

Richad Rivanto David, S.Kom

UPTP BPVP Ambon

Jabatan : Instruktur Ahli Pertama
Kejuruan : Teknologi Informasi dan Komunikasi

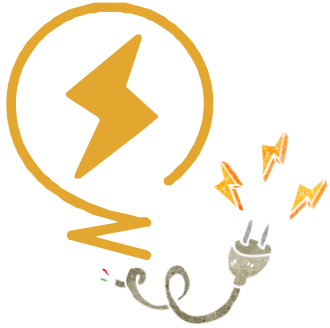
Pendidikan: S1 Sistem Informasi
Agama: Kristen
Wa/HP: 08111809859
Email: richadrivantodavid@gmail.com





SAFETY CONDUCTION

YOUR SAFETY IS PRIORITY



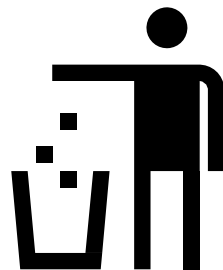


ROLE PLAY

LEARNING FOR FUN



**share &
care**



**keep
clean**



**no ring
phone**



**no
smoking**





ROLE PLAY

LEARNING FOR FUN



08.00-16.45



**rapi
sopan**



**class
leader**

APERSEPSI

Pada pertemuan sebelumnya, sudah dijelaskan bagaimana cara Mengimplementasikan Pemrograman Terstruktur

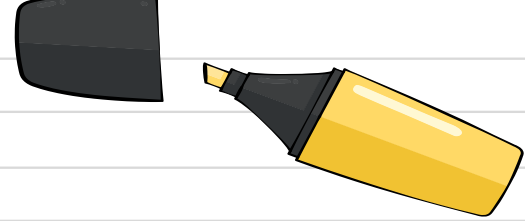


/Menyusun Fungsi, File atau Sumber Daya Pemrograman Lain dalam Organisasi yang Rapi

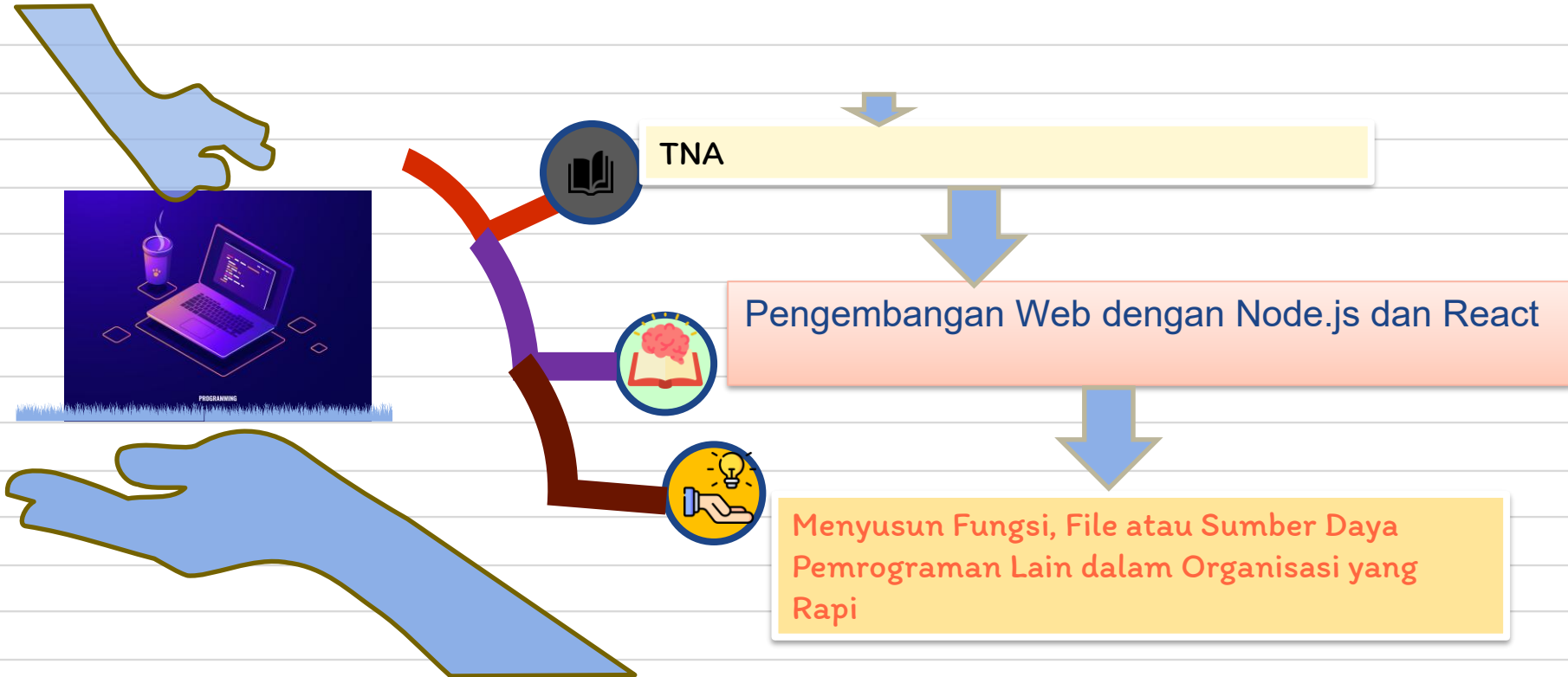


```
> node_modules
  ▼ src
    ▼ config
      JS db.js
    ▼ controllers
      JS nilaiControllers.js
    ▼ middleware
      JS validation.js
    ▼ routes
      JS main.js
  .env
```





