

LAPORAN PRAKTIKUM  
MOBILE & WEB SERVICE PRAKTIK

Pertemuan ke-5

Semester Ganjil TA. 2024/2025

**FIGMA, REST API, DAN API CRUD**

Dosen Pengampu: Suyud Widiono, S.Pd., M.Kom.



Disusun oleh :

Regita Cahya Arrahma (5220411359)

PROGRAM STUDI INFORMATIKA  
FAKULTAS SAINS DAN TEKNOLOGI  
UNIVERSITAS TEKNOLOGI YOGYAKARTA  
2024

# DAFTAR ISI

<b>DAFTAR ISI .....</b>	<b>i</b>
<b>BAB I Desain <i>User Interface</i> Menggunakan Figma .....</b>	<b>1</b>
<b>1.1    Dasar teori.....</b>	<b>1</b>
1.1.1    Pengertian Figma .....	1
<b>1.2    Praktikum .....</b>	<b>2</b>
1.2.1    Dokumentasi.....	2
1.2.2    Hasil.....	2
<b>BAB II REST API.....</b>	<b>6</b>
<b>2.1    Dasar Teori.....</b>	<b>6</b>
2.1.1    REST API.....	6
2.1.2    Postman .....	10
<b>2.2    Praktikum .....</b>	<b>12</b>
2.2.1    Kode program .....	12
2.2.2    Output.....	17
<b>BAB III API CRUD .....</b>	<b>18</b>
<b>3.1    Dasar Teori.....</b>	<b>18</b>
3.1.1    Pengertian API CRUD.....	18
<b>3.2    Praktikum .....</b>	<b>18</b>
3.2.1    Kode program.....	18
3.2.2    Output .....	33
<b>REFERENSI .....</b>	<b>37</b>

# **BAB I**

## **Desain *User Interface* Menggunakan Figma**

### **1.1 Dasar teori**

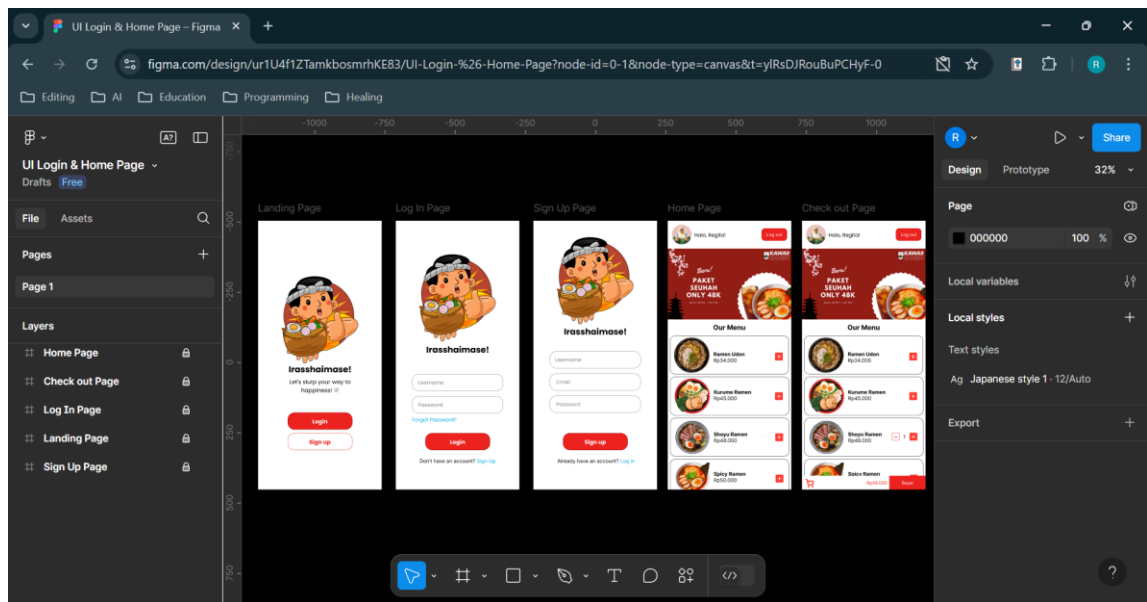
#### **1.1.1 Pengertian Figma**

Figma adalah desain digital dan alat prototyping. Ini adalah aplikasi desain UI dan UX yang dapat Anda gunakan untuk membuat situs web, aplikasi, atau komponen antarmuka pengguna yang lebih kecil yang dapat diintegrasikan ke dalam proyek lain. Dengan alat berbasis vektor yang hidup di cloud, Figma memungkinkan para penggunanya untuk bekerja di mana saja dari browser. Cara ini termasuk alat zippy yang dibuat untuk desain, pembuatan prototipe, kolaborasi, dan sistem desain organisasi.

Figma merupakan alat desain berbasis web yang solid, menawarkan paket gratis yang dapat diakses oleh siapa saja yang bekerja di ruang digital. Figma memiliki keunggulan dalam memfasilitasi proses desain dan prototipe, sehingga para desainer lebih mudah memasarkan hasil karyanya kepada klien maupun calon pemberi kerja. Dengan fitur kolaborasi real-time, Figma sangat cocok digunakan dalam tim yang aktif dan kolaboratif, sehingga meningkatkan efisiensi dan sinkronisasi kerja. Dibandingkan dengan alat desain lain, Figma menonjol berkat fitur-fitur kuncinya yang mampu menghemat waktu, mempermudah alur kerja, dan meningkatkan produktivitas tim dalam menyelesaikan proyek dengan lebih cepat dan efektif.

## 1.2 Praktikum

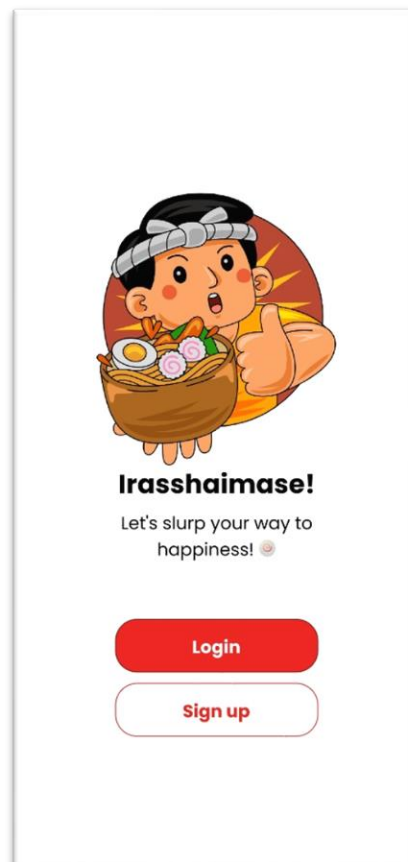
### 1.2.1 Dokumentasi



### 1.2.2 Hasil

Membuat desain *user interface* dari tugas widget pada pertemuan keempat menggunakan Figma. Berikut adalah hasil dari desain tersebut:

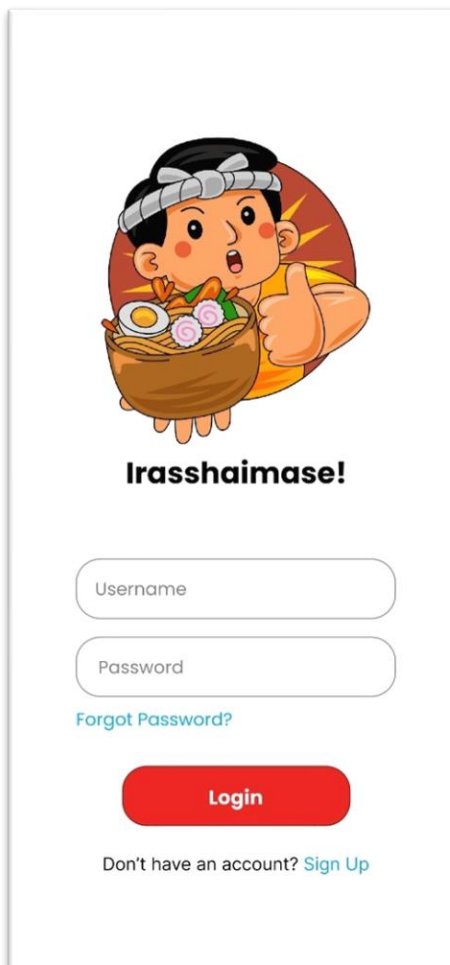
#### a. Landing page



Landing page ini merupakan pintu masuk untuk pengguna baru dan lama sebelum mereka dapat mengakses fitur-fitur utama dari aplikasi. Fungsi dari landing page ini adalah sebagai berikut:

