SISTEM OPERASIONAL PEMESANAN MAKANAN DENGAN QR "WAITING LINE METHOD"

VueJs

Ratu SukmaKomala
Isnaenti Nur Latifah
Rd. Nuraini Siti Fathonah

bukupedia

SISTEM OPERASIONAL PEMESANAN MAKANAN DENGAN QR *WAITING LINE METHOD*

Food Order Waiting Line Method Dengan VueJs

Ratu Sukmakomala
Isnaenti Nur Latifah
Rd. Nuraini Siti Fathonah

bukupedia

Sistem Operasional Pemesanan Makanan Dengan QR *Waiting Line Method*

Food Order Waiting Line Method Dengan VueJs

Penulis :
Ratu Sukmakomala
Isnaenti Nur Latifah
Rd. Nuraini Siti Fathonah

ISBN:

Editor:

Penyunting:

Desain Sampul dan Tata letak:
Ratu SukmaKomala
Isnaenti Nur Latifah

Penerbit:
Penerbit Buku Pedia

Redaksi:
Athena Residence Blok.E No. 1, Desa Ciwaruga,
Kec. Parongpong, Kab. Bandung Barat 40559
Tel. 628-775-2000-300
Email : penerbit@bukupedia.co.id

Distibutor:
Informatics Research CenterJl. Sariasih No. 54
Bandung 40151
Email : irc@ulbi.ac.id

Cetakan Pertama, 2023
Hak cipta dilindungi undang-undang
Dilarang memperbanyak karya tulis ini dalam bentuk dan
dengan cara apa pun tanpa ijin tertulis dari penerbit

PRAKATA

Pada setiap restoran umumnya masih menggunakan buku menu sebagai media pemesananan makanan tetapi masih terdapat kelemahan seperti tersedianya buku menu yang sedikit, jika ingin ada perubahan dalam menu sulit harus mencetak buku menu yang baru, dan apabila restoran sedang ramai catatan pemesanan pelanggan tidak berurutan sesuai dengan antriannya. Berdasarkan penelitian yang telah dilakukan oleh Universitas Islam Negeri Syarif Hidayatullah Jakarta dalam pengumpulan data pengaruh penggunaan kode QR dalam pemesanan menu makanan terhadap kepuasan customer dengan melakukan metode penelitian kuantitatif deskriptif. Dalam penelitiannya telah terkumpul hasil uji koefisien determinasi dengan nilai R square sebesar 0,089 (8,9%). Yang artinya 8,9% menunjukan hasil kepuasan customer terhadap penggunaan kode QR dari 105 orang sebagai sample dari objek observasi, kuesioner dan dokumentasi. Dengan ini penulis membuat buku yang berjudul SISTEM OPERASIONAL PEMESANAN MAKANAN DENGAN QR WAITING LINE METHOD untuk dapat mengatasi masalah yang terjadi sebelumnya.

Pada buku ini terdapat beberapa chapter diantaranya: pada chapter 1 membahas mengenai:Pengenalan Bahasa Pemrograman & Aplikasi, pada chapter 2 membahas mengenai: Sistem Pelayanan Pemesanan Makanan dengan Waiting Line Method, pada chapter3 membahas mengenai: Tatacara Instalasi Aplikasi Kebutuhan Pendukung dalam Pembuatan Website Pemesanan Makanan, pada chapter 4 membahas mengenai: Membuat Website Pemesanan Makanan, dan pada chapter 5 membahas mengenai: Debugging Aplikasi Pemesanan Makanan.

Penulis berharap buku ini dapat menambah pengetahuan dan pengalaman bagi pembaca untuk membuat website pemesanan makanan menggunakan scan barcode. Bahkan kami berharap lebih jauh lagi agar buku ini bisa pembaca praktekkan sesuai kebutuhan. Sebagai penulis merasa masih banyak kekurangan dalam penyusunan buku ini karena keterbatasan pengetahuan dan pengalaman kami. Untuk itu, penulis sangat mengharapkan kritik dan saran yang membangun dari pembaca demi kesempurnaan buku ini. Untuk memudahkan pembaca dalam mempelajari tutorial pada buku ini, kami telah menyediakan kode yang bisa diunduh. Kode untuk buku ini di Github dengan alamat

Bandung, 4 Desember 2023

Penulis

DAFTAR ISI

Contents

CHAPTER 1
PENGENALAN BAHASA PEMROGRAMAN & APLIKASI

1.1 JavaScript

JavaScript adalah bahasa pemrograman yang sering digunakan untuk mengembangkan aplikasi web interaktif. Ini adalah salah satu bahasa pemrograman yang paling umum digunakan di seluruh dunia. JavaScript berjalan pada sisi klien (di browser web pengguna) dan dapat digunakan untuk mengendalikan perilaku elemen-elemen HTML, berinteraksi dengan pengguna, mengambil data dari server, dan banyak lagi. JavaScript (biasanya disingkat menjadi JS) adalah bahasa pemrograman tingkat tinggi yang secara khusus digunakan dalam pengembangan aplikasi web.

JavaScript dapat digunakan untuk mengubah elemen-elemen HTML dan mengubah tampilan halaman dengan mengedit CSS. Misalnya, Anda dapat menambahkan elemen baru, mengubah teks, atau mengganti warna latar belakang dengan JavaScript. Selain itu javaScript memungkinkan pembuatan situs web yang dinamis dan interaktif. Anda dapat mengubah isi halaman web, merespons tindakan pengguna (seperti mengklik tombol), dan membuat efek animasi, semuanya secara langsung di browser.

1.2 Point Penting JavaScript

- Bahasa Pemrograman Sisi Klien : JavaScript berjalan pada sisi klien, yang berarti ia dieksekusi di browser web pengguna. Hal ini memungkinkan JavaScript untuk merespons tindakan pengguna, seperti mengklik tombol, mengisi formulir, dan sebagainya.
- Bahasa Skrip : JavaScript adalah bahasa pemrograman berjenis skrip, yang berarti kode JavaScript dapat dieksekusi tanpa kompilasi sebelumnya. Ini memungkinkan pengembang untuk mengedit dan menguji kode dengan cepat.
- Interaksi dengan HTML dan CSS : JavaScript digunakan untuk mengendalikan elemen-elemen HTML dan mengubah tampilan halaman web dengan mengubah CSS. Ini memungkinkan pembuatan tampilan dan efek yang dinamis.
- Variabel dan Tipe Data : JavaScript mendukung variabel dan tipe data seperti string, angka, boolean, dan objek. Ini memungkinkan pengembang untuk menyimpan dan memanipulasi data.
- Fungsi : Fungsi adalah blok kode yang dapat digunakan kembali dan dipanggil dengan nama tertentu. JavaScript memungkinkan pembuatan dan pemanggilan fungsi untuk mengorganisasi kode.

- Struktur Kontrol : JavaScript mendukung struktur kontrol seperti percabangan (if/else) dan pengulangan (for, while) untuk mengatur alur program.
- Objek : JavaScript adalah bahasa pemrograman berorientasi objek. Pengembang dapat membuat objek dan metode yang terkait dengan objek tersebut.
- Event Handling : JavaScript memungkinkan pengembang untuk menangani peristiwa (events) seperti klik tombol, pengisian formulir, dan lainnya. Ini memungkinkan pembuatan aplikasi web interaktif.
- AJAX : JavaScript digunakan dalam teknik AJAX (Asynchronous JavaScript and XML) untuk berkomunikasi dengan server tanpa harus me-refresh seluruh halaman web.
- Library dan Framework : Ada banyak perpustakaan (seperti jQuery) dan kerangka kerja (seperti React, Angular, dan Vue) yang dibangun di atas JavaScript untuk mempermudah pengembangan web.

1.3 Jenis Fitur dan Aspek Utama JavaScript

- Bahasa Pemrograman Serbaguna: adalah bahasa pemrograman yang serbaguna yang dapat digunakan untuk pengembangan aplikasi web di sisi klien (browser), di sisi server (menggunakan Node.js), dan dalam berbagai konteks lain, termasuk pengembangan aplikasi mobile dan desktop.
- Kecocokan dengan HTML dan CSS: JavaScript terintegrasi dengan baik dalam HTML dan dapat digunakan untuk mengendalikan elemen-elemen HTML serta mengubah tampilan dengan mengedit CSS. Hal ini memungkinkan Anda untuk menciptakan tampilan dan efek yang dinamis di halaman web.
- Sintaks Dinamis:JavaScript memungkinkan pembuatan kode yang bersifat dinamis. Anda dapat membuat, mengedit, dan menghapus elemen HTML dan kode JavaScript sendiri selama halaman web dijalankan.
- Asinkronitas: JavaScript mendukung pemrograman asinkron, yang memungkinkan operasi non-blokir. Ini berguna saat berurusan dengan pengiriman permintaan ke server, pengambilan data, atau operasi yang memerlukan waktu.
- Model Kejadian (Event Model):JavaScript memanfaatkan model kejadian yang memungkinkan Anda menangani tindakan pengguna dan respons terhadap peristiwa seperti klik tombol, pengisian formulir, atau pergerakan mouse.
- Manipulasi DOM:JavaScript memungkinkan Anda mengakses dan memanipulasi elemen-elemen HTML melalui DOM (Document Object Model). Anda dapat menambah, menghapus, atau mengubah elemen-elemen ini secara dinamis.
- Fungsi dan Modularitas:JavaScript memungkinkan Anda mendefinisikan dan memanggil fungsi. Ini membantu dalam mengorganisasi dan mengelola kode. Modularitas juga sangat didukung, memungkinkan kode yang dibagi menjadi modul yang dapat digunakan kembali.
- Pustaka dan Kerangka Kerja: Ada banyak pustaka (libraries) dan kerangka kerja (frameworks) JavaScript yang mempermudah pengembangan aplikasi web. Ini mencakup jQuery, React, Angular, Vue.js, dan banyak lainnya.
- Pemrograman Berbasis Komponen: Beberapa kerangka kerja JavaScript mempromosikan pemrograman berbasis komponen, di mana aplikasi dibangun dari komponen-komponen yang dapat digunakan kembali.
- Manajemen Ketergantungan: Alat seperti npm (Node Package Manager) memungkinkan Anda mengelola dan menginstal paket-paket pihak ketiga yang dibutuhkan dalam proyek Anda.
- Keamanan: JavaScript perlu diimplementasikan dengan hati-hati untuk menghindari kerentanan seperti Cross-Site Scripting (XSS). Ini melibatkan sanitasi input, validasi, dan penggunaan metode keamanan yang sesuai.
- Pengujian dan Debugging:Terdapat alat pengujian (testing tools) yang memungkinkan Anda untuk menguji kode JavaScript secara otomatis. Selain itu, alat pemecahan masalah (debugging tools) seperti DevTools di browser web membantu dalam mengidentifikasi dan memperbaiki kesalahan.
- Model Komunitas yang Kuat: JavaScript memiliki komunitas yang besar dan aktif. Anda dapat mengakses forum, tutorial, dan sumber daya online yang membantu dalam memahami, memecahkan masalah, dan memperluas pengetahuan Anda tentang JavaScript.

1.4 Apa Saja Variabel Dalam JavaScript

1. Variabel var: Variabel var digunakan untuk mendeklarasikan variabel dalam skrip JavaScript. Namun, variabel var cenderung memiliki cakupan yang lebih luas (function-scoped) daripada variabel let dan const.

```
var x = 10;
```

2. Variabel let: Variabel let digunakan untuk mendeklarasikan variabel yang memiliki cakupan blok (block-scoped). Ini berarti variabel hanya dapat diakses dalam blok di mana ia dideklarasikan.

```
let y = 20;
```

3. Variabel const: Variabel const digunakan untuk mendeklarasikan variabel yang memiliki cakupan blok dan nilainya tidak dapat diubah setelah diberikan nilai awal.

```
const z = 30;
```

4. Variabel Global: Variabel yang dideklarasikan di luar fungsi atau blok disebut variabel global. Ini berarti variabel dapat diakses dari seluruh skrip JavaScript.

```
var globalVar = "Ini adalah variabel global";
```

5. Variabel Lokal: Variabel yang dideklarasikan dalam fungsi atau blok memiliki cakupan lokal, yang berarti mereka hanya dapat diakses di dalam fungsi atau blok tersebut.

```
function exampleFunction() {
  var localVar = "Ini adalah variabel lokal";
}
```

6. Variabel Array: Variabel array digunakan untuk menyimpan kumpulan nilai dalam satu variabel. Nilai-nilai ini dapat diakses berdasarkan indeks.

```
var colors = ["merah", "hijau", "biru"];
```

7. Variabel Objek: Variabel objek digunakan untuk menyimpan data yang terstruktur dalam bentuk properti dan nilai. Properti objek dapat diakses dengan notasi titik (dot) atau notasi kurung siku.

```
var person = {
  name: "John",
  age: 30,
};
```

8. Variabel Tipe Data Primitif: JavaScript memiliki variabel yang menyimpan tipe data primitif seperti string, angka, boolean, null, undefined, symbol, dan bigint.

```
var name = "Alice";
var age = 25;
var isAdult = true;
var nothing = null;
var notDefined;
var uniqueSymbol = Symbol("unique");
var bigNumber = 123456789012345678901234567890123456789
```

1.5 Framework dan Library JavaScript

- React:React adalah sebuah library JavaScript yang dikembangkan oleh Facebook. Ini sangat populer untuk membangun antarmuka pengguna yang responsif dan dinamis. React menggunakan komponen untuk memisahkan tampilan menjadi bagian-bagian yang dapat dikelola secara independen.
- Angular: Angular adalah sebuah framework yang dikembangkan oleh Google. Ini adalah kerangka kerja lengkap untuk pengembangan aplikasi web dan aplikasi lintas platform. Angular menggunakan TypeScript sebagai bahasa pemrograman dan menawarkan berbagai alat untuk mengelola aspek front-end aplikasi.
- Vue.js:Vue.js adalah library JavaScript yang dirancang untuk memudahkan pengembangan aplikasi web yang dinamis. Ini sangat sederhana untuk dipelajari dan digunakan, dan sering digambarkan sebagai alternatif yang lebih ringan daripada React dan Angular.
- Ember.js: Ember.js adalah framework JavaScript yang dikembangkan untuk mempercepat pengembangan aplikasi web. Ini memiliki panduan konvensi yang kuat dan menyediakan alat-alat untuk membangun aplikasi yang besar dan kompleks.
- jQuery:jQuery adalah library JavaScript yang telah ada sejak lama. Meskipun tidak sepopuler di masa lalu, jQuery masih digunakan dalam berbagai proyek untuk menyederhanakan manipulasi DOM, animasi, dan interaktivitas pada situs web.
- Backbone.js:Backbone.js adalah library yang memungkinkan pengembang untuk membangun aplikasi web yang lebih terstruktur dengan pola desain Model-View-Controller (MVC). Ini memberikan dasar untuk pengembangan aplikasi yang lebih besar.
- Meteor: Meteor adalah platform yang lengkap untuk pengembangan aplikasi web dan mobile. Ini menggunakan JavaScript di sisi server (Node.js) dan di sisi klien, serta menyediakan alat untuk membangun aplikasi real-time.
- Express.js: Express.js adalah framework sisi server yang berjalan di atas Node.js. Ini digunakan untuk membangun aplikasi server dan API dengan mudah dan efisien.
- D3.js: D3.js adalah library JavaScript yang digunakan untuk visualisasi data. Ini memungkinkan pembuatan grafik interaktif dan visualisasi data yang kuat.
- Redux:Redux adalah library yang digunakan bersama-sama dengan React untuk mengelola keadaan (state) aplikasi. Ini membantu dalam mengatur data dan komunikasi antara komponen-komponen React.
- Electron: Electron adalah framework yang memungkinkan Anda untuk membangun aplikasi desktop lintas platform menggunakan HTML, CSS, dan JavaScript. Ini digunakan untuk membangun aplikasi seperti Visual Studio Code dan Slack.
- Next.js:Next.js adalah framework React yang dikhususkan untuk pengembangan aplikasi web yang dapat di-render di sisi server. Ini membantu dalam mempercepat waktu muat halaman web dan meningkatkan SEO.

1.6 Aplikasi Yang Menggunakan JavaScript

1. Aplikasi Web Umum:Hampir semua situs web modern menggunakan JavaScript untuk meningkatkan interaktivitas dan menghadirkan pengalaman pengguna yang lebih dinamis. Ini mencakup situs web berita, e-commerce, blog, jejaring sosial.
2. Aplikasi Sosial Media:Platform media sosial seperti Facebook, Twitter, dan Instagram menggunakan JavaScript untuk menghadirkan fitur-fitur seperti berbagi status, komentar, notifikasi real-time, dan interaksi antar pengguna.
3. Aplikasi E-commerce:Situs web e-commerce seperti Amazon dan eBay memanfaatkan JavaScript untuk mengelola keranjang belanja, filter produk, rekomendasi, dan pembayaran online.
4. Aplikasi Perbankan Online:Aplikasi perbankan online menggunakan JavaScript untuk memfasilitasi transfer dana, pengecekan saldo, pembayaran tagihan, dan manajemen akun pengguna.
5. Aplikasi Streaming Video:Layanan streaming video seperti YouTube, Netflix, dan Hulu menggunakan JavaScript untuk mengontrol pemutaran video, manajemen antrean, dan fitur berbagi.
6. Aplikasi Game Web:Ada banyak game web yang dibangun dengan JavaScript, seperti permainan puzzle, permainan kasual, dan game multiplayer.
7. Aplikasi Berbasis Lokasi: Aplikasi berbasis lokasi seperti Google Maps menggunakan JavaScript untuk menampilkan peta interaktif, mencari lokasi, dan menyediakan petunjuk arah.
8. Aplikasi Email:Layanan email seperti Gmail menggunakan JavaScript untuk membuat antarmuka email yang kaya dengan fitur seperti penyusunan pesan, kotak surat masuk, dan filter.
9. Aplikasi Komunikasi Real-Time:Aplikasi obrolan, telepon video, dan konferensi seperti WhatsApp, Zoom, dan Slack menggunakan JavaScript untuk menyediakan komunikasi real-time.
10. Aplikasi Mobile Cross-Platform: Framework seperti React Native dan NativeScript memungkinkan pengembang untuk menggunakan JavaScript untuk membangun aplikasi mobile yang berjalan di berbagai platform, termasuk iOS dan Android.
11. Aplikasi Desktop:Aplikasi desktop seperti Visual Studio Code dan Slack menggunakan teknologi Electron, yang memungkinkan JavaScript untuk membangun aplikasi desktop lintas platform.
12. Aplikasi Internet of Things (IoT):JavaScript digunakan dalam pengembangan aplikasi IoT untuk menghubungkan dan mengontrol perangkat fisik, seperti lampu pintar, kunci pintu pintar, dan thermostat cerdas.
13. Aplikasi Cloud:Layanan cloud computing seperti AWS Lambda dan Azure Functions memungkinkan pengembang untuk menulis fungsi serverless menggunakan JavaScript.
14. Aplikasi Game Mobile: Beberapa game mobile, terutama game puzzle dan kasual, menggunakan JavaScript dengan bantuan framework seperti Phaser.
15. Aplikasi AR/VR:Aplikasi Augmented Reality (AR) dan Virtual Reality (VR) sering menggunakan JavaScript untuk interaktivitas dan efek grafis.

1.7 Dasar Mengkoding Menggunakan JavaScript

- Mendeklarasikan Variabel:
  Variabel digunakan untuk menyimpan data. Anda dapat mendeklarasikan variabel menggunakan kata kunci 'var', 'let,'.

```
var nama = "John";
let usia = 30;
const PI = 3.14;
```

- Menggunakan Tipe Data:
  JavaScript memiliki beberapa tipe data dasar, seperti string (teks), angka, boolean (true/false), array (kumpulan data), objek (struktur data).

```
var nama = "Alice";
var umur = 25;
var sudahDewasa = true;
var angka = [1, 2, 3, 4, 5];
var orang = { nama: "Bob", usia: 28 };
```

- Operator:
  JavaScript mendukung operator aritmatika (+, -, *, /), operator perbandingan (<, >, ==, !=), operator logika (&&, ||).

```
var x = 10;
var y = 5;
var z = x + y; // z akan berisi 15
```

- Fungsi:
  Fungsi digunakan untuk mengelompokkan kode ke dalam blok yang dapat digunakan kembali. Anda dapat mendefinisikan fungsi dan memanggilnya.

```
function tambahkan(a, b) {
  return a + b;
}
var hasil = tambahkan(3, 4); // hasil akan berisi 7
```

- Percabangan (if/else):
  Percabangan memungkinkan Anda untuk menjalankan kode berdasarkan kondisi tertentu.

```
var usia = 20;
if (usia >= 18) {
  console.log("Anda sudah dewasa.");
} else {
  console.log("Anda masih anak-anak.");
}
```

❖ Pengulangan (for, while):
Pengulangan memungkinkan Anda menjalankan kode berulang kali.

```
for (var i = 0; i < 5; i++) {
  console.log(i);
}
```

❖ Manipulasi DOM:
JavaScript dapat digunakan untuk mengakses dan memanipulasi elemen-elemen HTML menggunakan DOM.

```
var elemen = document.getElementById("judul");
elemen.innerHTML = "Selamat Datang!";
```

❖ Peristiwa (Events):
JavaScript digunakan untuk menangani peristiwa seperti klik tombol atau pengisian formulir. Anda dapat menetapkan fungsi untuk dijalankan saat peristiwa terjadi.

```
var tombol = document.getElementById("tombolKlik");
tombol.addEventListener("click", function() {
  alert("Tombol telah diklik!");
});
```

❖ AJAX (Asynchronous JavaScript and XML):
AJAX memungkinkan Anda untuk berkomunikasi dengan server tanpa harus me-refresh seluruh halaman web. Ini memungkinkan aplikasi web menjadi lebih dinamis.

```
var xhr = new XMLHttpRequest();
xhr.open("GET", "data.json", true);
xhr.onload = function() {
  if (xhr.status == 200) {
    var data = JSON.parse(xhr.responseText);
    console.log(data);
  }
};
xhr.send();
```

❖ Komunitas dan Sumber Daya:
Pelajari lebih lanjut dengan membaca buku, kursus online, dan sumber daya online, dan berinteraksi dengan komunitas pengembang JavaScript

1.8 Dasar menanganani kesalahan menggunakan JavaScript

- Try-Catch Statement:
  Anda dapat menggunakan blok try dan catch untuk menangani kesalahan. Kode yang mungkin memunculkan kesalahan ditempatkan dalam blok try, dan jika kesalahan terjadi, kode dalam blok ‘catch’.

```
try {
  // Kode yang mungkin menyebabkan kesalahan
  var hasil = 10 / 0; // Membagi oleh nol akan menyebabk
} catch (error) {
  // Kode yang menangani kesalahan
  console.error("Terjadi kesalahan:", error.message);
}
```

- Penggunaan ‘throw’:
  Anda dapat menggunakan pernyataan throw untuk menghasilkan kesalahan secara manual. Ini berguna ketika Anda ingin mengendalikan aliran eksekusi berdasarkan kondisi tertentu.

```
function bagi(a, b) {
  if (b === 0) {
    throw new Error("Pembagian oleh nol tidak diperboleh
  }
  return a / b;
}

try {
  var hasil = bagi(10, 0);
} catch (error) {
  console.error("Terjadi kesalahan:", error.message);
}
```

- Penggunaan ‘finally’:
  Blok finally dapat digunakan bersama dengan try dan catch untuk mengeksekusi kode yang harus dijalankan, terlepas dari apakah kesalahan terjadi atau tidak.

```
try {
  // Kode yang mungkin menyebabkan kesalahan
} catch (error) {
  // Kode yang menangani kesalahan
} finally {
  // Kode yang selalu dijalankan, baik terjadi kesalahan
}
```

- Penggunaan ‘Error’ Object:
  Objek ‘Error’ adalah objek bawaan JavaScript yang dapat digunakan untuk membuat pesan kesalahan yang

lebih deskriptif dan rinci. Anda dapat membuat objek 'Error' dan melemparkannya menggunakan 'throw'

```
function validasiInput(nilai) {
  if (nilai < 0) {
    throw new Error("Nilai tidak boleh negatif");
  }
}

try {
  validasiInput(-5);
} catch (error) {
  console.error("Terjadi kesalahan:", error.message);
}
```

- Penanganan Kesalahan Spesifik:
  Anda dapat menangani kesalahan yang spesifik dengan menentukan tipe kesalahan yang akan ditangkap

```
try {
  // Kode yang mungkin menyebabkan kesalahan
} catch (error) {
  if (error instanceof TypeError) {
    // Kode untuk menangani kesalahan TypeError
  } else if (error instanceof ReferenceError) {
    // Kode untuk menangani kesalahan ReferenceError
  } else {
    // Kode untuk menangani jenis kesalahan lainnya
  }
}
```

.

