Dokumentasi Project Mobile IF-Kelas5

Nama : Lorensius Firngadi
NPM : 24382001P
Judul : Aplikasi Cetak Resi
Deskripsi Aplikasi :
Belajar informatika jadi lebih seru dengan "Informatika Kelas 5"! Aplikasi ini membantu siswa memahami materi lewat pembelajaran interaktif, latihan soal, dan fitur akun guru-siswa untuk memantau perkembangan belajar.

A. Membuat Project Backend (API)

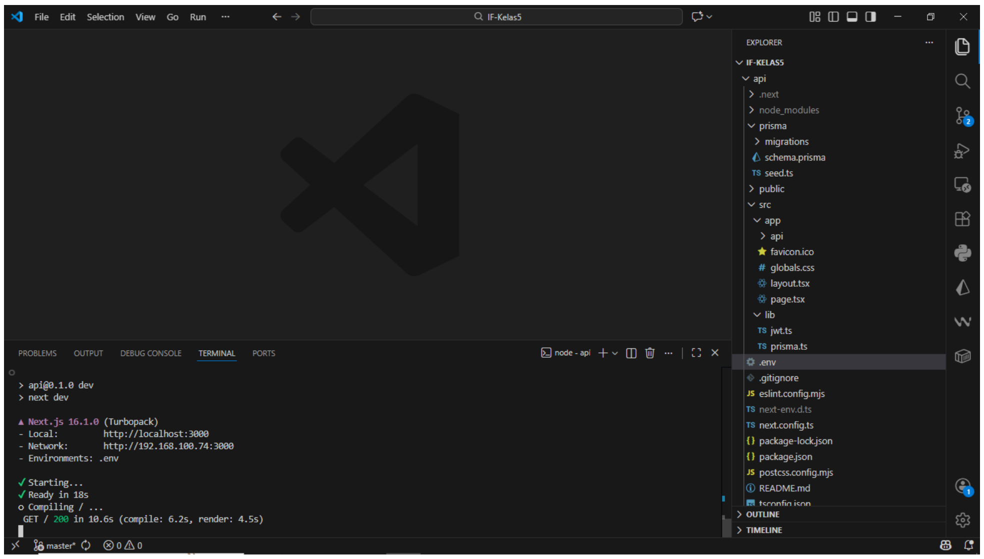
1. Inisialisasi Project Next.js
   Ketik kode di Terminal :

   ```
   npx create-next-app@latest api –typescript
   ```

   Pilih:
   - App Router ✅
   - TypeScript ✅
   - ESLint ✅
   - src/ directory ✅

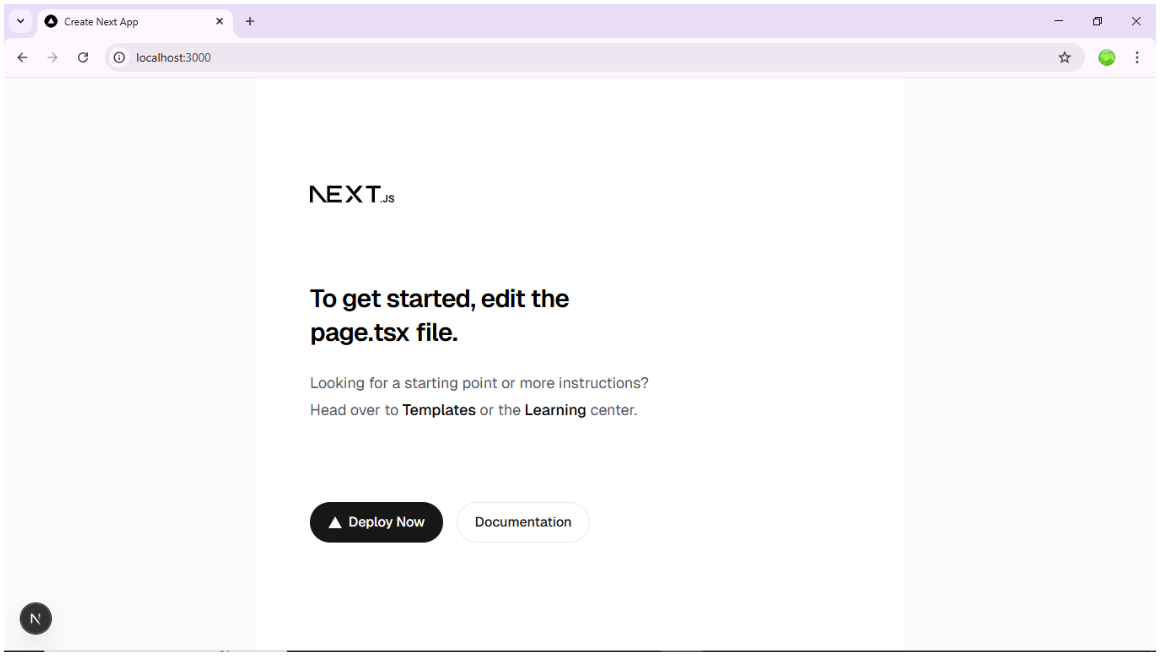
2. Menjalankan Server Awal
   Ketik kode di Terminal :

   ```
   npm run dev
   ```



```
> api@0.1.0 dev
> next dev

▲ Next.js 16.1.0 (Turbopack)
- Local:        http://localhost:3000
- Network:      http://192.168.100.74:3000
- Environments: .env

✓ Starting...
✓ Ready in 18s
○ Compiling / ...
 GET / 200 in 10.6s (compile: 6.2s, render: 4.5s)
```

Buka di Chrome atau Browser lainnya :
http://localhost:3000
Jika muncul halaman Next.js → berhasil



To get started, edit the page.tsx file.

Looking for a starting point or more instructions? Head over to Templates or the Learning center.

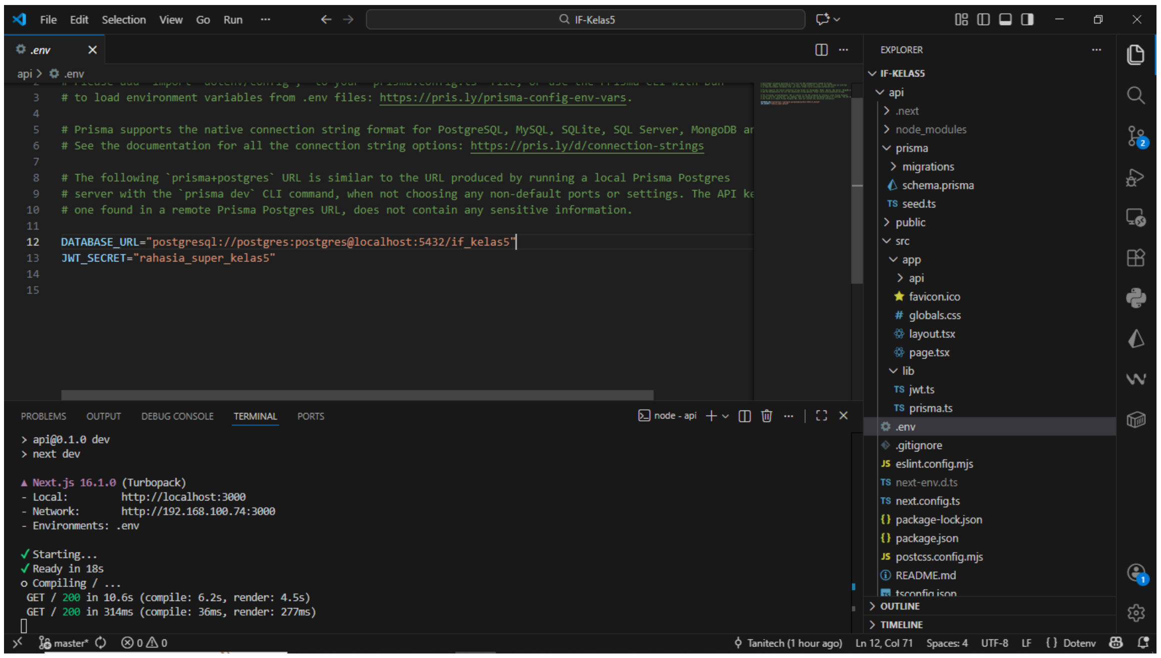
3. Install Prisma & Library Pendukung
   Ketik di terminal kode ini :
   npm install prisma @prisma/client
   dan
   npm install bcryptjs jsonwebtoken

   Untuk TypeScript:
   npm install -D @types/jsonwebtoken

4. Setup Prisma & Database
   Inisialisasi Prisma
   Kode :
   npx prisma init

   Setelah selesai akan dibuatkan folder : prisma/schema.prisma dan .env

5. Konfigurasi .env



```
# to load environment variables from .env files: https://pris.ly/prisma-config-env-vars.

# Prisma supports the native connection string format for PostgreSQL, MySQL, SQLite, SQL Server, MongoDB ar
# See the documentation for all the connection string options: https://pris.ly/d/connection-strings

# The following `prisma+postgres` URL is similar to the URL produced by running a local Prisma Postgres
# server with the `prisma dev` CLI command, when not choosing any non-default ports or settings. The API ke
# one found in a remote Prisma Postgres URL, does not contain any sensitive information.

DATABASE_URL="postgresql://postgres:postgres@localhost:5432/if_kelas5"
JWT_SECRET="rahasia_super_kelas5"
```

```
> api@0.1.0 dev
> next dev

▲ Next.js 16.1.0 (Turbopack)
- Local:        http://localhost:3000
- Network:      http://192.168.100.74:3000
- Environments: .env

✓ Starting...
✓ Ready in 18s
o Compiling / ...
 GET / 200 in 10.6s (compile: 6.2s, render: 4.5s)
 GET / 200 in 314ms (compile: 36ms, render: 277ms)
```

