POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

DIKTISAINTEK BERDAMPAK KAMPUS INOVASI



# React.js, Express.js, dan Implementasi API

## Workshop Pemrograman WEB

*Yunia Ikawati*

Teknik Informatika-PENS

# Pendahuluan

- Tujuan Pembelajaran Utama

  Mahasiswa memahami React.js berbasis komponen, peran Express.js dalam REST API, dan integrasi frontend-backend.

- Ruang Lingkup Materi
  - Materi meliputi pengenalan teknologi, arsitektur client-server, HTTP, JSON, API React, dan pembuatan API Express.
- Hasil Akhir Perkuliahan
  - Mahasiswa mampu membuat aplikasi web sederhana React yang terhubung dengan Express API secara mandiri.

# Pengenalan React.js

# Konsep Dasar dan Peran React.js dalam Frontend

- React sebagai Library UI
  - React digunakan untuk membangun antarmuka pengguna yang interaktif dan responsif di sisi frontend.
- Konsep Komponen Reusable
  - React memecah tampilan aplikasi menjadi komponen kecil yang dapat digunakan kembali dan dikelola secara mandiri.
- Pengambilan Data dari Backend
  - React mengambil data melalui HTTP request dari backend API, biasanya dalam format JSON, tanpa berkomunikasi langsung dengan database.
- Konsep Dasar React
  - JSX, functional component, useState, dan useEffect adalah fondasi penting dalam pengembangan aplikasi React.

# Konsep React js

| Konsep | Peran di React |
| --- | --- |
| JSX | Menulis tampilan UI |
| Functional Component | Struktur utama aplikasi |
| useState | Menyimpan & mengelola data |
| useEffect | Menangani proses tambahan (API, lifecycle) |

# Konsep React js : JSX

- JSX adalah cara menulis tampilan (UI) React yang mirip HTML di dalam JavaScript.
- JSX memudahkan kita membuat UI karena tampilannya lebih rapi dan mudah dibaca.
- Contoh:

```
<h1>Daftar Mahasiswa</h1>
```

# Konsep React js: Functional Component

- Functional Component adalah komponen React yang dibuat menggunakan fungsi JavaScript.
- Contoh:

```
function App() {
  return <h1>Hello React</h1>;
}
```

# Konsep React js: useState

- useState digunakan untuk menyimpan dan mengelola data (state) di dalam komponen React.
- Saat nilai state berubah, tampilan React akan otomatis diperbarui.
- Contoh:

```
const [mahasiswa, setMahasiswa] = useState([]);
```

# Konsep React js: useEffect

- useEffect digunakan untuk menjalankan proses tambahan (side effect) seperti:
  - Memanggil API
  - Mengambil data dari backend
  - Menjalankan kode saat komponen pertama kali tampil.
- Contoh:

```
useEffect(() => {
  fetch("http://localhost:5000/api/mahasiswa")
    .then(res => res.json())
    .then(data => setMahasiswa(data));
}, []);
```

# Pengenalan Node.js dan Express.js

Express.js sebagai Backend dan Pembuat API

## Framework Backend Express.js

Express.js menyederhanakan pembuatan server dan REST API di atas Node.js dengan fitur routing dan middleware.

## Penggunaan API dengan React

Express menyediakan API yang mengembalikan data JSON untuk dikonsumsi aplikasi React sebagai sumber data.

## Metode HTTP dan Pemisahan Tugas

Konsep HTTP GET dan POST penting untuk memisahkan tanggung jawab frontend dan backend secara efektif.

## Manajemen Logika Bisnis Server

Express menempatkan logika bisnis, validasi, dan komunikasi database di server agar frontend fokus pada UI.