1.9 Kelebihan JavaScript

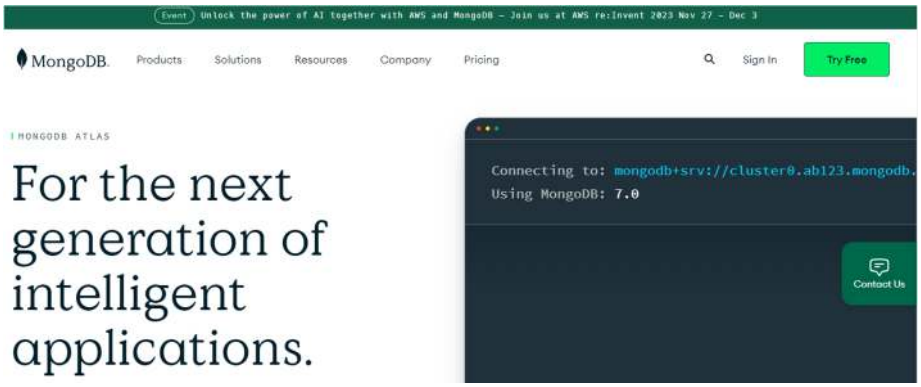
a. Dukungan Utama di Browser: JavaScript didukung oleh semua browser utama seperti Chrome, Firefox, Safari, dan Edge. Ini membuatnya menjadi bahasa standar untuk pengembangan web dan memastikan bahwa hampir setiap pengguna web dapat menjalankan kode JavaScript.
b. Sisi Klien (Client-Side): JavaScript berjalan di sisi klien (browser), yang memungkinkan aplikasi web untuk berjalan di perangkat pengguna tanpa memerlukan instalasi tambahan. Ini membuat pengembangan web lebih mudah diakses.
c. Bahasa Universal: JavaScript adalah bahasa pemrograman yang serbaguna, yang dapat digunakan dalam berbagai konteks, termasuk pengembangan web, aplikasi server (Node.js), aplikasi mobile (React Native), dan bahkan aplikasi desktop (Electron).
d. Ringan dan Cepat: JavaScript adalah bahasa yang ringan dan cepat dalam eksekusi. Ini membantu dalam memuat halaman web dengan cepat dan memberikan pengalaman pengguna yang responsif.
e. Interaktivitas: JavaScript memungkinkan pembuatan halaman web yang interaktif dan dinamis. Anda dapat mengubah isi halaman, menangani peristiwa pengguna, dan menghasilkan efek animasi dengan mudah.
f. DOM Manipulation: JavaScript memungkinkan Anda mengakses dan memanipulasi elemen-elemen HTML melalui DOM (Document Object Model). Anda dapat menambah, menghapus, atau mengubah elemen HTML secara dinamis.

g. Perkaya Pengalaman Pengguna: JavaScript memungkinkan pengembang untuk menyediakan pengalaman pengguna yang lebih kaya, termasuk validasi formulir, penggunaan geolokasi, audio, video, dan banyak lagi.
h. Asynchronous Programming: JavaScript mendukung pemrograman asinkron, yang memungkinkan aplikasi untuk melakukan operasi tanpa harus menunggu operasi sebelumnya selesai. Ini sangat berguna dalam mengambil data dari server atau menjalankan tugas-tugas latar belakang.
i. Komunitas yang Besar: JavaScript memiliki komunitas pengembang yang sangat besar dan aktif. Ini berarti ada banyak sumber daya, library, dan framework yang tersedia untuk membantu dalam pengembangan.
j. Cross-Platform Development: JavaScript dapat digunakan untuk membangun aplikasi yang berjalan di berbagai platform, termasuk desktop, web, dan mobile, dengan menggunakan alat-alat seperti Electron, React Native, dan NativeScript.
k. Transpilers: Ada alat seperti Babel yang memungkinkan Anda menulis kode JavaScript dengan versi terbaru (ECMAScript 6 dan seterusnya) dan kemudian mengonversinya menjadi versi JavaScript yang lebih lama untuk memastikan kompatibilitas dengan browser yang lebih lama.
l. Paket dan Manajemen Ketergantungan: Alat manajemen paket seperti npm (Node Package Manager) digunakan untuk mengelola dan menginstal paket-paket pihak ketiga yang diperlukan oleh proyek Anda.

1.10 Kekurangan JavaScript

a. Keterbatasan Keamanan: JavaScript memiliki risiko keamanan seperti Cross-Site Scripting (XSS), di mana penyerang dapat menyisipkan skrip berbahaya ke dalam halaman web, dan Cross-Site Request Forgery (CSRF), di mana penyerang dapat memanipulasi permintaan HTTP yang diajukan oleh pengguna. Pengembang perlu mengambil langkah-langkah ekstra untuk mengamankan aplikasi mereka.
b. Kinerja Tergantung pada Browser: Kinerja JavaScript dapat bervariasi tergantung pada browser yang digunakan. Beberapa fitur JavaScript mungkin berjalan lebih lambat di browser tertentu, dan perlu pengujian lintas browser yang ekstensif untuk memastikan kompatibilitas.
c. Mengganggu Pengguna: Beberapa pengguna menganggap penggunaan berlebihan JavaScript sebagai mengganggu dan mengganggu pengalaman pengguna. Oleh karena itu, penggunaan JavaScript yang bijak diperlukan untuk menjaga kepuasan pengguna.
d. Mengakibatkan Blok Konten: Ketika pengunduhan kode JavaScript yang besar diperlukan, hal ini dapat mengakibatkan penundaan dalam muatan halaman web. Ini bisa mengganggu pengalaman pengguna jika tidak dikelola dengan baik.
e. Keterbatasan SEO: Meskipun mesin pencari seperti Google semakin mampu mengindeks konten yang dibangun dengan JavaScript, beberapa aplikasi yang sepenuhnya bergantung pada JavaScript mungkin mengalami kesulitan dalam hal SEO (Search Engine Optimization) karena mesin pencari tidak selalu dapat membaca konten yang dibuat dengan JavaScript.
f. Tidak Dapat Menjalankan Di Lingkungan Terbatas: JavaScript tidak dapat dijalankan di beberapa lingkungan terbatas seperti server email atau perangkat seluler lama.
g. Membutuhkan Kemampuan: Bahasa JavaScript yang dinamis dan fleksibel juga berarti lebih mungkin membuat kesalahan dalam kode. Oleh karena itu, pengembang perlu berhati-hati dan memahami konsep dasar bahasa ini.
h. Sumber Daya Eksternal: Saat menggunakan library dan framework JavaScript, pengembang harus mengandalkan sumber daya eksternal yang mungkin dapat berhenti beroperasi atau berubah seiring waktu. Ini dapat memengaruhi keberlanjutan aplikasi.
i. Keterbatasan Sisi Klien: JavaScript berjalan di sisi klien, sehingga sebagian besar logika aplikasi harus dikelola di browser pengguna. Hal ini dapat membatasi kemampuan untuk melakukan operasi seperti mengakses sistem file lokal atau mengelola tugas server yang berat.
j. Pentingnya Keberlanjutan: Karena JavaScript terus berkembang dengan rilis baru (ECMAScript), pengembang perlu memastikan bahwa kode mereka tetap kompatibel dengan versi JavaScript yang lebih lama, atau merencanakan pembaruan yang diperlukan.

1.11 Mongodb



MongoDB adalah sebuah sistem manajemen basis data (DBMS) yang bersifat dokumen dan open source. Ini adalah jenis basis data NoSQL yang dirancang untuk menyimpan dan mengelola data dalam bentuk dokumen semi-struktural, yang biasanya disimpan dalam format JSON. MongoDB mengutamakan fleksibilitas, skalabilitas, dan kinerja tinggi.

Beberapa karakteristik utama MongoDB adalah sebagai berikut:

- Dokumen: Data disimpan dalam bentuk dokumen BSON (Binary JSON), yang memungkinkan Anda untuk menyimpan data yang berbeda dalam satu koleksi tanpa skema yang ketat. Hal ini memudahkan penambahan, penghapusan, atau perubahan bidang data tanpa memerlukan perubahan skema.
- Sistem yang Terdistribusi: MongoDB mendukung replikasi dan shard untuk skalabilitas horizontal dan toleransi kesalahan. Ini memungkinkan pengguna untuk mengelola basis data yang lebih besar dan meningkatkan ketersediaan.
- Query Ekspresif: MongoDB menyediakan bahasa kueri yang kuat untuk mencari, menyaring, dan mengakses data. Anda dapat melakukan pencarian berdasarkan atribut dalam dokumen, membuat indeks untuk meningkatkan kinerja, dan menjalankan berbagai operasi agregasi.
- Ketersediaan Tinggi: MongoDB mendukung replikasi data di berbagai server, sehingga jika satu server gagal, data masih dapat diakses dari server lainnya. Ini meningkatkan ketersediaan sistem.
- Skalabilitas Horizontal: MongoDB mendukung shard, yang memungkinkan data dibagi menjadi beberapa server untuk meningkatkan kapasitas dan kinerja basis data.
- Open Source: MongoDB tersedia sebagai perangkat lunak sumber terbuka dengan lisensi yang memungkinkan penggunaan, modifikasi, dan distribusi secara gratis.

1.12 Fungsi Utama Mongodb

a. Penyimpanan Data Dokumen: MongoDB memungkinkan Anda untuk menyimpan data dalam bentuk dokumen BSON (Binary JSON). Ini memungkinkan penyimpanan data semi-struktural yang lebih fleksibel daripada basis data relasional, di mana setiap dokumen dalam koleksi dapat memiliki struktur yang berbeda.
b. Query Ekspresif: MongoDB menyediakan bahasa kueri yang kuat untuk mencari, menyaring, dan mengakses data. Anda dapat melakukan pencarian berdasarkan atribut dalam dokumen, membuat indeks untuk meningkatkan kinerja, dan menjalankan berbagai operasi agregasi.
c. Indeks: MongoDB mendukung pembuatan indeks untuk mempercepat operasi pencarian dan pengurutan data. Indeks memungkinkan akses cepat ke dokumen yang sesuai dengan kriteria pencarian.
d. Replikasi: MongoDB mendukung replikasi data di berbagai server, yang membantu dalam meningkatkan ketersediaan sistem dan mengurangi risiko kehilangan data. Ini memungkinkan server cadangan untuk secara otomatis menggantikan server utama yang mengalami kegagalan.

e. Shard: MongoDB mendukung shard untuk skalabilitas horizontal. Ini memungkinkan data dibagi menjadi beberapa server (shard) untuk meningkatkan kapasitas dan kinerja basis data. Shard juga memungkinkan distribusi data yang lebih merata.
f. Ketersediaan Tinggi: MongoDB menyediakan berbagai opsi untuk meningkatkan ketersediaan sistem, termasuk replikasi, auto-failover, dan deteksi kesalahan otomatis.
g. Penyimpanan File: MongoDB memiliki fungsi GridFS yang memungkinkan Anda untuk menyimpan file biner besar seperti gambar, video, dan dokumen dalam basis data. Ini memungkinkan akses cepat dan manajemen file.
h. Fungsi Agregasi: MongoDB memiliki berbagai operasi agregasi yang memungkinkan Anda untuk melakukan perhitungan kompleks dan analisis data dalam koleksi, seperti grouping, sorting, dan counting.
i. Geospasial: MongoDB memiliki dukungan untuk data geospasial, yang memungkinkan penyimpanan dan pencarian data berdasarkan koordinat geografis.
j. Dukungan untuk Bahasa Pemrograman: MongoDB memiliki driver dan dukungan untuk berbagai bahasa pemrograman, sehingga Anda dapat mengintegrasikan basis data MongoDB dengan aplikasi yang ditulis dalam berbagai bahasa.

1.13 Penggunaan Driver MongoDB Dalam Beberapa Bahasa Pemrograman

- Python:
  Penggunaan driver PyMongo dalam Python untuk menghubungkan ke basis data MongoDB, memasukkan dokumen, dan melakukan pencarian sederhana:

```python
from pymongo import MongoClient

# Menghubungkan ke server MongoDB (default: localhost, port: 27017)
client = MongoClient()

# Mengakses basis data
db = client.mydatabase

# Mengakses koleksi
collection = db.mycollection

# Memasukkan dokumen
data = {"name": "John", "age": 30}
collection.insert_one(data)

# Melakukan pencarian
result = collection.find({"name": "John"})
for doc in result:
    print(doc)
```

- JavaScript (Node.js):
  Penggunaan driver MongoDB dalam Node.js menggunakan modul 'mongodb':

```
const MongoClient = require('mongodb').MongoClient;

// URL koneksi MongoDB (default: localhost, port: 27017)
const url = 'mongodb://localhost:27017';

// Menghubungkan ke server MongoDB
MongoClient.connect(url, function(err, client) {
    if (err) throw err;

    // Mengakses basis data
    const db = client.db('mydatabase');

    // Mengakses koleksi
    const collection = db.collection('mycollection');

    // Memasukkan dokumen
    const data = { name: 'John', age: 30 };
    collection.insertOne(data, function(err, result) {
        if (err) throw err;
        console.log('Dokumen telah dimasukkan');
    });

    // Melakukan pencarian
    collection.find({ name: 'John' }).toArray(function(err, docs) {
        if (err) throw err;
        console.log(docs);
        client.close(); // Tutup koneksi setelah selesai
    });
});
```

- Java:

  Penggunaan driver MongoDB dalam Java menggunakan MongoDB Java Driver:

```
import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;

// Membuat koneksi ke server MongoDB (default: localhost, port: 27017)
MongoClient mongoClient = new MongoClient();

// Mengakses basis data
MongoDatabase database = mongoClient.getDatabase("mydatabase");

// Mengakses koleksi
MongoCollection<Document> collection = database.getCollection("mycollection

// Memasukkan dokumen
Document doc = new Document("name", "John").append("age", 30);
collection.insertOne(doc);

// Melakukan pencarian
Document searchQuery = new Document("name", "John");
for (Document document : collection.find(searchQuery)) {
    System.out.println(document.toJson());
}
```

1.14 Kelebihan MongoDB

- Fleksibilitas Skema: MongoDB memungkinkan Anda untuk menyimpan data dengan struktur yang bervariasi dalam satu koleksi, sehingga Anda tidak perlu mendefinisikan skema tabel terlebih dahulu. Ini memungkinkan adaptasi yang mudah terhadap perubahan kebutuhan aplikasi.
- Kinerja Tinggi: MongoDB dirancang untuk memberikan kinerja tinggi. Dukungan untuk indeks, replikasi, dan shard memungkinkan Anda untuk meningkatkan kinerja dan skalabilitas sesuai dengan kebutuhan aplikasi.
- Skalabilitas Horizontal: Anda dapat dengan mudah menambahkan lebih banyak server (shard) untuk mengatasi peningkatan beban aplikasi. Ini memungkinkan Anda untuk mengatasi basis data yang sangat besar dan meningkatkan kinerja.
- Dukungan untuk Dokumen Semi-struktural: MongoDB mendukung penyimpanan dan pengambilan data dalam format dokumen BSON (Binary JSON), yang sesuai dengan data semi-struktural seperti dokumen JSON. Ini sangat berguna dalam pengembangan aplikasi yang membutuhkan fleksibilitas dalam penyimpanan data.
- Bahasa Kueri yang Kuat: MongoDB menyediakan bahasa kueri yang kuat dan ekspresif untuk melakukan operasi pencarian, penyaringan, dan agregasi data dengan mudah. Anda dapat melakukan pencarian berdasarkan atribut dalam dokumen dan melakukan operasi agregasi yang kompleks.
- Replikasi dan Ketersediaan Tinggi: MongoDB mendukung replikasi data untuk meningkatkan ketersediaan sistem dan memastikan bahwa data tetap tersedia bahkan jika satu server mengalami kegagalan. Fitur auto-failover memungkinkan server cadangan menggantikan server utama yang mengalami masalah.
- Dukungan untuk Indeks: MongoDB mendukung pembuatan indeks, yang mempercepat operasi pencarian dan pengurutan data. Anda dapat membuat indeks pada bidang yang paling sering digunakan dalam pencarian.
- Dukungan untuk Geospasial: MongoDB memiliki dukungan bawaan untuk data geospasial, yang memungkinkan Anda untuk menyimpan dan mencari data berdasarkan koordinat geografis.
- GridFS: MongoDB memiliki fungsi GridFS yang memungkinkan penyimpanan dan pengambilan file biner besar seperti gambar, video, dan dokumen dalam basis data.
- Dukungan Komunitas yang Kuat: MongoDB adalah perangkat lunak sumber terbuka dan memiliki komunitas pengguna yang aktif, yang berarti Anda dapat menemukan banyak sumber daya dan dukungan online.
- Bahasa Pemrograman yang Beragam: MongoDB memiliki driver dan SDK untuk berbagai bahasa pemrograman populer, sehingga Anda dapat mengintegrasikannya dengan aplikasi yang ditulis dalam bahasa tersebut.
- Dokumentasi yang Bagus: MongoDB menyediakan dokumentasi yang baik dan sumber daya pembelajaran yang mendukung pengembang dalam menguasai platform ini.

1.15 *Virtual Private Server (VPS)*

Sebuah Virtual Private Server (VPS) adalah bentuk hosting web yang menggabungkan elemen-elemen dari hosting bersama (shared hosting) dan server pribadi fisik (dedicated server). Dalam model ini, sebuah server fisik besar dibagi menjadi beberapa server virtual, di mana setiap server virtual memiliki lingkungan operasi yang terisolasi dan sumber daya yang dapat dialokasikan secara independen.

Berikut beberapa konsep kunci terkait VPS:

- Virtualization:
  Teknologi virtualisasi memungkinkan server fisik dibagi menjadi beberapa server virtual. Hypervisor adalah perangkat lunak yang digunakan untuk menciptakan dan mengelola mesin virtual ini.
- Isolation:
  Setiap VPS beroperasi secara terisolasi dari yang lain. Artinya, satu VPS tidak dapat mengakses atau mempengaruhi VPS lainnya. Isolasi ini memastikan keamanan dan kinerja yang lebih baik.
- Resource Allocation:
  Sumber daya seperti CPU, RAM, penyimpanan, dan bandwidth dibagi di antara VPS yang ada pada server fisik. Setiap VPS memiliki alokasi sumber daya yang dapat dikonfigurasi sesuai kebutuhan.

- Root Access:

  Pengguna VPS biasanya diberikan akses root atau administrator penuh ke lingkungan mereka. Ini memungkinkan pengguna untuk menginstal dan mengonfigurasi perangkat lunak, serta mengelola pengaturan server.
- Flexibility:
  VPS memberikan fleksibilitas yang lebih besar dibandingkan shared hosting. Pengguna memiliki kontrol lebih besar atas konfigurasi server dan dapat menginstal perangkat lunak sesuai kebutuhan mereka.
- Cost-Efficiency:
  Dibandingkan dengan dedicated hosting, VPS menawarkan solusi yang lebih terjangkau karena sumber daya server fisik dibagi di antara beberapa pengguna.
- Scalability:
  Pengguna dapat dengan mudah menyesuaikan sumber daya yang dialokasikan untuk VPS sesuai dengan kebutuhan mereka. Ini memungkinkan pengguna untuk mengatasi lonjakan lalu lintas atau memperbesar kapasitas server saat bisnis mereka berkembang.
- Operating System (OS) Choice:
  Pengguna dapat memilih sistem operasi yang sesuai dengan kebutuhan mereka. Berbagai distribusi Linux dan beberapa versi Windows umumnya tersedia.
- Control Panel:
  Sebagian besar penyedia VPS menyediakan kontrol panel yang memudahkan manajemen VPS, termasuk instalasi perangkat lunak, pemantauan kinerja, dan manajemen sumber daya.
- Security:
  Karena setiap VPS beroperasi secara terpisah, keamanan lebih mudah dijaga. Meskipun masih mungkin terjadi kerentanan, dampaknya terbatas pada satu VPS dan tidak memengaruhi yang lain.

1.16 Bahasa Pemrograman Yang Menggunakan *VPS*

Beberapa bahasa pemrograman yang dapat digunakan dengan VPS:

- PHP:
  PHP sangat umum digunakan dalam pengembangan web dan sering diimplementasikan di server VPS dengan dukungan untuk web server seperti Apache atau Nginx.

- JavaScript (Node.js):
  Node.js memungkinkan pengembangan server-side dengan JavaScript. Ini dapat dijalankan di server VPS menggunakan runtime Node.js.

- Python:

Python adalah bahasa pemrograman serbaguna yang dapat digunakan untuk berbagai keperluan, termasuk pengembangan web. Framework populer seperti Django dan Flask dapat dijalankan di server VPS.

- Ruby:
  Ruby dan framework web seperti Ruby on Rails dapat diimplementasikan di server VPS.

- Java:
  Java sering digunakan untuk pengembangan aplikasi berbasis server dan dapat dijalankan di server VPS dengan dukungan Java Runtime Environment (JRE).

- Go (Golang):
  Go adalah bahasa pemrograman yang dikembangkan oleh Google dan dapat digunakan untuk pengembangan aplikasi backend. Program Go dapat dijalankan di server VPS.

- C#:
  C# digunakan secara luas untuk pengembangan aplikasi berbasis .NET. Aplikasi berbasis .NET dapat dihosting di server VPS dengan dukungan .NET runtime.

- Perl:
  Perl adalah bahasa pemrograman yang sering digunakan untuk skrip dan pengolahan teks. Perl dapat dijalankan di server VPS yang mendukung interpreter Perl.

- Shell Scripting:
  Bahasa skrip shell seperti Bash sering digunakan untuk otomatisasi tugas di lingkungan server dan dapat dijalankan di VPS.