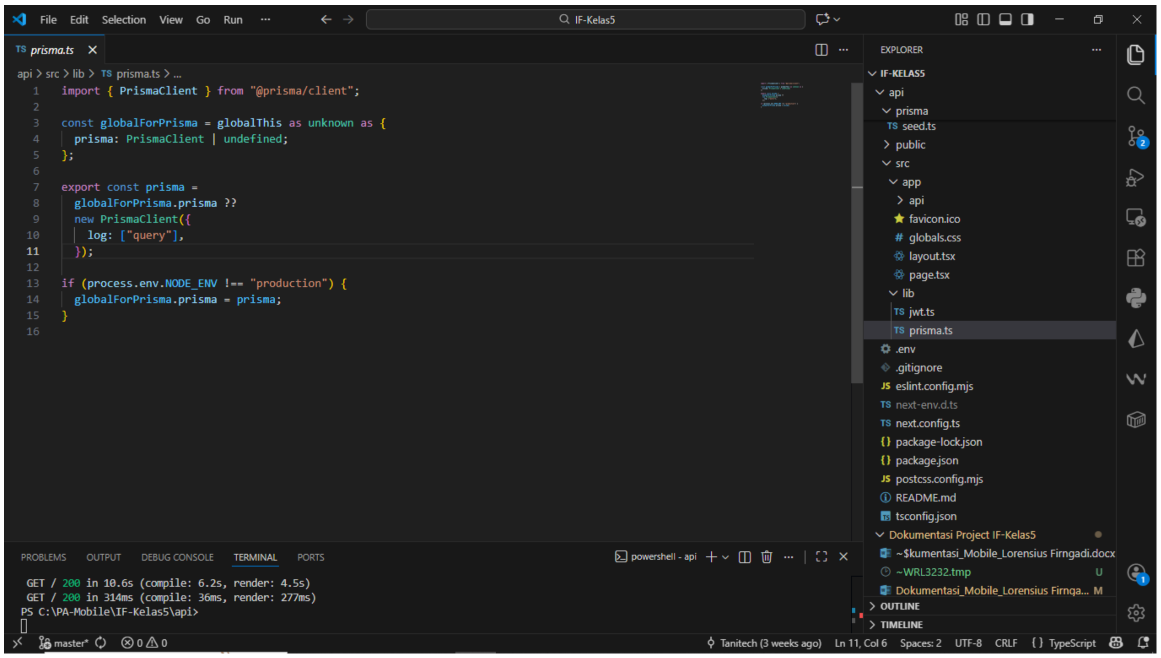
Untuk menyimpan data sensitif dan konfigurasi aplikasi secara terpisah dari kode sumber (source code). Menggunakan format untuk keamanan, Fleksibilitas dan Standar dalam pengembangan perangkat lunak modern.

6. Membuat Schema Database
   Kode :
   npx prisma migrate dev --name init

7. Migrasi Database

8. Membuka Prisma Studio
   Kode :
   npx prisma studio

9. Setup Prisma Client
   src/lib/prisma.ts

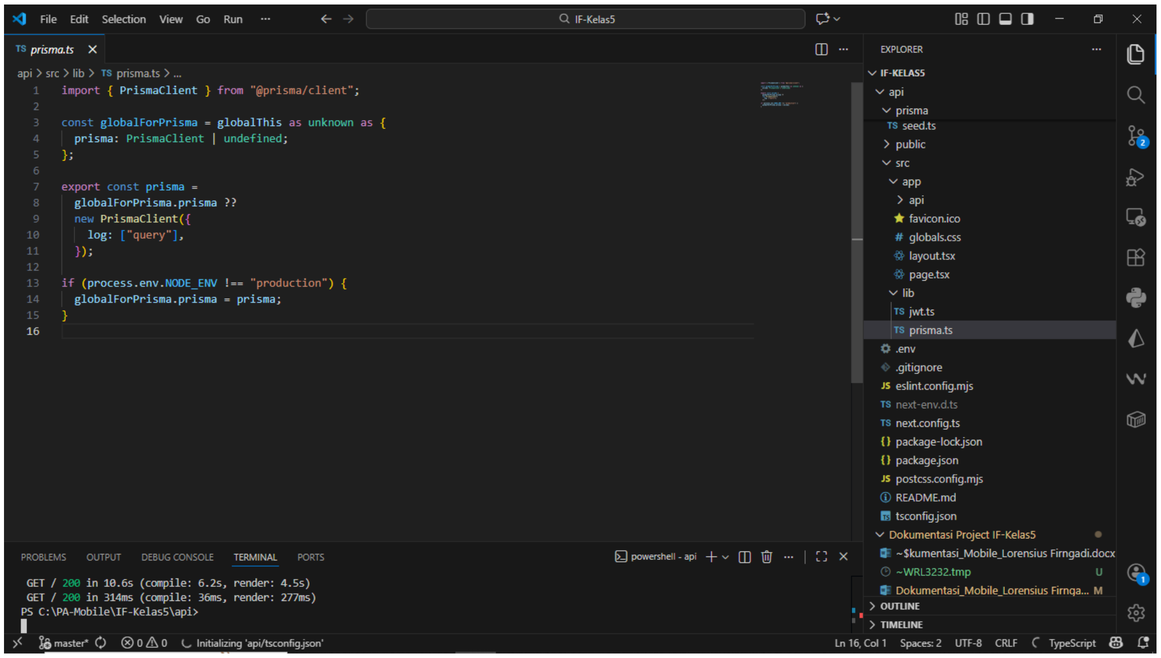


```
import { PrismaClient } from "@prisma/client";

const globalForPrisma = globalThis as unknown as {
  prisma: PrismaClient | undefined;
};

export const prisma =
  globalForPrisma.prisma ??
  new PrismaClient({
    log: ["query"],
  });

if (process.env.NODE_ENV !== "production") {
  globalForPrisma.prisma = prisma;
}
```

```
GET / 200 in 10.6s (compile: 6.2s, render: 4.5s)
GET / 200 in 314ms (compile: 36ms, render: 277ms)
PS C:\PA-Mobile\IF-Kelas5\api>
```

Menginisialisasi Prisma Client untuk menghubungkan kode program dengan database. Berfungsi : Membuat variabel yang bisa diimpor di file lain untuk melakukan query, dan Menghindari pembuatan instance baru setiap kali refresh saat proses development.

10. Setup JWT



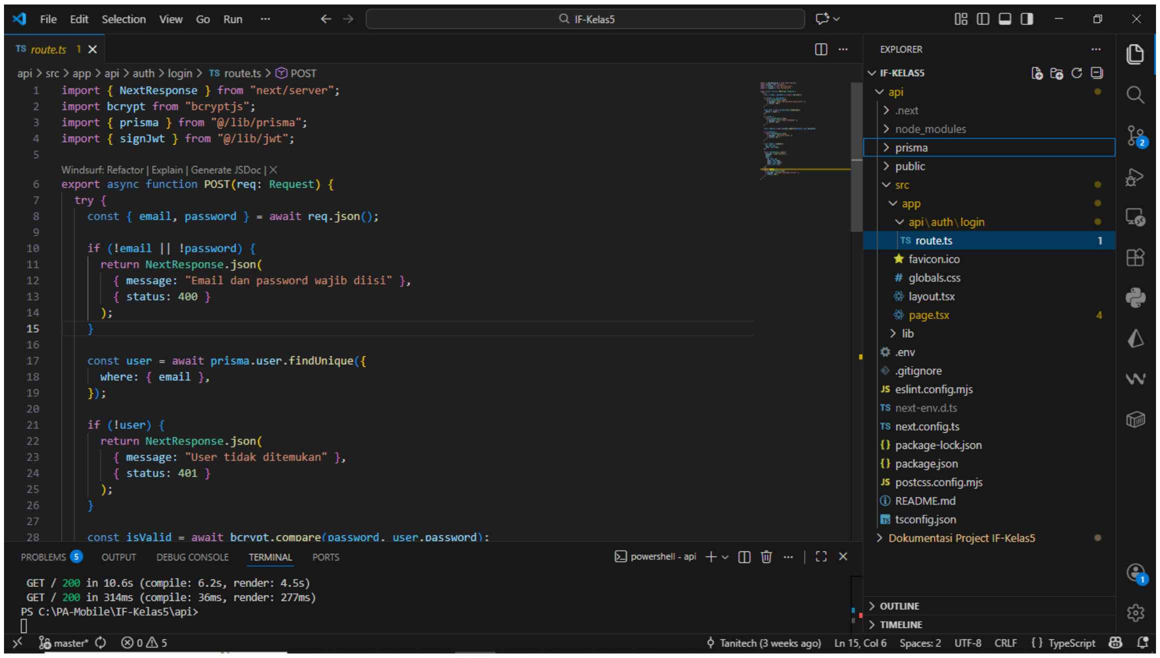
```
import { PrismaClient } from "@prisma/client";

const globalForPrisma = globalThis as unknown as {
  prisma: PrismaClient | undefined;
};

export const prisma =
  globalForPrisma.prisma ??
  new PrismaClient({
    log: ["query"],
  });

if (process.env.NODE_ENV !== "production") {
  globalForPrisma.prisma = prisma;
}
```

```
GET / 200 in 10.6s (compile: 6.2s, render: 4.5s)
GET / 200 in 314ms (compile: 36ms, render: 277ms)
PS C:\PA-Mobile\IF-Kelas5\api>
```

Sebuah Utility Module yang berfungsi untuk menangani sistem keamanan login menggunakan JSON Web Token (JWT). Mengambil kunci rahasia dari variabel lingkungan (.env).

11. Membuat API Login
    Struktur Folder API :
    src/app/api/auth/login/route.ts.

    Kode API Login
    route.ts



```
import { NextResponse } from "next/server";
import bcrypt from "bcryptjs";
import { prisma } from "@/lib/prisma";
import { signJwt } from "@/lib/jwt";

export async function POST(req: Request) {
  try {
    const { email, password } = await req.json();

    if (!email || !password) {
      return NextResponse.json(
        { message: "Email dan password wajib diisi" },
        { status: 400 }
      );
    }

    const user = await prisma.user.findUnique({
      where: { email },
    });

    if (!user) {
      return NextResponse.json(
        { message: "User tidak ditemukan" },
        { status: 401 }
      );
    }

    const isValid = await bcrypt.compare(password, user.password);
```