# Arsitektur React dan Express

# Alur Komunikasi Client dan Server

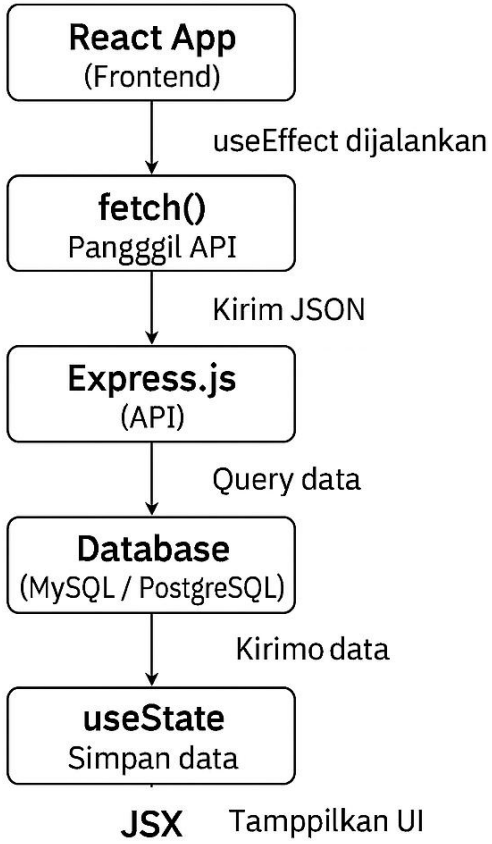
- Peran Client dan Server
  - React berfungsi sebagai client di browser, sementara Express berfungsi sebagai server di Node.js yang melayani permintaan data.
- Protokol HTTP
  - Komunikasi antara client dan server dilakukan melalui protokol HTTP yang mengirimkan request dan menerima response data JSON.
- Alur Data Dinamis
  - React mengirim permintaan data ke Express, menerima response, dan memperbarui tampilan UI secara dinamis sesuai data yang diterima.
- Keuntungan Arsitektur
  - Pemisahan tanggung jawab memudahkan pengembangan tim dan memberikan fleksibilitas untuk aplikasi skala besar.