ROAD OF MAP





Setelah selesai mengikuti pembelajaran ini, peserta mampu menyusun fungsi, file atau sumber daya pemrograman lain dalam organisasi yang rapi





Setelah selesai mengikuti pembelajaran ini, dapat menyusun struktur kode yang rapi, terorganisir, dan mudah dibaca dalam membuat aplikasi berbasis web.



SECTION 1

**/MENGELOLA SUMBER DAYA
PEMROGRAMAN SESUAI KARAKTER**



PART A

**/NAMA FILE, FUNGSI, VARIABEL,
KONSTANTA, DAN SUMBER DAYA
PEMROGRAMAN LAIN DIBUAT
SESUAI KONTEKS**



/KONVENSI PENAMAAN



/01 APA ITU KONVENSI PENAMAAN?

> Konvensi penamaan adalah **aturan atau kebiasaan dalam memberi nama** pada elemen dalam kode seperti **file, folder, variabel, fungsi, komponen, kelas, konstanta, dll**



/PENAMAAN FILE, FUNGSI, VARIABEL, KONSTANTA DAN SUMBER DAYA PEMROGRAMAN LAIN



/02 PENTING DILAKUKAN !

> Penamaan yang baik dalam pemrograman sangat penting karena kode akan dibaca lebih sering daripada ditulis. Nama yang jelas membantu kita dan tim memahami maksud dari fungsi, variabel, atau file tanpa harus membaca seluruh kode. Kode yang tidak memiliki penamaan yang konsisten dan deskriptif akan menyulitkan pemeliharaan dan kolaborasi.



/TUJUAN MEMBUAT NAMA FILE, FUNGSI, VARIABEL, KONSTANTA



- > - Membuat kode lebih mudah dibaca dan dimengerti.
- Mencegah kesalahan penulisan dan penggunaan kode.
- Mempermudah kerja kolaboratif dalam tim.
- Mengikuti standar profesional dunia kerja.



/ ILUSTRASI



File Itu Penting?



Bayangkan mencari dokumen di dalam lemari arsip tanpa label

Resep Masakan Tanpa Nama Bahan

Campur bahan A dan bahan B
Masukkan bahan C, panaskan selama X menit
Sajikan



- Kalau bahan A itu mie, B itu bawang, dan C itu kecap, kenapa tidak disebut langsung?
- Begitu juga variabel yang diberi nama a, b, c — tidak jelas artinya