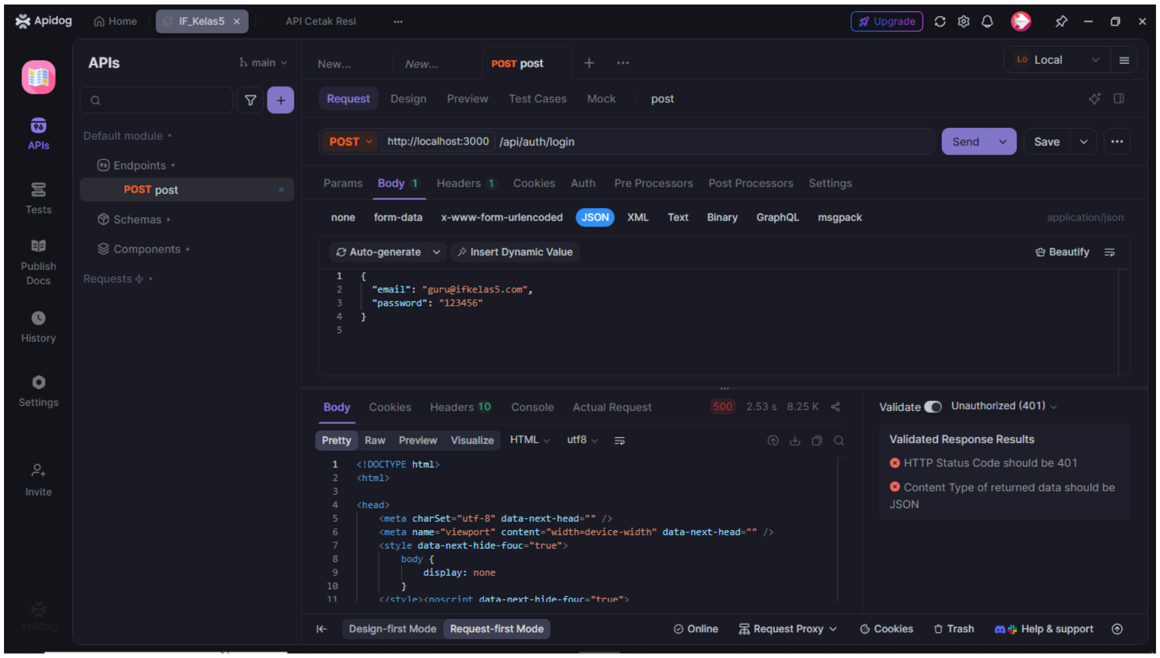
```
 GET / 200 in 10.6s (compile: 6.2s, render: 4.5s)
 GET / 200 in 314ms (compile: 36ms, render: 277ms)
PS C:\PA-Mobile\IF-Kelas5\api>
```

Logika Backend untuk Proses Login fungsi utamanya :

1. Validasi & Keamanan: Memastikan email/password sudah diisi dan memeriksa apakah password yang dimasukkan cocok dengan yang ada di database (menggunakan enkripsi bcrypt).
2. Identifikasi User: Mencari data pengguna di database menggunakan Prisma berdasarkan email yang dimasukkan.
3. Pemberian Izin (Token): Jika data cocok, fungsi ini membuatkan JWT (JSON Web Token) sebagai "kartu akses" digital agar user bisa mengakses fitur lain tanpa harus login ulang terus-menerus.

Alurnya: Input Data > Cek Database > Verifikasi Password > Kirim Token Akses.

12. Testing API dengan Apidog



POST http://localhost:3000 /api/auth/login

Body: JSON

```
{
  "email": "guru@ifkelas5.com",
  "password": "123456"
}
```

500 · 2.53 s · 8.25 K

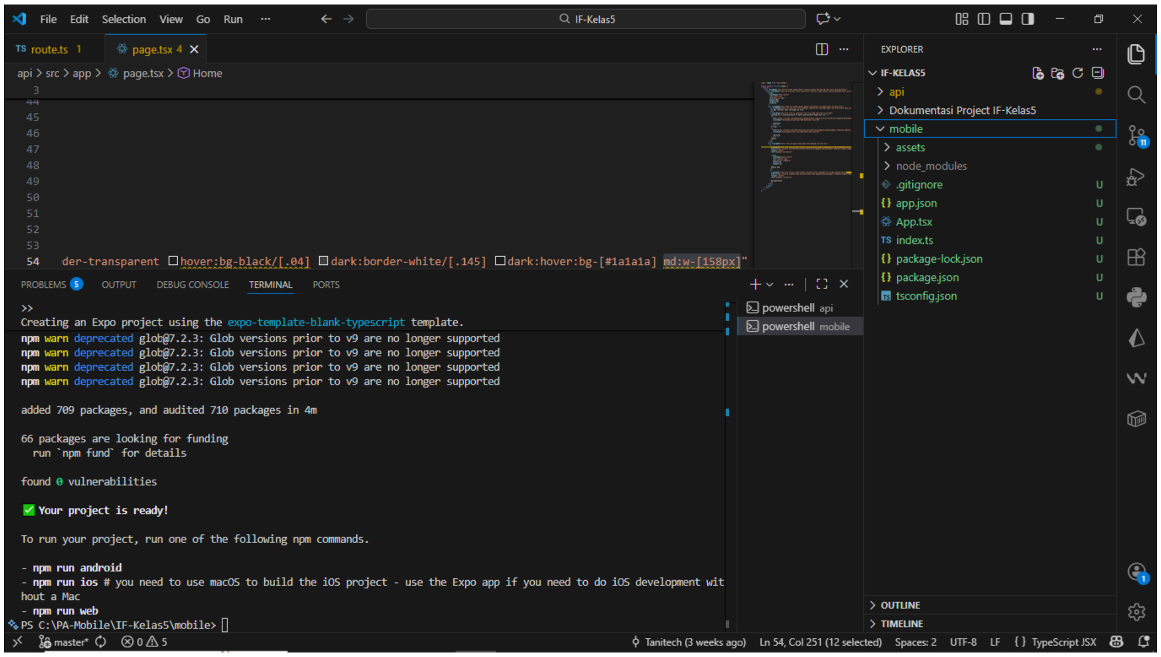
```
<!DOCTYPE html>
<html>

<head>
    <meta charSet="utf-8" data-next-head="" />
    <meta name="viewport" content="width=device-width" data-next-head="" />
    <style data-next-hide-fouc="true">
        body {
            display: none
        }
    </style><noscript data-next-hide-fouc="true">
```

Validated Response Results
- HTTP Status Code should be 401
- Content Type of returned data should be JSON

Untuk sementara apidog masih error.

B. Membuat Project Frontend (Folder mobile)

1. Membuat Project React Native (Expo)
   Jalankan pada terminal folder mobile :
   npx create-expo-app . -t expo-template-blank-typescript



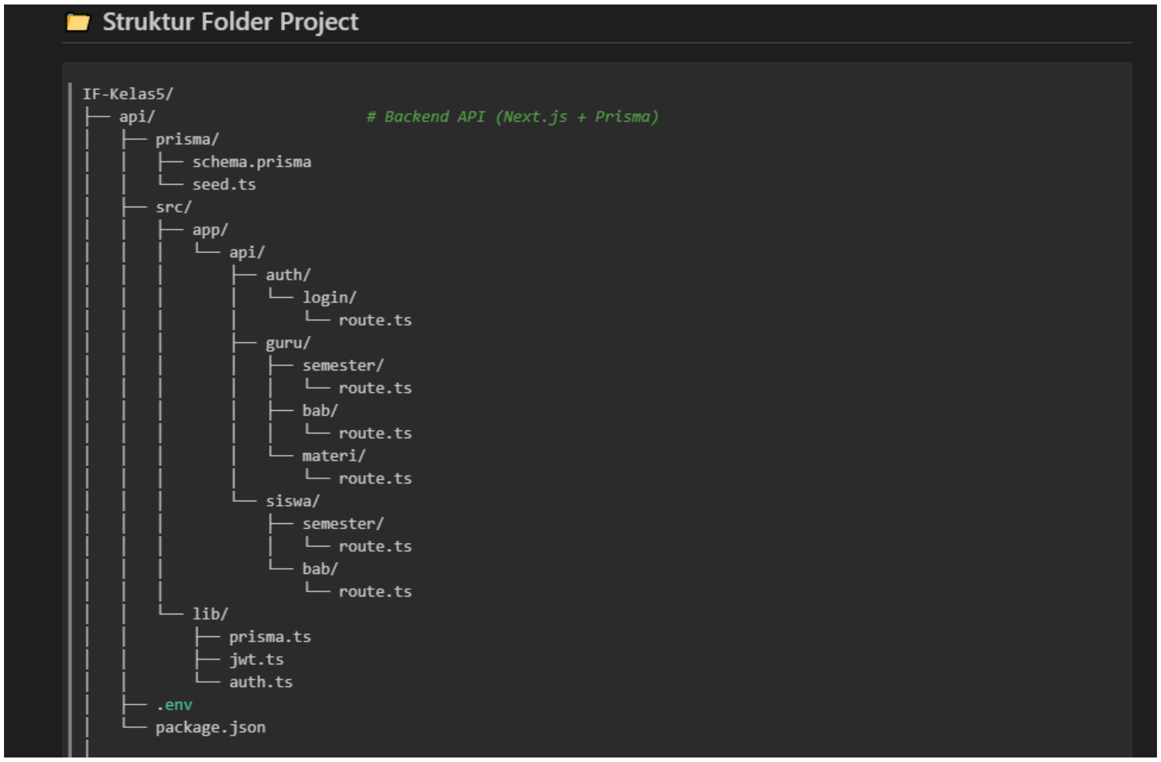
```
>>
Creating an Expo project using the expo-template-blank-typescript template.
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported

added 709 packages, and audited 710 packages in 4m

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

found 0 vulnerabilities

✅ Your project is ready!

To run your project, run one of the following npm commands.

- npm run android
- npm run ios # you need to use macOS to build the iOS project - use the Expo app if you need to do iOS development wit
hout a Mac
- npm run web
PS C:\PA-Mobile\IF-Kelas5\mobile> 
```

Proses install expo selesai.