- **Sambutan Pengguna:** Halaman ini menyapa pengguna dengan kata-kata "*Irasshaimase!*" (sapaan khas di restoran Jepang yang artinya "Selamat datang"). Diikuti dengan slogan "*Let's slurp your way to happiness!*" yang menggugah selera pengguna dan menciptakan suasana yang ramah.
- **Pilihan untuk Login atau Sign Up:** Pada landing page, terdapat dua tombol utama yaitu:
  - *Login*: Mengarahkan pengguna yang sudah memiliki akun untuk masuk ke aplikasi.
  - *Sign Up*: Mengarahkan pengguna yang belum memiliki akun untuk mendaftar.

b. Login page



**Irasshaimase!**

Username

Password

[Forgot Password?](#)

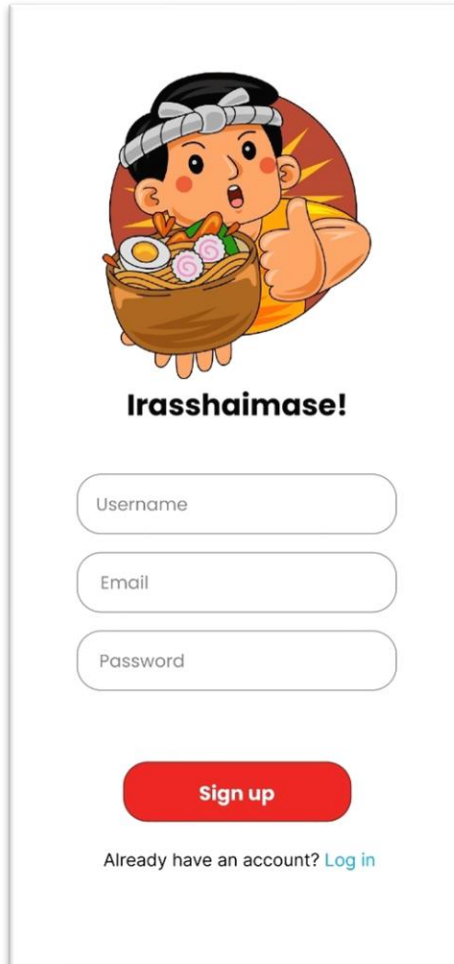
**Login**

Don't have an account? [Sign Up](#)

Halaman ini berfungsi untuk membiarkan pengguna masuk ke akun mereka. Pengguna diminta untuk memasukkan username dan password mereka. Jika mereka

lupa kata sandi, terdapat pilihan "Forgot Password?" untuk membantu memulihkan akses. Jika pengguna belum memiliki akun, tersedia opsi untuk mendaftar melalui link "Sign Up".

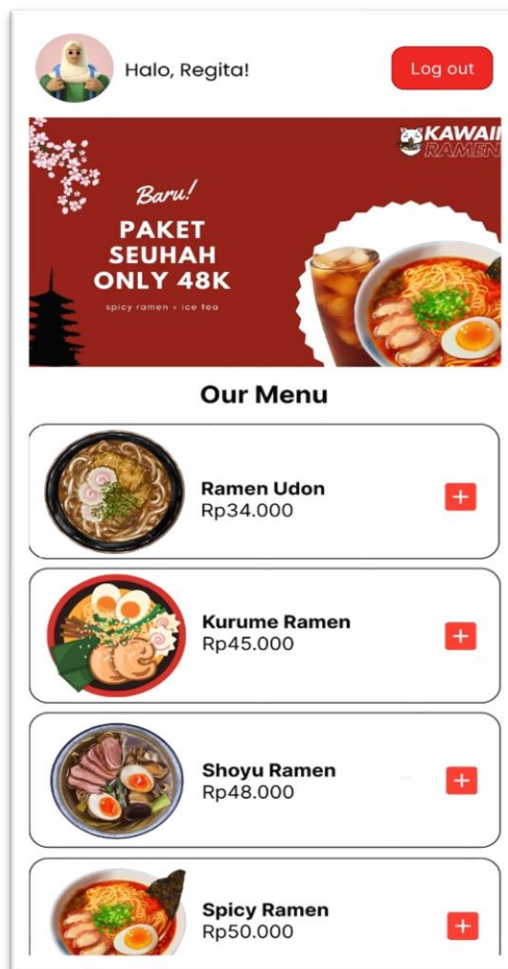
c. Sign up page



The image shows a sign-up page for a website. At the top, there is a cartoon illustration of a person with black hair and a white headband, holding a bowl of ramen with chopsticks. Below the illustration, the text "Irasshaimase!" is displayed. Underneath, there are three input fields labeled "Username", "Email", and "Password". A red button labeled "Sign up" is positioned below the input fields. At the bottom, there is a link that says "Already have an account? Log in".

Halaman ini digunakan untuk pengguna baru yang ingin membuat akun. Formulir pendaftaran ini meminta informasi seperti email, username, dan password. Jika pengguna sudah memiliki akun, tersedia opsi untuk langsung masuk melalui link "Log in".

d. Home page



Halaman ini menampilkan sambutan seperti "Halo, [Nama Pengguna]" dan memberikan opsi untuk *log out*. Pada halaman ini, tersedia menu makanan yang terdiri dari berbagai pilihan ramen, seperti *Kurume Ramen*, *Shoyu Ramen*, dan *Spicy Ramen*, dengan harga yang tertera di bawah setiap item. Di sini, pengguna dapat melihat menu dan melanjutkan pembelian makanan.

## **BAB II**

### **REST API**

#### **2.1 Dasar Teori**

##### **2.1.1 REST API**

REST API (Representational State Transfer Application Programming Interface) adalah antarmuka yang memungkinkan komunikasi dan pertukaran data antara berbagai perangkat lunak melalui jaringan. REST API didasarkan pada prinsip arsitektur REST yang menekankan kesederhanaan, skalabilitas, dan keterbacaan. Dengan menggunakan metode HTTP seperti GET, POST, PUT, dan DELETE, REST API berinteraksi dengan sumber daya yang diidentifikasi melalui URL. Klien mengirimkan permintaan yang mencakup metode HTTP, URL, serta informasi tambahan seperti parameter, header, atau body untuk diolah oleh server.

Setelah menerima permintaan, server REST API merespons dengan mengirimkan status HTTP, data, dan informasi tambahan melalui header sebagai hasil pemrosesan. REST API memungkinkan aplikasi berkomunikasi dan bertukar data melalui jaringan, misalnya untuk mengambil data cuaca, berbagi pesan di media sosial, atau melakukan pembayaran online. Dengan fungsinya, REST API mendukung integrasi antar aplikasi dan akses ke berbagai layanan dan database eksternal secara efisien.

##### **2.1.1.1 Komponen utama REST API**

###### **a. Resources (sumber daya)**

Resources dalam REST API mengacu pada objek atau data yang dapat diakses, diubah, atau dimanipulasi melalui API. Sumber daya ini diwakili oleh URL (Uniform Resource Locator), di mana setiap URL secara unik mengidentifikasi sebuah resource. Sebagai contoh, sebuah URL dapat mengidentifikasi data pengguna, produk, atau layanan dalam sebuah aplikasi. Sumber daya dalam REST API tidak selalu berbentuk fisik, tetapi bisa berupa entitas abstrak yang disediakan server dan dapat diakses melalui permintaan HTTP dari klien. Format data yang sering digunakan untuk berkomunikasi dalam REST API biasanya adalah JSON atau XML.

###### **b. HTTP Methods**



REST API menggunakan metode HTTP standar untuk berinteraksi dengan resources. Setiap metode HTTP memiliki peran yang spesifik dalam manipulasi data, yaitu:

- GET: Mengambil atau membaca sumber daya dari server tanpa mengubahnya.
- POST: Menambah data atau membuat sumber daya baru di server.
- PUT: Memperbarui atau mengganti sumber daya yang ada dengan data baru.
- DELETE: Menghapus sumber daya dari server. Metode ini memungkinkan klien dan server untuk berkomunikasi dengan cara yang jelas dan konsisten sesuai dengan operasi yang dibutuhkan.

#### 2.1.1.2 Karakteristik REST API

- Client-server

Arsitektur REST API mengikuti pola client-server, di mana terdapat pemisahan yang jelas antara klien dan server. Klien bertanggung jawab untuk antarmuka pengguna dan pengalaman pengguna, sementara server menangani penyimpanan data dan logika bisnis. Pemisahan ini memungkinkan setiap pihak untuk berkembang secara independen. Klien dapat diubah atau diperbarui tanpa memengaruhi server, dan sebaliknya, server dapat diperbarui tanpa mengubah klien.