1.17 Scrip Code Menggunakan VPS

1. Shell Script (Bash):

```bash
#!/bin/bash
echo "Hello, World! This script is running on a VPS."
```

2. Python Script:

```python
#!/usr/bin/env python3
print("Hello, World! This script is running on a VPS.")
```

3. Node.js Script:

```javascript
// hello.js
console.log("Hello, World! This script is running on a VPS.");
```

4. PHP Script:

```php
<?php
echo "Hello, World! This script is running on a VPS.";
?>
```

5. Java Program:

```java
// HelloWorld.java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World! This program is running on a VPS.");
    }
}
```

1.18 Kelebihan Virtual Private Server

- Isolasi Sumber Daya:
  Setiap VPS beroperasi secara terisolasi dari yang lain di server fisik yang sama. Ini berarti sumber daya seperti CPU, RAM, dan ruang disk dihindarkan dari penggunaan dan pengaruh yang tidak diinginkan oleh VPS lainnya.
- Kontrol Penuh:
  Pengguna VPS memiliki akses root atau administrator penuh, memberikan kontrol penuh terhadap konfigurasi server, instalasi perangkat lunak, dan pengelolaan pengaturan server.
- Skalabilitas:
  VPS memungkinkan pengguna untuk dengan mudah menyesuaikan sumber daya yang dialokasikan sesuai dengan kebutuhan mereka. Ini membuatnya lebih mudah untuk menangani lonjakan lalu lintas atau memperbesar kapasitas server saat bisnis atau aplikasi tumbuh.
- Kemampuan Menyesuaikan Lingkungan:
  Pengguna dapat memilih sistem operasi, konfigurasi server, dan lingkungan pengembangan yang sesuai dengan kebutuhan mereka. Ini memberikan fleksibilitas yang lebih besar dibandingkan dengan shared hosting.
- Kinerja Lebih Baik:

Dibandingkan dengan shared hosting, VPS umumnya menawarkan kinerja yang lebih baik karena sumber daya server tidak dibagi dengan banyak pengguna. Setiap VPS memiliki alokasi sumber daya yang dapat dikonfigurasi sesuai kebutuhan.

- Keamanan yang Lebih Baik:
  Isolasi antara VPS menciptakan lapisan keamanan tambahan. Jika satu VPS mengalami masalah keamanan atau terinfeksi malware, itu tidak langsung memengaruhi yang lain.
- Akses SSH dan Shell:
  VPS memungkinkan pengguna untuk mengakses server mereka melalui protokol SSH, memberikan akses ke shell. Ini mempermudah administrasi server dan eksekusi perintah dari jarak jauh.
- Dukungan untuk Berbagai Aplikasi:
  VPS dapat digunakan untuk menjalankan berbagai jenis aplikasi, termasuk situs web, aplikasi web, basis data, server game, dan sebagainya. Ini membuatnya cocok untuk berbagai kebutuhan pengguna.
- Kemudahan Manajemen:
  Banyak penyedia VPS menyediakan antarmuka pengguna atau panel kontrol yang memudahkan manajemen server, instalasi perangkat lunak, dan pemantauan kinerja.
- Harga yang Terjangkau:
  Dibandingkan dengan dedicated hosting, VPS menawarkan solusi yang lebih terjangkau, memungkinkan pengguna untuk memanfaatkan sejumlah besar sumber daya tanpa biaya yang sama tinggi dengan dedicated server.

1.19 Kekurangan Virtual Private Server

1. Harga Lebih Tinggi Dibandingkan Shared Hosting:
   Meskipun lebih terjangkau daripada dedicated server, VPS masih cenderung lebih mahal daripada shared hosting. Ini bisa menjadi kendala bagi pengguna yang memiliki anggaran hosting yang sangat terbatas.
2. Tingkat Keterampilan Diperlukan:
   Mengelola VPS membutuhkan tingkat keterampilan teknis yang lebih tinggi daripada shared hosting. Pengguna harus memiliki pemahaman tentang konfigurasi server, keamanan, dan manajemen sumber daya.
3. Keterbatasan Sumber Daya Fisik:
   Meskipun VPS menawarkan isolasi sumber daya, mereka masih berbagi sumber daya fisik dengan VPS lain di server yang sama. Jika server fisik mengalami masalah atau overloading, dapat memengaruhi kinerja semua VPS di dalamnya.
4. Keamanan Tergantung pada Pengaturan Pengguna:
   Keamanan VPS sepenuhnya tergantung pada pengaturan dan tindakan keamanan yang diambil oleh pengguna. Jika pengguna tidak memperbarui perangkat lunak, mengelola izin dengan benar, atau melaksanakan praktik keamanan yang baik, VPS dapat menjadi rentan terhadap ancaman keamanan.
5. Keterbatasan Sumber Daya di Beberapa Penyedia:
   Beberapa penyedia VPS mungkin memberikan sumber daya terbatas dalam paket dasar mereka, dan peningkatan sumber daya dapat menjadi mahal. Pengguna perlu memilih paket yang sesuai dengan kebutuhan mereka atau mempertimbangkan penyedia dengan skala yang lebih baik.
6. Kesulitan Skala Besar:
   Meskipun VPS mudah diukur, jika aplikasi atau situs web Anda tumbuh secara signifikan, mungkin lebih mudah dan lebih efisien untuk beralih ke lingkungan dedicated hosting atau cloud hosting yang dapat diukur lebih mudah.
7. Ketergantungan pada Penyedia Layanan:
   Pengguna VPS bergantung pada penyedia layanan untuk menyediakan dan mengelola infrastruktur fisik. Jika penyedia mengalami masalah atau kegagalan, ini dapat memengaruhi ketersediaan dan kinerja VPS.
8. Keterbatasan Kapasitas Hardware:
   VPS dapat terbatas oleh kapasitas hardware fisik server tempat mereka dihosting. Jika server fisik mencapai batas kapasitasnya, ini dapat mempengaruhi kinerja semua VPS yang ada di dalamnya.
9. Kinerja Terpengaruh oleh Sesama Pengguna:

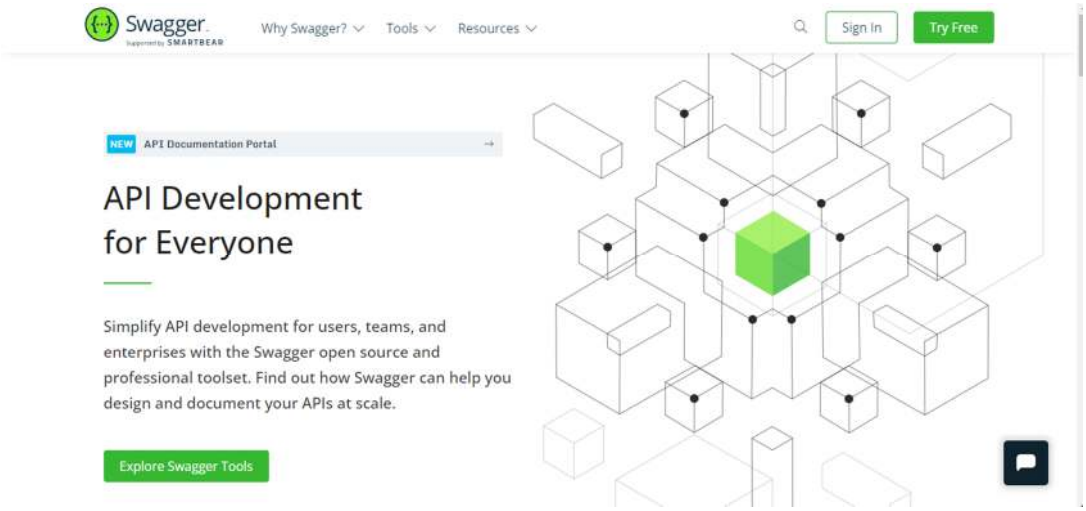
Meskipun terisolasi, jika satu atau beberapa VPS di server fisik menggunakan sumber daya secara berlebihan, ini dapat memengaruhi kinerja VPS lainnya pada server yang sama.

1.20 Seberapa Banyak VPS Digunakan?

Jumlah VPS yang digunakan dapat bervariasi secara signifikan tergantung pada kebutuhan dan skala pengguna atau bisnis tertentu. Faktor dapat mempengaruhi jumlah VPS yang diperlukan, termasuk:

a. Jenis Aplikasi atau Layanan:
   Aplikasi atau layanan yang lebih besar atau memiliki permintaan kinerja yang tinggi mungkin memerlukan lebih banyak sumber daya, dan oleh karena itu, dapat memerlukan lebih banyak VPS.
b. Volume Lalu Lintas:
   Situs web atau aplikasi dengan volume lalu lintas tinggi mungkin memerlukan lebih banyak sumber daya untuk menangani permintaan pengguna secara efisien.
c. Skala Bisnis:
   Bisnis yang berkembang mungkin perlu menambah jumlah VPS untuk menangani peningkatan beban kerja atau untuk memisahkan fungsi-fungsi tertentu, seperti database atau server aplikasi.
d. Kebutuhan Spesifik Aplikasi:
   Beberapa aplikasi atau layanan mungkin memiliki persyaratan atau kebutuhan khusus yang memerlukan penyesuaian jumlah dan konfigurasi VPS.
e. Distribusi Tugas dan Kinerja:
   Beberapa pengguna mungkin memilih untuk mendistribusikan tugas dan layanan di beberapa VPS untuk meningkatkan keamanan, isolasi, dan kinerja.
f. Jenis Hosting:
   Pilihan untuk menggunakan VPS di cloud atau di lingkungan hosting tertentu dapat mempengaruhi fleksibilitas dan kemampuan untuk dengan cepat menambah atau mengurangi jumlah VPS.
g. Strategi Ketersediaan dan Keandalan:
   Strategi ketersediaan tinggi atau pengelolaan beban kerja yang baik dapat memerlukan penggunaan lebih dari satu VPS untuk memastikan bahwa layanan tetap aktif dan responsif.
h. Anggaran:
   Anggaran yang tersedia juga dapat membatasi jumlah VPS yang dapat digunakan. Beberapa pengguna mungkin memilih untuk memulai dengan jumlah VPS yang lebih kecil dan menambahnya seiring waktu seiring pertumbuhan bisnis atau kebutuhan aplikasi.

1.21 *Swagger*

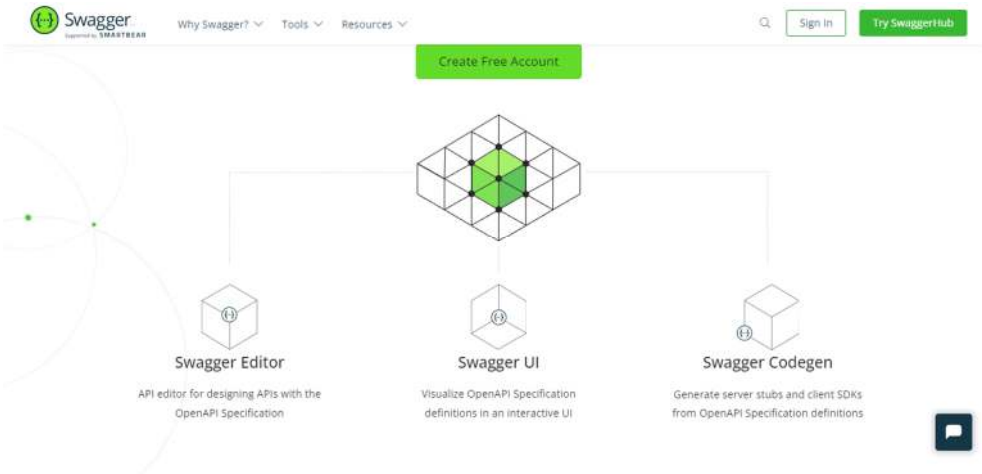


Swagger adalah sebuah alat (tool) dan kerangka kerja (framework) pengembangan aplikasi yang digunakan untuk merancang, membangun, dan mendokumentasikan layanan web RESTful. Swagger memungkinkan pengembang untuk mendefinisikan, mengonfigurasi, dan mendokumentasikan API secara terstruktur.

Fitur utama dari Swagger melibatkan:

1. Spesifikasi API: Swagger memungkinkan pengembang untuk membuat spesifikasi API secara terstruktur menggunakan format YAML (YAML Ain't Markup Language) atau JSON (JavaScript Object Notation). Spesifikasi ini mencakup detail seperti endpoint API, parameter yang diperlukan, respons yang diharapkan, dan informasi lainnya.
2. UI Interaktif: Swagger menyediakan antarmuka pengguna grafis interaktif yang memungkinkan pengguna untuk menjelajahi dan menguji endpoint API secara langsung dari browser. Ini memudahkan pengembang untuk memahami dan menguji fungsionalitas API tanpa harus membuat permintaan HTTP secara manual.
3. Generasi Kode Otomatis: Swagger dapat menghasilkan kode sumber otomatis untuk klien API dalam berbagai bahasa pemrograman berdasarkan spesifikasi yang dibuat. Ini membantu dalam pengembangan klien yang konsisten dengan API yang dijelaskan oleh spesifikasi Swagger.
4. Dokumentasi Otomatis: Swagger menciptakan dokumentasi otomatis untuk API berdasarkan spesifikasi yang dibuat. Dokumentasi ini dapat membantu pengguna atau pengembang lain untuk memahami cara menggunakan API dengan benar.
5. Ketersediaan Ekosistem: Swagger memiliki ekosistem yang kaya dengan berbagai alat pendukung dan integrasi dengan kerangka kerja pengembangan perangkat lunak populer.

1.22 Sejarah Swagger



Swagger pertama kali diperkenalkan pada tahun 2011 oleh perusahaan bernama Wordnik. Awalnya, proyek ini disebut "Swagger," tetapi kemudian berkembang dan menjadi lebih dikenal sebagai "OpenAPI Specification." Wordnik memperkenalkan Swagger sebagai proyek sumber terbuka dengan tujuan untuk memfasilitasi dokumentasi dan penggunaan API RESTful. Swagger/OpenAPI Specification adalah standar spesifikasi API terbuka yang memungkinkan pengembang mendefinisikan, mendokumentasikan, dan merinci fitur-fitur API mereka secara konsisten. Sejak itu, proyek ini telah mendapatkan dukungan luas dalam komunitas pengembang perangkat lunak. Pada tahun 2015, SmartBear Software mengakuisisi teknologi Swagger dari Wordnik, dan Swagger berkembang lebih lanjut sebagai proyek open source di bawah naungan Linux Foundation.

Pada tahun 2016, Swagger digabungkan dengan proyek OpenAPI, dan spesifikasi resmi API tersebut disebut sebagai "OpenAPI Specification," sementara alat-alat yang terkait masih sering disebut sebagai "Swagger tools. "Sejak itu, Swagger/OpenAPI Specification telah menjadi salah satu standar industri yang umum digunakan untuk merancang, mendokumentasikan, dan mengimplementasikan API RESTful. Banyak perusahaan besar dan komunitas pengembang mengadopsi Swagger/OpenAPI untuk meningkatkan interoperabilitas dan pemahaman terhadap API mereka.

1.23 Keuntungan Swagger

- Dokumentasi Otomatis: Swagger memberikan dokumentasi otomatis untuk API berdasarkan spesifikasi yang dihasilkan. Ini meningkatkan keterbacaan dan pemahaman terhadap API.
- Antarmuka Pengguna Grafis Interaktif: Swagger menyediakan antarmuka pengguna grafis yang memungkinkan pengembang menjelajahi dan menguji API secara langsung dari browser.

- Spesifikasi Terstruktur: Menggunakan format YAML atau JSON untuk mendefinisikan spesifikasi API membantu menciptakan spesifikasi yang terstruktur dan jelas.
- Generasi Kode Otomatis: Swagger memungkinkan pengembang untuk menghasilkan kode sumber otomatis untuk klien API dalam berbagai bahasa pemrograman, meningkatkan efisiensi pengembangan.
- Konsistensi Pengembangan: Swagger membantu dalam menciptakan konsistensi pengembangan dengan memastikan bahwa tim bekerja dengan pandangan yang seragam terhadap API.
- Pemeliharaan Mudah: Dengan dokumentasi otomatis dan spesifikasi yang jelas, pemeliharaan API menjadi lebih mudah karena perubahan dapat segera dicatat dan dokumentasi diperbarui secara otomatis.
- Integrasi Ekosistem: Swagger memiliki ekosistem yang kaya dengan berbagai alat pendukung dan integrasi dengan kerangka kerja pengembangan populer.
- Peningkatan Kolaborasi Tim: Swagger dapat meningkatkan kolaborasi tim dengan memberikan pandangan yang konsisten dan terdokumentasi tentang API yang sedang dibangun.

1.24 Kekurangan Swagger

- Kompleksitas Awal: Implementasi Swagger mungkin memerlukan waktu dan usaha awal untuk memahami konsep dan mengonfigurasi spesifikasi.
- Keterbatasan Spesifikasi: Beberapa fitur atau kasus penggunaan mungkin sulit atau tidak dapat dijelaskan dengan baik menggunakan spesifikasi Swagger.
- Overhead File Konfigurasi: Dalam beberapa kasus, file konfigurasi Swagger dapat menjadi besar dan memerlukan pemeliharaan tambahan.
- Ketergantungan pada Spesifikasi: Jika spesifikasi API tidak didefinisikan dengan benar atau tidak dijaga pemeliharaannya, dokumentasi otomatis dan generasi kode dapat menjadi tidak akurat.
- Ketergantungan pada Ekosistem Swagger: Ketergantungan pada ekosistem Swagger dapat membuat sulit beralih ke kerangka kerja dokumentasi API lainnya jika diperlukan.
- Kurangnya Kesadaran Pengembang: Meskipun Swagger populer di kalangan pengembang, beberapa tim mungkin kurang sadar tentang manfaatnya atau tidak memprioritaskan dokumentasi API.
- Tidak Cocok untuk Semua Kasus Penggunaan: Swagger mungkin tidak cocok untuk semua kasus penggunaan, terutama untuk proyek-proyek kecil atau sederhana di mana overhead konfigurasi tidak dibutuhkan.

1.25 Penggunakan Swagger untuk API sederhana menggunakan Node.js dan Express

1. Instalasi Dependensi:
   Pertama, instal dependensi yang diperlukan. Jika Anda menggunakan Node.js, Anda dapat menggunakan npm. Buka terminal dan jalankan perintah berikut:

```
npm init -y
npm install express swagger-jsdoc swagger-ui-express
```

2. Buat File Server (app.js atau server.js):

```
const express = require('express');
const swaggerUi = require('swagger-ui-express');
const swaggerJsdoc = require('swagger-jsdoc');

const app = express();
const PORT = 3000;

// Swagger options
const options = {
  definition: {
    openapi: '3.0.0',
    info: {
      title: 'Simple API with Swagger',
      version: '1.0.0',
      description: 'A simple API with Swagger documentation',
    },
  },
  apis: ['./routes/*.js'], // Path to the API routes
};
```

```
const specs = swaggerJsdoc(options);

app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(specs));

// Sample API route
app.get('/api/greet', (req, res) => {
  /**
   * @openapi
   * /api/greet:
   *   get:
   *     summary: Returns a greeting message
   *     responses:
   *       200:
   *         description: A successful response with a greeting message
   *         content:
   *           application/json:
   *             example: { message: 'Hello, Swagger!' }
   */
  res.json({ message: 'Hello, Swagger!' });
});

app.listen(PORT, () => {
  console.log('Server is running on http://localhost:${PORT}');
});
```

3. Jalankan Aplikasi:
   Simpan file di atas dengan nama app.js atau server.js dan jalankan aplikasi dengan perintah:

```
node app.js
```

Aplikasi Anda akan berjalan di http://localhost:3000. Anda dapat mengakses dokumentasi Swagger di http://localhost:3000/api-docs.

Pastikan untuk menyesuaikan spesifikasi Swagger sesuai kebutuhan proyek Anda dan menambahkan endpoint API yang sesuai. Juga, pastikan untuk memahami format YAML atau JSON yang digunakan dalam spesifikasi Swagger.

1.26 Bahasa Pemrograman Yang Menggunakan Swagger

Bahasa pemrograman yang dapat bekerja dengan Swagger/OpenAPI:

1. **JavaScript/Node.js:**
   JavaScript adalah bahasa pemrograman serbaguna yang paling umum digunakan untuk pengembangan web. JavaScript berjalan di sisi klien (client-side) dan dapat disematkan langsung ke dalam HTML untuk menyediakan fungsionalitas dinamis pada halaman web. Biasanya, JavaScript digunakan untuk meningkatkan interaktivitas halaman web dan memberikan pengalaman pengguna yang dinamis.
   Node.js, di sisi lain, adalah lingkungan runtime yang memungkinkan Anda menjalankan JavaScript di sisi server. Sebelum adanya Node.js, JavaScript hanya dapat dieksekusi di browser. Dengan Node.js, Anda dapat menggunakan JavaScript untuk membuat skrip di sisi server, membantu pengembang membangun aplikasi jaringan yang dapat diskalakan dan layanan backend.

2. **Java:**
   Java adalah bahasa pemrograman yang didesain untuk menjadi sederhana, portabel, dan dapat dijalankan di berbagai platform tanpa perlu kompilasi ulang. Diciptakan oleh James Gosling dan timnya di Sun Microsystems, Java pertama kali diperkenalkan pada tahun 1995.

3. **Python**
   Python adalah bahasa pemrograman tingkat tinggi yang didesain untuk kesederhanaan, kejelasan sintaksis, dan keterbacaan kode. Diciptakan oleh Guido van Rossum dan pertama kali diperkenalkan pada tahun 1991, Python telah menjadi salah satu bahasa pemrograman yang sangat populer dan digunakan secara luas di berbagai bidang.

4. **C#: ASP.NET Core**
   C# dan ASP.NET Core adalah dua teknologi yang berhubungan erat dan sering digunakan bersama untuk pengembangan aplikasi web.

5. **Ruby:**
   Ruby adalah bahasa pemrograman dinamis dan berorientasi objek yang pertama kali dikembangkan oleh Yukihiro "Matz" Matsumoto pada awal 1990-an di Jepang. Ruby dirancang untuk mendukung pemrograman yang produktif dan menyenangkan, dengan fokus pada kejelasan sintaksis dan keterbacaan kode.

6. **Go:**
   Go, juga dikenal sebagai Golang, adalah bahasa pemrograman yang dikembangkan oleh Google pada tahun 2007 dan pertama kali dirilis pada tahun 2009. Go dirancang dengan tujuan menyediakan bahasa yang efisien, mudah dipahami, dan cocok untuk pengembangan perangkat lunak yang bersifat bersih, efisien, dan dapat diskalakan

7. **PHP:**
   PHP adalah bahasa pemrograman skrip yang umumnya digunakan untuk pengembangan aplikasi web.

8. **Kotlin dan Scala:**
   Kotlin dan Scala adalah dua bahasa pemrograman yang berjalan di atas Java Virtual Machine (JVM) dan memiliki fokus pada konsep pemrograman berorientasi objek dan fungsional.

9. **Swift dan Objective-C:**
   Swift dan Objective-C adalah dua bahasa pemrograman yang digunakan untuk pengembangan aplikasi di platform Apple, seperti iOS, macOS, watchOS, dan tvOS.

10. **Perl:**
    Pemrograman Perl adalah kegiatan menulis kode menggunakan bahasa pemrograman Perl. Perl adalah bahasa pemrograman yang serbaguna dan berorientasi teks, diciptakan oleh Larry Wall pada tahun 1987.

11. **Rust:**
    Pemrograman Rust adalah kegiatan menulis kode menggunakan bahasa pemrograman Rust. Rust adalah bahasa pemrograman yang dikembangkan oleh Mozilla yang bertujuan untuk menyediakan tingkat kontrol tinggi pada sistem, kinerja tinggi, dan keamanan memori tanpa mengorbankan produktivitas pengembang.

12. **Elixir dan Erlang:**
    Elixir dan Erlang adalah dua bahasa pemrograman yang sering dikaitkan karena Elixir dikembangkan di atas Erlang dan berjalan pada mesin virtual Erlang (BEAM).

13. **Haskell:**
    Haskell adalah bahasa pemrograman fungsional murni yang pertama kali dikembangkan pada awal 1990-an.