/KONVENSI PENAMAAN SECARA UMUM FILE DALAM NODE.JS, DAN REACT



| No | Kategori | Gaya Penamaan | Format Contoh | Tujuan / Isi File | Umumnya Digunakan di |
|----|-----------------------|---------------|--------------------|---|----------------------|
| 1 | Router | kebab-case | user-routes.js | Mengatur endpoint URL dan HTTP method | Node.js |
| 2 | Controller | kebab-case | auth-controller.js | Menangani logika request/response | Node.js |
| 3 | Service | kebab-case | email-service.js | Berisi logika bisnis atau helper function | Node.js |
| 4 | Middleware | kebab-case | auth-middleware.js | Berisi fungsi penghubung request-response | Node.js |
| 5 | Config | kebab-case | database-config.js | Konfigurasi database, env, setup awal | Node.js |
| 6 | Komponen React | PascalCase | LoginForm.js | Komponen UI yang digunakan dalam React | React |
| 7 | Halaman/Page | PascalCase | DashboardPage.js | Menampilkan halaman lengkap | React |
| 8 | Utilitas | kebab-case | api-helper.js | Fungsi pembantu (fetch, format, dsb) | React/Node.js |
| 9 | Gaya CSS | kebab-case | login-form.css | Styling komponen atau halaman tertentu | React |
| 10 | Aset/Gambar | kebab-case | user-avatar.svg | Aset gambar/icon yang digunakan di aplikasi | React |





/camelCase



- > Huruf pertama kecil, kata berikutnya diawali huruf kapital, tanpa spasi atau tanda hubung.

```
let userName = "Richad";  
function getUserAge(birthYear) { return 2025 - birthYear; }
```

- > Biasanya digunakan untuk :
 - Variabel (let Username)
 - Fungsi (function)





/PascalCase



> Seperti camelCase, tapi huruf pertama juga kapital.

```
class ProductList {}  
function RegisterUser() {}
```

Biasanya Digunakan untuk :

- > - Kelas (class App)
- Komponen React





/snake_case



- > Semua huruf kecil, antar kata dipisahkan oleh underscore (_)

```
let user_id = 1023;  
const order_list = ["Apple", "Banana"];
```

- > Biasanya Digunakan untuk :
 - Nama file di beberapa penulisan (auth_controller.js)





/kebab-case



- > Semua huruf kecil, antar kata dipisahkan oleh underscore (-)

```
file: user-profile.js
```

- > **Ingat** Nama file di sistem operasi *case sensitive*





/UPPER_CASE



- > Semua huruf besar, biasanya digunakan untuk sesuatu yang tidak berubah (konstanta)

```
CORS_ALLOW_LIST=http://localhost:5713,http://localhost:3000
```



✗ CONTOH PENAMAAN BURUK (HINDARI INI):



| ✗ Nama Buruk | Kenapa? |
|--|------------------------------------|
| a, x1, data2 | Tidak jelas arti fungsinya |
| myfunction, func1 | Tidak deskriptif |
| test.js, script1.js | Tidak menunjukkan konteks kegunaan |
| Userlist, userList (tidak konsisten huruf besar kecil) | Inkonstisten |





/ CONTOH PENAMAAN BAIK:



> File

```
/components/UserCard.jsx  
/pages/ProductList.jsx  
/utils/formatDate.js
```

> Variabel

```
let totalPrice = 5000;  
let isLoggedIn = true;
```

> Fungsi

```
function calculateDiscount(price, percent) {  
  return price * percent / 100;  
}
```

> Konstanta

```
CORS_ALLOW_LIST=http://localhost:5713,http://localhost:3000  
DATABASE_URL="mysql://tugasprisma:randompassword@localhost:3306/db_tugas_prisma_jwt"  
JWT_SECRET="2b10sQouC2z/Ds89hj6oM090ZAUg1GXzY47UPi6AZjjw929KUvr490u"
```



PART B

**/SETIAP FUNGSI ATAU PROSEDUR
ATAU PROGRAM DILENGKAPI
DENGAN PENULISAN KOMENTAR DI
AWAL**





/PENAMAAN FILE, FUNGSI, VARIABEL, KONSTANTA DAN SUMBER DAYA PEMROGRAMAN LAIN



/02 PENTING DILAKUKAN !

- > Penamaan yang baik dalam pemrograman sangat penting karena kode akan dibaca lebih sering daripada ditulis. Nama yang jelas membantu kita dan tim memahami maksud dari fungsi, variabel, atau file tanpa harus membaca seluruh kode. Kode yang tidak memiliki penamaan yang konsisten dan deskriptif akan menyulitkan pemeliharaan dan kolaborasi.