# Contoh Implementasi API sederhana yang terhubung dengan Database MySQL

# Case: API Data Mahasiswa

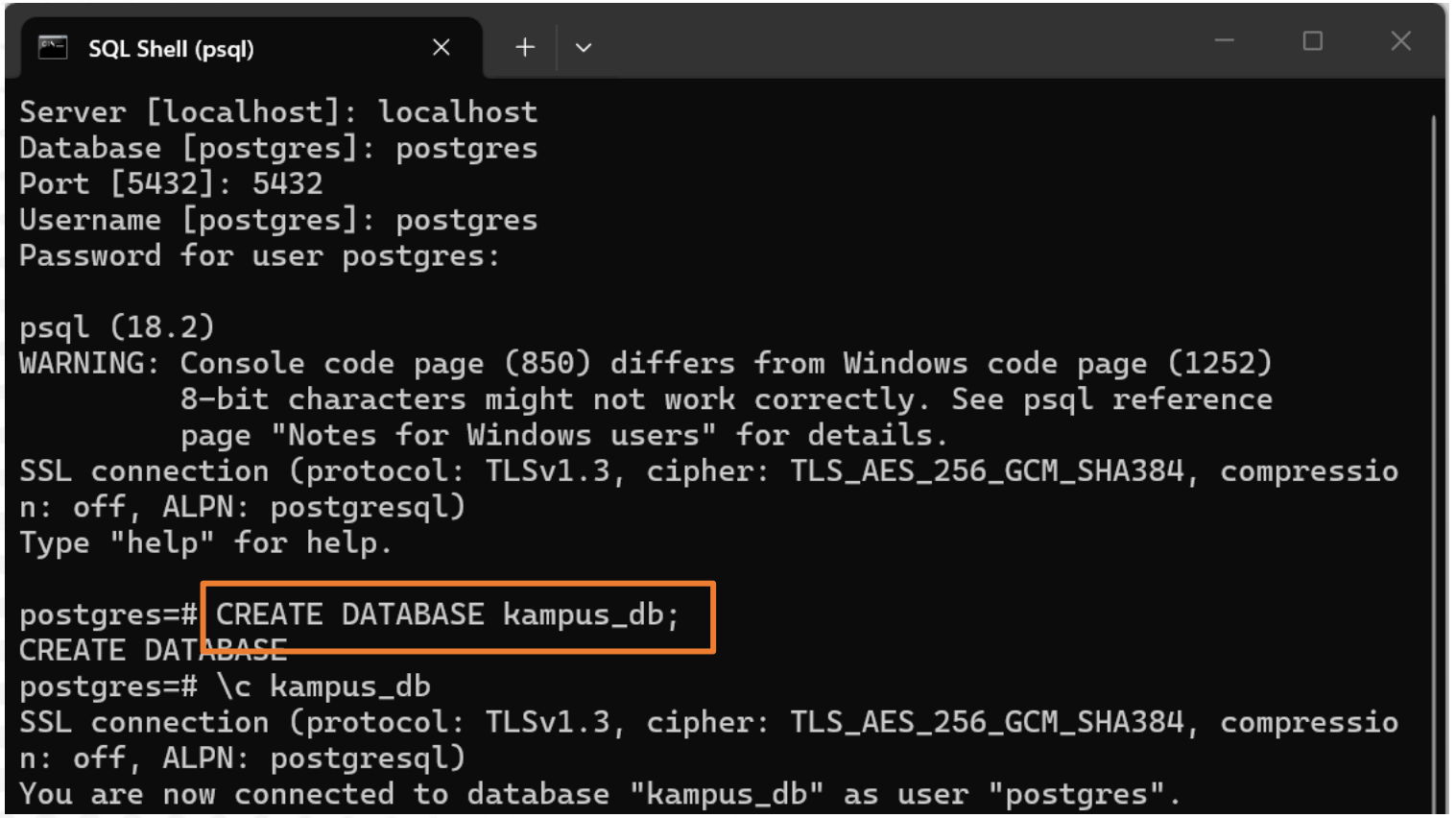
- Kebutuhan:
  - Menyimpan data mahasiswa di database
  - Menyediakan API untuk:
    - Mengambil daftar mahasiswa
    - Menambahkan mahasiswa
- Field Mahasiswa:
  - Ide
  - Nama
  - Nim
  - jurusan

# ARSITEKTUR APLIKASI



# BAGIAN 1 DATABASE POSTGRESQL

- Buat Database ➔ buka terminal psql



# BAGIAN 1 DATABASE POSTGRESQL

- Masuk Database kampus_db

```
postgres=# \c kampus_db
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compressio
n: off, ALPN: postgresql)
You are now connected to database "kampus_db" as user "postgres".
```

- buat table “Mahasiswa”

```
kampus_db=# CREATE TABLE mahasiswa(id SERIAL PRIMARY KEY, nama VARCHAR(100),
nim VARCHAR(20), jurusan VARCHAR(100));
CREATE TABLE
kampus_db=#
```

# BAGIAN 1 DATABASE POSTGRESQL

- Tambahkan isi data table “Mahasiswa”

```
kampus_db=# INSERT INTO mahasiswa (nama, nim, jurusan) VALUES
kampus_db-# ('Andi', '2023001', 'Informatika'),
kampus_db-# ('Siti', '2023002', 'Sistem Informasi');
INSERT 0 2
```