- Stateless

REST API bersifat stateless, artinya setiap permintaan dari klien ke server harus mengandung semua informasi yang diperlukan untuk memproses permintaan tersebut. Server tidak menyimpan informasi tentang status klien di antara permintaan. Dengan kata lain, setiap interaksi bersifat independen. Karakteristik ini meningkatkan skalabilitas, karena server tidak perlu menyimpan sesi atau status, yang dapat menghambat kinerja.

- Cacheable

REST API mendukung caching untuk meningkatkan kinerja dan efisiensi. Respons dari server dapat di-cache di klien atau di intermediate caches (seperti proxy), sehingga mengurangi jumlah permintaan yang harus dilakukan ke server. Dengan menggunakan cache, klien dapat mengakses data lebih cepat dan mengurangi beban pada server, karena tidak perlu memproses permintaan yang sama berulang kali. Penggunaan cache juga membantu mengurangi latensi dan meningkatkan pengalaman pengguna.

- Uniform Interface

REST API memiliki antarmuka yang seragam, yang memungkinkan interaksi antara klien dan server dilakukan dengan cara yang konsisten dan terstandarisasi. Ini mencakup penggunaan metode HTTP (seperti GET, POST, PUT, DELETE) untuk berkomunikasi dengan sumber daya yang ada. Antarmuka yang seragam membuat sistem lebih mudah dipahami dan diintegrasikan, karena pengembang dapat berasumsi bahwa setiap interaksi mengikuti pola yang sama.

- Layered System

Arsitektur REST API terdiri dari beberapa lapisan yang terpisah. Ini berarti bahwa klien tidak perlu mengetahui detail tentang server yang sebenarnya melayani permintaan, karena ada lapisan perantara yang dapat mengelola permintaan dan respons. Lapisan ini dapat mencakup server proxy, gateway, atau load balancer, yang membantu dalam mengelola beban, meningkatkan keamanan, dan menyederhanakan skala sistem. Pemisahan ini juga meningkatkan keamanan dan fleksibilitas, karena perubahan pada satu lapisan tidak mempengaruhi lapisan lainnya.

### 2.1.1.3 Keuntungan menggunakan REST API

- Skalabilitas

Salah satu keuntungan utama dari menggunakan REST API adalah kemampuannya untuk skalabilitas. REST API dirancang untuk menangani sejumlah besar permintaan dari banyak klien secara bersamaan tanpa penurunan kinerja. Karena arsitektur yang berbasis stateless, server tidak perlu menyimpan informasi sesi, sehingga lebih mudah untuk menambah atau mengurangi sumber daya server sesuai kebutuhan. Ini memungkinkan sistem untuk tumbuh dan beradaptasi dengan peningkatan beban, menjadikannya pilihan yang ideal untuk aplikasi yang memerlukan pertumbuhan yang cepat.

- Fleksibilitas

REST API menawarkan fleksibilitas yang tinggi dalam pengembangan aplikasi. Dengan mendukung berbagai format data (seperti JSON, XML, dan HTML), pengembang dapat memilih format yang paling sesuai untuk kebutuhan aplikasi mereka. Selain itu, arsitektur REST memungkinkan berbagai jenis klien (web, mobile, desktop) untuk berinteraksi dengan server secara konsisten. Pengembang juga dapat memperbarui atau mengubah

komponen tanpa mengganggu fungsionalitas keseluruhan aplikasi, yang meningkatkan efisiensi dan produktivitas.

- Portabilitas

Keuntungan lain dari REST API adalah portabilitasnya. Karena REST API menggunakan protokol HTTP yang umum, aplikasi yang dibangun di atas REST API dapat dengan mudah dipindahkan antara berbagai platform dan lingkungan. Ini memungkinkan integrasi yang mudah dengan sistem dan layanan lain, serta memberikan fleksibilitas dalam memilih teknologi yang digunakan. Pengembang dapat menggunakan alat dan bahasa pemrograman yang berbeda tanpa khawatir tentang kompatibilitas, yang membuat REST API menjadi pilihan yang sangat portabel untuk pengembangan aplikasi.

- Kinerja

REST API dapat memberikan kinerja yang baik berkat beberapa faktor. Pertama, penggunaan caching memungkinkan pengulangan permintaan untuk mengakses data lebih cepat, mengurangi waktu respons dan beban server. Selain itu, dengan menggunakan metode HTTP yang efisien, REST API dapat meminimalkan ukuran data yang ditransfer antara klien dan server. Arsitektur yang sederhana juga berarti lebih sedikit overhead dalam komunikasi, memungkinkan aplikasi untuk berjalan lebih lancar dan responsif. Kinerja yang baik sangat penting untuk pengalaman pengguna yang positif, terutama dalam aplikasi yang memerlukan interaksi real-time.

#### 2.1.1.4 Tantangan dalam implementasi REST API

- Keamanan

Salah satu tantangan terbesar dalam implementasi REST API adalah memastikan keamanan data dan komunikasi. REST API terbuka untuk akses publik, yang berarti bahwa tanpa perlindungan yang memadai, mereka rentan terhadap berbagai serangan, seperti injeksi SQL, serangan DDoS, dan pengambilalihan sesi.

- Pengelolaan visual

Pengelolaan visual REST API dapat menjadi tantangan, terutama ketika API memiliki banyak endpoint dan fungsionalitas. Tanpa alat pengelolaan yang baik, pengembang dan tim dapat kesulitan memahami dan memelihara API.

- Dokumentasi

Dokumentasi yang jelas dan komprehensif adalah kunci untuk keberhasilan REST API. Tantangan yang sering dihadapi adalah menghasilkan dokumentasi yang selalu diperbarui dan mencakup semua fitur API. Tanpa dokumentasi yang baik, pengguna API (baik pengembang internal maupun eksternal) mungkin kesulitan memahami cara menggunakan API secara efektif.

### 2.1.2 Postman

Postman adalah aplikasi kolaboratif yang dirancang khusus untuk memfasilitasi pengujian dan pengembangan API. Dengan Postman, pengembang dapat dengan mudah membuat, menguji, dan mengelola permintaan HTTP terhadap API. Sebagai alat yang mendukung metode pengembangan berbasis API (Application Programming Interface), Postman memainkan peran penting dalam memastikan keberhasilan implementasi API.

Fungsi utama Postman adalah sebagai berikut:

- Mengirim HTTP Request kepada Server

HTTP Request adalah permintaan yang dikirimkan dari sisi klien ke server menggunakan protokol HTTP (Hypertext Transfer Protocol). Protokol ini digunakan untuk komunikasi di web, memungkinkan klien dan server untuk bertukar informasi. Dalam konteks REST API, terdapat beberapa metode yang umum digunakan untuk melakukan HTTP Request, masing-masing dengan fungsi yang spesifik:

- GET: Metode ini digunakan untuk mengambil data dari server. Ketika klien mengirimkan permintaan GET, server akan mengembalikan representasi dari sumber daya yang diminta. Metode GET tidak mengubah data di server dan umumnya digunakan untuk meminta data yang bersifat read-only.
- POST: Metode ini digunakan untuk mengirimkan data ke server, sering kali untuk membuat sumber daya baru. Ketika klien menggunakan metode POST, data yang dikirim biasanya akan ditambahkan ke database atau sumber daya lain di server. Metode POST dapat menyebabkan perubahan di server, dan karena itu tidak dianggap idempotent (dapat memberikan hasil yang berbeda jika dijalankan berkali-kali).
- PUT: Metode PUT digunakan untuk memperbarui data yang sudah ada di server. Ketika klien mengirimkan permintaan PUT, ia biasanya menyertakan seluruh representasi sumber daya yang ingin diperbarui.

Metode ini juga dianggap idempotent, artinya jika permintaan yang sama dikirim beberapa kali, hasilnya akan tetap sama.

- DELETE: Metode DELETE digunakan untuk menghapus sumber daya dari server. Ketika klien mengirimkan permintaan DELETE, server akan menghapus sumber daya yang diminta. Seperti metode PUT, DELETE juga bersifat idempotent; menghapus sumber daya yang sama berkali-kali akan memberikan hasil yang sama setelah sumber daya pertama kali dihapus.

- Pengujian Terhadap Response Yang Diterima

Postman API menyediakan alat pengujian yang kuat untuk memvalidasi respons API. Anda dapat memeriksa status kode, header, dan data yang diterima dari permintaan API. Ini memungkinkan kamu untuk menguji fungsionalitas dan memastikan bahwa API bekerja dengan benar.