1.27 Postman

Postman adalah sebuah platform kolaborasi yang memungkinkan pengembang untuk merancang, menguji, dan memahami API (Application Programming Interface). Berikut adalah penjelasan mengenai Postman, termasuk kegunaan, manfaat, kelebihan, kekurangan, dan cara penggunaannya:

1.28 Kegunaan Postman:

1. Desain dan Pembuatan API:
   Postman memudahkan pengembang dalam merancang dan membuat API dengan menyediakan antarmuka grafis yang intuitif.
2. Pengujian API:
   Fasilitas pengujian API yang lengkap memungkinkan pengguna untuk mengirim permintaan HTTP dan melihat tanggapan secara langsung, memudahkan pengujian dan debugging.
3. Kolaborasi Tim:
   Memungkinkan kolaborasi tim dengan menyediakan lingkungan berbagi dan dokumentasi yang mudah dipahami.
4. Automatisasi Pengujian:
   Postman mendukung otomatisasi pengujian API, memungkinkan pengguna untuk membuat skenario pengujian dan mengintegrasikan mereka dalam alur kerja CI/CD (Continuous Integration/Continuous Deployment).
5. Monitoring API:
   Menyediakan fitur monitoring untuk melacak performa API dan mendeteksi potensi masalah.
6. Dokumentasi API:
   Memungkinkan pembuatan dokumentasi API yang terperinci dan mudah dimengerti.

1.29 Manfaat Postman:

1. Efisiensi Pengembangan:
   Membantu pengembang dalam membuat, menguji, dan mengelola API dengan cepat dan efisien.
2. Ketepatan Pengujian:
   Memudahkan pengujian API dengan memberikan kontrol penuh terhadap permintaan dan respons HTTP.
3. Kolaborasi yang Mudah:
   Tim dapat bekerja sama dengan mudah menggunakan fitur berbagi dan dokumentasi terintegrasi.
4. Dokumentasi yang Baik:
   Membantu dalam membuat dokumentasi API yang baik dan terorganisir.

1.30 Kelebihan Postman:

1. Antarmuka Pengguna yang Ramah:
   Antarmuka pengguna yang intuitif dan mudah digunakan.
2. Pengujian Otomatis yang Kuat:
   Kemampuan untuk membuat skenario pengujian otomatis dan mengintegrasikannya dengan alur kerja CI/CD.
3. Kolaborasi Tim:
   Dukungan untuk kolaborasi tim dengan fitur berbagi dan kontrol akses.
4. Dokumentasi Terintegrasi:
   Kemampuan untuk membuat dan mengelola dokumentasi API secara langsung.

1.31 Kekurangan Postman:

1. Ketergantungan pada GUI:
   Beberapa pengembang mungkin lebih suka menggunakan alat berbasis teks dan mungkin merasa terbatas oleh antarmuka grafis Postman.
2. Beban Kerja Berat:
   Pada proyek besar, Postman dapat memiliki beban kerja yang tinggi jika tidak dikelola dengan baik.

1.32 Cara Penggunaan Postman:

1. Instalasi:
   Unduh dan instal aplikasi Postman dari situs web resmi.
2. Membuat Permintaan:
   Buat permintaan HTTP GET, POST, PUT, atau DELETE dengan mengisi detail permintaan seperti URL, header, dan payload.

3. Pengujian dan Debugging:
   Gunakan fitur pengujian untuk menguji dan mendebung API, melihat respons, dan mengevaluasi hasilnya.
4. Kolaborasi Tim:
   Gunakan fitur berbagi untuk mengizinkan anggota tim mengakses dan berkontribusi pada koleksi dan skrip pengujian.
5. Dokumentasi:
   Buat dokumentasi API menggunakan fitur dokumentasi terintegrasi.

1.33 Paseto

Paseto pertama kali diperkenalkan oleh Scott Arciszewski pada tahun 2018 sebagai alternatif yang lebih aman dan sederhana untuk JSON Web Tokens (JWT). Scott Arciszewski adalah seorang pengembang keamanan yang bekerja untuk Paragon Initiative Enterprises dan memiliki latar belakang dalam keamanan aplikasi dan kriptografi.

Seiring dengan kekhawatiran keamanan yang muncul terkait dengan JWT, terutama terkait dengan implementasi RSA dan beberapa serangan yang mempengaruhi keamanannya, Paseto diusulkan sebagai solusi untuk mengatasi kelemahan-kelemahan tersebut. Nama "Paseto" sendiri berasal dari bahasa Esperanto, yang berarti "passing token," mencerminkan tujuan desainnya untuk memberikan token keamanan yang aman dan dapat dipertukarkan.

Paseto dirancang sebagai format token keamanan yang dapat diimplementasikan di berbagai bahasa pemrograman. Sebagai format yang bersifat independen dari bahasa pemrograman, Paseto dapat dihasilkan dan diverifikasi oleh berbagai implementasi di sejumlah bahasa.

1.34 Bahasa Pemrograman Yang Bisa Menggunakan Paseto

- PHP:
  Implementasi referensi Paseto dibuat dalam PHP oleh Scott Arciszewski, pencipta Paseto. Implementasi PHP ini disebut "paragonie/paseto."
- JavaScript/Node.js:
  Ada beberapa implementasi Paseto untuk JavaScript dan Node.js yang dapat digunakan dalam proyek-proyek berbasis web. Contohnya adalah "sambego/paseto" dan "lukechilds/paseto.js."
- Python:
  Terdapat implementasi Paseto untuk Python yang disebut "pymacaroons-paseto."
- Go:
  Beberapa proyek Go juga menyediakan dukungan untuk Paseto, seperti "o1egl/paseto" dan "sethvargo/go-paseto."
- Java:
  Terdapat implementasi Paseto untuk Java dengan proyek "gokberkisik/paseto."
- Ruby:
  Implementasi Paseto untuk Ruby dapat ditemukan dalam proyek "ruby_paseto."
- C#:
  Ada beberapa implementasi Paseto untuk C#, termasuk "brandonroberts/Paseto" dan "mbdavid/Paseto."
- Rust:
  Terdapat proyek implementasi Paseto untuk Rust yang disebut "quininer/paseto-rs."

1.35 Membuat Paseto Menggunakan PHP

Menggunakan Paseto melibatkan beberapa langkah umum, tergantung pada peran token tersebut dalam aplikasi Anda. Berikut adalah panduan umum untuk membuat dan memverifikasi token Paseto.

1. Instalasi Pustaka:
   Pastikan pustaka Paseto yang sesuai sudah diinstal. Misalnya, dalam PHP, Anda dapat menggunakan pustaka "paragonie/paseto" dengan composer:

```
composer require paragonie/paseto
```

2. Membuat Token Paseto (Local Mode):

```php
use ParagonIE\Paseto\Builder;
use ParagonIE\Paseto\Keys\SymmetricKey;

$key = SymmetricKey::generateNewKey();
$builder = Builder::local($key)
    ->set('data', 'Hello, Paseto!')
    ->setExpiration(new DateTime('+1 day'));

$token = $builder->toString();
```

3. Membuat Token Paseto (Public Mode):

```php
use ParagonIE\Paseto\Builder;
use ParagonIE\Paseto\Keys\AsymmetricPublicKey;
use ParagonIE\Paseto\Keys\AsymmetricSecretKey;

$privateKey = AsymmetricSecretKey::generateNewKey();
$publicKey = $privateKey->getPublicKey();

$builder = Builder::public($privateKey)
    ->set('data', 'Hello, Paseto!')
    ->setExpiration(new DateTime('+1 day'));

$token = $builder->toString();
```

1.36 Memverifikasi Token Paseto Menggunakan PHP

1. Memverifikasi Token Paseto (Local Mode):

```php
use ParagonIE\Paseto\Parser;

$parser = (new Parser())
    ->setKey($key);

$parsedToken = $parser->parse($token);
```

2. Memverifikasi Token Paseto (Public Mode):

```php
use ParagonIE\Paseto\Parser;

$parser = (new Parser())
    ->setKey($publicKey);

$parsedToken = $parser->parse($token);
```

3. Mengambil Data dari Token:

```php
$data = $parsedToken->get('data');
```

1.37 Apa Saja Keungtungan Menggunakan Paseto ?

Menggunakan Paseto memiliki beberapa keuntungan dibandingkan dengan menggunakan JSON Web Tokens (JWT) atau beberapa format token keamanan lainnya.

- Keamanan yang Lebih Baik:
  Paseto dirancang dengan fokus pada keamanan, dan sejumlah desainnya membantu mengatasi beberapa masalah keamanan yang ditemui pada implementasi JWT. Misalnya, Paseto tidak menggunakan skema yang rentan terhadap serangan "Padding Oracle" yang dapat mempengaruhi JWT.
- Tidak Bergantung pada Implementasi RSA:
  Paseto tidak bergantung pada implementasi RSA atau algoritma kriptografi yang seringkali rumit dan dapat menimbulkan risiko keamanan jika tidak diimplementasikan dengan benar. Ini dapat membantu mengurangi kompleksitas dan meningkatkan keamanan.
- Tidak Mengandung Informasi Tersembunyi:
  Paseto dirancang agar tidak menyimpan informasi tersembunyi dalam token yang dapat menyebabkan masalah keamanan. Semua informasi dapat dibaca langsung dari token tanpa perlu melakukan dekripsi.
- Mendukung Versi (Versioning):
  Paseto mendukung versi, memungkinkan evolusi format tanpa menghancurkan ekosistem pengguna. Ini memudahkan perubahan dan pembaruan tanpa memerlukan migrasi besar-besaran.
- Mendukung Beberapa Scheme Kriptografi:
  Paseto mendukung beberapa skema kriptografi, termasuk skema lokal (kriptografi simetris) dan skema publik (kriptografi asimetris). Ini memberikan fleksibilitas dalam memilih metode kriptografi yang sesuai dengan kebutuhan aplikasi.
- Bekerja Dengan String Biner:
  Paseto beroperasi langsung pada string biner tanpa memerlukan operasi encoding dan decoding JSON seperti yang dibutuhkan oleh JWT. Hal ini dapat mempercepat proses dan mengurangi overhead.
- Open Standard:
  Paseto adalah standar terbuka dan memiliki spesifikasi yang terdefinisi dengan baik, memungkinkan pengembang untuk mengimplementasikannya dengan konsisten di berbagai bahasa pemrograman.
- Dukungan Implementasi di Berbagai Bahasa:
  Terdapat implementasi Paseto untuk berbagai bahasa pemrograman, sehingga dapat diintegrasikan dengan mudah dalam berbagai ekosistem.

1.38 Manfaat Menggunakan Paseto

- Keamanan yang Lebih Tinggi:
  Paseto dirancang dengan fokus pada keamanan, dan banyak aspek desainnya diarahkan untuk mengatasi masalah keamanan yang mungkin muncul pada implementasi JWT. Ini termasuk menghindari skema yang rentan terhadap serangan tertentu, seperti "Padding Oracle."
- Tidak Bergantung pada Implementasi RSA:

Menggunakan Paseto menghilangkan ketergantungan pada implementasi RSA atau algoritma kriptografi yang mungkin rumit dan rentan terhadap berbagai jenis serangan. Ini dapat memudahkan implementasi dan mengurangi risiko keamanan.

- ❖ Pemahaman yang Lebih Mudah:
  Desain Paseto yang sederhana dan jelas membuatnya lebih mudah dipahami daripada beberapa format token keamanan lainnya. Ini dapat mempermudah pengembangan, pemeliharaan, dan pemecahan masalah.
- ❖ Tidak Mengandung Informasi Tersembunyi:
  Paseto tidak menyimpan informasi tersembunyi dalam token, yang dapat menjadi sumber masalah keamanan. Semua informasi dapat dibaca langsung dari token tanpa memerlukan dekripsi.
- ❖ Versi (Versioning) yang Didukung:
  Paseto mendukung versi, memungkinkan format untuk berkembang seiring waktu tanpa mempengaruhi aplikasi yang masih menggunakan versi sebelumnya. Ini memfasilitasi pembaruan dan perbaikan tanpa mengganggu ekosistem pengguna.
- ❖ Fleksibilitas Kriptografi:
  Paseto mendukung beberapa skema kriptografi, termasuk kriptografi simetris dan asimetris. Ini memberikan fleksibilitas kepada pengembang untuk memilih metode kriptografi yang paling sesuai dengan kebutuhan keamanan aplikasi.
- ❖ Tidak Memerlukan Operasi Terhadap String JSON:
  Paseto beroperasi langsung pada string biner, menghilangkan kebutuhan untuk operasi encoding dan decoding JSON seperti yang terjadi pada JWT. Hal ini dapat meningkatkan efisiensi dan mengurangi overhead.
- ❖ Open Standard:
  Paseto adalah standar terbuka dengan spesifikasi yang terdefinisi dengan baik, memastikan konsistensi dan interoperabilitas di berbagai lingkungan pengembangan.

1.39 FrameWork Laravel

Laravel adalah salah satu framework pengembangan aplikasi web berbasis PHP yang sangat populer dan kuat. Dikembangkan oleh Taylor Otwell, Laravel dirancang dengan konsep "elegan, ekspresif, dan menyenangkan" untuk mempermudah proses pengembangan. Berikut adalah beberapa poin penting tentang Laravel:

Fitur Utama Laravel:

- Eloquent ORM:
  Laravel menyertakan ORM (Object-Relational Mapping) yang disebut Eloquent. Ini menyederhanakan koneksi dan manipulasi basis data dengan menggunakan model dan relasi antarmuka objek.
- Blade Templating Engine:
  Blade adalah mesin templating yang ringan dan ekspresif yang memungkinkan pengembang untuk menulis tampilan HTML dengan sintaks yang bersih dan jelas.
- Artisan Console:
  Artisan adalah CLI (Command Line Interface) bawaan untuk Laravel yang menyediakan sejumlah perintah bawaan untuk membantu dalam pengembangan, seperti membuat model, kontroler, migrasi basis data, dan banyak lagi.
- Migrations dan Seeders:
  Laravel menyediakan migrasi untuk memfasilitasi manajemen basis data dan struktur tabel. Seeder digunakan untuk mengisi basis data dengan data dummy atau awal.

- Middleware:
  Middleware memungkinkan Anda memfilter permintaan HTTP yang masuk ke aplikasi sebelum mencapai rute atau kontroler. Ini membantu dalam otentikasi, autorisasi, dan pemrosesan permintaan lainnya.
- Routing:
  Laravel menyediakan sistem routing yang kuat dan sederhana. Anda dapat mendefinisikan rute menggunakan sintaks yang mudah dimengerti.
- Authentication dan Authorization:
  Laravel menyediakan otentikasi pengguna dan kontrol akses yang mudah diimplementasikan dengan beberapa perintah Artisan.
- Testing:
  Laravel mendukung pengujian otomatis dengan menggunakan PHPUnit. Tes dapat dilakukan pada berbagai tingkatan, termasuk pengujian unit, pengujian fungsional, dan pengujian penerimaan.
- Paket Composer dan Dependency Injection:
  Laravel menggunakan Composer untuk mengelola dependensi dan paket-paket eksternal. Framework ini juga mendukung Dependency Injection untuk manajemen ketergantungan yang efisien.
- Redis dan Cache:
  Laravel menyertakan dukungan untuk Redis, sistem basis data berkinerja tinggi yang digunakan untuk menyimpan cache dan sesi.

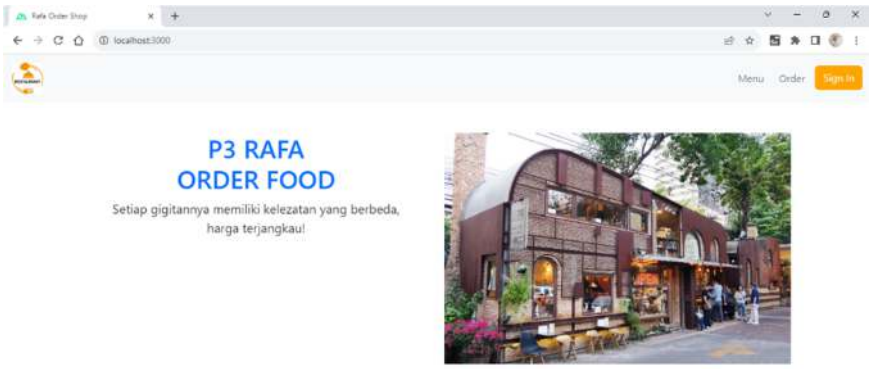
Keuntungan Menggunakan Laravel:

a. Produktivitas Tinggi:
   Laravel dirancang untuk meningkatkan produktivitas pengembang dengan menyediakan fitur-fitur yang dapat digunakan secara cepat dan mudah.
b. Kode yang Jelas dan Bersih:
   Laravel mendorong pengembangan dengan menggunakan kode yang bersih, terstruktur, dan mudah dimengerti.
c. Komunitas yang Aktif:
   Laravel memiliki komunitas yang besar dan aktif, dengan banyak sumber daya, tutorial, dan paket tambahan yang tersedia.
d. Dokumentasi yang Baik:
   Laravel memiliki dokumentasi yang sangat baik yang menjelaskan setiap aspek framework dengan rinci.
e. Skalabilitas
   Laravel dapat digunakan untuk membangun aplikasi kecil hingga besar dan skalabel. Dengan fitur-fitur seperti Eloquent dan Redis, ia dapat menangani beban kerja yang signifikan.
f. Keamanan:
   Laravel menyertakan berbagai fitur keamanan bawaan seperti perlindungan CSRF, hashing password, dan perlindungan terhadap SQL injection.

Laravel adalah pilihan populer di kalangan pengembang web karena kombinasi kekuatan dan kemudahan penggunaannya. Pembaruan reguler dan inovasi terus-menerus membuatnya relevan dan diminati dalam komunitas pengembangan web.

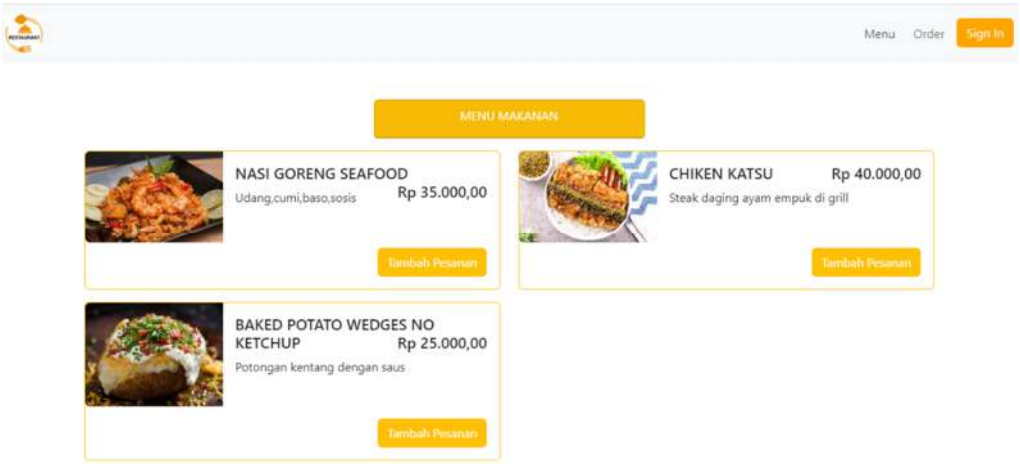
CHAPTER 2
SISTEM PELAYANAN PEMESANAN MAKANAN DENGAN WAITING LINE METHOD

2.1 Pelayanan Dengan Waiting Line Method



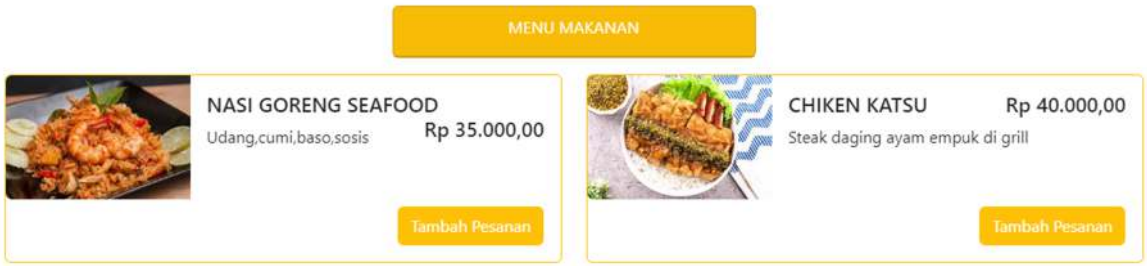
Restoran adalah bisnis atau tempat yang menyediakan makanan dan minuman untuk dijual kepada pelanggan. Restoran dapat memiliki berbagai jenis masakan, konsep, dan tingkat pelayanan, mulai dari warung makan sederhana hingga restoran mewah dengan masakan gourmet.
Berikut beberapa karakteristik umum dari restoran:

- Menu: Restoran memiliki menu yang berisi beragam hidangan dan minuman yang ditawarkan kepada pelanggan. Menu bisa mencakup berbagai jenis makanan, seperti hidangan utama, hidangan pembuka, hidangan penutup, minuman jus atau teh, dan lainnya.



- Tempat Makan: Restoran biasanya menyediakan area tempat makan yang nyaman bagi pelanggan. Ini bisa berupa meja dan kursi di dalam ruangan atau di luar ruangan, bar, atau bahkan meja pemesanan untuk makanan yang akan dibawa pulang.
- Pelayanan Pelanggan: Staf restoran, seperti pelayan, pelayan bar, koki, dan staf dapur, bertanggung jawab melayani pelanggan. Pelayanan pelanggan mencakup menerima pesanan, membawa makanan, menjawab pertanyaan pelanggan, dan memberikan pengalaman yang baik.

- Konsep dan Gaya: Restoran dapat memiliki berbagai konsep atau gaya, seperti restoran cepat saji, restoran keluarga, restoran fine dining, restoran tematik, restoran etnis, dan banyak lagi. Konsep ini mencerminkan masakan, dekorasi, harga, dan atmosfer restoran.
- Hidangan Khusus: Beberapa restoran menawarkan hidangan khusus atau spesialisasi tertentu yang membuat mereka dikenal. Ini bisa berupa hidangan signature, masakan etnik, atau keahlian tertentu dalam memasak.
- Layanan Minuman: Banyak restoran juga menyediakan layanan minuman, termasuk berbagai jenis minuman alkohol dan non-alkohol yang dapat disajikan kepada pelanggan.
- Harga dan Pembayaran: Restoran memiliki harga berbeda untuk makanan dan minuman mereka, dan biasanya menerima pembayaran dalam bentuk uang tunai, kartu kredit, atau metode pembayaran lainnya.
- Lokasi dan Dekorasi: Lokasi restoran dan dekorasi interior eksterior dapat sangat bervariasi. Restoran dapat ditemukan di pusat kota, di tepi pantai, di pinggiran kota, atau di daerah pedesaan. Dekorasi interior bisa sederhana atau mewah, sesuai dengan konsep restoran.
- Kebersihan dan Keselamatan: Restoran harus mematuhi standar kebersihan dan keselamatan makanan untuk menjaga kualitas dan keamanan makanan yang disajikan kepada pelanggan.



Pelayanan pemesanan makanan dengan metode waiting line atau antrian adalah sistem yang digunakan oleh restoran atau penyedia layanan makanan untuk mengatur dan mengelola proses pemesanan makanan dan pengiriman pesanan kepada pelanggan dengan menggunakan konsep antrian. Sistem ini dirancang untuk mengurangi waktu tunggu pelanggan, meningkatkan efisiensi operasional, dan memberikan pengalaman yang lebih baik kepada pelanggan. Restoran yang menggunakan sistem ini biasanya memperhatikan faktor-faktor seperti kapasitas restoran, waktu puncak kunjungan, dan ketersediaan staf untuk mengatur antrian dan memastikan pemesanan dan pengiriman makanan berjalan dengan lancar. Hal ini sangat penting dalam industri makanan dan minuman yang kompetitif dan berorientasi pada layanan pelanggan.