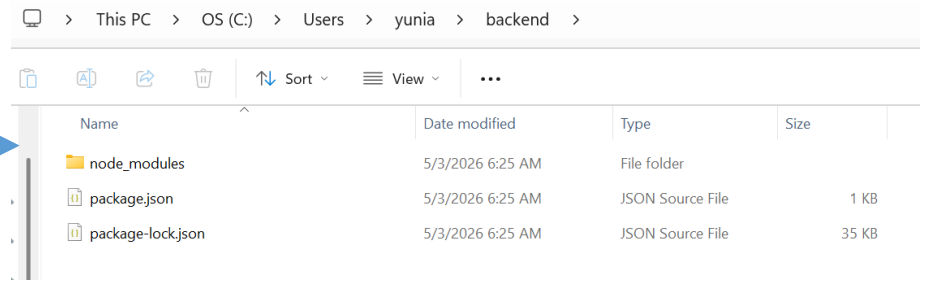
- Cek isi table “Mahasiswa”

```
kampus_db=# SELECT *FROM mahasiswa;
 id | nama |   nim   |     jurusan
----+------+---------+-----------------
  1 | Andi | 2023001 | Informatika
  2 | Siti | 2023002 | Sistem Informasi
(2 rows)
```

# BAGIAN 2 BACKEND (EXPRESS.JS + API)

- Setup Project Backend

```
Command Prompt

C:\Users\yunia>mkdir backend

C:\Users\yunia>cd backend

C:\Users\yunia\backend>npm init -y
Wrote to C:\Users\yunia\backend\package.json:

{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs"
}



C:\Users\yunia\backend>npm install express cors pg

added 81 packages, and audited 82 packages in 4s

23 packages are looking for funding
  run `npm fund` for details
```

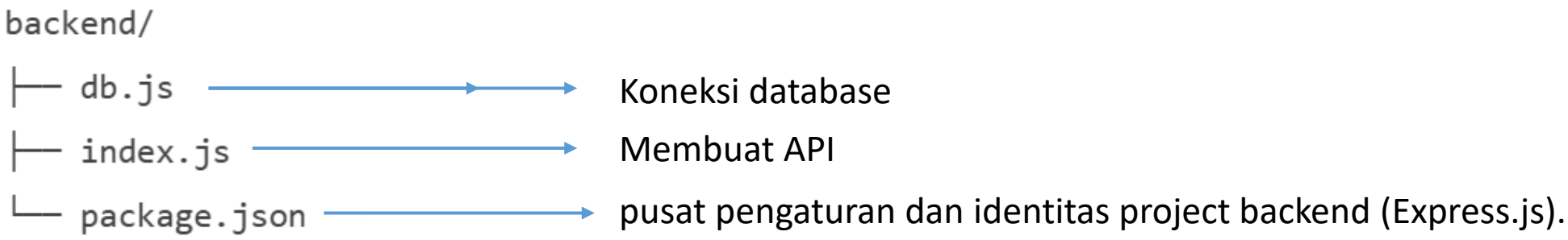



📌 Penjelasan:

- express → API server
- cors → izinkan React mengakses API
- pg → koneksi PostgreSQL

# BAGIAN 2 BACKEND (EXPRESS.JS + API)

- Struktur Folder Backend




- Buatlah isi backend diatas menggunakan Visual Code, awalnya silahkan buka folder backend di VS

# BAGIAN 2 BACKEND (EXPRESS.JS + API)

- index.js

```
const express = require("express");
const cors = require("cors");
const pool = require("./db");

const app = express();
const PORT = 5000;

// Middleware
app.use(cors());
app.use(express.json());

/* ==========================
   ROOT ROUTE (cek server)
========================== */
app.get("/", (req, res) => {
  res.send("Backend Express berjalan ✅");
});

/* ==========================
   GET Data Mahasiswa
========================== */
app.get("/api/mahasiswa", async (req, res) => {
  try {
    const result = await pool.query("SELECT * FROM mahasiswa");
    res.json(result.rows);
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: "Gagal mengambil data mahasiswa" });
  }
});

/* ==========================
   POST Mahasiswa Baru
========================== */
app.post("/api/mahasiswa", async (req, res) => {
  try {
    const { nama, nim, jurusan } = req.body;

    await pool.query(
      "INSERT INTO mahasiswa (nama, nim, jurusan) VALUES ($1, $2, $3)",
      [nama, nim, jurusan]
    );

    res.json({ message: "Mahasiswa berhasil ditambahkan ✅" });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: "Gagal menambahkan mahasiswa" });
  }
});

/* ==========================
   Jalankan Server
========================== */
app.listen(PORT, () => {
  console.log(`Server berjalan di http://localhost:${PORT}`);
});
```