2. Install Dependency
   Navigasi & API
   Kode :
   npm install axios
   npm install @react-navigation/native
   npm install @react-navigation/native-stack

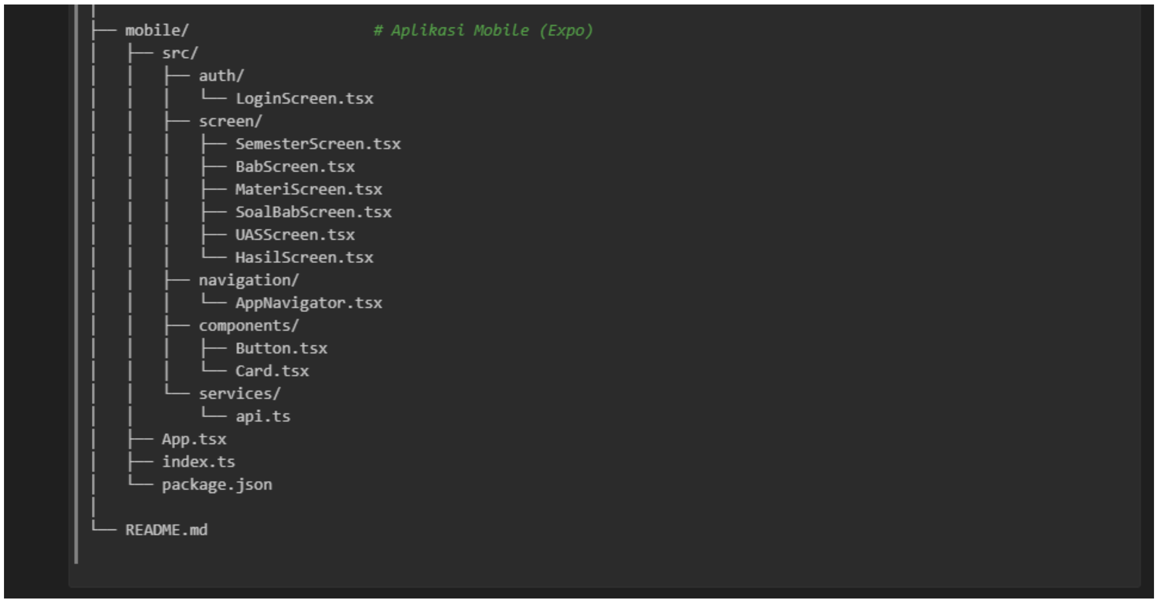
   Dependency Expo Navigation
   npx expo install react-native-screens react-native-safe-area-context

3. Susunan Struktur Folder



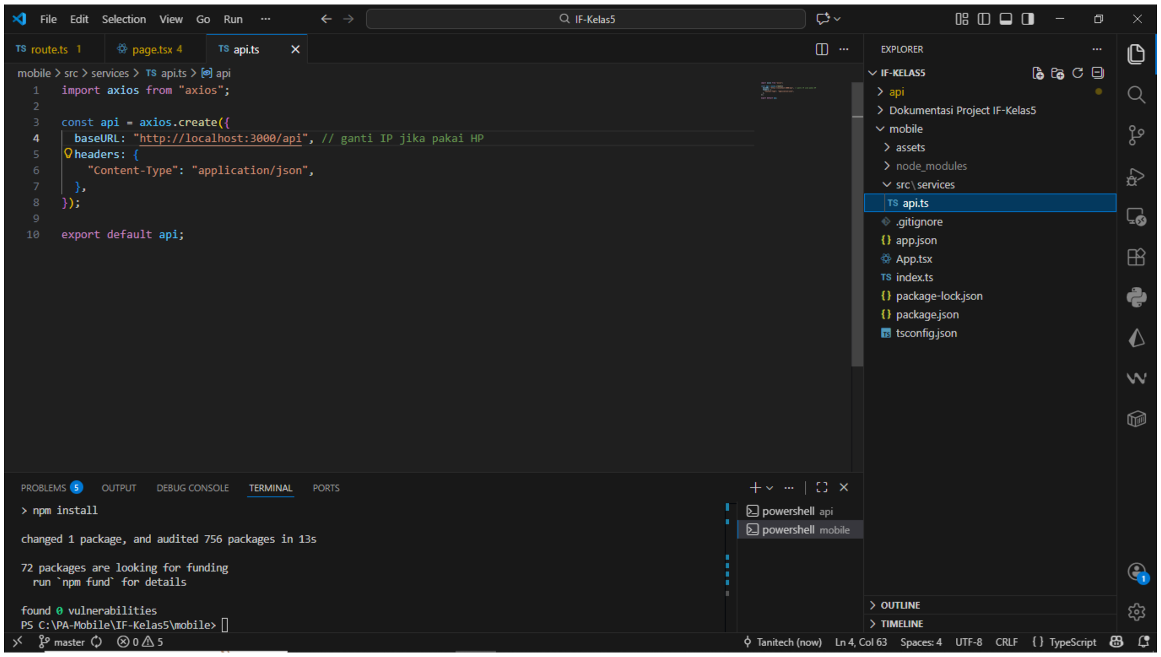
Struktur Folder Project

```
IF-Kelas5/
├── api/                        # Backend API (Next.js + Prisma)
│   ├── prisma/
│   │   ├── schema.prisma
│   │   └── seed.ts
│   ├── src/
│   │   ├── app/
│   │   │   └── api/
│   │   │       ├── auth/
│   │   │       │   └── login/
│   │   │       │       └── route.ts
│   │   │       ├── guru/
│   │   │       │   ├── semester/
│   │   │       │   │   └── route.ts
│   │   │       │   ├── bab/
│   │   │       │   │   └── route.ts
│   │   │       │   └── materi/
│   │   │       │       └── route.ts
│   │   │       └── siswa/
│   │   │           ├── semester/
│   │   │           │   └── route.ts
│   │   │           └── bab/
│   │   │               └── route.ts
│   │   └── lib/
│   │       ├── prisma.ts
│   │       ├── jwt.ts
│   │       └── auth.ts
│   ├── .env
│   └── package.json
│
```



```
│
├── mobile/                    # Aplikasi Mobile (Expo)
│   ├── src/
│   │   ├── auth/
│   │   │   └── LoginScreen.tsx
│   │   ├── screen/
│   │   │   ├── SemesterScreen.tsx
│   │   │   ├── BabScreen.tsx
│   │   │   ├── MateriScreen.tsx
│   │   │   ├── SoalBabScreen.tsx
│   │   │   ├── UASScreen.tsx
│   │   │   └── HasilScreen.tsx
│   │   ├── navigation/
│   │   │   └── AppNavigator.tsx
│   │   ├── components/
│   │   │   ├── Button.tsx
│   │   │   └── Card.tsx
│   │   └── services/
│   │       └── api.ts
│   ├── App.tsx
│   ├── index.ts
│   └── package.json
│
└── README.md
```

4. Konfigurasi API (services/api.ts)

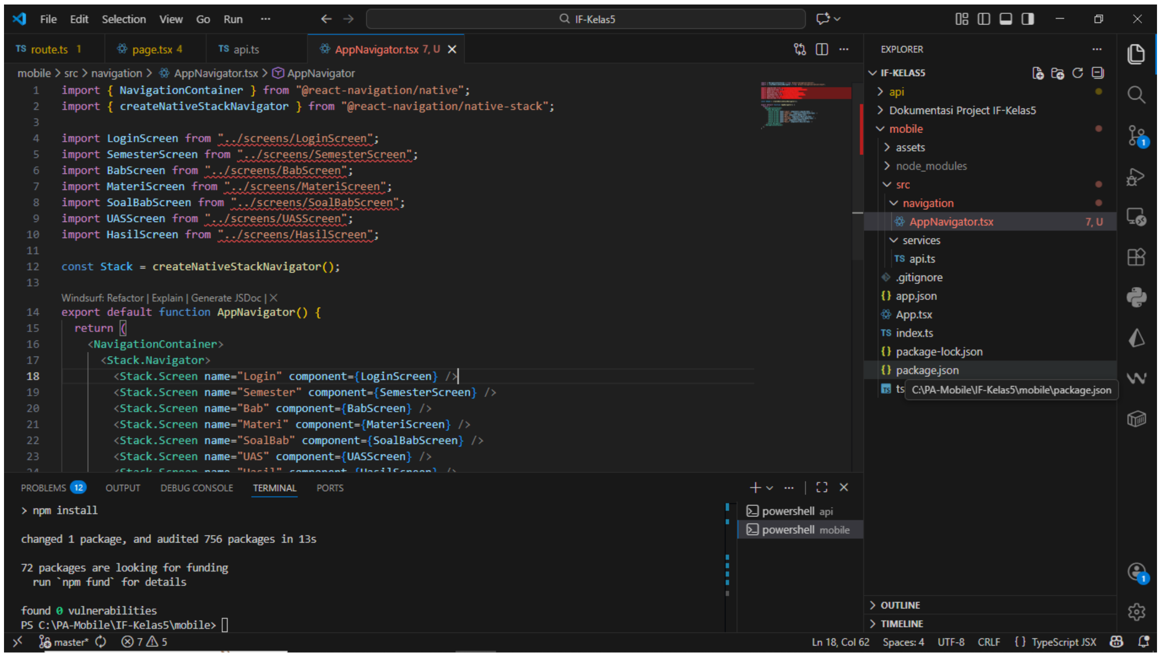


```ts
import axios from "axios";

const api = axios.create({
  baseURL: "http://localhost:3000/api", // ganti IP jika pakai HP
  headers: {
    "Content-Type": "application/json",
  },
});

export default api;
```

```
> npm install

changed 1 package, and audited 756 packages in 13s

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

found 0 vulnerabilities
PS C:\PA-Mobile\IF-Kelas5\mobile>
```

Berfungsi sebagai konfigurasi terpusat (instance) untuk melakukan HTTP request (seperti GET, POST, DELETE) dari aplikasi ke server backend.

5. Navigasi Aplikasi (navigation/AppNavigator.tsx)

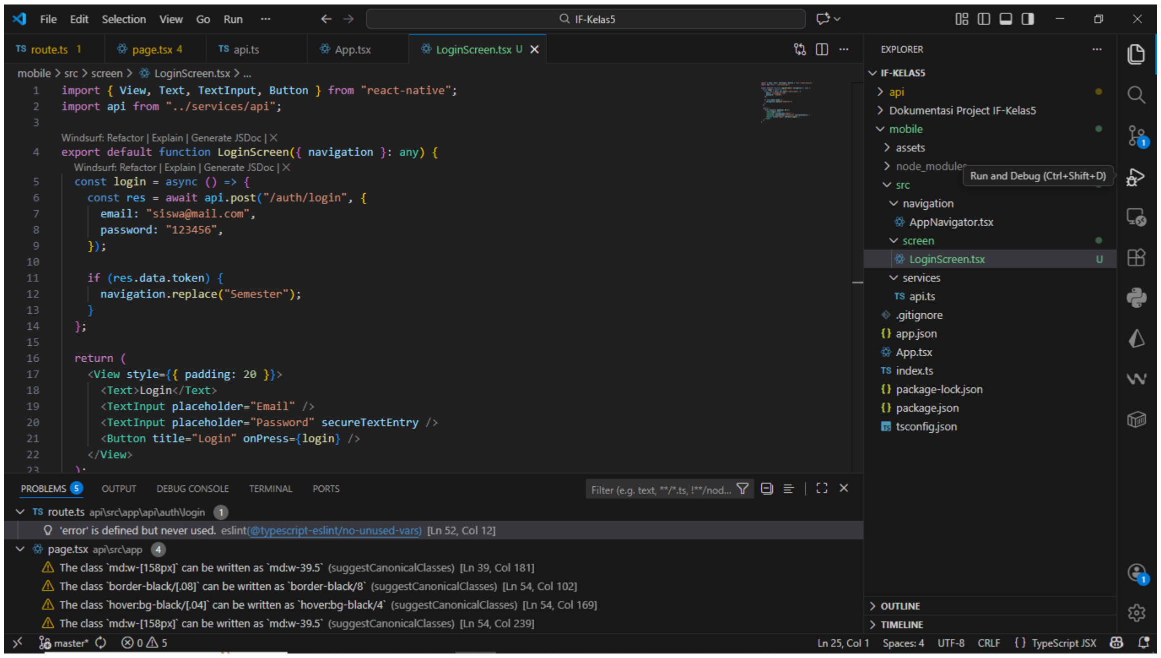


```tsx
import { NavigationContainer } from "@react-navigation/native";
import { createNativeStackNavigator } from "@react-navigation/native-stack";

import LoginScreen from "../screens/LoginScreen";
import SemesterScreen from "../screens/SemesterScreen";
import BabScreen from "../screens/BabScreen";
import MateriScreen from "../screens/MateriScreen";
import SoalBabScreen from "../screens/SoalBabScreen";
import UASScreen from "../screens/UASScreen";
import HasilScreen from "../screens/HasilScreen";

const Stack = createNativeStackNavigator();

export default function AppNavigator() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Login" component={LoginScreen} />
        <Stack.Screen name="Semester" component={SemesterScreen} />
        <Stack.Screen name="Bab" component={BabScreen} />
        <Stack.Screen name="Materi" component={MateriScreen} />
        <Stack.Screen name="SoalBab" component={SoalBabScreen} />
        <Stack.Screen name="UAS" component={UASScreen} />
```

```
> npm install

changed 1 package, and audited 756 packages in 13s

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

found 0 vulnerabilities
PS C:\PA-Mobile\IF-Kelas5\mobile>
```