- Membuat Mock Server

Mock server pada Postman adalah fitur yang memungkinkan untuk membuat server tiruan yang dapat mensimulasikan respons API tanpa perlu mengakses server asli atau melibatkan pengembangan penuh dari sisi server. Dengan menggunakan mock server, kamu dapat menguji permintaan API dan merancang respons yang diharapkan tanpa perlu menunggu atau mengandalkan server yang sebenarnya. Mock server memungkinkan pengguna untuk menciptakan berbagai skenario respons API berdasarkan permintaan yang diterima. Pengguna dapat menentukan respons yang ingin dikirimkan oleh mock server, termasuk status kode, header, dan isi respons. Hal ini memungkinkan pengguna untuk mensimulasikan berbagai kasus pengujian dan mengamati bagaimana klien berinteraksi dengan respons yang berbeda.

- Membuat Dokumentasi API

Dokumentasi API di Postman adalah fitur yang memungkinkan pengguna untuk membuat dokumentasi yang terstruktur dan mudah dipahami untuk API yang sedang dikembangkan atau digunakan. Dokumentasi ini berfungsi sebagai panduan dan referensi bagi pengembang atau pengguna lain yang ingin berinteraksi dengan API tersebut. Dokumentasi API pada Postman membantu dalam proses berbagi dan memahami penggunaan API. Dengan adanya dokumentasi yang baik, pengguna atau pengembang lain dapat dengan mudah memahami cara menggunakan API, melihat contoh permintaan dan

respons, serta memahami parameter dan header yang diperlukan. Ini memfasilitasi kolaborasi dalam tim dan meningkatkan adopsi serta penggunaan API secara keseluruhan.

- Mengatur Environment pada Postman

Dalam Postman, pengguna dapat membuat lingkungan (environment) untuk menyimpan nilai variabel yang spesifik untuk setiap lingkungan, seperti pengembangan, produksi, atau pengujian. Misalnya, jika bekerja dengan API yang memiliki URL berbeda di lingkungan pengembangan dan produksi, pengguna dapat menyimpan URL tersebut sebagai variabel dalam setiap lingkungan. Dengan cara ini, pengguna dapat dengan mudah mengubah nilai variabel sesuai dengan lingkungan yang sedang digunakan. Selain URL, variabel lain yang umum disimpan dalam lingkungan Postman mencakup token autentikasi, header, parameter, dan isi dari permintaan HTTP. Pengguna dapat merujuk ke variabel-variabel ini dengan menggunakan sintaks `{{nama_variabel}}` dalam permintaan mereka, dan Postman akan secara otomatis menggantikan variabel-variabel ini dengan nilai yang sesuai saat permintaan dijalankan.

## 2.2 Praktikum

### 2.2.1 Kode program

- UserModel

```
class UserModel { //menyimpan informasi tentang pengguna
  final Int? id; //Nomor identifikasi pengguna.
  final String? avatar; //Gambar profil pengguna.
  final String? email; //Alamat email pengguna.
  final String? firstName; //Nama depan pengguna.
  final String? lastName; //Nama belakang pengguna.

  const UserModel( //konstruktor, atau cara untuk membuat objek
    UserModel
    {required this.id, //required yang berarti wajib diisi
    required this.avatar,
    required this.email,
    required this.firstName,
    required this.lastName});

  //Bagian ini adalah cara untuk membuat objek UserModel dari
  data berbentuk JSON.
  factory UserModel.fromJson({required Map<String, dynamic>
    json}) => UserModel(
```

```

        id: json['id'] ?? 0,
        avatar: json['avatar'] ?? '-',
        email: json['email'] ?? '-',
        firstName: json['first_name'] ?? '-',
        lastName: json['last_name'] ?? '-',
    );
}

```

- Membuat UserService

//Bagian ini mengimpor paket dio, yang merupakan paket untuk melakukan permintaan HTTP di Dart, dan mengimpor user\_model.dart, file yang berisi kelas UserModel yang digunakan untuk menangani data pengguna.

```

import 'package:dio/dio.dart';
import 'package:project/user_model.dart';

```

```

class UserService { //Kelas yang bertanggung jawab untuk mengambil data pengguna dari API.
    final Dio _dio = Dio(); //Membuat instance Dio, yang digunakan untuk mengirim permintaan HTTP.
    static const _baseUrl = 'https://reqres.in/api'; //Menyimpan URL dasar dari API
    static const _users = 'users'; //Menyimpan bagian endpoint (users) untuk mempermudah pembentukan URL Lengkap.

```

//Metode untuk mengambil data pengguna dari API dan mengembalikannya dalam bentuk daftar UserModel.

```

    Future<List<UserModel>> fetchUser() async {
        try {
            //Mengirim permintaan GET ke endpoint yang berisi URL dasar dan endpoint, ditambah parameter halaman.
            final response = await _dio.get('$_baseUrl/$_users',
            queryParameters: {
                'page': '1',
            });
            //Memeriksa apakah respons HTTP berhasil (kode status 200 atau 201). Jika berhasil, maka data dari respons akan diproses lebih lanjut.
            if (response.statusCode == 200 || response.statusCode == 201)
            {
                //Mengambil bagian data dari respons JSON yang berisi daftar pengguna.
                final data = response.data['data'];
                //Mengonversi data JSON menjadi daftar objek UserModel.
                List<UserModel> users =
                    List.from(data.map((user) => UserModel.fromJson(
                    user))));
                return users;
            }

```

```

    }
    //Jika kode status bukan 200 atau 201, maka program melempar
    pengecualian.
    throw Exception();
  } catch (e) {
    //Menangani pengecualian dan melempar ulang agar bisa
    ditangani di luar fungsi ini.
    rethrow;
  }
}
}
}

```

- Membuat UserPage

```

//Mengimpor paket-paket yang diperlukan untuk membangun aplikasi
Flutter, model UserModel, dan layanan UserService.
import 'package:flutter/material.dart';
import 'package:project/user_model.dart';
import 'package:project/user_service.dart';

void main() {
  runApp(const MyApp()); //Menjalankan aplikasi dengan widget utama
  MyApp.
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  //Widget utama aplikasi yang mengatur tema dan halaman utama
  aplikasi.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note', //Menentukan judul aplikasi.
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(
          seedColor: Colors.deepPurple), //Mengatur skema warna
          utama.
        useMaterial3: true, //Mengaktifkan gaya Material Design 3.
      ),
      home:
        const UserPage(), //Menampilkan halaman pengguna sebagai
        halaman utama.
    );
  }
}

class UserPage extends StatefulWidget {
  const UserPage({super.key});
}

```



```

    //Membuat state untuk UserPage untuk manage data dan tampilan
    halaman pengguna.
    @override
    State<UserPage> createState() => _UserPageState();
}

class _UserPageState extends State<UserPage> {
    List<UserModel> listUser =
        <UserModel>[]; //Variabel untuk menyimpan daftar pengguna dari
    API.

    //initState dipanggil ketika widget dibangun pertama kali dan
    memuat data pengguna dari API.
    @override
    void initState() {
        super.initState();
        getDataUser();
    }

    //Fungsi untuk mengambil data pengguna dari API menggunakan
    UserService.
    void getDataUser() async {
        UserService userService =
            UserService(); //Membuat instance dari UserService.
        List<UserModel> listUserTemp =
            await userService.fetchUser(); //Mengambil daftar pengguna
    dari API.
        if (listUserTemp.isNotEmpty) {
            //Jika data pengguna berhasil diambil.
            setState(() {
                listUser = listUserTemp; //Memperbarui listUser dengan data
    pengguna.
            });
        }
    }

    //Membangun tampilan halaman yang menampilkan daftar pengguna.
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: const Text(
                    'Users', //Menampilkan judul "Users" di AppBar.
                    style: TextStyle(fontSize: 16.0, fontWeight:
    FontWeight.w600),
                ),
            ),
        ),
    }
}