# BAGIAN 2 BACKEND (EXPRESS.JS + API)

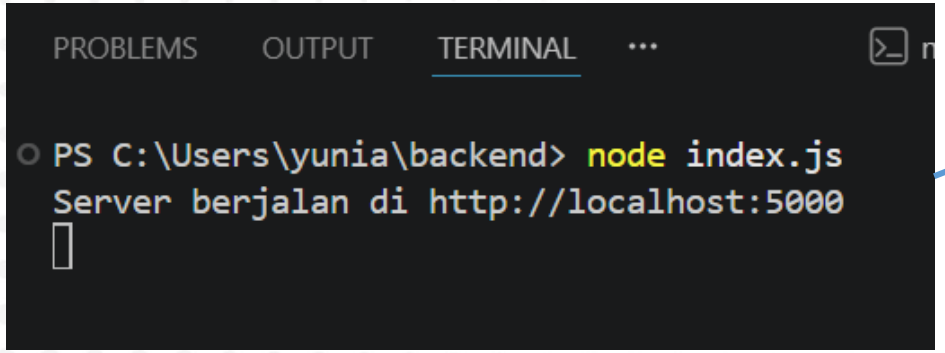
- db.js

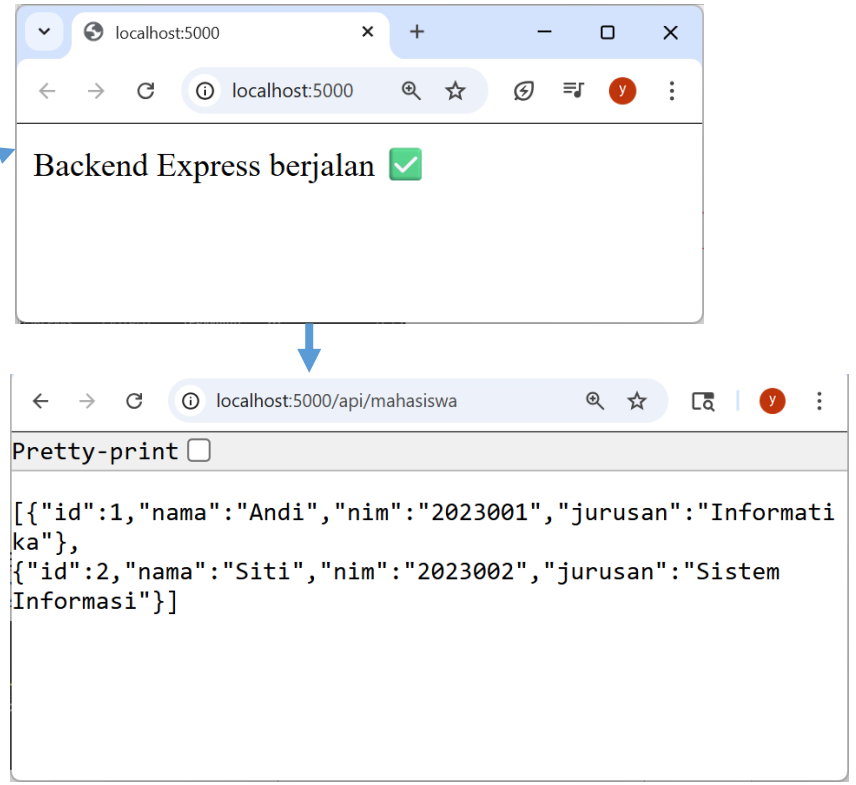
```
const { Pool } = require("pg");

const pool = new Pool({
  user: "postgres",
  host: "localhost",
  database: "kampus_db",
  password: "",
  port: 5432,
});

module.exports = pool;
```