Sistem Navigasi Utama (App Navigator) pada aplikasi. Berfungsi sebagai pengatur alur perpindahan halaman (routing) di dalam aplikasi React Native menggunakan library React Navigation.

6. Screen LOGIN (LoginScreen.tsx)

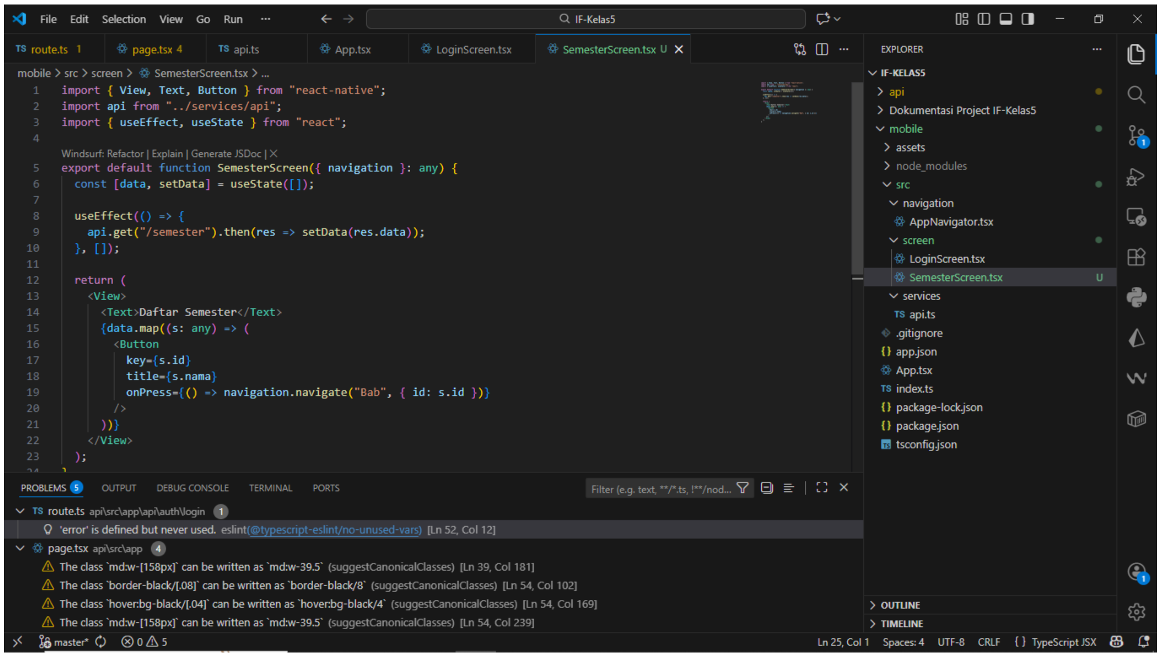


```tsx
import { View, Text, TextInput, Button } from "react-native";
import api from "../services/api";

export default function LoginScreen({ navigation }: any) {
  const login = async () => {
    const res = await api.post("/auth/login", {
      email: "siswa@mail.com",
      password: "123456",
    });

    if (res.data.token) {
      navigation.replace("Semester");
    }
  };

  return (
    <View style={{ padding: 20 }}>
      <Text>Login</Text>
      <TextInput placeholder="Email" />
      <TextInput placeholder="Password" secureTextEntry />
      <Button title="Login" onPress={login} />
    </View>
```

```
route.ts api\src\app\api\auth\login 1
  'error' is defined but never used. eslint(@typescript-eslint/no-unused-vars) [Ln 52, Col 12]
page.tsx api\src\app 4
  The class `md:w-[158px]` can be written as `md:w-39.5` (suggestCanonicalClasses) [Ln 39, Col 181]
  The class `border-black/[.08]` can be written as `border-black/8` (suggestCanonicalClasses) [Ln 54, Col 102]
  The class `hover:bg-black/[.04]` can be written as `hover:bg-black/4` (suggestCanonicalClasses) [Ln 54, Col 169]
  The class `md:w-[158px]` can be written as `md:w-39.5` (suggestCanonicalClasses) [Ln 54, Col 239]
```

Berfungsi sebagai gerbang masuk (autentikasi) pengguna ke dalam aplikasi. Alurnya : Proses Autentikasi (login function) dan Antarmuka Pengguna (UI).

7. Ambil Data Semester (SemesterScreen.tsx)

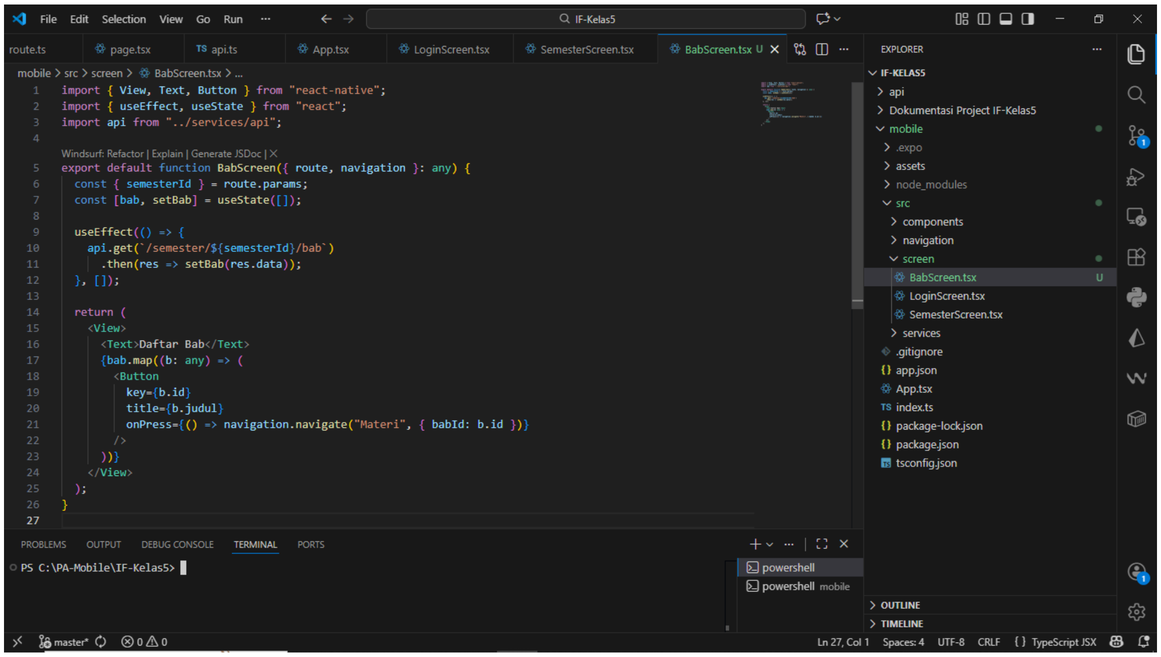


```tsx
import { View, Text, Button } from "react-native";
import api from "../services/api";
import { useEffect, useState } from "react";

export default function SemesterScreen({ navigation }: any) {
  const [data, setData] = useState([]);

  useEffect(() => {
    api.get("/semester").then(res => setData(res.data));
  }, []);

  return (
    <View>
      <Text>Daftar Semester</Text>
      {data.map((s: any) => (
        <Button
          key={s.id}
          title={s.nama}
          onPress={() => navigation.navigate("Bab", { id: s.id })}
        />
      ))}
    </View>
  );
```

```
PROBLEMS 5
route.ts api\src\app\api\auth\login 1
  'error' is defined but never used. eslint(@typescript-eslint/no-unused-vars) [Ln 52, Col 12]
page.tsx api\src\app 4
  The class `md:w-[158px]` can be written as `md:w-39.5` (suggestCanonicalClasses) [Ln 39, Col 181]
  The class `border-black/[.08]` can be written as `border-black/8` (suggestCanonicalClasses) [Ln 54, Col 102]
  The class `hover:bg-black/[.04]` can be written as `hover:bg-black/4` (suggestCanonicalClasses) [Ln 54, Col 169]
  The class `md:w-[158px]` can be written as `md:w-39.5` (suggestCanonicalClasses) [Ln 54, Col 239]
```

Untuk menampilkan daftar pilihan semester yang datanya diambil langsung dari database melalui API. Pengambilan Data (Data Fetching), Rendering Dinamis (Mapping), dan Navigasi dengan Parameter.