```

```

body: SafeArea(
  child: ListView.separated(
    itemCount: listUser
      .length, //Menentukan jumlah item dalam daftar
    berdasarkan jumlah pengguna.
    shrinkWrap: true,
    physics:
      const BouncingScrollPhysics(), //Menambahkan efek
    bouncing pada scroll.
    padding: const EdgeInsets.symmetric(vertical: 16.0),
    itemBuilder: (context, index) => ListTile(
      leading: ClipOval(
        child: Image.network(
          listUser[index].avatar ?? '-', //Menampilkan avatar
          pengguna.

          width: 52.0,
          height: 52.0,
          fit: BoxFit.cover,
        ),
      ),
      title: Text(
        '${listUser[index].firstName}
        ${listUser[index].lastName}', //Menampilkan nama depan dan belakang
        pengguna.

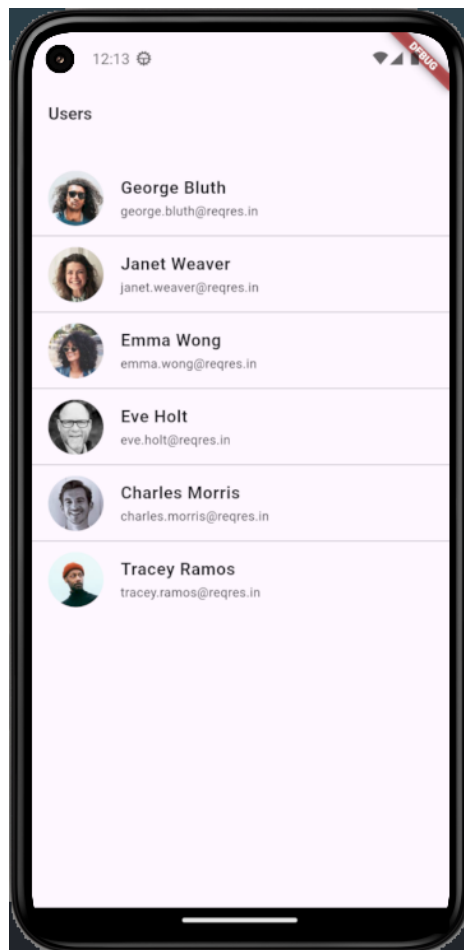
        style:
          const TextStyle(fontSize: 16.0, fontWeight:
FontWeight.w600),
      ),
      subtitle: Text(
        listUser[index].email ?? '-', //Menampilkan email
        pengguna.

        style: const TextStyle(
          fontSize: 12.0,
        ),
      ),
      separatorBuilder: (context, index) => const Divider(
        height: 0.0, //Menambahkan garis pemisah antara item
        pengguna.

      ),
    ),
  );
}

```

## 2.2.2 Output



## BAB III

### API CRUD

#### 3.1 Dasar Teori

##### 3.1.1 Pengertian API CRUD

CRUD API adalah sebuah antarmuka pemrograman aplikasi (API) yang memungkinkan pengguna untuk melakukan operasi dasar pada data di dalam sistem basis data atau aplikasi. CRUD adalah singkatan dari empat operasi dasar yang sering digunakan dalam pengelolaan data, yaitu:

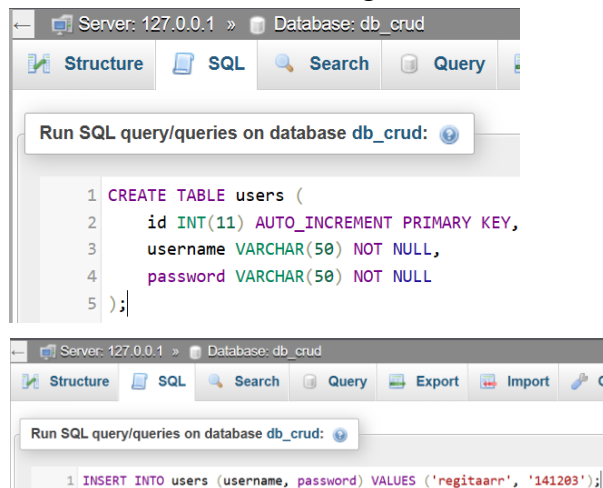
- Create - Membuat data baru.
- Read - Membaca atau mengambil data yang ada.
- Update - Memperbarui data yang sudah ada.
- Delete - Menghapus data.

Dengan menggunakan CRUD API, pengguna atau pengembang aplikasi dapat berinteraksi dengan data melalui HTTP request (permintaan HTTP) seperti GET, POST, PUT, dan DELETE.

#### 3.2 Praktikum

##### 3.2.1 Kode program

- Membuat database dan mengisi record



- Folder MWSP:
  - folder includes:
    - db.php

```
<?php
$host = "localhost";           // Server host
$user = "root";                // Username database
$password = "";                // Password database (kosong jika
                                default di XAMPP)
```

```

$database = "db_crud";    // Nama database

// Membuat koneksi ke database
$conn = new mysqli($host, $user, $password, $database);

// Cek koneksi
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>

```

- api.php

```

<?php
header("Content-Type: application/json");

// Menyertakan file db.php yang berada di dalam folder includes.
require 'includes/db.php';

// Mendapatkan metode permintaan
$request_method = $_SERVER["REQUEST_METHOD"];

switch($request_method) {
    case 'GET':
        if (isset($_GET['id'])) {
            get_user($conn, $_GET['id']);
        } else {
            get_users($conn);
        }
        break;
    case 'POST':
        create_user($conn);
        break;
    case 'PUT':
        update_user($conn, $_GET['id']);
        break;
    case 'DELETE':
        delete_user($conn, $_GET['id']);
        break;
    default:
        echo json_encode(["status" => "error", "message" => "Invalid request method"]);
        break;
}

$conn->close();

// Fungsi untuk mendapatkan semua pengguna
function get_users($conn) {
    $sql = "SELECT * FROM users";
}

```

```

$result = $conn->query($sql);
$users = [];
while ($row = $result->fetch_assoc()) {
    $users[] = $row;
}
echo json_encode($users);
}

// Fungsi untuk mendapatkan pengguna berdasarkan ID
function get_user($conn, $id) {
    $sql = "SELECT * FROM users WHERE id = $id";
    $result = $conn->query($sql);
    $user = $result->fetch_assoc();
    if ($user) {
        echo json_encode($user);
    } else {
        echo json_encode(["status" => "error", "message" => "User not
found"]);
    }
}

// Fungsi untuk membuat pengguna baru
function create_user($conn) {
    $data = json_decode(file_get_contents("php://input"));
    $username = $data->username;
    $password = $data->password;

    if ($username && $password) {
        $sql = "INSERT INTO users (username, password) VALUES
('$username', '$password')";
        if ($conn->query($sql) === TRUE) {
            echo json_encode(["status" => "success", "message" =>
"User created successfully"]);
        } else {
            echo json_encode(["status" => "error", "message" =>
"Error: " . $conn->error]);
        }
    } else {
        echo json_encode(["status" => "error", "message" => "Username
and password required"]);
    }
}

// Fungsi untuk memperbarui pengguna
function update_user($conn, $id) {
    $data = json_decode(file_get_contents("php://input"));
    $username = $data->username;
    $password = $data->password;

```

```

        if ($username && $password) {
            $sql = "UPDATE users SET username='$username',
password='$password' WHERE id=$id";
            if ($conn->query($sql) === TRUE) {
                echo json_encode(["status" => "success", "message" =>
                "User updated successfully"]);
            } else {
                echo json_encode(["status" => "error", "message" =>
                "Error: " . $conn->error]);
            }
        } else {
            echo json_encode(["status" => "error", "message" => "Username
and password required"]);
        }
    }

    // Fungsi untuk menghapus pengguna
    function delete_user($conn, $id) {
        $sql = "DELETE FROM users WHERE id=$id";
        if ($conn->query($sql) === TRUE) {
            echo json_encode(["status" => "success", "message" => "User
deleted successfully"]);
        } else {
            echo json_encode(["status" => "error", "message" => "Error:
" . $conn->error]);
        }
    }
}
?>

```

#### - login.php

```

<?php
header("Content-Type: application/json");

// Menyertakan file db.php yang berisi konfigurasi koneksi ke
database.
require 'includes/db.php';

// Mengambil data dari permintaan (request) dalam format JSON.
$data = json_decode(file_get_contents("php://input"));

// Memastikan data username dan password dikirim dalam permintaan.
if (isset($data->username) && isset($data->password)) {
    $username = $data->username; // Mengambil nilai username dari
data.
    $password = $data->password; // Mengambil nilai password dari
data.
}

```

```

    // Query SQL untuk memeriksa apakah ada pengguna dengan username
    dan password yang sesuai.
    $sql = "SELECT * FROM users WHERE username = '$username' AND
password = '$password'";
    $result = $conn->query($sql); // Menjalankan query ke database.

    // Memeriksa apakah ada baris yang cocok (login berhasil).
    if ($result->num_rows > 0) {
        // Jika login berhasil, mengembalikan respons JSON dengan
        status sukses.
        echo json_encode(["status" => "success", "message" => "Login
successful"]);
    } else {
        // Jika login gagal, mengembalikan respons JSON dengan status
        error.
        echo json_encode(["status" => "error", "message" => "Invalid
username or password"]);
    }
} else {
    // Jika username atau password tidak dikirim, mengembalikan
    respons JSON dengan status error.
    echo json_encode(["status" => "error", "message" => "Username and
password required"]);
}

// Menutup koneksi ke database.
$conn->close();
?>

```

- o Folder project:

- pubspec.yaml

```

30 dependencies:
31   flutter:
32     sdk: flutter
33
34
35   # The following adds the Cupertino Icons font to your application.
36   # Use with the CupertinoIcons class for iOS style icons.
37   cupertino_icons: ^1.0.8
38   dio: ^5.0.0
39   http: ^1.2.2

```

Menambahkan kode program `http: ^1.2.2`, berfungsi untuk memungkinkan aplikasi Flutter berkomunikasi dengan API dan server, sehingga memudahkan pengambilan dan pengiriman data secara efisien.