/MENGELOLA SUMBER DAYA PEMROGRAMAN SESUAI KARAKTER



/01 APA ITU KOMENTAR?

- > Komentar di awal fungsi/prosedur/program berarti kita menuliskan **penjelasan tentang apa yang dilakukan fungsi tersebut, parameter yang digunakan, dan apa hasil akhirnya (return).**





/TUJUAN MEMBUAT KOMENTAR AWAL



- Menjelaskan **fungsi tersebut untuk apa**
- Apa **input (parameter)** dan apa **output (return)**
- > - Mempermudah orang lain (atau diri sendiri) membaca ulang kode
- Menunjukkan profesionalisme dan tanggung jawab



/APA YANG SEBAIKNYA DITULISKAN DALAM KOMENTAR AWAL?



| Elemen | Penjelasan |
|--------------------------|---|
| Deskripsi | Jelaskan fungsi ini bertugas apa |
| Input/Param | Data apa yang masuk, tipenya, dan maknanya |
| Output/Return | Apa yang dihasilkan oleh fungsi tersebut |
| Author (opsional) | Nama penulis, tanggal, versi (untuk proyek besar) |
| | |



/JENIS – JENIS KOMENTAR DI NODE.JS



| Jenis | Sintaks | Keterangan |
|----------------------------|----------------|---|
| Satu Baris | // komentar | Untuk penjelasan singkat |
| Multi Baris | /* komentar */ | Untuk menjelaskan blok kode |
| Dokumentasi (JSDoc) | /** ... */ | Untuk mendeskripsikan fungsi, parameter, dll. |





/KOMENTAR SATU BARIS



```
// Menampilkan hasil ke console  
console.log(price);
```





/KOMENTAR MULTI BARIS




```
/*  
    Fungsi ini memeriksa apakah user memiliki akses  
    Berdasarkan status login dan role yang dimiliki  
*/  
function checkAccess(user) {  
    return user.isLoggedIn && user.role === 'admin';  
}
```



/ CONTOH KOMENTAR BURUK



```
let total = price * quantity; // Mengalikan price dengan quantity
```

 Komentar hanya mengulangi apa yang sudah terlihat jelas dari kodenya



/ CONTOH KOMENTAR BURUK



```
// cek dta
if (usr != null) {
    // gk null masuk
    next();
}
```

✗ Menggunakan singkatan berantakan (cek, dta, gk) sehingga sulit dipahami programmer lain



/JENIS – JENIS KOMENTAR DI REACT



| No | Jenis Komentar | Lokasi Penggunaan | Sintaks | Contoh Penggunaan |
|----|----------------------|-------------------------|---|--|
| 1 | Komentar Satu Baris | Di luar JSX | // komentar | // Mengimpor komponen |
| 2 | Komentar Multi Baris | Di luar JSX | /* komentar */ | /* Ini komentar banyak baris */ |
| 3 | Komentar JSX | Di dalam JSX (return) | {/* komentar */} | {/* Menampilkan nama user */} |
| 4 | Komentar JSDoc | Di atas fungsi/komponen | <pre>/** ... @param ... @returns ... */</pre> | /** Komponen untuk menampilkan info */ |



PART C

**/BADAN SOURCECODE DILENGKAPI
DENGAN KOMENTAR/KETERANGAN** →



/MENGAPA HARUS MEMBUAT KOMENTAR DI BADAN SOURCE ?



- > Agar setiap bagian penting dari kode (terutama fungsi, logika, dan blok proses) diberikan **penjelasan singkat** dalam bentuk komentar, agar pembaca tahu **apa yang dilakukan baris tersebut** dan **mengapa**.