8. BabScreen.tsx



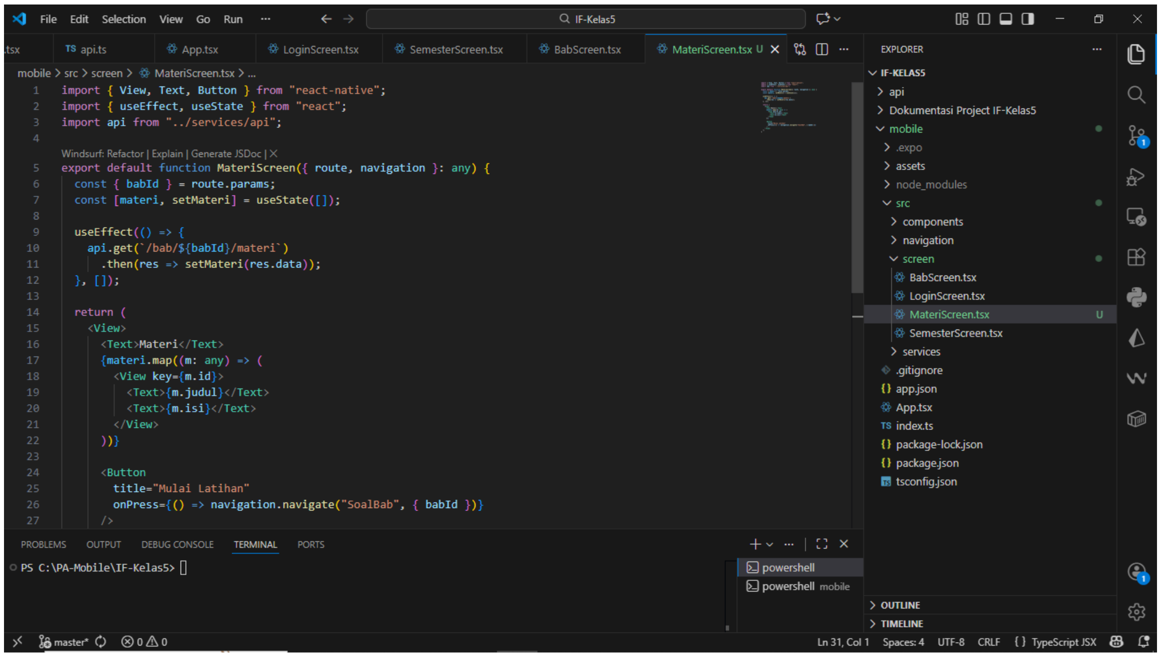
```tsx
import { View, Text, Button } from "react-native";
import { useEffect, useState } from "react";
import api from "../services/api";

export default function BabScreen({ route, navigation }: any) {
  const { semesterId } = route.params;
  const [bab, setBab] = useState([]);

  useEffect(() => {
    api.get(`/semester/${semesterId}/bab`)
      .then(res => setBab(res.data));
  }, []);

  return (
    <View>
      <Text>Daftar Bab</Text>
      {bab.map((b: any) => (
        <Button
          key={b.id}
          title={b.judul}
          onPress={() => navigation.navigate("Materi", { babId: b.id })}
        />
      ))}
    </View>
  );
}
```

```
PS C:\PA-Mobile\IF-Kelas5>
```

Screen Component pada React Native bernama BabScreen yang berfungsi untuk menampilkan daftar bab berdasarkan semester yang dipilih.
Berikut adalah penjelasan poin-poin utamanya:

- Penerimaan Data (route): Kode ini mengambil semesterId dari parameter rute (route.params). Ini berarti layar ini diekspektasikan menerima data dari layar sebelumnya.
- Pengambilan Data API: Menggunakan useEffect untuk memicu pengambilan data (fetching) dari API segera setelah layar dimuat. Endpoint yang diakses adalah /semester/${semesterId}/bab.
- Manajemen State: Menggunakan useState untuk menyimpan daftar bab yang diterima dari server ke dalam variabel bab.
- Iterasi Data (Mapping): Data bab yang berupa array diulang menggunakan fungsi .map() untuk merender komponen Button bagi setiap itemnya.
- Navigasi: Setiap tombol memiliki fungsi onPress yang akan mengarahkan pengguna ke layar "Materi" sambil membawa data babId.

9. MateriScreen.tsx



```tsx
import { View, Text, Button } from "react-native";
import { useEffect, useState } from "react";
import api from "../services/api";

export default function MateriScreen({ route, navigation }: any) {
  const { babId } = route.params;
  const [materi, setMateri] = useState([]);

  useEffect(() => {
    api.get(`/bab/${babId}/materi`)
      .then(res => setMateri(res.data));
  }, []);

  return (
    <View>
      <Text>Materi</Text>
      {materi.map((m: any) => (
        <View key={m.id}>
          <Text>{m.judul}</Text>
          <Text>{m.isi}</Text>
        </View>
      ))}

      <Button
        title="Mulai Latihan"
        onPress={() => navigation.navigate("SoalBab", { babId })}
      />
```

```
PS C:\PA-Mobile\IF-Kelas5>
```

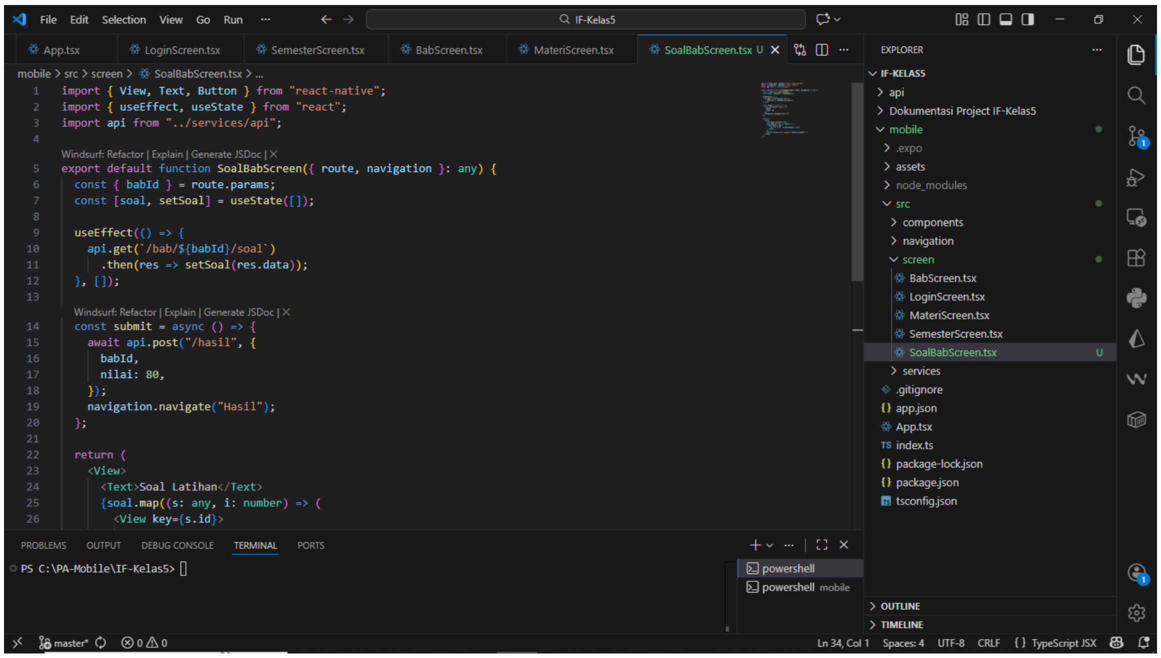
MateriScreen, yaitu layar yang berfungsi untuk menampilkan isi materi pembelajaran berdasarkan bab yang dipilih dan menyediakan akses ke latihan soal.

Berikut adalah deskripsi alur kerjanya:

- Integrasi Data: Mengambil babId dari layar sebelumnya via route.params untuk memastikan materi yang muncul sesuai dengan bab yang diklik pengguna.

- Pengambilan Konten: Menggunakan useEffect untuk melakukan fetch data dari endpoint API /bab/${babId}/materi. Data tersebut kemudian disimpan dalam state materi.
- Penyajian Informasi: Menggunakan metode .map() untuk menampilkan daftar materi. Berbeda dengan layar sebelumnya, di sini aplikasi menampilkan judul dan isi materi secara detail di dalam komponen View.
- Navigasi Lanjutan: Di bagian bawah daftar materi, terdapat tombol "Mulai Latihan" yang akan mengarahkan pengguna ke layar SoalBab dengan membawa babId sebagai referensi soal yang harus dimuat.

10. SoalBabScreen.tsx



```
import { View, Text, Button } from "react-native";
import { useEffect, useState } from "react";
import api from "../services/api";

export default function SoalBabScreen({ route, navigation }: any) {
  const { babId } = route.params;
  const [soal, setSoal] = useState([]);

  useEffect(() => {
    api.get(`/bab/${babId}/soal`)
      .then(res => setSoal(res.data));
  }, []);

  const submit = async () => {
    await api.post("/hasil", {
      babId,
      nilai: 80,
    });
    navigation.navigate("Hasil");
  };

  return (
    <View>
      <Text>Soal Latihan</Text>
      {soal.map((s: any, i: number) => (
        <View key={s.id}>
```

```
PS C:\PA-Mobile\IF-Kelas5>
```

SoalBabScreen sebagai modul evaluasi atau latihan soal untuk menguji pemahaman pengguna terhadap bab yang baru saja dipelajari.

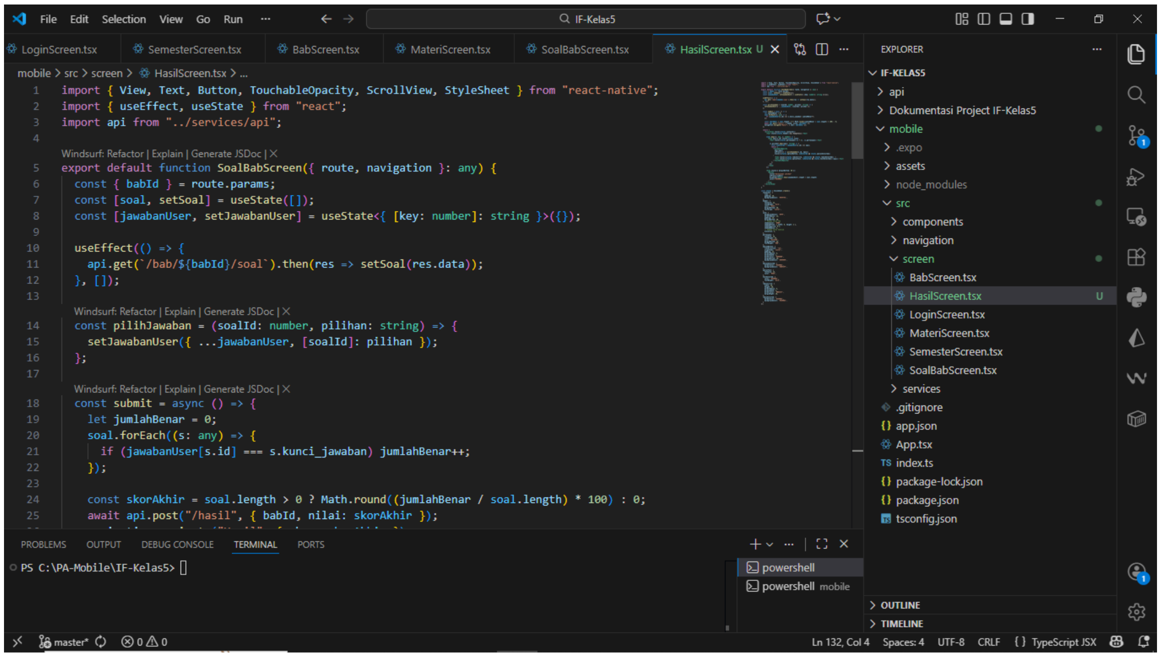
Berikut adalah rincian fungsionalitasnya:

- Pemuatan Soal: Mengambil babId dari parameter navigasi dan menggunakan useEffect untuk memanggil API guna mendapatkan daftar pertanyaan terkait bab tersebut.