Isi password postgresql kalian

# BAGIAN 2 BACKEND (EXPRESS.JS + API)

- Jalankan index.js di terminal VS ➔ Pada bagian atas pilih terminal – New terminal dan jalankan node index.js, lalu tampilkan http://localhost:5000 di browser

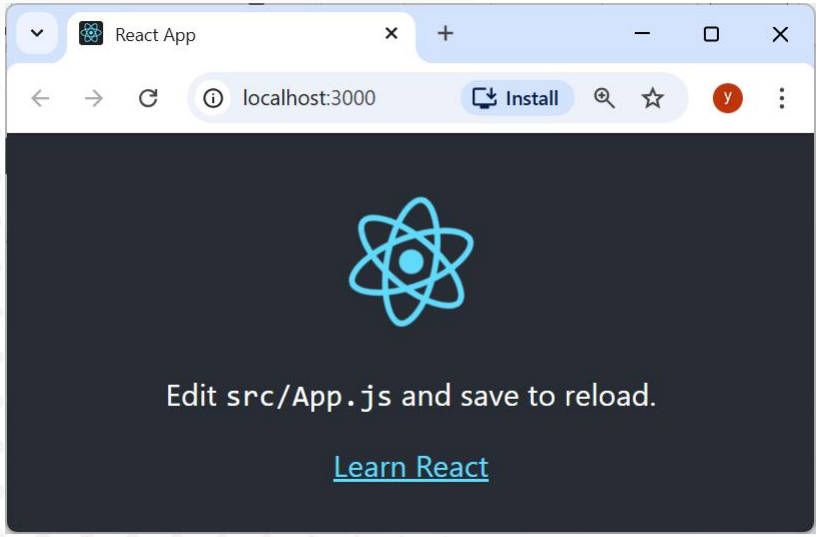


# BAGIAN 3 FRONTEND (REACT.JS)

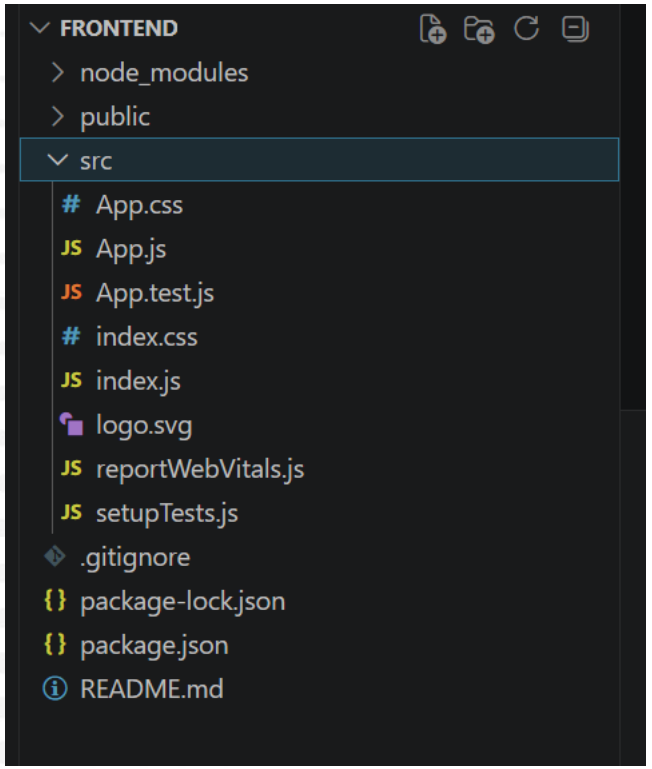
- Buat project react ➔ di terminal

```
npx create-react-app frontend
cd frontend
npm start
```