2.2 Manfaat Dari Sistem Pelayanan Pemesanan Makanan Dengan Waiting Line Method

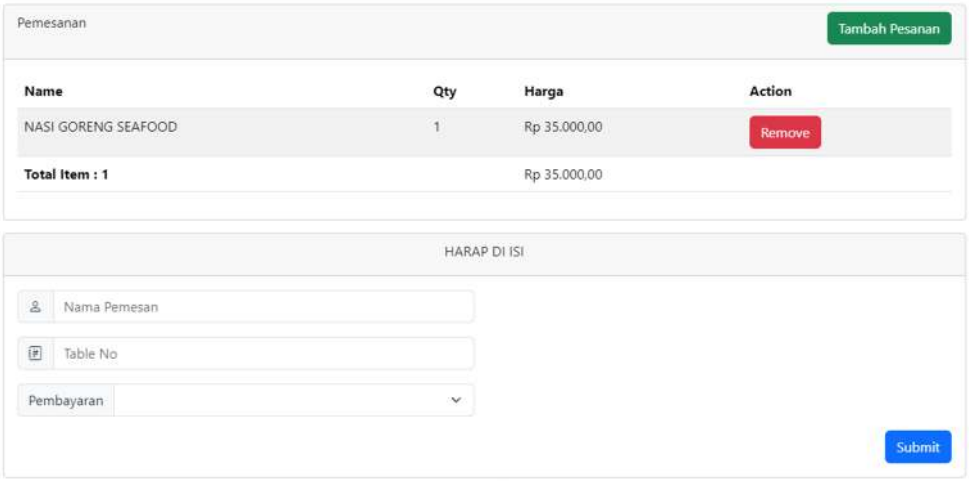
- Pengurangan Waktu Tunggu: Sistem ini membantu mengurangi waktu tunggu pelanggan karena pesanan dikelola dalam antrian dan diproses secara berurutan.
- Efisiensi Operasional: Restoran dapat mengatur staf dan sumber daya dengan lebih efisien, menghindari kelebihan kerja atau kekurangan staf saat jam sibuk.
- Peningkatan Kualitas Layanan: Dengan mengatur antrian dan proses pemesanan, restoran dapat memberikan layanan yang lebih baik kepada pelanggan dan mencegah kesalahan dalam pemesanan.
- Pemantauan Kinerja: Sistem ini memungkinkan restoran untuk memantau kinerja staf dan waktu pelayanan, sehingga mereka dapat melakukan perbaikan jika diperlukan.
- Analisis Data: Data dari sistem ini dapat digunakan untuk menganalisis tren pemesanan dan preferensi pelanggan, membantu restoran dalam perencanaan menu dan strategi pemasaran.

2.3 Komponen Utama Sistem Pelayanan Pemesanan Makanan

1. Pelanggan: Pelanggan adalah individu atau kelompok yang datang ke restoran atau tempat makan untuk memesan dan menikmati makanan. Pelanggan adalah elemen utama dalam sistem ini.

2. Antrian (Queue): Antrian adalah tempat atau mekanisme yang digunakan untuk mengatur giliran pelanggan. Antrian ini bisa berupa barisan fisik atau digital, seperti aplikasi pemesanan makanan.
3. Staf Layanan: Staf restoran atau penyedia makanan adalah individu yang bertanggung jawab menerima pesanan, memproses makanan, dan menyajikan pesanan kepada pelanggan.
4. Proses Pemesanan: Proses di mana pelanggan memesan makanan mereka kepada staf restoran. Pesanan bisa berupa memesan menu langsung di lokasi atau melalui aplikasi pemesanan makanan.
5. Pemrosesan Pesanan: Setelah menerima pesanan, staf restoran memprosesnya dengan memasak atau menyiapkan makanan sesuai pesanan.
6. Pengiriman Pesanan: Pesanan yang telah siap kemudian disajikan kepada pelanggan. Ini bisa melibatkan meja pemesanan di restoran atau pengiriman ke alamat pelanggan jika layanan pengiriman tersedia.

2.4 Keuntungan Sistem Pelayanan Pemesanan Makanan Dengan Waiting Line Method



a. Pengurangan Waktu Tunggu Pelanggan: Salah satu manfaat utama dari sistem ini adalah pengurangan waktu tunggu pelanggan. Dengan mengatur antrian dan memproses pesanan secara berurutan, restoran dapat mengurangi waktu yang dihabiskan pelanggan untuk menunggu pesanannya. Hal ini meningkatkan kepuasan pelanggan karena mereka tidak perlu menunggu terlalu lama.
b. Efisiensi Operasional: Sistem antrian membantu restoran mengelola sumber daya dan staf dengan lebih efisien. Mereka dapat merencanakan jumlah staf yang dibutuhkan pada waktu tertentu berdasarkan jumlah pelanggan dalam antrian. Hal ini membantu menghindari kelebihan kerja atau kekurangan staf.
c. Peningkatan Kualitas Layanan: Dengan mengorganisir pemesanan dan pengiriman makanan, restoran dapat memberikan layanan yang lebih baik kepada pelanggan. Pesanan dapat lebih tepat dan akurat, dan pelanggan lebih puas dengan pengalaman mereka.
d. Pemantauan Kinerja: Sistem ini memungkinkan manajemen restoran untuk memantau kinerja staf dan waktu pelayanan secara lebih efektif. Mereka dapat mengidentifikasi area yang memerlukan perbaikan dan mengambil langkah-langkah untuk meningkatkan efisiensi operasional.
e. Analisis Data: Data yang dikumpulkan dari sistem ini dapat digunakan untuk menganalisis tren pemesanan dan preferensi pelanggan. Informasi ini dapat membantu restoran dalam mengadaptasi menu mereka, merencanakan promosi, dan meningkatkan strategi pemasaran.
f. Keseragaman Layanan: Dengan mengatur proses dengan antrian, restoran dapat mencapai tingkat keseragaman layanan yang lebih tinggi. Hal ini memastikan bahwa setiap pelanggan menerima pengalaman yang konsisten dan baik.
g. Manajemen Kapasitas: Restoran dapat mengelola kapasitas mereka lebih baik, terutama saat jam sibuk. Mereka dapat mencegah situasi di mana restoran penuh sesak atau terlalu sepi.

h. Kemampuan Perencanaan: Dengan sistem ini, restoran dapat merencanakan lebih baik untuk acara atau promosi khusus, serta memproyeksikan permintaan pelanggan di masa depan.
i. Peningkatan Efisiensi Dapur: Bagi dapur restoran, sistem antrian memungkinkan pengelolaan pesanan yang lebih baik. Ini membantu meminimalkan kesalahan dalam pemesanan dan meningkatkan efisiensi dapur.

2.5 Cara Membuat Sistem Pelayanan Pemesanan Makanan Dengan Waiting Line Method

- Analisis Kebutuhan:
  Identifikasi jenis restoran atau tempat makan yang Anda kelola, termasuk kapasitasnya, jenis menu, dan volume pelanggan yang diharapkan. Evaluasi apakah sistem waiting line cocok untuk restoran Anda berdasarkan karakteristik tersebut.
- Pemilihan Sistem Antrian:
  Pilih jenis sistem antrian yang akan Anda gunakan. Ini bisa berupa antrian fisik dengan nomor urut atau aplikasi pemesanan makanan yang memungkinkan pelanggan memesan dari ponsel mereka.
- Pelatihan Staf:
  Latih staf restoran Anda untuk menggunakan sistem antrian dengan benar, baik itu staf pemesanan, koki, atau pelayan. Mereka harus memahami bagaimana mengatur antrian dan mengelola pesanan dengan baik.
- Pendaftaran Pelanggan:
  Berikan cara bagi pelanggan untuk mendaftar dalam antrian. Ini bisa melalui sistem pengambilan nomor antrian fisik atau dengan meminta mereka mengunduh aplikasi pemesanan makanan jika Anda menggunakan solusi digital.
- Pemesanan Makanan:
  Selain pendaftaran dalam antrian, pastikan pelanggan memiliki cara untuk memesan makanan, seperti melalui staf restoran atau melalui aplikasi pemesanan makanan.
- Antrian Pemrosesan Pesanan:
  Setelah pesanan diterima, pesanan dimasukkan ke dalam antrian pemrosesan. Pastikan staf tahu cara mengorganisir pesanan sesuai urutan waktu atau jenis makanan yang dipesan.
- Pemrosesan Pesanan:
  Staf restoran memasak atau menyiapkan makanan sesuai pesanan. Pastikan kualitas dan akurasi pesanan dijaga dengan baik.
- Pengiriman Pesanan:
  Setelah pesanan selesai, makanan harus disajikan kepada pelanggan sesuai urutan dalam antrian. Pastikan pelanggan mendapatkan pesanannya dengan benar.
- Monitoring dan Perbaikan:
  Pantau kinerja sistem secara terus-menerus. Amati waktu tunggu pelanggan, tingkat kepuasan pelanggan, dan efisiensi operasional.
- Terapkan perbaikan atau peningkatan jika ada masalah yang muncul, seperti pengelolaan antrian yang lebih baik, pelatihan staf tambahan, atau peningkatan dalam penggunaan teknologi.
- Analisis Data: Gunakan data yang diperoleh dari sistem antrian untuk menganalisis tren pemesanan dan perilaku pelanggan. Ini dapat membantu Anda merencanakan perbaikan lebih lanjut dan mengoptimalkan operasional restoran.
- Penyuluhan Pelanggan:
  Berikan informasi kepada pelanggan tentang sistem antrian Anda, baik melalui papan informasi di restoran atau melalui aplikasi pemesanan makanan. Pastikan mereka tahu cara menggunakan sistem tersebut dengan benar.
- Pengembangan Solusi Digital (Opsional):

Jika memungkinkan, Anda dapat mengembangkan atau menggunakan solusi digital, seperti aplikasi pemesanan makanan, untuk memudahkan pelanggan dalam memesan makanan dan mengelola antrian secara lebih efisien.

2.6 Faktor Pendukung Sistem Pelayanan Pemesanan Makanan Dengan Waiting Line Method

1. Teknologi yang Tepat: Penggunaan teknologi yang tepat adalah kunci dalam mengelola sistem pemesanan dengan waiting line method. Hal ini termasuk aplikasi pemesanan makanan yang mudah digunakan, sistem manajemen antrian yang canggih, dan perangkat keras yang dapat memantau pesanan dan antrian dengan efisien.
2. Pelatihan Staf: Staf yang terlatih dengan baik adalah aset penting dalam sistem ini. Mereka harus memahami bagaimana mengatur antrian, memproses pesanan, dan memberikan pelayanan pelanggan yang berkualitas.
3. Analisis Data: Kemampuan untuk mengumpulkan dan menganalisis data dari sistem antrian dapat membantu dalam mengidentifikasi tren, kebutuhan pelanggan, dan area di mana perbaikan diperlukan.
4. Manajemen Operasional yang Baik: Manajemen yang baik dalam mengelola kapasitas, staf, dan sumber daya adalah kunci dalam menjaga sistem pemesanan berjalan dengan lancar. Manajemen harus mampu merencanakan dan menyesuaikan operasi sesuai dengan permintaan pelanggan.
5. Pengaturan Fisik yang Sesuai: Restoran harus memiliki pengaturan fisik yang mendukung sistem antrian, seperti area antrian yang nyaman, ruang dapur yang efisien, dan ruang tempat makan yang sesuai.
6. Komunikasi yang Baik dengan Pelanggan: Memberikan informasi yang jelas dan komunikasi yang baik kepada pelanggan tentang sistem antrian dan prosedur pesanan membantu menghindari kebingungan dan frustrasi pelanggan.
7. Keamanan Data dan Privasi Pelanggan: Jika Anda menggunakan solusi digital, pastikan bahwa data pelanggan aman dan bahwa privasi pelanggan dijaga dengan baik.
8. Kapasitas dan Infrastruktur yang Memadai: Pastikan bahwa restoran memiliki kapasitas yang cukup untuk mengakomodasi jumlah pelanggan yang diharapkan dan infrastruktur yang mendukung operasi yang efisien.
9. Edukasi Pelanggan: Pelanggan mungkin memerlukan edukasi tentang cara menggunakan sistem antrian, khususnya jika Anda menggunakan solusi digital. Pastikan mereka memahami cara memesan makanan dan memahami urutan prosesnya.
10. Kualitas Makanan yang Konsisten: Memastikan bahwa kualitas makanan yang disajikan konsisten adalah faktor kunci dalam memuaskan pelanggan. Pemesanan dengan waiting line method harus mendukung pengiriman makanan berkualitas tinggi dalam waktu yang wajar.
11. Ketersediaan Layanan Pelanggan: Memiliki staf atau layanan pelanggan yang dapat membantu pelanggan yang memiliki pertanyaan atau masalah dengan pesanan mereka.
12. Adaptasi Terhadap Perubahan: Sistem ini harus dapat beradaptasi dengan perubahan dalam permintaan pelanggan, tren makanan, atau perubahan dalam operasional restoran.

2.7 Konsep Penting Dalam Penerapan Sistem Waiting Line Method

a. Pemahaman Proses:

Pahami dengan baik seluruh proses dari pemesanan hingga pengiriman makanan. Ini melibatkan identifikasi titik-titik antrian potensial dalam operasi restoran Anda.

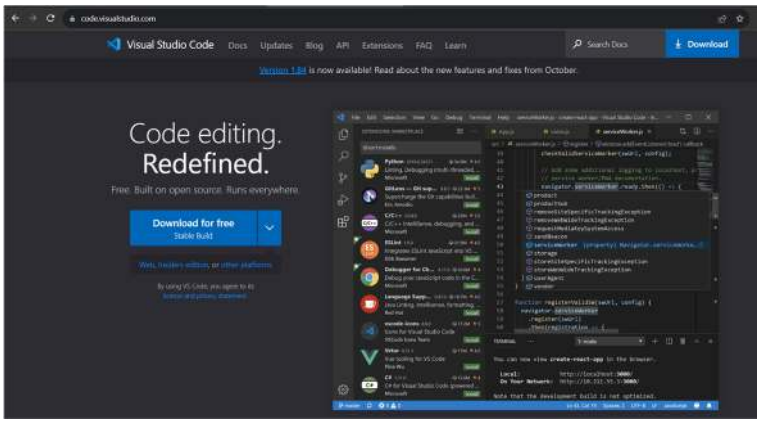
b. Desain Antrian yang Efisien:
   Desain sistem antrian yang efisien dengan mempertimbangkan kapasitas restoran, jumlah staf yang tersedia, dan pola lalu lintas pelanggan. Pastikan bahwa antrian mudah dipahami dan diakses oleh pelanggan.
c. Penggunaan Teknologi:
   Pertimbangkan penggunaan teknologi, seperti aplikasi pemesanan makanan atau perangkat lunak manajemen antrian, untuk mengelola sistem waiting line dengan lebih efisien dan memberikan pelanggan opsi pemesanan digital.
d. Pelatihan Staf:
   Berikan pelatihan kepada staf restoran tentang cara mengelola antrian, prosedur pemesanan, dan penggunaan teknologi jika diperlukan.
e. Komunikasi dengan Pelanggan:
   Komunikasikan dengan jelas kepada pelanggan tentang bagaimana sistem antrian bekerja, bagaimana mereka dapat memesan makanan, dan apa yang bisa mereka harapkan dalam hal waktu tunggu.
f. Manajemen Kapasitas:
   Manajemen kapasitas adalah kunci. Pastikan bahwa Anda dapat mengakomodasi jumlah pelanggan yang diharapkan tanpa membuat restoran menjadi terlalu penuh sesak.
g. Pengukuran Kinerja:
   Pantau kinerja sistem waiting line secara terus-menerus. Ukur waktu tunggu rata-rata, kepuasan pelanggan, dan efisiensi operasional.
h. Fleksibilitas dan Adaptasi:
   Jadilah fleksibel dan siap beradaptasi dengan perubahan dalam permintaan pelanggan atau operasional. Misalnya, Anda mungkin perlu menyesuaikan jumlah staf saat jam sibuk.
i. Perbaikan Berkelanjutan:
   Selalu berusaha untuk memperbaiki sistem waiting line. Gunakan data yang diperoleh untuk mengidentifikasi masalah dan kesempatan perbaikan, dan terapkan perubahan yang diperlukan.
j. Peningkatan Layanan Pelanggan:
   Manfaatkan sistem waiting line untuk meningkatkan kualitas layanan pelanggan. Pastikan pesanan disajikan dengan akurat dan dalam kondisi terbaik.
k. Penghargaan untuk Pelanggan Tetap:
   Pertimbangkan untuk memberikan penghargaan atau program loyalitas kepada pelanggan tetap yang sering menggunakan sistem waiting line Anda.
l. Keamanan dan Privasi Data Pelanggan (Jika Menggunakan Solusi Digital):
   Jika Anda menggunakan solusi digital, pastikan data pelanggan aman dan privasi pelanggan dijaga dengan baik.

CHAPTER 3
TATACARA INSTALASI APLIKASI KEBUTUHAN PENDUKUNG DALAM PEMBUATAN WEBSITE PEMESANAN MAKANAN

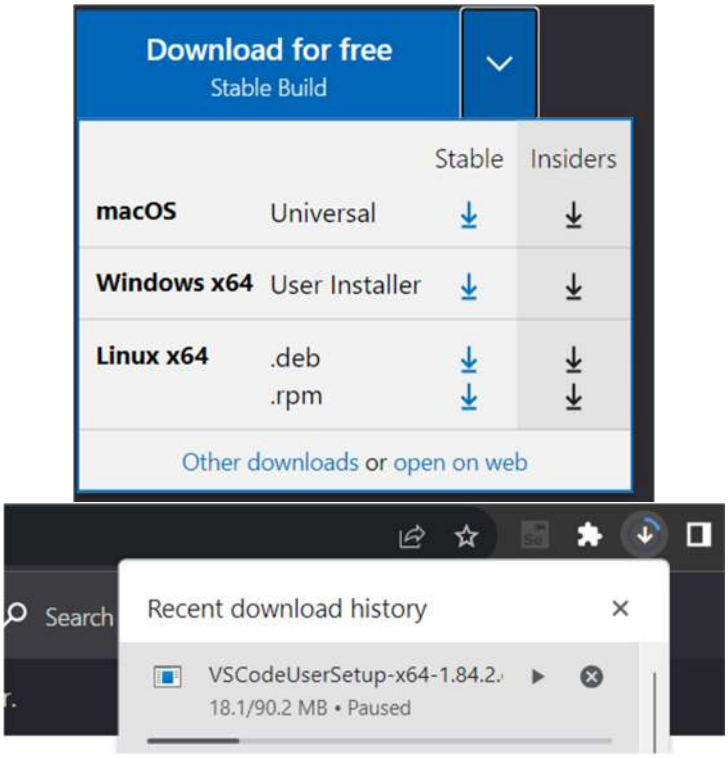
3.1 Visual Studio Code

Ikuti langkah-langkah berikut untuk menginstal Visual Studio Code:

1. Di browser web, buka https://code.visualstudio.com/



2. Unduh versi untuk sistem operasi Anda, mendukung Windows, Linux, dan macOS.

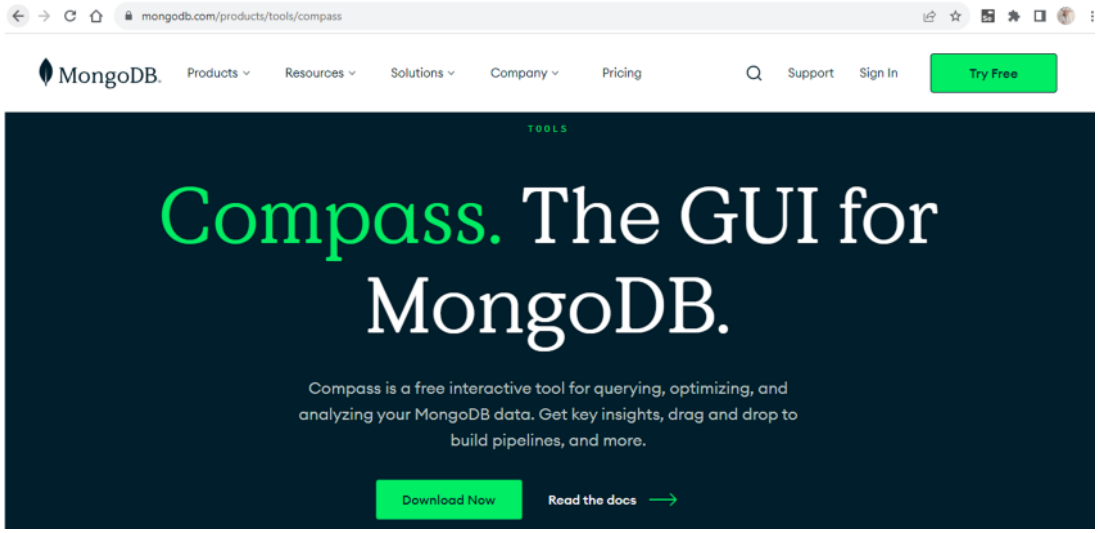


3. Setelah diunduh, jalankan penginstal. Ini akan memerlukan waktu beberapa menit.

3.2 MongoDB

Instalasi MongoDB melibatkan beberapa langkah, tergantung pada sistem operasi yang anda gunakan. Berikut adalah panduan langkah-langkah umum untuk instalasi MongoDB:

1. Pilih Versi MongoDB
   Kunjungi situs resmi MongoDB untuk mendownload versi terbaru: [MongoDB Download Center] (https://www.mongodb.com/try/download/community)



2. Pilih Tipe Instalasi:
   MongoDB menyediakan beberapa tipe instalasi, termasuk Community Server, Atlas (MongoDB di cloud), dan Ops Manager. Pada umumnya, untuk pengembangan lokal atau uji coba, Anda akan memilih "Community Server".
3. Pilih Sistem Operasi
   Pilih sistem operasi yang Anda gunakan (Windows, macOS, atau Linux) dan versi arsitektur yang sesuai (32-bit atau 64-bit).
4. Download dan Install:
   - Windows:
     Unduh installer MongoDB untuk Windows.
     Jalankan installer dan ikuti petunjuknya.
     Pastikan Anda menambahkan MongoDB ke PATH selama proses instalasi (opsional, tapi disarankan).
   - macOS:
     Unduh installer MongoDB untuk macOS.
     Ekstrak file ZIP dan pindahkan aplikasi ke direktori yang diinginkan (biasanya `/usr/local`).
   - Linux:
     Pada distribusi Linux yang berbeda, cara instalasinya dapat bervariasi. Umumnya, gunakan manajer paket seperti **`apt`** (Ubuntu/Debian) atau **`yum`** (CentOS/RHEL).
     Contoh untuk Ubuntu/Debian:

```bash
sudo apt-get update
sudo apt-get install -y mongodb
```

5. Konfigurasi MongoDB:
   MongoDB dapat dijalankan tanpa konfigurasi tambahan, tetapi Anda mungkin perlu mengonfigurasi beberapa opsi tergantung pada kebutuhan Anda.
   Berkas konfigurasi utama MongoDB biasanya bernama **`mongod.conf`** dan terletak di direktori instalasi MongoDB.
6. Jalankan MongoDB:

- Windows:
  Buka Command Prompt atau PowerShell sebagai administrator.
  Jalankan perintah:

```
bash
mongod
```

- macOS atau Linux:
  Buka terminal.
  Jalankan perintah:

```
bash
mongod
```

7. Jalankan Shell MongoDB:
   Buka terminal atau Command Prompt baru.
   Jalankan perintah:

```
bash
mongo
```

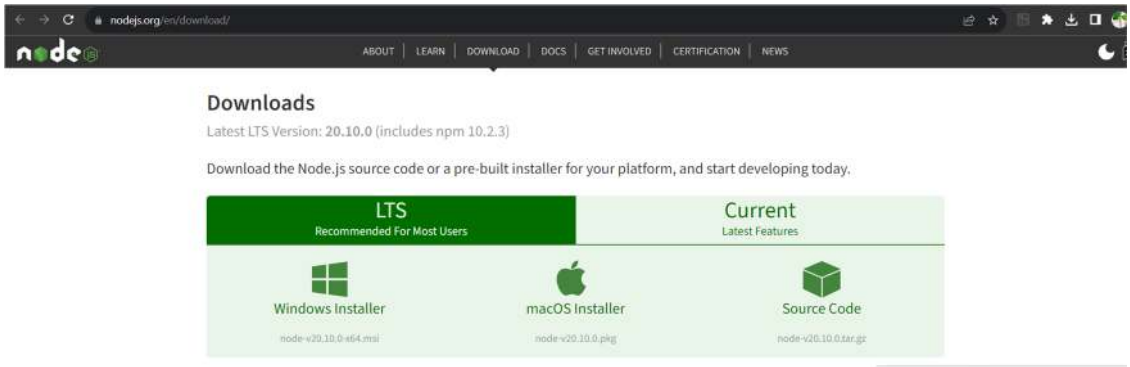
8. Selesai: Anda sekarang harus dapat mengakses server MongoDB lokal Anda dan mengelola basis data menggunakan shell MongoDB atau alat pengelolaan GUI.