- tugas\_pertemuan5.dart

```

import 'dart:convert'; // Mengimpor pustaka untuk bekerja dengan JSON
import 'package:http/http.dart'

```



```

    as http; // Mengimpor paket http untuk melakukan permintaan HTTP
import 'package:flutter/material.dart'; // Mengimpor paket Flutter
untuk membangun antarmuka pengguna

void main() {
  runApp(
    const MyApp()); // Menjalankan aplikasi dengan widget MyApp
    sebagai root
  }

class MyApp extends StatelessWidget {
  // Kelas MyApp adalah widget Stateless yang menjadi root aplikasi
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    // Metode ini membangun antarmuka pengguna
    return MaterialApp(
      title: 'Login Navigation', // Judul aplikasi
      theme: ThemeData(
        primarySwatch: Colors.purple, // Skema warna utama aplikasi
      ),
      debugShowCheckedModeBanner: false, // Menonaktifkan banner debug
      home: const LoginPage(), // Menetapkan halaman awal aplikasi ke
LoginPage
    );
  }
}

class LoginPage extends StatefulWidget {
  // Kelas LoginPage adalah widget Stateful untuk menangani status
  const LoginPage({super.key});

  @override
  State<LoginPage> createState() =>
    _LoginPageState(); // Mengembalikan instance dari
_LoginPageState
}

class _LoginPageState extends State<LoginPage> {
  final TextEditingController _usernameController =
    TextEditingController(); // Mengontrol input username
  final TextEditingController _passwordController =
    TextEditingController(); // Mengontrol input password

  void _login() async {
    // Metode untuk menangani proses login
    const String apiUrl =

```

```

        "http://10.0.2.2/mwsp/login.php"; // URL endpoint API Login

final response = await http.post(
  // Mengirim permintaan POST ke API
  Uri.parse(apiUrl),
  headers: <String, String>{
    'Content-Type':
      'application/json; charset=UTF-8', // Menetapkan header
Content-Type
  },
  body: jsonEncode(<String, String>{
    // Mengonversi data input menjadi format JSON
    'username': _usernameController.text,
    'password': _passwordController.text,
  })),
);

if (response.statusCode == 200) {
  // Memeriksa apakah respons berhasil
  final Map<String, dynamic> responseData =
    json.decode(response.body); // Mengurai respons JSON

  if (responseData['status'] == 'success') {
    // Jika Login berhasil
    Navigator.push(
      // Mendorong halaman DashboardPage ke tumpukan navigasi
      context,
      MaterialPageRoute(
        builder: (context) => DashboardPage(
          username: _usernameController.text, onLogout:
_logout),
      ));
  } else {
    // Jika Login gagal
    ScaffoldMessenger.of(context).showSnackBar(
      // Menampilkan pesan kesalahan
      SnackBar(content: Text(responseData['message'])),
    );
  }
} else {
  // Jika status kode bukan 200
  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(
      content: Text(
        'Terjadi kesalahan pada server!'), // Menampilkan
pesan kesalahan server
      ),
    );
}

```

```

}

void _logout() {
  // Metode untuk menangani proses Logout
  _usernameController.clear(); // Menghapus input username
  _passwordController.clear(); // Menghapus input password

  Navigator.pop(context); // Kembali ke halaman Login
}

@override
Widget build(BuildContext context) {
  // Metode untuk membangun tampilan halaman Login
  return Scaffold(
    appBar: AppBar(toolbarHeight: 10), // Membuat AppBar dengan
    tinggi 10
    body: Padding(
      padding: const EdgeInsets.all(
        30.0), // Menetapkan padding di sekitar isi halaman
      child: SingleChildScrollView(
        // Memungkinkan scroll jika konten melebihi ukuran layar
        child: Column(
          mainAxisAlignment: MainAxisAlignment
            .center, // Mengatur konten agar berada di tengah
          children: [
            SizedBox(
              height: 300,
              child: Image.asset('assets/images/logo.png',
                fit: BoxFit.contain), // Menampilkan logo aplikasi
            ),
            // Teks Login
            const Align(
              alignment: Alignment.centerLeft,
              child: Text(
                'Login',
                style: TextStyle(
                  fontSize: 40,
                  fontWeight: FontWeight.bold), // Mengatur gaya
            ),
            // teks login
            const SizedBox(height: 10),
            // Input Username
            SizedBox(
              width: double.infinity,
              child: TextField(
                controller:

```

```

        _usernameController, // Menghubungkan TextField
dengan controller username
        decoration: const InputDecoration(
            labelText: 'Username', // Label untuk input
username
            border: OutlineInputBorder(), // Border untuk
TextField
            contentPadding: EdgeInsets.symmetric(
                vertical: 15,
                horizontal: 16), // Padding di dalam TextField
        ),
        style: const TextStyle(fontSize: 20), // Mengatur
ukuran teks
    ),
),
const SizedBox(height: 20),

// TextField Password
SizedBox(
    width: double.infinity,
    child: TextField(
        controller:
            _passwordController, // Menghubungkan TextField
dengan controller password
            obscureText: true, // Mengaburkan teks untuk
keamanan
            decoration: const InputDecoration(
                labelText: 'Password', // Label untuk input
password
                border: OutlineInputBorder(), // Border untuk
TextField
                contentPadding: EdgeInsets.symmetric(
                    vertical: 15,
                    horizontal: 16), // Padding di dalam TextField
            ),
            style: const TextStyle(fontSize: 20), // Mengatur
ukuran teks
        ),
    ),
    // "Forgot Password" di sebelah kanan dengan warna biru
const SizedBox(height: 5),
Align(
    alignment: Alignment.centerRight,
    child: TextButton(
        onPressed: () {
            // Forgot password action (tindakan ketika "Lupa
kata sandi" ditekan)
        },

```

```

        child: const Text(
          'Forgot Password?',
          style: TextStyle(
            color: Colors.blue, // Mengubah warna menjadi
            fontSize: 20), // Mengatur ukuran teks
        ),
      ),
    ),
    // Tombol Login dengan warna kuning, teks italic hitam
    dan panjang sesuai keinginan
    const SizedBox(height: 15),
    SizedBox(
      width: 250, // Panjang tombol sesuai keinginan
      height: 50, // Tinggi tombol tetap 50
      child: ElevatedButton(
        onPressed:
          _login, // Memanggil fungsi _login saat tombol
        style: ElevatedButton.styleFrom(
          backgroundColor:
            Colors.red, // Mengubah warna tombol menjadi
        ),
        child: const Text(
          'Login',
          style: TextStyle(
            color: Colors.white, // Warna teks menjadi putih
            fontWeight: FontWeight.bold, // Mengatur bobot
            fontSize: 20,
          ),
        ),
      ),
    ),
    // Spasi di atas dan bawah teks "Or"
    const SizedBox(height: 17), // Spasi di atas
    const Text('Or', style: TextStyle(fontSize: 20)), //
    // Tombol Register dengan warna biru
    TextButton(
      onPressed: () {
        // Register action (tindakan ketika tombol register
        ditekan)
      },
      child: const Text(
        'Register',
        style: TextStyle(

```

```

        color: Colors.blue, // Mengubah warna menjadi
        biru
        fontSize: 20), // Mengatur ukuran teks
    ),
),
],
),
),
),
);
}
}

class DashboardPage extends StatelessWidget {
    // Kelas DashboardPage adalah widget Stateless untuk menampilkan
    halaman dashboard
    final String username; // Variabel untuk menyimpan nama pengguna
    final VoidCallback onLogout; // Callback untuk Logout

    const DashboardPage(
        {required this.username, required this.onLogout, super.key});

    @override
    Widget build(BuildContext context) {
        // Metode untuk membangun tampilan halaman dashboard
        return Scaffold(
            appBar: AppBar(
                toolbarHeight: 15, // Mengatur tinggi toolbar
                automaticallyImplyLeading:
                    false, // Menonaktifkan navigasi kembali di AppBar
            ),
            body: SingleChildScrollView(
                // Memungkinkan scroll jika konten melebihi ukuran layar
                child: Padding(
                    padding: const EdgeInsets.all(
                        16.0), // Menetapkan padding di sekitar isi halaman
                    child: Column(
                        crossAxisAlignment:
                            CrossAxisAlignment.start, // Mengatur alignment konten
                            ke kiri
                        children: [
                            Row(
                                mainAxisAlignment: MainAxisAlignment
                                    .spaceBetween, // Mengatur jarak antar elemen
                                    dalam baris
                                children: [
                                    Row(
                                        children: [