- Tampilan Dinamis: Menggunakan fungsi .map() untuk merender daftar pertanyaan. Teks pertanyaan ditampilkan dengan nomor urut otomatis menggunakan indeks array (i + 1).
- Proses Pengiriman (submit):
  - ✓ Fungsi ini bersifat asynchronous (async/await).

    - ✓ Mengirimkan data hasil latihan ke endpoint /hasil melalui metode POST.
    - ✓ Dalam contoh ini, nilai dikirim secara statis (nilai: 80) sebagai placeholder.
- Alur Navigasi: Setelah proses pengiriman data ke server selesai, aplikasi secara otomatis mengarahkan pengguna ke layar "Hasil".

11. HasilScreen.tsx



```tsx
import { View, Text, Button, TouchableOpacity, ScrollView, StyleSheet } from "react-native";
import { useEffect, useState } from "react";
import api from "../services/api";

export default function SoalBabScreen({ route, navigation }: any) {
  const { babId } = route.params;
  const [soal, setSoal] = useState([]);
  const [jawabanUser, setJawabanUser] = useState<{ [key: number]: string }>({});

  useEffect(() => {
    api.get(`/bab/${babId}/soal`).then(res => setSoal(res.data));
  }, []);

  const pilihJawaban = (soalId: number, pilihan: string) => {
    setJawabanUser({ ...jawabanUser, [soalId]: pilihan });
  };

  const submit = async () => {
    let jumlahBenar = 0;
    soal.forEach((s: any) => {
      if (jawabanUser[s.id] === s.kunci_jawaban) jumlahBenar++;
    });

    const skorAkhir = soal.length > 0 ? Math.round((jumlahBenar / soal.length) * 100) : 0;
    await api.post("/hasil", { babId, nilai: skorAkhir });
```

```
PS C:\PA-Mobile\IF-Kelas5>
```

HasilScreen, yang berfungsi sebagai layar penyelesaian atau ringkasan skor setelah pengguna menyelesaikan latihan soal pada bab tertentu.

12. UASScreen.tsx

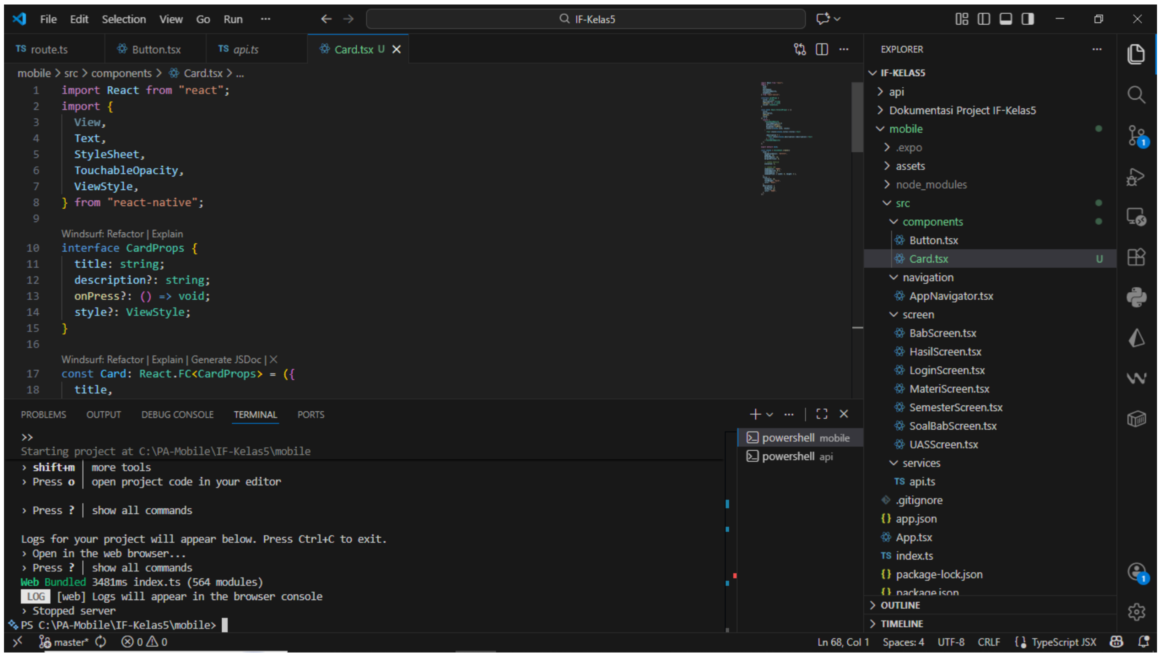
UASScreen, yang merupakan modul ujian akhir untuk mengevaluasi seluruh materi dalam satu semester.

alur kerjanya sebagai berikut :

- Pengambilan Data Terpusat: Berbeda dengan latihan per bab, layar ini memanggil endpoint /semester/1/uas, yang berarti sistem mengambil kumpulan soal ujian khusus untuk semester 1.
- State Management: Daftar soal disimpan dalam state soal dan akan diperbarui secara otomatis setelah data berhasil ditarik dari API menggunakan useEffect.
- Penyajian Soal: Menggunakan metode .map() untuk menampilkan daftar pertanyaan secara berurutan dengan nomor otomatis (i + 1).

- Aksi Selesai: Saat ini, tombol "Selesai" hanya memicu fungsi alert sederhana sebagai penanda bahwa ujian telah berakhir.

13. Card.tsx

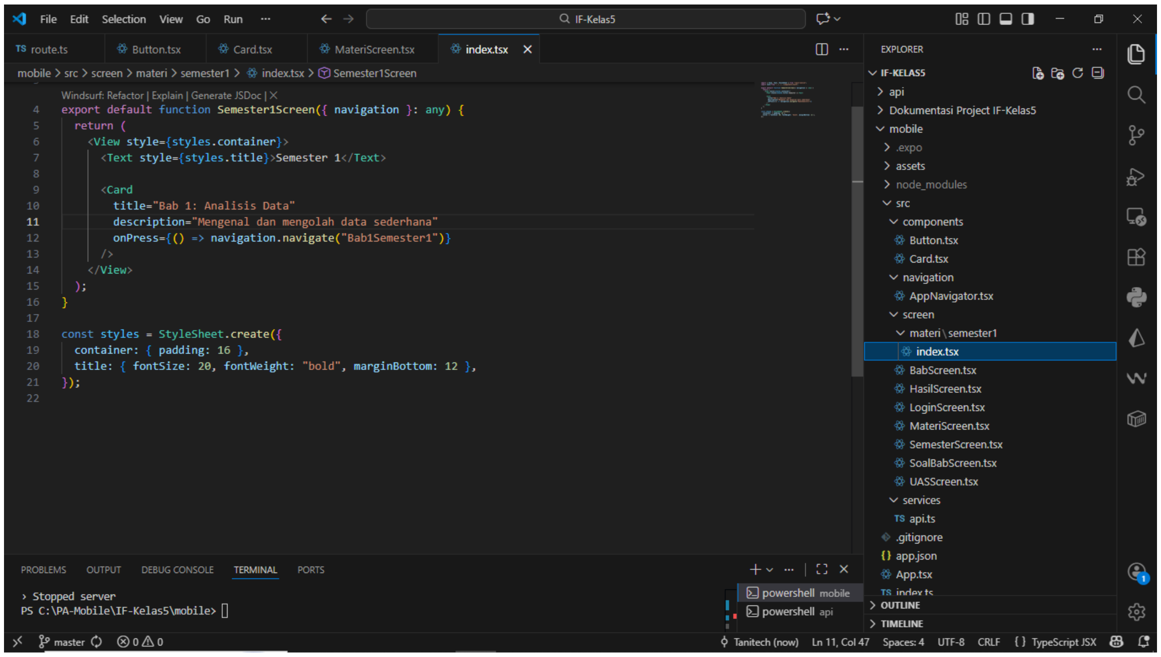


```tsx
import React from "react";
import {
  View,
  Text,
  StyleSheet,
  TouchableOpacity,
  ViewStyle,
} from "react-native";

interface CardProps {
  title: string;
  description?: string;
  onPress?: () => void;
  style?: ViewStyle;
}

const Card: React.FC<CardProps> = ({
  title,
```

```
>>
Starting project at C:\PA-Mobile\IF-Kelas5\mobile
› shift+m │ more tools
› Press o │ open project code in your editor

› Press ? │ show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
› Open in the web browser...
› Press ? │ show all commands
Web Bundled 3481ms index.ts (564 modules)
 LOG  [web] Logs will appear in the browser console
› Stopped server
PS C:\PA-Mobile\IF-Kelas5\mobile>
```

Komponen ini berfungsi untuk menampilkan sebuah kotak informasi (kartu) yang memiliki judul, deskripsi opsional, dan bisa diklik (interaktif).