   Pastikan untuk membaca dokumentasi resmi MongoDB untuk informasi lebih lanjut dan panduan spesifik untuk sistem operasi yang Anda gunakan.

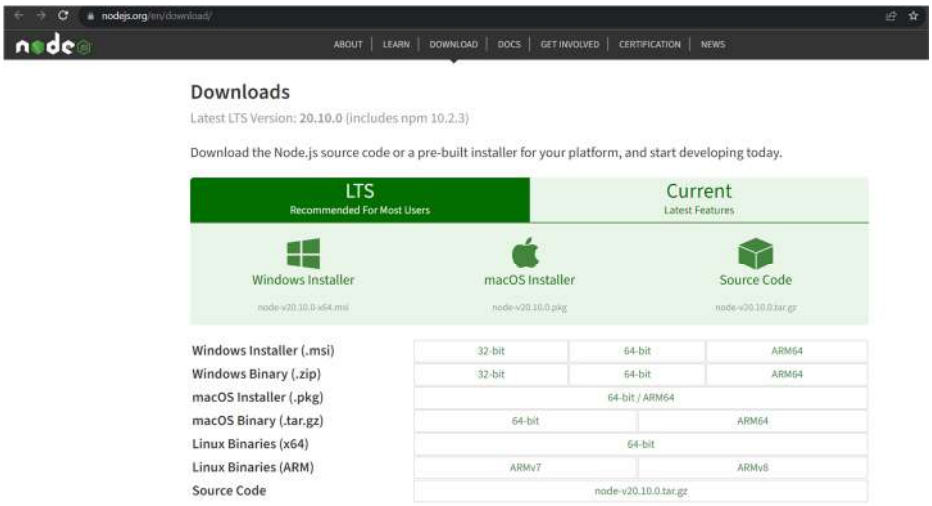
3.2 NodeJS

Langkah cara penginstalannya sebahgai berikut:

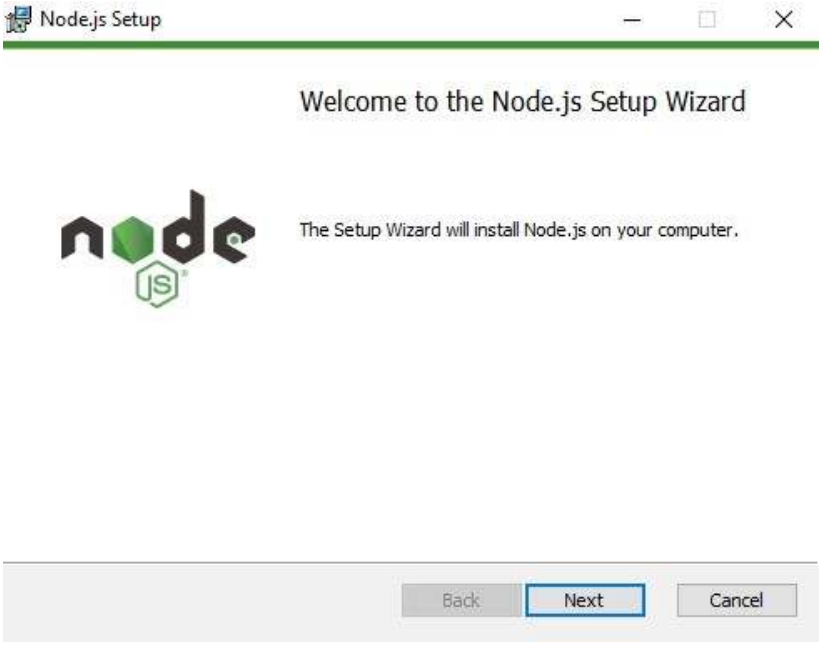
1. Cari pada browser link berikut https://nodejs.org/en/download/



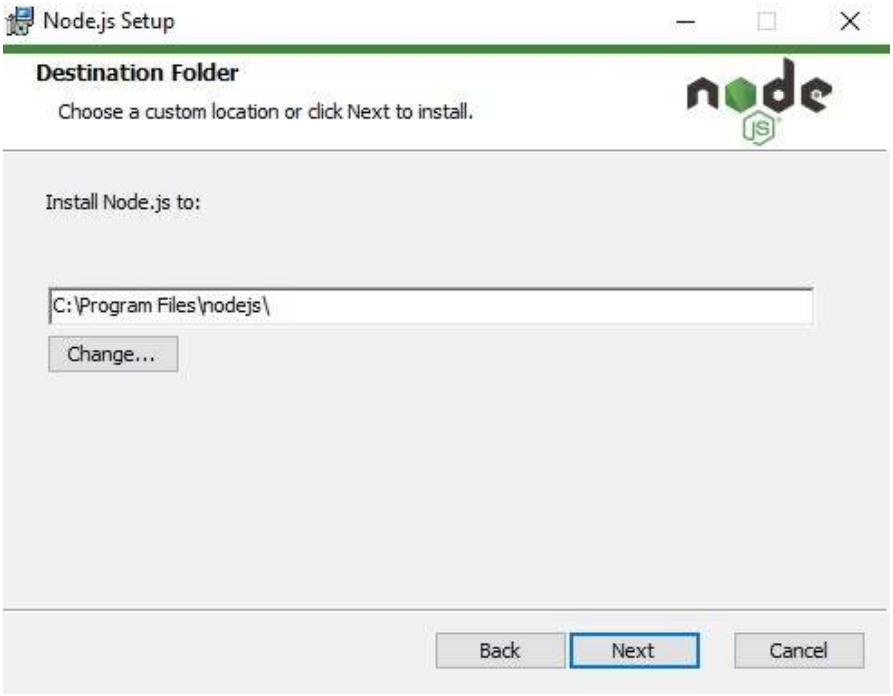
2. Kemudian pilih sesuai kebutuhan pada laptop, disini penulis memilah Windows installer(.msi) 64-bit



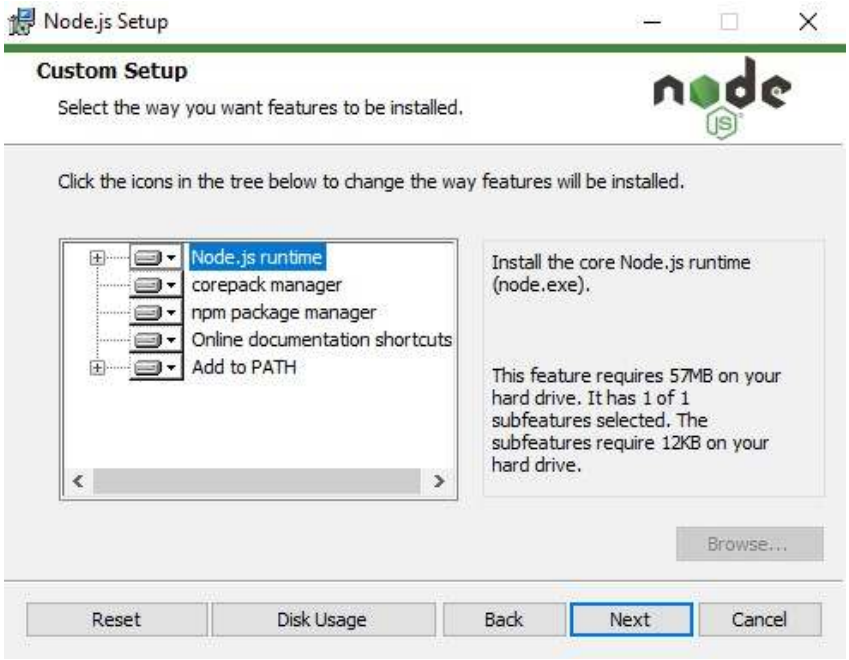
3. Setelah menunggu proses intallasi selesai buka file Node.js Setup akan terbuka. Klik **Next**.



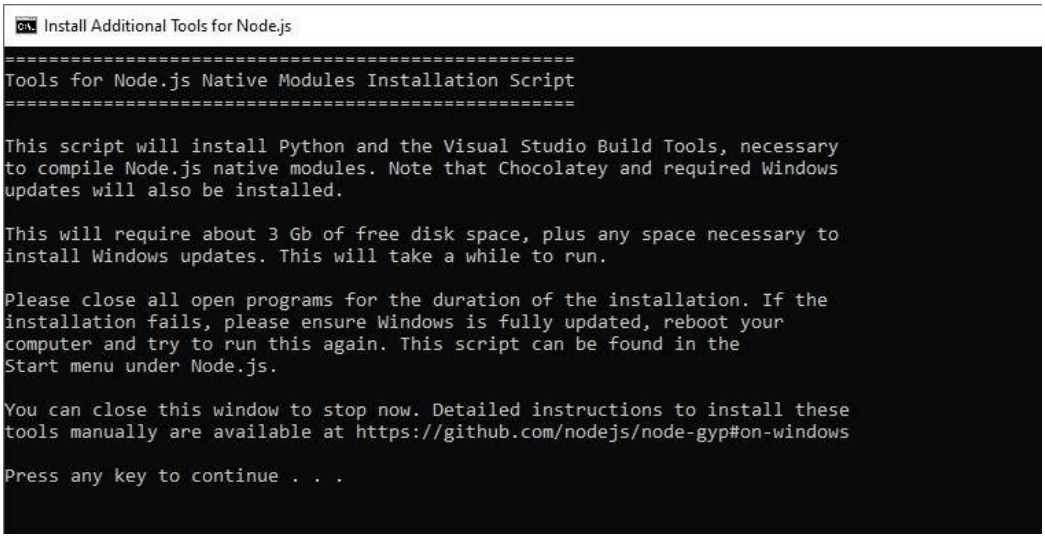
4. Baca EULA program dan centang **I accept the terms in the License Agreement**. Klik **Next** lagi.
5. Pilih folder tujuan, lalu klik **Next**.



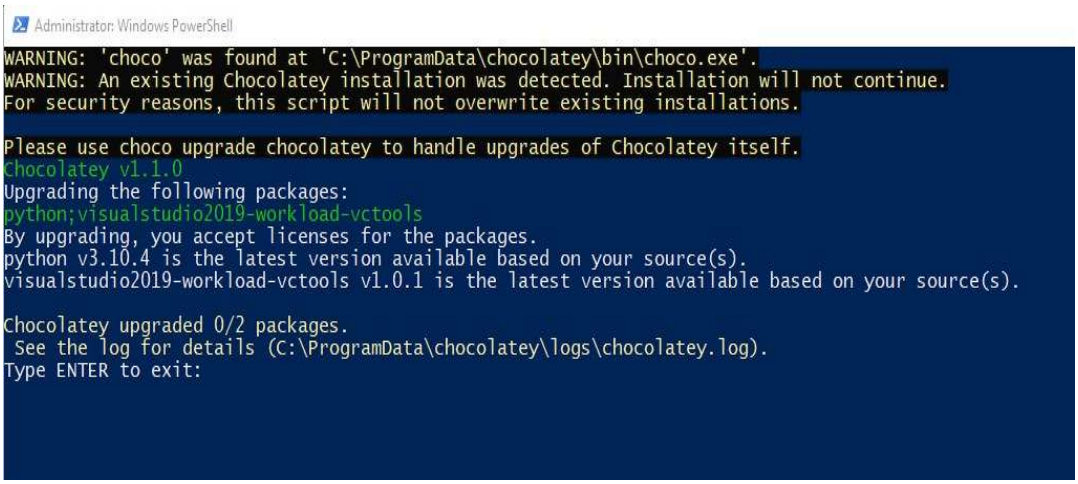
6. Pilih fitur yang akan diinstal. Kalau tidak yakin, biarkan nilai defaultnya. Klik **Next**.



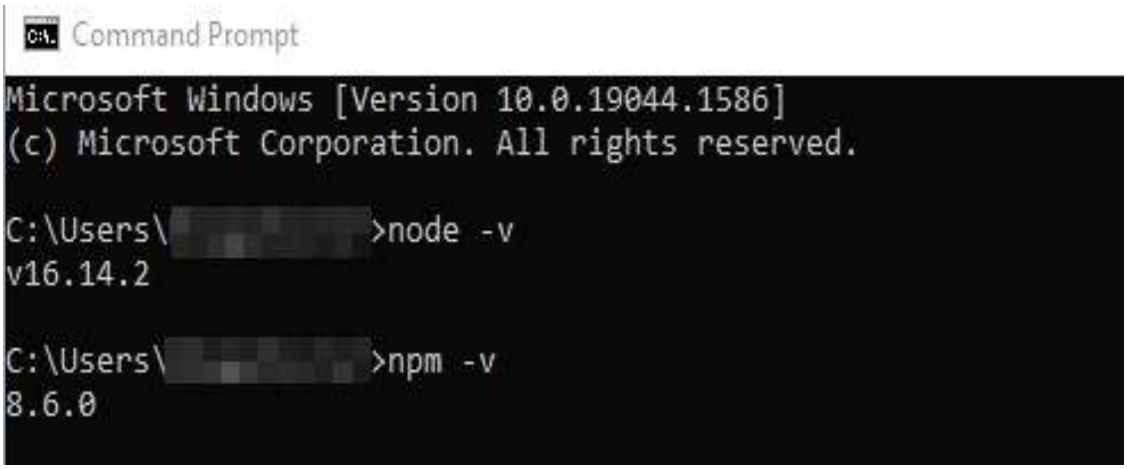
7. Di halaman berikutnya, centang **Automatically install the necessary tools**. Klik **Next**, lalu **Install**. Mungkin akan muncul prompt yang menanyakan apakah Anda mengizinkan program Setup melakukan perubahan pada PC Anda. Pilih **Yes** kalau bersedia.
8. Setelah penginstalan default selesai, Command Prompt akan terbuka dan menampilkan setting tambahan. Tekan dua kali tombol apa pun untuk melanjutkan.



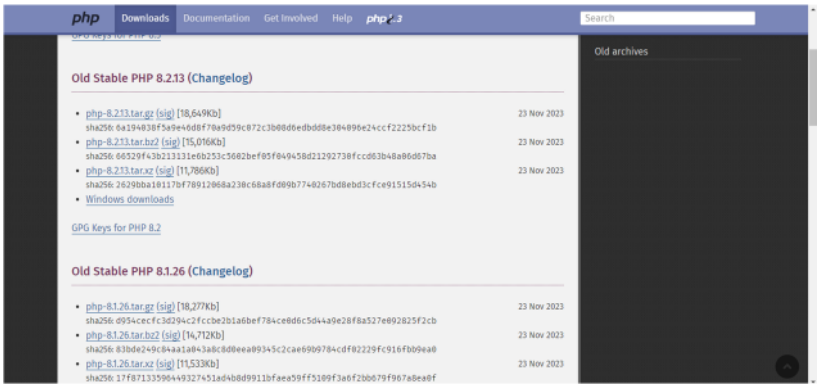
9. Setelah proses selesai, Command Prompt akan meminta Anda menekan **Enter** untuk menutup jendela.



10. Setelah proses selesai, Command Prompt akan meminta Anda menekan **Enter** untuk menutup jendela.



3.4 PHP 8.2.13



Instalasi PHP pada Windows:

1. Unduh PHP:
   Kunjungi PHP Windows Download Page dan pilih versi PHP yang sesuai dengan sistem operasi dan arsitektur (x86 atau x64).
2. Ekstrak File ZIP:
   Ekstrak file ZIP yang telah diunduh ke direktori yang diinginkan, misalnya **C:\php.**
3. Konfigurasi php.ini:
   Salin file **php.ini-development** menjadi **php.ini.**
   Buka **php.ini** dan sesuaikan pengaturan sesuai kebutuhan Anda. Pastikan untuk mengatur path ekstensi PHP dan konfigurasi lainnya.
4. Tambahkan PHP ke PATH (Opsional):
   Tambahkan path ke direktori PHP ke dalam variabel lingkungan PATH. Ini memudahkan untuk menjalankan perintah PHP dari mana saja di Command Prompt atau PowerShell.
5. Uji Instalasi:
   Buka Command Prompt atau PowerShell dan jalankan perintah:

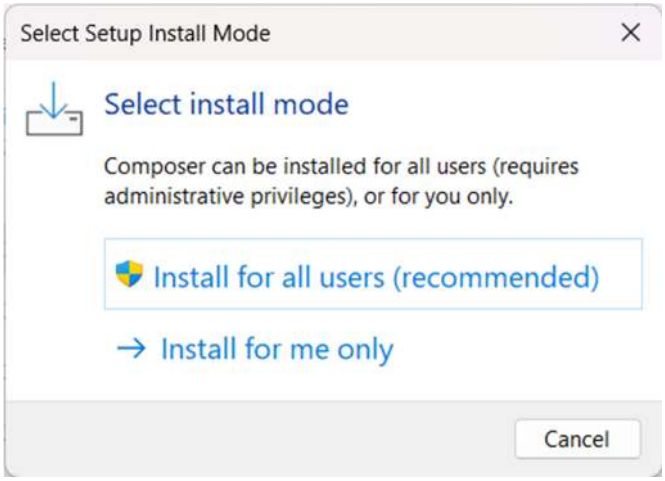
```
bash

php -v
```

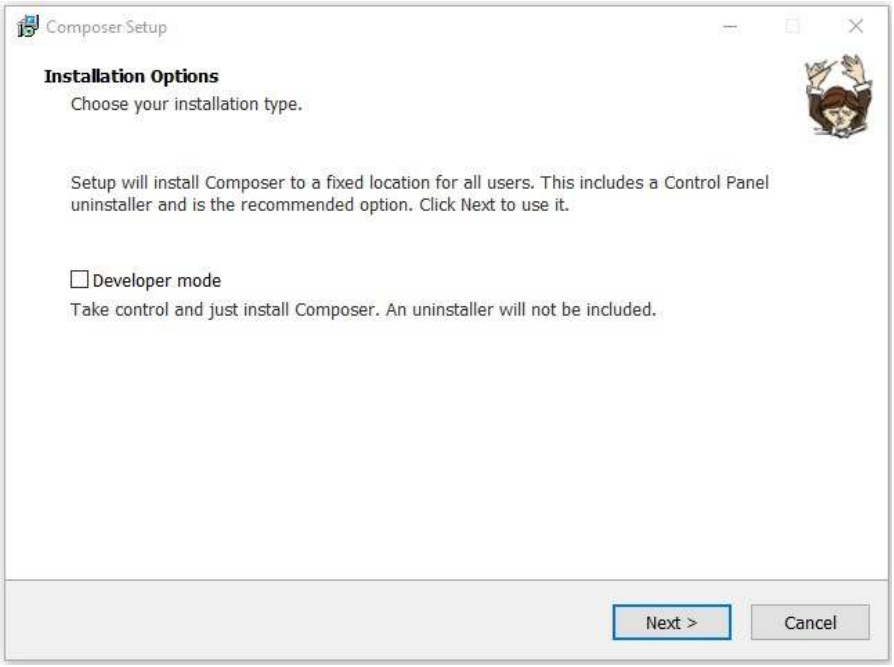
3.5 Composer Setup

Cara mendowload composer sebagai berikut:

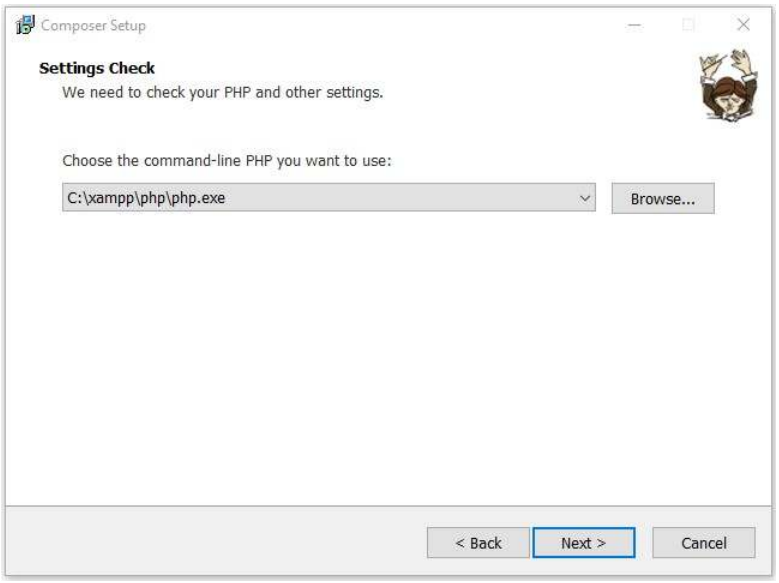
1. Download file installer dari web resminya di Getcomposer.org



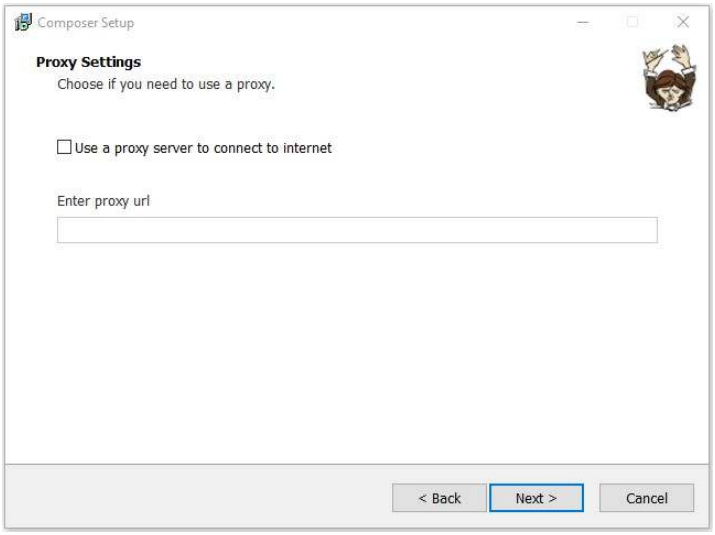
2. Untuk mulai proses instalasi, double klik file `Composer-Setup.exe`. Di jendela pertama, biarkan checkbox **Developer mode** kosong dan langsung saja klik tombol Next



3. Composer akan mencari dimana file php.exe berada, klik tombol **Next**.

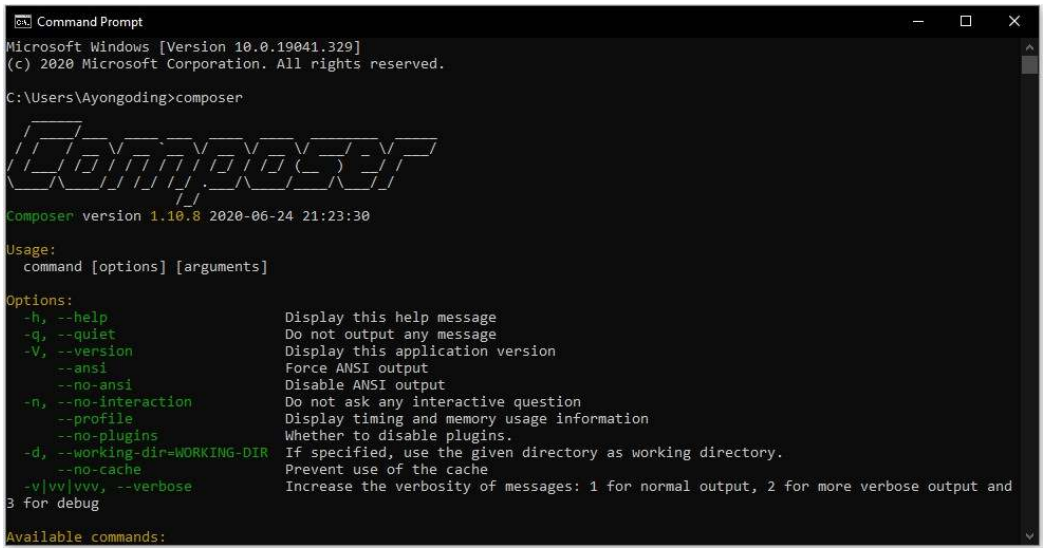


4. Untuk jendela inputan *proxy* tidak perlu diisi dan langsung saja klik tombol **Next**.

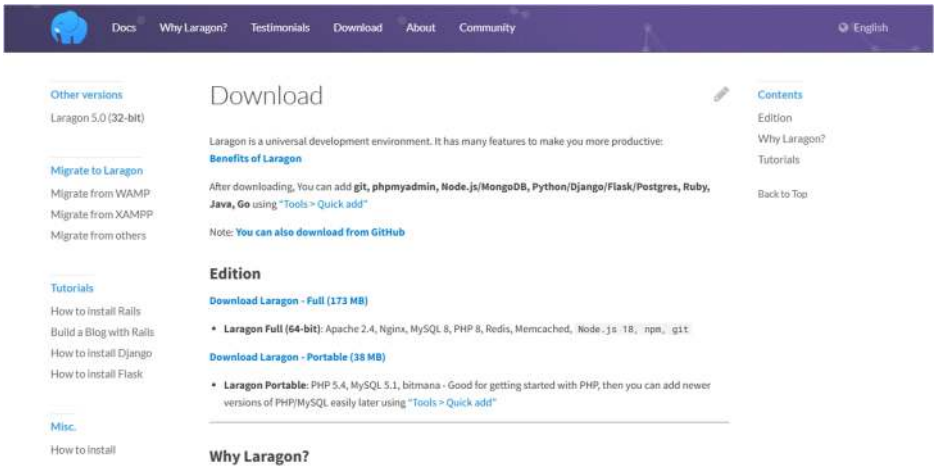


5. Kemudian klik tombol **Install**. Tunggu beberapa saat, klik tombol **Next** dan **Finish**. Maka composer sudah sukses terinstall.

