempt()** digunakan untuk proses autentikasi, yaitu apakah email dan password yang di-submit ada dalam database pada tabel **users**. Jika email dan password cocok maka akan menghasilkan **true**. Hal yang harus diperhatikan adalah ketika menambahkan data User ke database, pastikan isi dari kolom password di-hash dengan metode **Bcrypt** agar dapat menggunakan library Auth ini. Laravel sudah menyediakan fungsi **bcrypt('x')** untuk mempermudahnya.

Langkah ketiga, dalam View yang sebelumnya menggunakan **@if(session('x'))** dapat kita ganti menjadi directive **@auth** untuk pengecekan apakah user telah terautentikasi, dan dapat menggunakan **Auth::user()** untuk mengakses data user yang login. Berikut contohnya dalam **template.blade.php**:

```

...
<body style="width:95%">
    @auth
    <div class="row justify-content-end" style="margin-top:2%">
        <div class="col-3">
            {{ Auth::user()->name }}
            <a href="/logout" class="btn btn-warning">Logout</a>
        </div>
    </div>
    @endauth
    <div class="row justify-content-center" style="margin-top:10%">
        @yield('content')
    </div>
...

```

13.3 Model Relasi

Model Eloquent menyediakan fasilitas agar dua Model yang berelasi dapat langsung memanggil satu sama lain. Kita akan mencoba salah satunya yaitu relasi one to many, dengan menggunakan contoh kasus relasi **Product** dengan **Variant**. Tiap **Product** dapat memiliki banyak **Variant**, tetapi tiap **Variant** hanya dimiliki oleh satu **Product**. Langkah awal yaitu membuat Model dan file migration-nya, maka perintahnya:

```
php artisan make:model Variant -m
```

Setelah itu edit file **xxx_create_variants_table.php** pada folder **database/migrations** untuk mendefinisikan kolomnya:

```

...
Schema::create('variants', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->text('description');
    $table->string('processor');
    $table->string('memory');
    $table->string('storage');
    $table->foreignId('product_id')->constrained();
    $table->timestamps();
});
...

```

Untuk kebutuhan foreign key, kita menggunakan **foreignId** dan **constrained** jika tabel entitas yang diacu mengikuti konvensi Model Eloquent (PK bernama “id”, tabel database ditambah akhiran “s”, dsb). Jika berbeda, maka pendefinisian foreign key harus menggunakan **foreign**, **references**, dan **on**. Contoh:

```

...
$table->integer('product_id');
$table->foreign('product_id')->references('id_product')->on('product');
...

```

Kemudian generate tabel **variants** dengan perintah seperti sebelumnya (php artisan migrate). Selanjutnya edit file Model **Variants**, tambahkan fungsi berikut:

```

...
public function product() {
    return $this->belongsTo(Product::class);
}
...

```

Fungsi **belongsTo()** secara otomatis akan memanggil object **Product** yang berelasi dengan dirinya berdasarkan **product_id**. Kemudian tambahkan juga fungsi ke dalam Model **Product**:

```

...
public function variants() {
    return $this->hasMany(Variant::class);
}
...

```

Fungsi **hasMany()** secara otomatis akan memanggil semua object **Variant** yang berelasi dengan dirinya berdasarkan **product_id**. Untuk mencobanya kita akan menambahkan satu kolom pada tampilan tabel di **index.blade.php** untuk menampilkan data variant dari tiap product:

```

...
<th>Harga</th>
<th>Variant</th>
<t

    h>Aksi</th>
</tr>
@foreach($list as $d)
<tr>
    <td>{{ $d->name }}</td>
    <td>{{ $d->price }}</td>

```

```

<td>
<ul>
@foreach($d->variants()->get() as $var)
    <li>{{ $var->name }}</li>
    Desc: {{ $var->description }} <br /> Proc:
    {{ $var->processor }} <br />
    RAM: {{ $var->memory }} <br /> Strg:
    {{ $var->storage }} <br /> Product: {{
    $var->product->name }}
    @endforeach
</ul>
</td>
<td align="center">

```

...

Tiap data product dapat langsung mengakses semua data variant-nya dengan fungsi variants() dan tiap variant dapat mengakses product yang berelasi dengannya dengan menggunakan atribut product. Uji dengan memasukkan data variants di database dan membuat form untuk menambahkan data variants ke database.

MODUL 14. Coding on the Spot 2 (COTS 2)

Tujuan Praktikum

1. Praktikan mampu membangun arsitektur web, pemrograman web dasar, teknologi yang mendukung pemrograman web, dan pemrograman web lanjut dengan framework.

14.1. Program Learning Outcomes

Menguasai metode dan proses analisis, perencanaan, pengelolaan, serta evaluasi yang berkaitan.

14.2. Course Learning Outcomes

Mahasiswa mampu merancang aplikasi dari studi kasus.

14.3. Komponen Penilaian

Komponen penilaian dari MONEV 1 adalah sebagai berikut:

NO	Komponen Penilaian	Nilai				
		0 Point	10 Point	15 Point	20 Point	30 Point
1	Menggunakan Framework Bootstrap	Tidak ada	Sebagian	Ada	-	-
2	3 Halaman (Form , table, CRUD)	Tidak ada	Sebagian halaman	Setengah halaman	Sebagian besar dari halaman	Lengkap
3	Poin 2 menggunakan Framework CI	Tidak ada	Ada	-	Jika menggunakan NodeJS	-
4	JQuery Plugin	Tidak ada	Sebagian halaman	Sebagian besar dari halaman	Lengkap	-
5	Menggunakan data JSON untuk table dan datatable (jQuery)	Tidak menggunakan	Sebagian halaman	Lengkap	-	-
6	Dokumentasi	Tidak ada	Ada	-	-	-

MODUL 15. Responsi Dan Evaluasi Tugas Besar 2

Tujuan Praktikum
1. Praktikan mampu membangun arsitektur web, pemrograman web dasar, teknologi yang mendukung pemrograman web, dan pemrograman web lanjut dengan framework.

Pada modul 13 melakukan asistensi dengan asprak yang sudah diplot. Pastikan progress project anda sekitar 80 - 100%.

DAFTAR PUSTAKA

- [1] R. W. Sebesta, Programming the World Wide Web, Addison-Wesley, 2014.
- [2] "W3Schools," December 2017. [Online]. Available: <https://www.w3schools.com/>.
- [3] "Bootstrap," January 2018. [Online]. Available: <https://getbootstrap.com/>.
- [4] "CodeIgniter," January 2018. [Online]. Available: <https://codeigniter.com/docs>.
- [5] A. Solichin, Pemrograman Web dengan PHP dan MySQL, Budi Luhur, 2016.
- [6] A. Subagia, Membangun Aplikasi dengan Codeigniter dan Database SQL Server, Jakarta: PT Elex Media Komputindo, 2017.
- [7] "Memulai Git - Dasar Git," Januari 2019. [Online]. Available: <https://git scm.com/book/id/v1/Memulai-Git-Dasar-Git>.
- [8] "Mengenal Restful API," Januari 2019. [Online]. Available: <https://kudo.co.id/engineering/2016/09/15/mengenal-restful-api/>.
- [9] "Pengertian Node.js," Januari 2019. [Online]. Available: <https://teknojurnal.com/pengertianapa-itu-node-js/>.
- [10] "Cara Menginstall XAMPP di Windows," Januari 2019. [Online]. Available: <https://www.duniaikom.com/tutorial-belajar-wordpress-cara-menginstall-xampp-diwindows/>.

https://drive.google.com/file/d/1ppX75ZdWTjJbVBjYR9d-fT_m5o1XV7pDx/view?usp=dr