/CONTOH BADAN SOURCE TIDAK DILENGKAPI KOMENTAR (NODE.JS) ● ● ●

>

```
app.post('/total', (req, res) => {  
  const { price, qty } = req.body;  
  if (!price || !qty) {  
    return res.status(400).json({ message: 'Harga dan jumlah wajib diisi' });  
  }  
  const total = price * qty;  
  res.json({ total });  
});
```

SANGAT SULIT DIPAHAMI





/CONTOH BADAN SOURCE DILENGKAPI KOMENTAR NODE.JS



```
// Import library express
const express = require('express');

// Inisialisasi aplikasi express
const app = express();

/**
 * Fungsi middleware untuk parsing JSON
 */
app.use(express.json());

/**
 * Endpoint untuk menghitung total harga
 *
 * @route POST /total
 * @body { price: number, qty: number }
 * @returns { total: number }
 */
app.post('/total', (req, res) => {
  // Ambil data dari body request
  const { price, qty } = req.body;
```

```
// Validasi input: pastikan keduanya diisi
if (!price || !qty) {
  return res.status(400).json({ message: 'Harga dan jumlah wajib diisi' });
}

// Hitung total harga
const total = price * qty;

// Kirimkan hasil ke client
res.json({ total });
});

// Jalankan server di port 3000
app.listen(3000, () => {
  console.log('Server berjalan di http://localhost:3000');
});
```





/CONTOH BADAN SOURCE TIDAK DILENGKAPI KOMENTAR (REACT) ● ● ●

SANGAT SULIT DIPAHAMI

```
function App() {  
  // function  
  const x = 10;  
  
  // component  
  return (  
    <div>  
      { /* div */}  
      <h1>Halo</h1>  
    </div>  
  );  
}
```





/CONTOH BADAN SOURCE DILENGKAPI KOMENTAR REACT



```
import React, { useState, useEffect } from "react";

/**
 * Komponen sederhana untuk menampilkan daftar pengguna
 * Data diambil dari API dan ditampilkan dalam list
 */
function UserList() {
  // State untuk menyimpan data user
  const [users, setUsers] = useState([]);

  // Fetch data dari API saat komponen pertama kali dirender
  useEffect(() => {
    // Ambil data dari API publik
    fetch('https://jsonplaceholder.typicode.com/users')
      .then((res) => res.json())
      .then((data) => {
        // Simpan data user ke dalam state
        setUsers(data);
      });
  }, []);
}
```



SECTION 2

**/MENGORGANISASI KAN SUMBER
DAYA PEMROGRAMAN SESUAI
KONTEKS**



PART A

**/FOLDER DAN SUB- SUB FOLDER
DISUSUN SESUAI KONTEKS DAN
ISINYA**



/KENAPA STRUKTUR FOLDER ITU PENTING?



/02 PENTING DILAKUKAN !

- Memudahkan navigasi dan pencarian file
- Membantu kolaborasi tim
- Menghindari tumpang tindih atau file tidak terorganisir



/ STRUKTUR FOLDER PADA NODE.JS (BACKEND)



```
/src
├─ controllers/      # Logika respon API (fungsi endpoint)
├─ routes/           # Definisi URL endpoint (GET, POST, dst)
├─ services/         # Logika pemrosesan / layanan bisnis
├─ models/           # Struktur data/database (misalnya mongoose)
├─ middleware/       # Fungsi perantara (contoh: cek token)
├─ config/           # Pengaturan environment, database
├─ utils/            # Fungsi bantu (helper function)
├─ validations/      # Validasi input user
└─ app.js            # File utama aplikasi (entry point)
```

/ STRUKTUR FOLDER PADA REACT (FRONTEND)



```
/src
├── components/      # Komponen kecil (reusable): tombol, kartu, dsb
│   └── Button.js
├── pages/           # Halaman lengkap: Home, Login, Dashboard
│   └── HomePage.js
├── layout/          # Struktur umum seperti Navbar, Sidebar
├── services/        # API call ke backend
├── utils/           # Fungsi bantu (format tanggal, dll)
├── assets/          # Gambar, ikon, font, dll
├── styles/          # File CSS
├── App.js           # Root komponen utama
└── index.js         # Entry point render React
```