6. Untuk menguji apakah composer sudah berhasil di install atau tidak, buka cmd, ketik composer dan tekan Enter. Jika sukses mengintal, composer akan muncul tampilan text dibawah ini:



3.6 Laragon



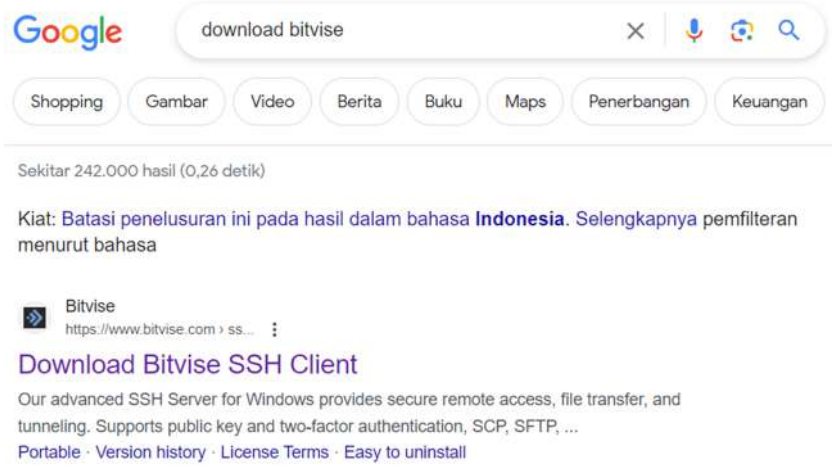
Laragon adalah lingkungan pengembangan lokal untuk PHP yang mudah digunakan dan dapat diinstal dengan cepat. Berikut adalah langkah-langkah untuk mendownload dan menginstal Laragon:

Langkah-langkah Instalasi Laragon di Windows:

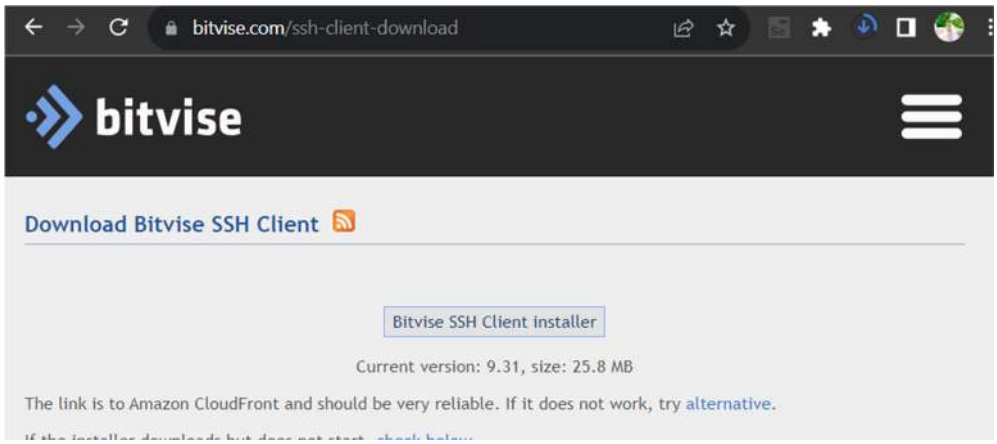
1. Unduh Laragon:
   Kunjungi situs web resmi Laragon dan unduh versi terbaru sesuai dengan kebutuhan Anda (Laragon Full atau Laragon Portable).
2. Ekstrak File:
   Setelah selesai mengunduh, ekstrak file ZIP Laragon ke direktori yang diinginkan di sistem Anda.
3. Jalankan Instalator:
   Buka direktori tempat Anda mengekstrak Laragon, dan jalankan instalator (biasanya memiliki nama seperti laragon.exe).
4. Pilih Tipe Instalasi:
   Pada layar instalasi, pilih jenis instalasi yang sesuai dengan kebutuhan Anda (Full, Portable, atau Lite). Full adalah pilihan yang umumnya direkomendasikan untuk pengembangan PHP lokal yang komprehensif.
5. Pilih Lokasi Instalasi:
   Pilih direktori tempat Anda ingin menginstal Laragon. Defaultnya biasanya berada di direktori C:\laragon.
6. Selesaikan Instalasi:
   Ikuti langkah-langkah selanjutnya pada instalator hingga instalasi selesai.
7. Jalankan Laragon:
   Setelah instalasi selesai, jalankan Laragon dari ikon yang muncul di desktop atau dari direktori instalasi.
8. Uji Instalasi:
   Buka Laragon dan pastikan bahwa server Apache dan MySQL telah diaktifkan. Anda dapat melihat status server di bagian atas jendela Laragon.
9. Buka Proyek PHP:
   Anda dapat membuka proyek PHP eksisting atau membuat proyek baru di direktori www di dalam direktori instalasi Laragon.
10. Selesai:
    Sekarang, Anda sudah memiliki Laragon terinstal dan siap digunakan untuk pengembangan PHP lokal.

3.7 Bitvise

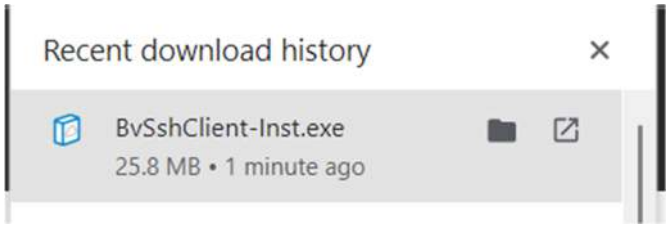
1. Klik link berikut untuk ke website resminya Download Bitvise SSH Client



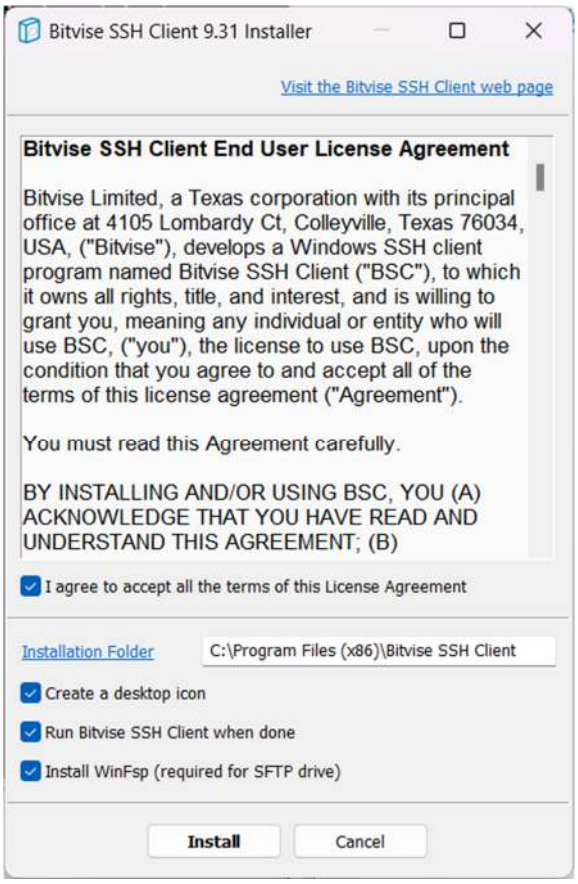
2. Pilih Bitvise SSH Client installer



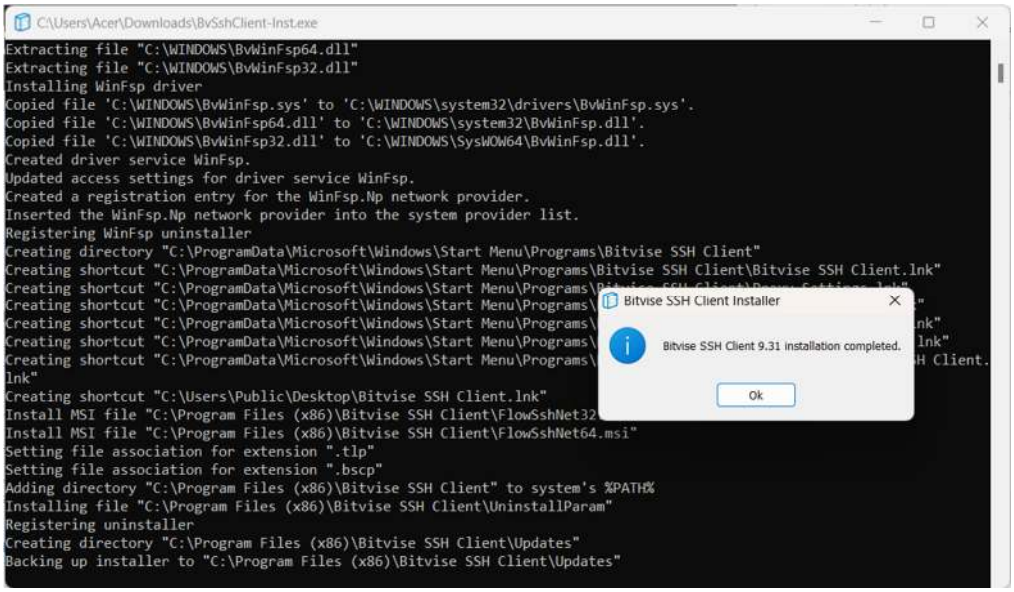
3. Klik file yang telah di download



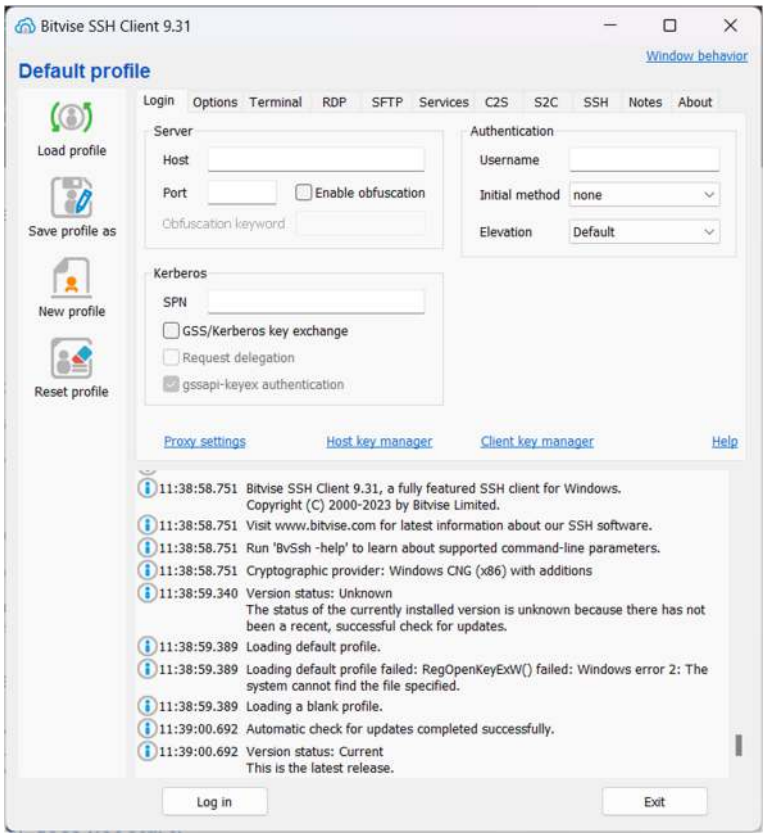
4. Muncul tampilan aplikasi bitvice ssh client, ceklis tanda I agree to accept all the terms of this License Agreement, pilih Install.



5. Tunggu proses instal sampai selesai jika sudah klik ok.



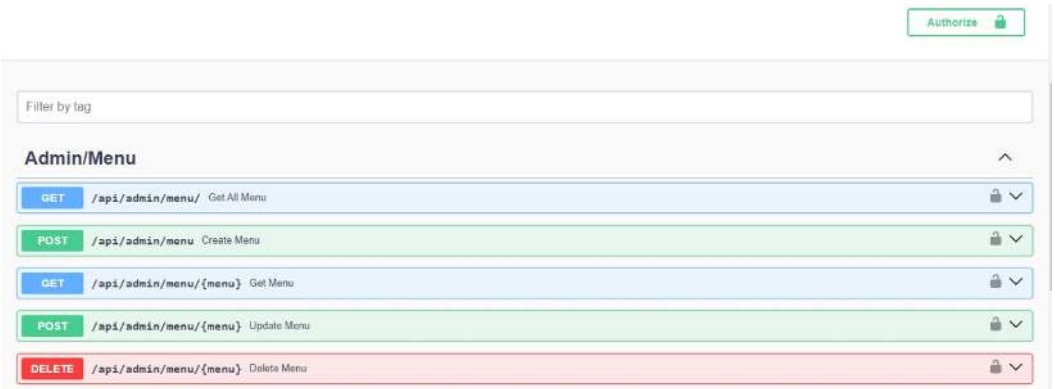
6. Akan muncul Aplikasi Bitvise SSH Client dan versi nya maka proses install selesai



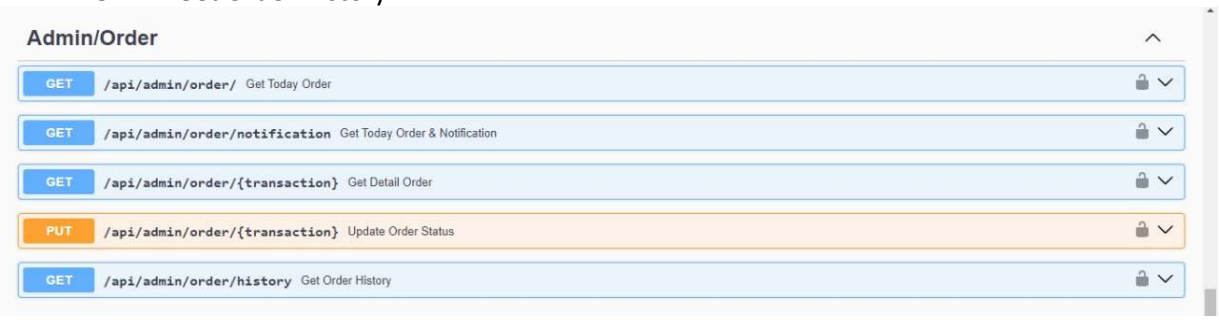
3.8 Swagger

Penulis membuat API melalui tools swagger dengan menggunakan bahasa PHP framework Laravel.

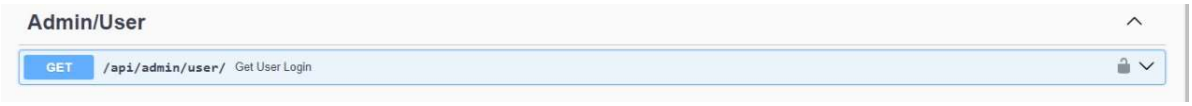
1. Penulis membuat API pada bagian Admin fitur menu yang dibutuhkan dalam pembuatan CRUD diantarnya:
   - GET -> Get All Menu
   - POST -> Create Menu
   - GET -> Get Menu
   - POST -> Update Menu
   - DELETE -> Delete Menu



2. Penulis membuat API pada bagian admin untuk sistem order yang dibutuhkan dalam pemesanan makanan diantaranya:
   - GET -> Get Today Order
   - GET -> Get Today & Notification
   - GET -> Get Detail Order
   - PUT -> Update Order Status
   - GET -> Get Order History



3. Penulis membuat API untuk kebutuhan admin untuk memasuki akun log in website diantaranya:
   - GET-> Get User Login



4. Penulis membuat API untuk kebutuhan Autentikasi token untuk memasuki log in diantaranya:
   - POST -> Login Function a User

5. Penulis membuat API untuk kebutuhan tampilan menu dan order pesanan yang dilakukan oleh customer diantaranya:
   - GET -> Get All Menu
   - POST -> Create an Order

CHAPTER 4
MEMBUAT WEBSITE PEMESANAN MAKANAN

4.1 FRONTEND

4.1.1 Tampilan Frontend Customer

Dalam pembuatan frontend penulis memilih menggunakan bahasa pemrograman VueJs, berikut source code dalam pembuatan website:

4.1.1.1Membuat Dashboard

Dalam pembuatan Dasboard yang pertama yaitu membuat index.vue untuk tampilan utama website

```
<script setup>
</script>

<template>
  <div>
    <div class="row">
      <div class="col-md-5 col-lg-6 text-center">
        <h1 class="text-primary">P3 RAFA <br class="d-none d-xl-block" />ORDER FOOD<br
/></h1>
         <p class="mb-5 fs-5">Setiap gigitannya memiliki kelezatan yang berbeda, <br
class="d-none d-xl-block" />harga terjangkau!</p>
      </div>
      <div class="col-md-7 col-lg-6">
        <img class="pt-7 pt-md-0 w-100" src="/img/profilresto.webp" alt="hero-header" />
      </div>
    </div>
  </div>
</template>

<style scoped>

</style>
```

Penjelasan dari source code diatas adalah:

Kode yang diberikan adalah komponen Vue 3 yang ditulis menggunakan Composition API, yang digunakan untuk membangun antarmuka pengguna pada aplikasi web menggunakan Vue.js. Mari kita jelaskan kode tersebut dengan bahasa Indonesia:

1. `<script setup>`: Ini adalah bagian dari Vue 3 yang memungkinkan penggunaan Composition API dengan cara yang lebih ringkas. Ini secara otomatis menangani kode boilerplate sehingga definisi komponen menjadi lebih bersih.
2. <template>`: Bagian ini berisi markup HTML dari komponen Vue. Komponen ini mendefinisikan tata letak sederhana dengan dua kolom menggunakan kelas-kelas Bootstrap (`col-md-5`, `col-lg-6`, dll.). Kontennya mencakup judul dengan teks "P3 RAFA ORDER FOOD" dan sebuah paragraf dengan deskripsi. Selain itu, ada gambar dengan atribut sumber yang menunjuk ke "/img/profilresto.webp."
3. `<style scoped>`: Bagian ini berisi gaya khusus komponen. Atribut `scoped` memastikan bahwa gaya tersebut hanya berlaku untuk komponen ini dan tidak bocor ke komponen lain.

Secara ringkas, kode ini mendefinisikan komponen Vue 3 menggunakan Composition API, membuat tata letak sederhana dengan judul, paragraf, dan gambar. Gaya khusus komponen juga disertakan untuk menghias elemen-elemen dalam komponen tersebut.

Kemudian pembuatan Dasboard yaitu membuat header.vue untuk tampilan utama website yang diterapkan kesemua tampilan pada halaman cutomer.

```
<script lang="ts" setup>

</script>

<template>
  <div>
    <nav class="navbar navbar-expand-lg bg-body-tertiary">
      <div class="container-fluid">
        <NuxtLink to="/" class="navbar-brand">
          <img src="/img/logo.jpg" alt="logo" width="50px" height="50px">
        </NuxtLink>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent"
                aria-controls="navbarSupportedContent"  aria-expanded="false"  aria-
label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
          <ul class="navbar-nav ms-auto mb-2 mb-lg-0">
            <li class="nav-item">
              <NuxtLink class="nav-link" to="/menu">Menu</NuxtLink>
            </li>
            <li class="nav-item mx-2">
              <NuxtLink class="nav-link" to="/order">Order</NuxtLink>
            </li>
            <li class="nav-item">
              <NuxtLink class="btn btn-orange" to="/login"> Log In</NuxtLink>
            </li>
          </ul>
        </div>
      </div>
    </nav>
  </div>
</template>

<style scoped>
.btn-orange {
  color: white;
  background-color: orange;
}
</style>
```

Penjelasan source code diatas adalah:
Kode tersebut adalah komponen Vue 3 yang ditulis menggunakan TypeScript dan menggunakan fitur baru yaitu

1. `<script lang="ts" setup>`: Ini adalah fitur baru pada Vue 3 yang memungkinkan penulisan komponen dengan cara yang lebih ringkas dan mudah dibaca, serta mendukung penggunaan TypeScript. Fitur ini secara otomatis menangani boilerplate code yang biasanya diperlukan untuk komponen Vue.
2. `<template>`: Bagian ini berisi markup HTML dari komponen. Komponen ini merupakan navigasi (navbar) untuk aplikasi web. Terdapat logo dengan tautan ke beranda ("/") menggunakan `<NuxtLink>`, tombol toggler untuk menu responsif, dan daftar navigasi dengan tautan ke "/menu", "/order", dan tombol "Sign In" untuk menuju "/login". Gaya Bootstrap digunakan untuk memperindah tampilan navigasi.
3. `<style scoped>`: Bagian ini berisi gaya khusus untuk komponen ini. Di dalamnya, terdapat definisi gaya untuk tombol dengan kelas `.btn-orange`, yang mengatur warna teks putih dan latar belakang orange.