```

```

const CircleAvatar(
  // Menampilkan avatar bulat
  radius: 30, // Mengatur radius avatar
  child: Icon(Icons.person, size: 40), //
Menampilkan ikon
),
const SizedBox(width: 15), // Jarak antara
avatar dan teks
Column(
  crossAxisAlignment: CrossAxisAlignment
    .start, // Mengatur alignment kolom ke
kiri
  children: [
    Text(
      'Halo, $username', // Menampilkan pesan
sambutan dengan nama pengguna
      style: const TextStyle(
        fontSize: 18), // Mengatur ukuran teks
    ),
    const Text(
      'Itadakimasu.', // Menampilkan teks
tambahan
      style: TextStyle(
        fontSize: 23,
        fontWeight: FontWeight
          .bold), // Mengatur ukuran dan
bobot teks
    ),
  ],
),
ElevatedButton(
  onPressed:
    onLogout, // Memanggil fungsi onLogout saat
tombol ditekan
  style: ElevatedButton.styleFrom(
    backgroundColor:
      Colors.red, // Mengubah warna tombol menjadi
merah
  ),
  child: const Text(
    'Log out', // Teks tombol Logout
    style: TextStyle(
      color: Colors.white, // Warna teks menjadi
putih
      fontWeight: FontWeight.bold, // Mengatur bobot
teks

```

```

        fontSize: 20,
      ),
    ),
  ],
),
const SizedBox(height: 20),

// Centering kedai.png
Center(
  child: Image.asset(
    'assets/images/kedai.png', // Menampilkan gambar
    height: 260,
    fit: BoxFit.fill),
),

const SizedBox(height: 20),

// Menambahkan text "Our Menu" ditengah
const Center(
  child: Text(
    '-- Our Menu --',
    style: TextStyle(
      fontSize: 24,
      fontWeight:
        FontWeight.bold, // Mengatur ukuran dan bobot
    ),
  ),
),
const SizedBox(height: 10),

// Ramen Udon Row
Row(
  children: [
    SizedBox(
      width: 130,
      height: 130,
      child: Image.asset(
        'assets/images/udon.png', // Menampilkan
        fit: BoxFit.cover),
    ),
    const SizedBox(width: 30),
    const Column(
      crossAxisAlignment: CrossAxisAlignment
        .start, // Mengatur alignment kolom ke kiri

```



```

        children: [
            Text('Ramen Udon',
                style: TextStyle(
                    fontSize: 20,
                    fontWeight:
                        FontWeight.bold)), // Menampilkan
nama menu
            Text('Rp34.000',
                style: TextStyle(fontSize: 20)), //
Menampilkan harga
        ],
    ),
],
),
const SizedBox(height: 10),

// Kurume Ramen Row
Row(
    children: [
        SizedBox(
            width: 130,
            height: 130,
            child: Image.asset(
                'assets/images/kurume.png', // Menampilkan
gambar ramen kurume
                fit: BoxFit.cover),
        ),
        const SizedBox(width: 30),
        const Column(
            crossAxisAlignment: CrossAxisAlignment
                .start, // Mengatur alignment kolom ke kiri
            children: [
                Text('Kurume Ramen',
                    style: TextStyle(
                        fontSize: 20,
                        fontWeight:
                            FontWeight.bold)), // Menampilkan
nama menu
                Text('Rp45.000',
                    style: TextStyle(fontSize: 20)), //
Menampilkan harga
            ],
        ),
    ],
),
const SizedBox(height: 10),

// Shoyu Ramen Row

```

```

        Row(
          children: [
            SizedBox(
              width: 130,
              height: 130,
              child: Image.asset(
                'assets/images/shoyu.png', // Menampilkan
gambar ramen shoyu
                fit: BoxFit.cover),
            ),
            const SizedBox(width: 30),
            const Column(
              crossAxisAlignment: CrossAxisAlignment
                .start, // Mengatur alignment kolom ke kiri
              children: [
                Text('Shoyu Ramen',
                  style: TextStyle(
                    fontSize: 20,
                    fontWeight:
nama menu
                      FontWeight.bold)), // Menampilkan
                Text('Rp48.500',
                  style: TextStyle(fontSize: 20)), //
Menampilkan harga
              ],
            ),
          ],
        ),
      ],
    ),

    // Spicy Ramen Row
    Row(
      children: [
        SizedBox(
          width: 130,
          height: 130,
          child: Image.asset(
            'assets/images/spicy.png', // Menampilkan
gambar ramen pedas
            fit: BoxFit.cover),
        ),
        const SizedBox(width: 30),
        const Column(
          crossAxisAlignment: CrossAxisAlignment
            .start, // Mengatur alignment kolom ke kiri
          children: [
            Text('Spicy Ramen',
              style: TextStyle(
                fontSize: 20,

```

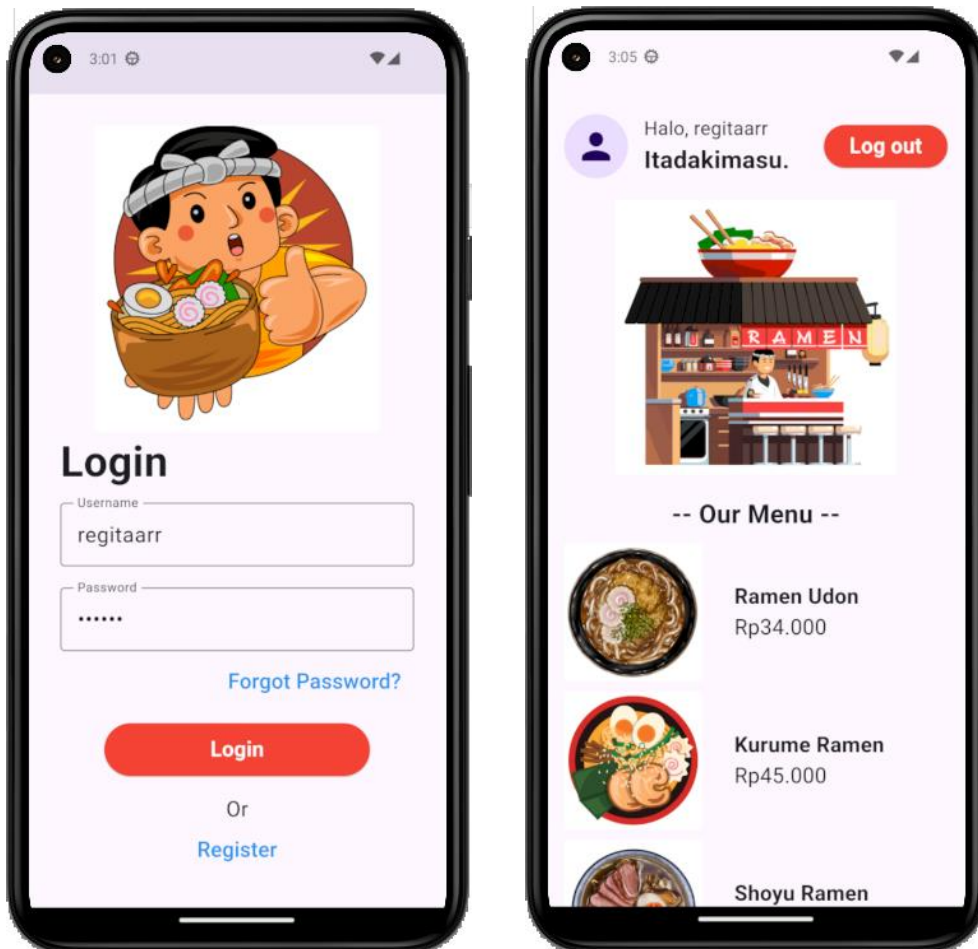
```

fontWeight:
    FontWeight.bold)), // Menampilkan
nama menu
    Text('Rp50.000',
        style: TextStyle(fontSize: 20)), //
Menampilkan harga
    ],
    ),
    ],
    ),
    ],
    ),
    ),
    );
}
}