# BAGIAN 3 FRONTEND (REACT.JS)

Arsitektur Kode React (app.js) ada di src



# BAGIAN 3 FRONTEND (REACT.JS)

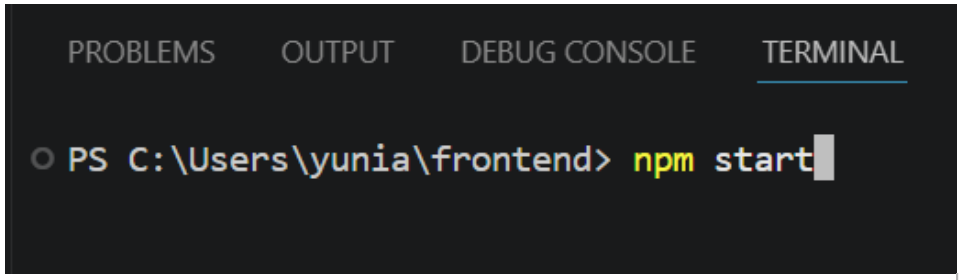
## 2. Kode React (app.js)

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

function App() {
  const [mahasiswa, setMahasiswa] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetch("http://localhost:5000/api/mahasiswa")
      .then((res) => res.json())
      .then((data) => {
        console.log("DATA DARI API:", data); // WAJIB ADA
        setMahasiswa(data);
        setLoading(false);
      })
      .catch((error) => {
        console.error("ERROR FETCH:", error);
        setLoading(false);
      });
  }, []);
```

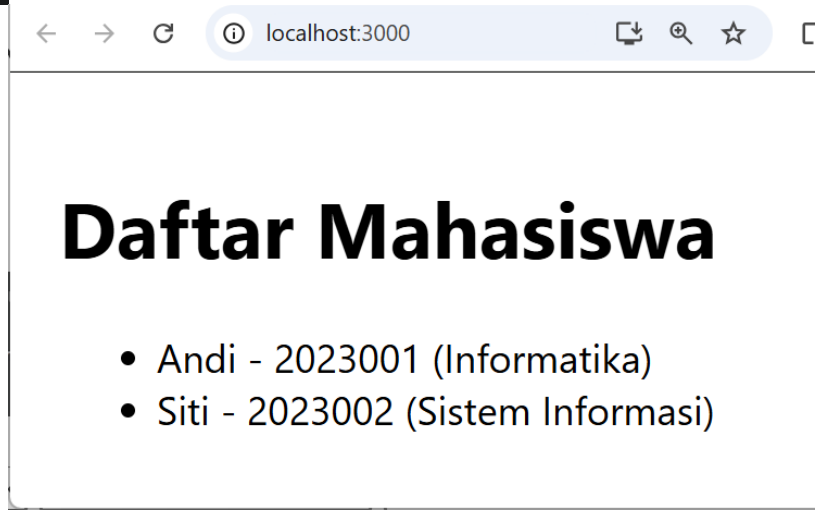
```jsx
function App() {

  if (loading) {
    return <h3>Loading data...</h3>;
  }

  return (
    <div style={{ padding: "20px" }}>
      <h1>Daftar Mahasiswa</h1>

      {mahasiswa.length === 0 ? (
        <p>Data mahasiswa kosong</p>
      ) : (
        <ul>
          {mahasiswa.map((mhs) => (
            <li key={mhs.id}>
              {mhs.nama} - {mhs.nim} ({mhs.jurusan})
            </li>
          ))}
        </ul>
      )}
    </div>
  );
}

export default App;
```

# BAGIAN 3 FRONTEND (REACT.JS)

- Jalankan React




- Browser akan otomatis terbuka di:http://localhost:3000



# Latihan Praktikum

# LATIHAN

- Buatlah study kasus yang lain tentang materi diatas Dimana react js, express js dan database terintegrasi jadi satu untuk menampilkan aplikasi sederhana