14. Bab Semester 1 (semester1/index.tsx)



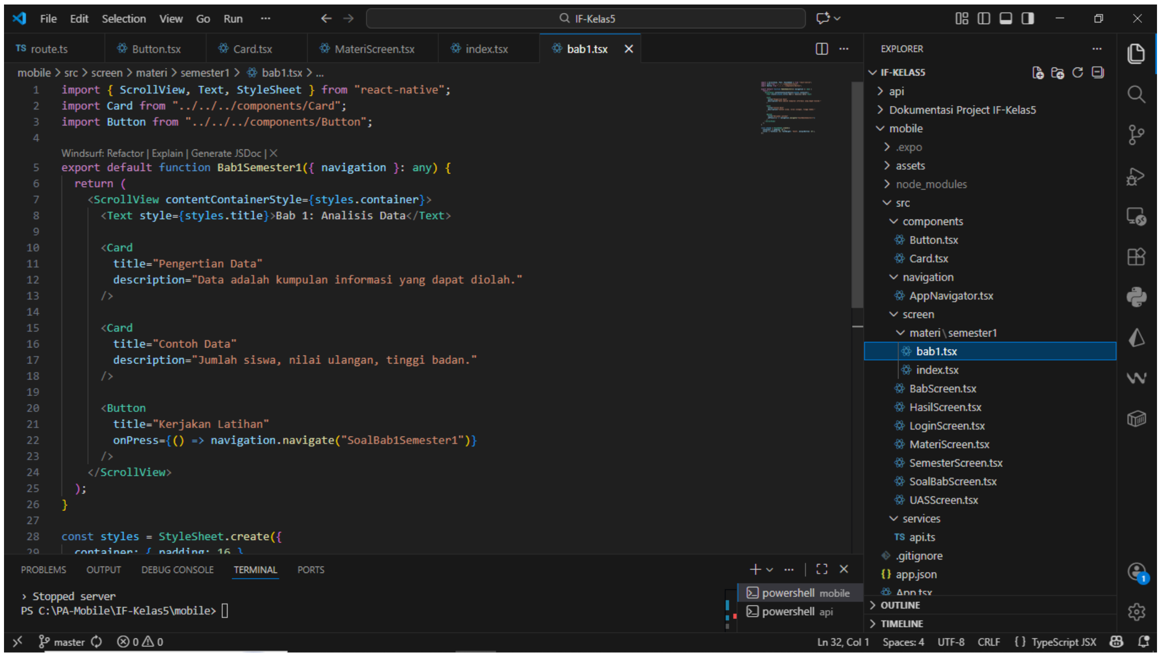
```tsx
export default function Semester1Screen({ navigation }: any) {
  return (
    <View style={styles.container}>
      <Text style={styles.title}>Semester 1</Text>

      <Card
        title="Bab 1: Analisis Data"
        description="Mengenal dan mengolah data sederhana"
        onPress={() => navigation.navigate("Bab1Semester1")}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: { padding: 16 },
  title: { fontSize: 20, fontWeight: "bold", marginBottom: 12 },
});
```

```
› Stopped server
PS C:\PA-Mobile\IF-Kelas5\mobile>
```

Berfungsi untuk menampilkan daftar materi pembelajaran pada Semester 1.

- Implementasi Komponen Reusable: Layar ini menggunakan komponen kustom <Card /> yang dipanggil dari folder komponen luar untuk menampilkan materi bab secara konsisten.
- Navigasi Interaktif: Terdapat fungsi onPress pada kartu yang akan mengarahkan pengguna ke layar detail materi Bab1Semester1 menggunakan navigation.navigate.
- Struktur Sederhana: * Judul: Menampilkan teks Semester 1 dengan gaya tebal (bold).
- Konten: Kartu berisi informasi spesifik seperti Bab 1: Analisis Data beserta ringkasan materinya.
- Styling: Menggunakan StyleSheet untuk memberikan ruang (padding) di sekeliling layar dan mengatur jarak antar elemen agar terlihat rapi.

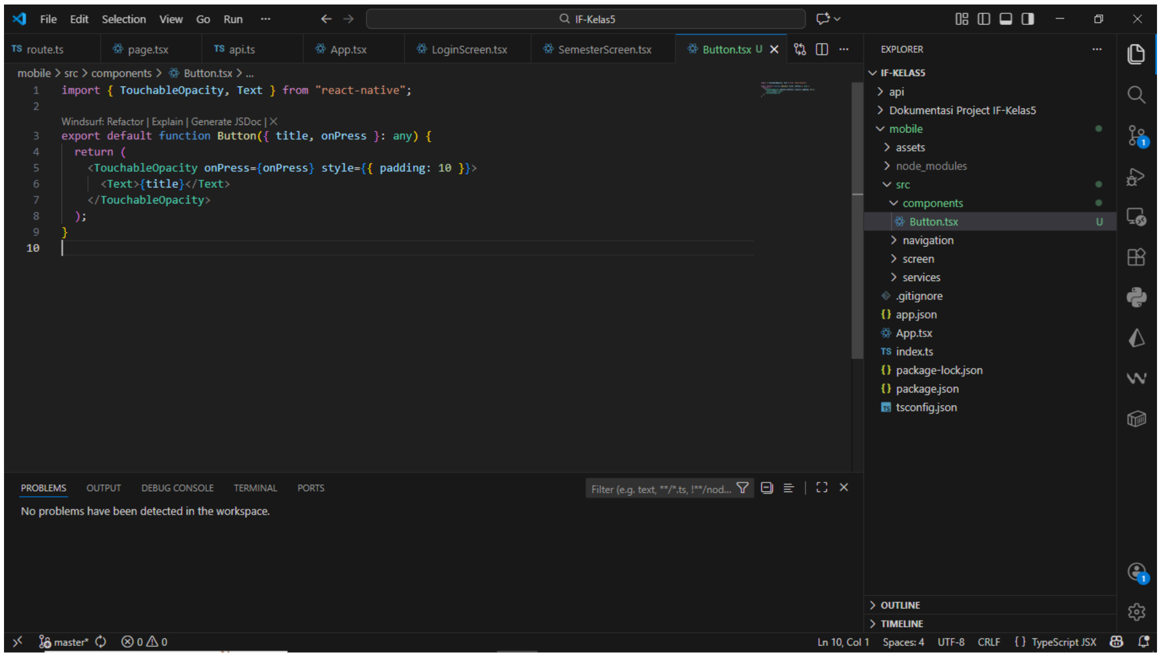
15. Bab Semester 1 (semester1/bab1.tsx)



```tsx
import { ScrollView, Text, StyleSheet } from "react-native";
import Card from "../../../components/Card";
import Button from "../../../components/Button";

export default function Bab1Semester1({ navigation }: any) {
  return (
    <ScrollView contentContainerStyle={styles.container}>
      <Text style={styles.title}>Bab 1: Analisis Data</Text>

      <Card
        title="Pengertian Data"
        description="Data adalah kumpulan informasi yang dapat diolah."
      />

      <Card
        title="Contoh Data"
        description="Jumlah siswa, nilai ulangan, tinggi badan."
      />

      <Button
        title="Kerjakan Latihan"
        onPress={() => navigation.navigate("SoalBab1Semester1")}
      />
    </ScrollView>
  );
}

const styles = StyleSheet.create({
  container: { padding: 16 },
```

```
> Stopped server
PS C:\PA-Mobile\IF-Kelas5\mobile>
```

16. Komponen Reusable



```
import { TouchableOpacity, Text } from "react-native";

export default function Button({ title, onPress }: any) {
  return (
    <TouchableOpacity onPress={onPress} style={{ padding: 10 }}>
      <Text>{title}</Text>
    </TouchableOpacity>
  );
}
```

No problems have been detected in the workspace.

Komponen Button kustom yang bersifat reusable (dapat digunakan kembali) dalam aplikasi React Native.

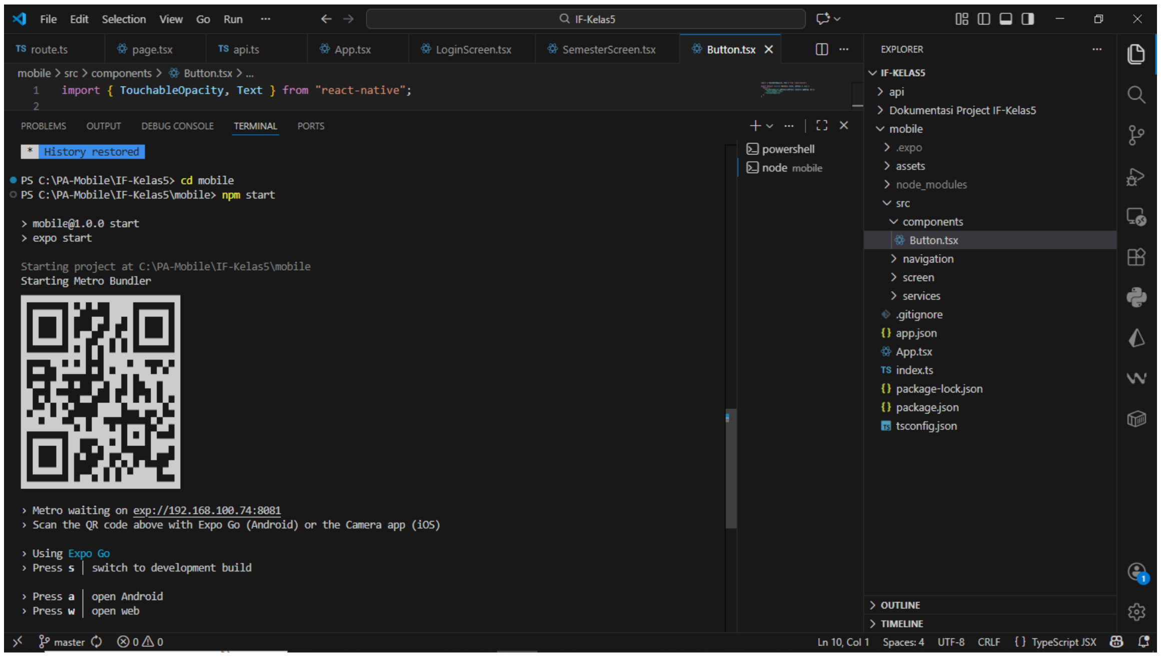
Struktur Komponen:

- Impor: Mengambil komponen dasar dari library react-native.
- Definisi: Menggunakan functional component dengan TypeScript (ditandai dengan tipe : any).
- Ekspor: Menggunakan export default agar komponen ini bisa diimpor dan digunakan dengan mudah di file lain.

17. Menjalankan aplikasinya

    Kode :

    npm start



```
import { TouchableOpacity, Text } from "react-native";
```

```
* History restored

PS C:\PA-Mobile\IF-Kelas5> cd mobile
PS C:\PA-Mobile\IF-Kelas5\mobile> npm start

> mobile@1.0.0 start
> expo start

Starting project at C:\PA-Mobile\IF-Kelas5\mobile
Starting Metro Bundler

> Metro waiting on exp://192.168.100.74:8081
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

> Using Expo Go
> Press s │ switch to development build

> Press a │ open Android
> Press w │ open web
```

    Lalu:

    Tekan a → Android Emulator

    Scan QR → Expo Go

    Download aplikasi Expo Go di PlayStore lalu install aplikasi dan scan barcodenya.

18. Alur Aplikasi

Login

↓

Semester

↓

Bab

↓

Materi

↓

Soal Bab

↓

Hasil

↓

UAS