```

### 3.2.2 Output

- Login berhasil



Berikut adalah penjelasan cara kerja API ketika login berhasil:

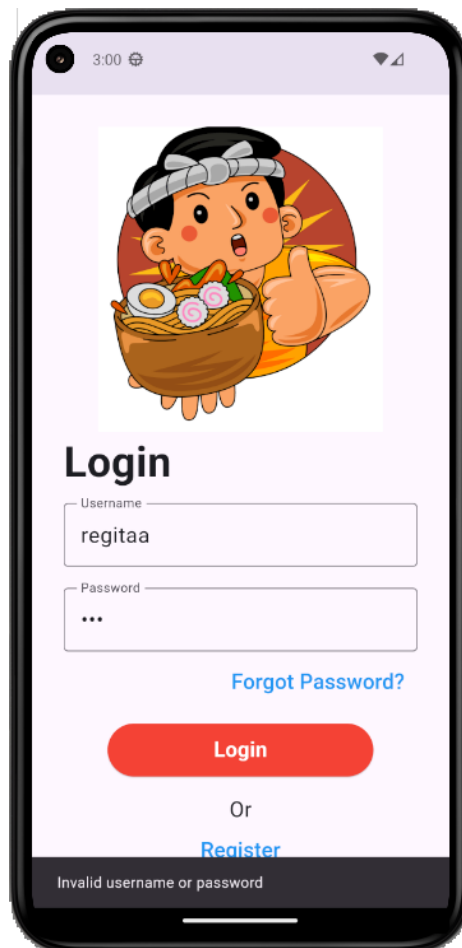
- Pengiriman Permintaan Login

- Ketika pengguna mengisi username dan password, kemudian menekan tombol "Login," metode `_login()` di dalam kelas `_LoginPageState` akan dipanggil.
  - Di dalam metode ini, URL API yang digunakan untuk proses login ditetapkan (`apiUrl`), yang dalam contoh ini adalah `http://10.0.2.2/mwsp/login.php`.
- Mengirim Permintaan HTTP POST
- Permintaan POST dikirim ke API menggunakan `http.post()`, dengan menyertakan:
    - **URL:** `apiUrl`.
    - **Headers:** Header `Content-Type` diatur ke `application/json`; `charset=UTF-8` untuk memberi tahu server bahwa data yang dikirim dalam format JSON.
    - **Body:** Data berupa JSON yang berisi username dan password, yang diambil dari kontroler `TextEditingController`:
- ```
dart
Copy code
body: jsonEncode(<String, String>{
  'username': _usernameController.text,
  'password': _passwordController.text,
}),
```
- Menerima Respons dari Server
- Setelah mengirimkan permintaan, aplikasi menunggu respons dari server:
- ```
dart
Copy code
if (response.statusCode == 200) {
```
- Jika status kode respons adalah 200 (yang berarti permintaan berhasil), maka respons dalam bentuk JSON diurai:
- ```
dart
Copy code
final Map<String, dynamic> responseData =
  json.decode(response.body);
```
- Memeriksa Status Respons
- Setelah mengurai respons, aplikasi memeriksa apakah status login berhasil dengan mengecek nilai dari `responseData['status']`:
- ```
dart
Copy code
if (responseData['status'] == 'success') {
```
- Jika statusnya adalah `success`, ini menandakan bahwa login berhasil. Dalam hal ini:

- Aplikasi kemudian akan menavigasi pengguna ke halaman `DashboardPage`, menggunakan `Navigator.push()`, dan mengoper username pengguna dan fungsi logout:

```
dart
Copy code
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => DashboardPage(
      username: _usernameController.text, onLogout:
_logout),
  ));
```

- Login tidak berhasil



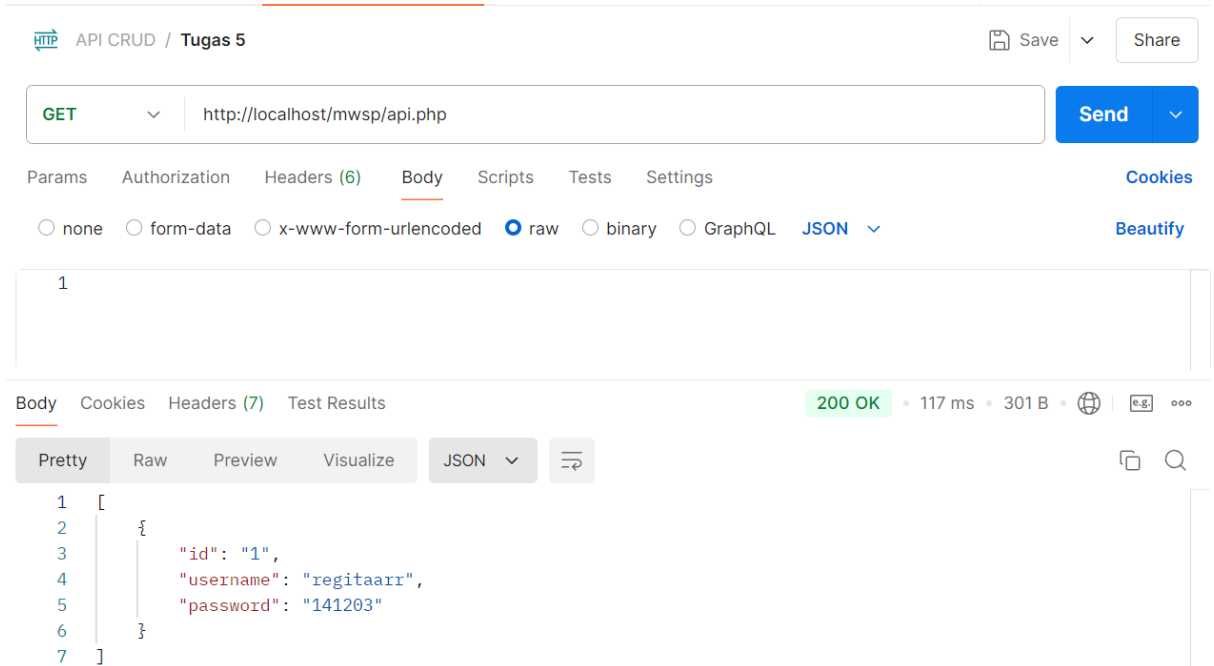
- Jika status bukan `success`, maka aplikasi akan menampilkan pesan kesalahan yang diterima dari server:

```
dart
Copy code
ScaffoldMessenger.of(context).showSnackBar(
```

```
SnackBar (content: Text (responseData ['message'])),
);
```

- Ini menampilkan pesan kesalahan kepada pengguna, memberitahu mereka tentang masalah yang terjadi saat mencoba login (misalnya, username atau password yang salah).

- **Postman**



Metode POST dalam login digunakan untuk mengirimkan data pengguna, seperti username dan password, ke server dengan aman. Saat login menggunakan metode POST, data tersebut tidak terlihat di URL (tidak seperti metode GET), sehingga lebih aman untuk mengirimkan informasi sensitif.

## REFERENSI

Admin. (2023, August 30). *Apa Itu REST API dan Bedanya dengan API Biasa?* Midtrans. <https://midtrans.com/id/blog/rest-api>

Anggara, A., & Pratama, D. (2024). *Modul Praktikum Mobile & Web Service*. Yogyakarta: Universitas Tekonologi Yogyakarta.

BuildWithAngga. (2023). 5 Fungsi Utama Software Postman API. BuildWithAngga. <https://buildwithangga.com/tips/5-fungsi-utama-software-postman-api>

Content IDCloudHost. (2020, July 14). Mengenal Apa itu Figma : Fitur, Fungsi, Cara Kerja / Menggunakannya. IDCloudHost. <https://idcloudhost.com/blog/mengenal-apa-itu-figma-fitur-fungsi-cara-kerja-menggunakannya/>

CRUD API (Buat, Baca, Perbarui, Hapus) | AppMaster. (2023). @Appmaster\_io; AppMaster - ultimate all-in no-code platform. <https://appmaster.io/id/glossary/crud-api-buat-baca-perbarui-hapus>

MTARGET. (2023, August 23). REST API: Definisi, Cara Kerja dan Contohnya. MTARGET Blog. <https://mtarget.co/blog/apa-itu-rest-api/>

Sebastian, A. (2023, November 15). Apa itu Postman: Pengertian, Fungsi dan 3 Perbedaan dengan Insomnia! Prismalink Payment Gateway Indonesia. <https://prismalink.co.id/apa-itu-postman/>