/ CONTOH PENYUSUNAN SESUAI KONTEN & KONTEXT



```
fullstack-app/  
├── client/                # Folder frontend React  
│   ├── public/  
│   └── src/  
│       ├── components/  
│       │   └── Button.js  
│       ├── pages/  
│       │   └── HomePage.js  
│       ├── layout/  
│       │   └── Navbar.js  
│       ├── services/  
│       │   └── api.js  
│       ├── utils/  
│       │   └── formatDate.js  
│       ├── styles/  
│       │   └── app.css  
│       ├── App.js  
│       └── index.js  
└── server/              # Folder backend Node.js  
    ├── controllers/  
    │   └── user-controller.js  
    ├── routes/  
    │   └── user-routes.js  
    ├── services/  
    │   └── email-service.js  
    ├── models/  
    │   └── user-model.js  
    ├── middleware/  
    │   └── auth-middleware.js
```

```
├── config/  
│   └── db-config.js  
├── utils/  
│   └── helper.js  
├── app.js  
└── package.json  
├── README.md  
└── package.json (root)    # Untuk mengatur workspace fullstack
```



PART B

/FILE “README”





/README



/01 APA ITU FILE “Readme”??






- > Dokumen teks yang berisi **penjelasan utama tentang sebuah proyek.**
- > Biasanya file ini bernama

`README.md`





/FUNGSI UTAMA FILE README

| FUNGSI | PENJELASAN |
|--|--|
|  Mengenalkan Proyek | Menjelaskan tujuan dan fungsi proyek |
|  Panduan Penggunaan | Menjelaskan cara menginstal, menjalankan, dan menggunakan proyek |
|  Struktur Folder | Menjelaskan isi dan fungsi masing-masing folder/file |
|  Teknologi yang Digunakan | Menyebutkan library/framework (contoh: Node.js, React, MongoDB, dll) |
|  Siapa Pembuatnya | Bisa mencantumkan nama pembuat, kontributor, atau author proyek |



/ISI FILE README







README.md > [1/1] # Nama Proyek > [1/1] ## Author

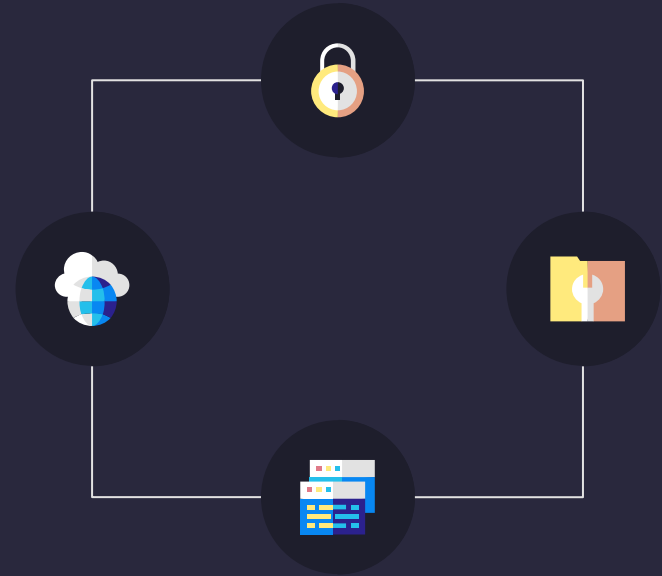
```
1  # Nama Proyek
2
3  Deskripsi singkat tentang aplikasi ini.
4
5  ## Struktur Folder
6  - /client → frontend React
7  - /server → backend Node.js
8
9  ## Teknologi
10 - Node.js
11 - Express
12 - React
13 - MongoDB
14
15 ## Cara Menjalankan
16 1. `npm install`
17 2. `npm start`
18
19 ## Author
20 - Nama: Richad
21 - Kelas: Pengembangan Web dengan Node.js dan React 2025
22
```



/KESIMPULAN

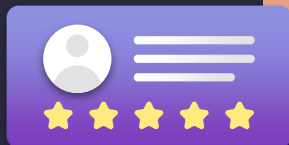


-  Penamaan yang Baik Adalah Kunci Keterbacaan Kode. Penamaan variabel, fungsi, file, dan folder harus jelas, konsisten, dan sesuai konvensi.
-  Struktur Folder Harus Disusun Sesuai Konteks dan Fungsi
-  Komentar Kode Membantu Pemahaman dan Dokumentasi
-  File README Adalah Wajah Utama Proyek





/PRAKTEK JOBSHEET



/EVALUASI





/THANKS!

/DO YOU HAVE ANY QUESTIONS?

richadrivantodavid@gmail.com
08111809859



Coding is fun