Secara keseluruhan, komponen ini menyediakan navigasi (navbar) dengan tautan menuju beranda, menu, dan halaman order, serta tombol "Log In". Gaya khusus diaplikasikan untuk mendefinisikan penampilan khusus tombol dengan warna latar belakang oranye.

4.1.1.2 Tampilan Menu Customer

Dalam membuat tampilan menu customer ini, ada beberapa pilihan menu yang dapat dipilih oleh customer diantaranya terdapat makanan dan minuman.

Menu index.vue

```
<script setup>
import { reactive, watch, onMounted, ref } from 'vue';
const showToast = ref(false);

const { data: resFoods } = await useApiFetch('/home/menu?type=makanan')
const foods = resFoods.value

const { data: resDrinks } = await useApiFetch('/home/menu?type=minuman')
const drinks = resDrinks.value

const shoppingCart = reactive([])

onMounted(() => {
  const storedCart = JSON.parse(localStorage.getItem('shoppingCart') || '[]')
  shoppingCart.push(...storedCart)
});

watch(shoppingCart, (newValue) => {
  localStorage.setItem('shoppingCart', JSON.stringify(newValue))
});

function addToCart(menu) {
  let exists = false;

  for (const cartItem of shoppingCart) {
    if (cartItem._id === menu._id) {
      cartItem.qty += 1
      exists = true;
      break;
    }
  }
```

```
  if (!exists) {
    shoppingCart.push({
      ...menu,
      qty: 1
    })
  }

  showToast.value = true

  setTimeout(() => {
    showToast.value = false
  }, 2000)
}
</script>

<template>
  <div>
    <Alert v-if="showToast" title="Pesanan Berhasil Ditambahkan"/>
    <div class="row mb-3">
      <div class="col-md-4"></div>
      <div class="col-md-4">
        <div class="card text-bg-warning text-center">
          <div class="card-header">
            <div class="card-title text-white">MENU MAKANAN</div>
          </div>
        </div>
      </div>
      <div class="col-md-4"></div>
    </div>
    <div class="row">
      <div class="col-md-6" v-for="(food, index) in foods" :key="'food-'+index">
        <div class="card border-warning mb-3" style="max-width: 600px;">
          <div class="row g-0">
            <div class="col-md-4">
              <img :src="food.image" class="img-fluid rounded-start" :alt="food.name">
            </div>
            <div class="col-md-8">
              <div class="card-body">
                  <h5 class="card-title">{{ food.name }}<span class="float-end"><Rupiah
:angka="food.price"/></span></h5>
                <p class="card-text">{{ food.desc }}</p>
                                    <p    class="card-text"><small    class="text-body-
secondary"> </small></p>
                <button type="button" class="btn btn-warning float-end text-white mb-3"
@click="addToCart(food)">Tambah Pesanan</button>
              </div>
            </div>
          </div>
        </div>
      </div>
```

```
    </div>

    <div class="row mb-3">
      <div class="col-md-4"></div>
      <div class="col-md-4">
        <div class="card text-bg-warning text-center">
          <div class="card-header">
            <div class="card-title text-white">MENU MINUMAN</div>
          </div>
        </div>
      </div>
      <div class="col-md-4"></div>
    </div>
    <div class="row">
      <div class="col-md-6" v-for="(drink, index) in drinks" :key="'drink-'+index">
        <div class="card border-warning mb-3" style="max-width: 600px;">
          <div class="row g-0">
            <div class="col-md-4">
              <img :src="drink.image" class="img-fluid rounded-start" :alt="drink.name">
            </div>
            <div class="col-md-8">
              <div class="card-body">
                 <h5 class="card-title">{{ drink.name }}<span class="float-end"><Rupiah
:angka="drink.price"/></span></h5>
                <p class="card-text">{{ drink.desc }}</p>
                                    <p    class="card-text"><small    class="text-body-
secondary"> </small></p>
                   <button type="button" class="btn btn-warning float-end text-white"
@click="addToCart(drink)">Tambah Pesanan</button>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

<style scoped></style>
```

Penjelasan dari source code diatas adalah :

Kode tersebut adalah sebuah komponen Vue 3 yang menggunakan Composition API dengan format `<script setup>`. Berikut adalah penjelasan kode tersebut:

a. `<script setup>`: Ini adalah fitur baru pada Vue 3 yang memungkinkan penulisan komponen dengan lebih ringkas dan mudah dibaca, serta mendukung penggunaan TypeScript.
b. Import Statements: Dalam blok `<script setup>`, beberapa modul Vue diimpor, seperti `reactive`, `watch`, `onMounted`, dan `ref`. Modul ini akan digunakan untuk manajemen state dan lifecycle komponen.
c. Pengambilan Data: Kode ini melakukan pengambilan data untuk menu makanan dan minuman melalui API dengan menggunakan `useApiFetch`. Data tersebut disimpan dalam variabel `foods` dan `drinks`.

d. State Management: Sebuah state `shoppingCart` yang reactive (reaktif) digunakan untuk menyimpan pesanan. Pada saat komponen dimount (`onMounted`), data dari localStorage diambil dan dimasukkan ke dalam `shoppingCart`. Selanjutnya, menggunakan `watch`, setiap perubahan pada `shoppingCart` akan disimpan kembali ke localStorage.
e. Fungsi `addToCart`: Fungsi ini digunakan untuk menambahkan item ke dalam keranjang belanja (`shoppingCart`). Jika item sudah ada dalam keranjang, kuantitasnya akan ditambah. Jika tidak, item baru akan ditambahkan. Selain itu, sebuah pesan notifikasi (`showToast`) akan ditampilkan selama 2 detik setelah menambahkan pesanan.
f. Template: Bagian template mendefinisikan tampilan komponen. Terdapat dua bagian untuk menu makanan dan minuman. Setiap item menu ditampilkan dalam bentuk kartu dengan informasi seperti nama, gambar, harga, dan deskripsi. Tombol "Tambah Pesanan" dipasang untuk setiap item menu, yang akan memanggil fungsi `addToCart` ketika diklik.
g. Komponen Rupiah dan Alert: Terlihat bahwa kode menggunakan komponen `Rupiah` dan `Alert`, yang kemungkinan merupakan komponen tambahan di dalam proyek Vue yang digunakan untuk memformat mata uang dan menampilkan pemberitahuan.
h. Gaya (`<style scoped>`): Tidak ada aturan gaya yang diatur dalam komponen ini.

Secara keseluruhan, komponen ini bertanggung jawab untuk menampilkan menu makanan dan minuman, memberikan kemampuan untuk menambahkan item ke dalam keranjang belanja, dan menangani manajemen status keranjang belanja.

4.1.1.3 Tampilan Order Cutomer

Dalam tampilan order penulis membuat sebuah fitur tambah pesanan yang di dalamnnya berisikan pesanan customer dan data customer seperti nama, nomor meja. Selanjutnya cutomer dapat mensubmit pesanannya dan nantinya pesanan cutomer akan masuk ke tampilan admin restoran berupa informasi notifikasi dari customer.

```
<script setup>
import { reactive, onMounted, watch, computed, ref } from 'vue';

const shoppingCart = reactive([])
const table_number = ref()
const order_name = ref()
const payment = ref()
const showToast = ref(false);

onMounted(() => {
  const storedCart = JSON.parse(localStorage.getItem('shoppingCart') || '[]')
  shoppingCart.push(...storedCart)
});

watch(shoppingCart, (newValue) => {
  localStorage.setItem('shoppingCart', JSON.stringify(newValue))
});

const totalPrice = computed(() => {
  let sum = 0

  for (const menu of shoppingCart) {
    sum += menu.price * menu.qty
  }

  return sum;
})
```

```
const totalItem = computed(() => {
  let sum = 0

  for (const menu of shoppingCart) {
    sum += 1
  }

  return sum;
})

function removeFromCart(product) {
    const productIndex = shoppingCart.findIndex(item => item._id === product._id)
    shoppingCart[productIndex].qty -= 1;
    if (shoppingCart[productIndex].qty < 1) {
      shoppingCart.splice(productIndex, 1)
    }
  }

async function sendOrder() {
  const orders = shoppingCart.map((item) => {
    return {
      'menu_id' : item._id,
      'qty': item.qty
    }
  })

  const reqOrder = {
    table_number :table_number.value,
    order_name : order_name.value,
    payment: payment.value,
    orders
  }

  await useApiFetch('/home/order', {
    method: 'POST',
    body: reqOrder,
  })

  table_number.value = ''
  order_name.value = ''
  payment.value = ''

  localStorage.setItem('shoppingCart', JSON.stringify([]))

  showToast.value = true

  setTimeout(() => {
    showToast.value = false
  }, 2000)
```

```
}
</script>

<template>
  <div>
    <Alert v-if="showToast" title="Pesanan Berhasil DIbuat"/>
    <div class="card mb-3">
      <div class="card-header">
        <div class="card-title">
          <div class="row">
            <div class="col-md-6">Pemesanan</div>
            <div class="col-md-6">
                      <NuxtLink  to="menu"  class="btn  btn-success  float-end">Tambah
Pesanan</NuxtLink>
            </div>
          </div>
        </div>
      </div>
      <div class="card-body">
        <div class="table-responsive">
          <table class="table table-border table-striped">
            <thead>
              <tr>
                <th>Name</th>
                <th>Qty</th>
                <th>Harga</th>
                <th>Action</th>
              </tr>
            </thead>
            <tbody>
              <tr v-for="(menu, index) in shoppingCart" :key="'menu-'+index">
                <td>{{ menu.name }}</td>
                <td>{{ menu.qty }}</td>
                <td><Rupiah :angka="menu.price"/></td>
                <td>
                                    <button  type="button"  class="btn  btn-danger"
@click="removeFromCart(menu)">Remove</button>
                </td>
              </tr>
              <tr>
                <td class="fw-bold" colspan="2">Total Item : {{ totalItem }}</td>
                <td colspan="2"><Rupiah :angka="totalPrice"/></td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </div>
    <div class="card">
      <div class="card-header">
```

```
        <div class="card-title" style="text-align: center;">HARAP DI ISI</div>
      </div>
      <div class="card-body">
        <form @submit.prevent="sendOrder">
          <div class="row">
            <div class="col-md-6">
              <div class="input-group mb-3">
              <span class="input-group-text" id="order_name">
                <i class="bi bi-person"></i>
              </span>
                 <input type="text" class="form-control" placeholder="Nama Pemesan" v-
model="order_name">
            </div>
            </div>
          </div>
          <div class="row">
            <div class="col-md-6">
              <div class="input-group mb-3">
              <span class="input-group-text" id="table_number">
                <i class="bi bi-journal-richtext"></i>
              </span>
                  <input type="number" class="form-control" placeholder="Table No" v-
model="table_number">
            </div>
            </div>
          </div>
          <div class="row">
            <div class="col-md-6">
              <div class="input-group mb-3">
              <label class="input-group-text" for="payment">Pembayaran</label>
              <select class="form-select" id="payment" v-model="payment">
                <option selected>Choose...</option>
                <option value="cash">Cash</option>
                <option value="debit">Debit</option>
              </select>
            </div>
            </div>
          </div>
          <button type="submit" class="btn btn-primary float-end">Submit</button>
        </form>
      </div>
    </div>
  </div>
</template>

<style scoped></style>
```

Penjelasan source code diatas adalah :

Source code ini adalah bagian dari sebuah komponen Vue yang ditulis menggunakan Composition API dengan format `<script setup>`.

1. State dan Pengelolaan Keranjang Belanja:
   - `shoppingCart`: Merupakan objek reactive yang menyimpan daftar pesanan. Data ini diambil dari local storage pada saat komponen di-mount.
   - `table_number`, `order_name`, dan `payment`: Ref yang digunakan untuk menyimpan nomor meja, nama pemesan, dan metode pembayaran.
   - `showToast`: Ref yang menunjukkan apakah pesan notifikasi ("Pesanan Berhasil Dibuat") harus ditampilkan atau tidak.
2. Computed Properties:
   - `totalPrice`: Menghitung total harga semua item dalam keranjang belanja.
   - `totalItem`: Menghitung total jumlah item dalam keranjang belanja.
3. Watchers:
   Mengawasi perubahan pada `shoppingCart` dan menyimpannya ke dalam local storage setiap kali terjadi perubahan.
4. Fungsi `removeFromCart`:
   Mengurangi jumlah qty dari item di keranjang belanja. Jika qty menjadi kurang dari 1, item tersebut dihapus dari keranjang.
5. Fungsi `sendOrder`:
   - Mengirim pesanan ke backend menggunakan API POST.
   - Data yang dikirim mencakup nomor meja, nama pemesan, metode pembayaran, dan daftar pesanan.
   - Setelah pengiriman berhasil, mereset nilai-nilai ref dan local storage, menampilkan notifikasi pesan berhasil, dan menyembunyikan notifikasi setelah 2 detik.
6. Template:
   - Menampilkan elemen-elemen HTML yang dibentuk dengan menggunakan data dari state dan computed properties.
   - Menggunakan komponen `Alert` untuk menampilkan notifikasi pesan berhasil.
   - Tabel menampilkan daftar pesanan dengan tombol "Remove" untuk setiap item.
   - Form untuk pengisian data pemesanan seperti nama pemesan, nomor meja, dan metode pembayaran.
7. Style (Gaya):
   Komponen ini menggunakan gaya terkait yang berlaku secara lokal (scoped), tetapi tidak ada aturan gaya yang ditentukan dalam kode ini.

Secara keseluruhan, komponen ini mengelola keranjang belanja, menampilkan daftar pesanan, menghitung total harga dan jumlah item, dan memungkinkan pengguna untuk mengirim pesanan ke backend. Juga, memberikan antarmuka untuk menghapus item dari keranjang belanja dan menampilkan notifikasi pesan berhasil.

4.1.1.4 Tampilan Log in Admin

Didalam tampilan login ini hanya digunakan oleh admin restoran saja, sedangkan untuk costomer tidak memiliki akses untuk melakukan log in, cutomer dapat langsung mengunjungi website.

```
<script setup>
import { ref } from 'vue'

const email = ref('')
const password = ref('')

const auth = useAuthStore()

async function login() {
  const credentials = {
    email: email.value,
    password: password.value
  }
```

```
  const error = await auth.login(credentials)

  // const { data:res } = await useApiFetch('/auth/login', {
  //   method: 'POST',
  //   body: credentials
  // })

  // localStorage.setItem('authToken', res.value.token)

  await navigateTo('/admin')
}

definePageMeta({
  layout: 'login'
})
</script>

<template>
  <div class="container">
    <div class="screen">
      <div class="screen__content">
        <form class="login" id="login-form" @submit.prevent="login">
          <div class="login__field">
            <i class="login__icon fas fa-user"></i>
              <input type="email" class="login__input" id="username" placeholder="User
Email" v-model="email">
          </div>
          <div class="login__field">
            <i class="login__icon fas fa-lock"></i>
                        <input   type="password"   class="login__input"   id="password"
placeholder="Password" v-model="password">
          </div>
          <button class="button login__submit" type="submit">
            <span class="button__text">Login</span>
            <i class="button__icon fas fa-chevron-right"></i>
          </button>
          <p id="error-message" style="color: red; text-align: center;"></p>
        </form>
      </div>
      <div class="screen__background">
        <span class="screen__background-shape screen__background-shape4"></span>
        <span class="screen__background-shape screen__background-shape3"></span>
        <span class="screen__background-shape screen__background-shape2"></span>
        <span class="screen__background-shape screen__background-shape1"></span>
      </div>
    </div>
  </div>
</template>

<style scoped>
```

```css
@import url('https://fonts.googleapis.com/css?family=Raleway:400,700');

* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
  font-family: Raleway, sans-serif;
}

body {
  background: linear-gradient(90deg, #C7C5F4, #776BCC);
}

.container {
  display: flex;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
}

.screen {
  background: linear-gradient(90deg, white, #7C78B8);
  position: relative;
  height: 600px;
  width: 360px;
  box-shadow: 0px 24px #5C5696;
}

.screen__content {
  z-index: 1;
  position: relative;
  height: 100%;
}

.screen__background {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  z-index: 0;
  -webkit-clip-path: inset(0 0 0 0);
  clip-path: inset(0 0 0 0);
}

.screen__background-shape {
  transform: rotate(45deg);
  position: absolute;
}
```

```css
.screen__background-shape1 {
  height: 520px;
  width: 520px;
  background: #FFF;
  top: -50px;
  right: 120px;
  border-radius: 0 72px 0 0;
}

.screen__background-shape2 {
  height: 220px;
  width: 220px;
  background: #6C63AC;
  top: -172px;
  right: 0;
  border-radius: 32px;
}

.screen__background-shape3 {
  height: 540px;
  width: 190px;
  background: linear-gradient(270deg, maroon, #6A679E);
  top: -24px;
  right: 0;
  border-radius: 32px;
}

.screen__background-shape4 {
  height: 400px;
  width: 200px;
  background: #7E7BB9;
  top: 420px;
  right: 50px;
  border-radius: 60px;
}

.login {
  width: 320px;
  padding: 30px;
  padding-top: 156px;
}

.login__field {
  padding: 20px 0px;
  position: relative;
}

.login__icon {
  position: absolute;
  top: 30px;
```

```
  color: #7875B5;
}

.login__input {
  border: none;
  border-bottom: 2px solid pink;
  background: none;
  padding: 10px;
  padding-left: 24px;
  font-weight: 700;
  width: 75%;
  transition: .2s;
}

.login__input:active,
.login__input:focus,
.login__input:hover {
  outline: none;
  border-bottom-color: white;
}

.login__submit {
  background: #fff;
  font-size: 14px;
  margin-top: 30px;
  padding: 16px 20px;
  border-radius: 26px;
  border: 1px solid #D4D3E8;
  text-transform: uppercase;
  font-weight: 700;
  display: flex;
  align-items: center;
  width: 100%;
  color: #4C489D;
  box-shadow: 0px 2px whitesmoke;
  cursor: pointer;
  transition: .2s;
}

.login__submit:active,
.login__submit:focus,
.login__submit:hover {
  border-color: white;
  outline: none;
}

.button__icon {
  font-size: 24px;
  margin-left: auto;
  color: white;
```

```
}

.social-login {
  position: absolute;
  height: 140px;
  width: 160px;
  text-align: center;
  bottom: 0px;
  right: 0px;
  color: #fff;
}

.social-icons {
  display: flex;
  align-items: center;
  justify-content: center;
}

.social-login__icon {
  padding: 20px 10px;
  color: #fff;
  text-decoration: none;
  text-shadow: 0px 8px white;
}

.social-login__icon:hover {
  transform: scale(1.5);
}
</style>
```

Penjelasan source code diatas adalah :

1. State dan Pengelolaan Login:
   email dan password: Ref yang digunakan untuk menyimpan nilai dari input email dan password.
   auth: Menggunakan store untuk manajemen otentikasi (useAuthStore()).
2. Fungsi login:
   - Membuat objek credentials yang berisi email dan password dari ref.
   - Menggunakan fungsi auth.login(credentials) dari store otentikasi untuk melakukan login.
   - Menggunakan fungsi navigateTo('/admin') untuk mengarahkan pengguna ke halaman admin setelah login berhasil.
   - Menampilkan pesan error jika login gagal (komentar yang dinonaktifkan mengandung contoh penggunaan API).
3. Template:
   - Menampilkan formulir login dengan input untuk email dan password.
   - Menggunakan CSS untuk menambahkan gaya visual yang menarik, termasuk animasi latar belakang dan desain formulir yang estetis.
   - Menambahkan pesan error yang akan muncul jika terjadi kesalahan saat login.
4. Style (Gaya):
   - Mendefinisikan gaya untuk tata letak, warna, dan animasi elemen-elemen halaman login.
   - Menggunakan font dari Google Fonts (Raleway) untuk estetika.
5. Latar Belakang dan Bentuk-Bentuk:
   Menerapkan latar belakang linear gradient dan beberapa bentuk geometris yang memperindah tampilan halaman login.

6. Efek Hover dan Fokus:
   Menambahkan efek pada elemen input dan tombol saat di-hover atau di-fokus untuk memberikan umpan balik visual.

```
import { defineStore } from 'pinia'
import { useApiFetch } from '~/composables/useApiFetch'

type User = {
  id: string,
  name: string,
  email: string
}

type Credentials = {
  email: string,
  password: string
}

export const useAuthStore = defineStore('auth', () => {
  const user = ref<User | null>(null)
  const token = ref(null)
  const isLoggedIn = computed(() => !!user.value)

  async function login(credentials: Credentials) {
    const { data:res, error } = await useApiFetch('/auth/login', {
      method: 'POST',
      body: credentials
    })

    const myCookieToken = useCookie('token', { maxAge: 60 * 60 * 24 * 7 })
    myCookieToken.value = res.value.token as string

    // await fetchUser()
    // user.value = res.value.user as User

    return error
  }

  async function fetchUser() {
    // const { data:res } = await useApiFetch('/admin/user', {
    //   headers: {
    //     'Authorization' : 'Bearer '+token.value
    //   }
    // })

    // user.value = res.value.user as User
  }

  return { user, token, login, isLoggedIn, fetchUser }
})
```

Kode tersebut adalah implementasi sebuah store untuk manajemen autentikasi (authentication) menggunakan Pinia, yang merupakan state management library untuk Vue.js. Mari kita jelaskan bagian-bagian utamanya:

1. Import Statements:

```
import { defineStore } from 'pinia'
import { useApiFetch } from '~/composables/useApiFetch'
```

Kode mengimpor fungsi `defineStore` dari pustaka Pinia untuk mendefinisikan store, dan `useApiFetch` dari `useApiFetch` untuk berinteraksi dengan API.

2. Type Definitions:

```
type User = {
  id: string,
  name: string,
  email: string
}

type Credentials = {
  email: string,
  password: string
}
```

Mendefinisikan dua tipe data TypeScript, yaitu `User` yang merepresentasikan informasi pengguna, dan `Credentials` yang merepresentasikan data kredensial (email dan password) yang diperlukan untuk login.

3. Store Definition:

```
export const useAuthStore = defineStore('auth', () => {
```

Mendefinisikan store dengan nama "auth" menggunakan fungsi `defineStore`. Fungsi ini menerima dua argumen: nama store dan fungsi untuk menginisialisasi state dan method-methodnya.

4. State Variables:

```
  const user = ref<User | null>(null)
  const token = ref(null)
```

Menggunakan `ref` untuk membuat reactive variables `user` dan `token` dalam store. `user` menyimpan informasi pengguna setelah login, dan `token` menyimpan token hasil autentikasi.

5. Computed Property:

```
  const isLoggedIn = computed(() => !!user.value)
```

Membuat computed property `isLoggedIn` yang akan memberikan nilai `true` jika `user.value` tidak null, dan `false` sebaliknya. Ini adalah cara yang nyaman untuk memeriksa status login.

6. Login Method:

```
  async function login(credentials: Credentials) {
    const { data:res, error } = await useApiFetch('/auth/login', {
      method: 'POST',
      body: credentials
    })
```

Membuat metode `login` untuk melakukan proses autentikasi. Fungsi ini menggunakan `useApiFetch` untuk melakukan permintaan login ke API. Setelah berhasil login, token disimpan dalam cookie menggunakan `useCookie` dan kemudian panggilan ke `fetchUser` dilakukan untuk mengambil data pengguna.

7. Fetch User Method:

```
async function fetchUser() {
```

Metode ini seharusnya mengambil data pengguna dari API menggunakan token yang tersimpan. Namun, saat ini bagian ini di-comment (dikarenakan tidak aktif) dan perlu diaktifkan sesuai kebutuhan.

8. Return Statement:

```
  return { user, token, login, isLoggedIn, fetchUser }
})
```

Mengembalikan objek yang berisi state dan metode-metode yang didefinisikan di dalam store. Ini memungkinkan komponen Vue untuk mengakses dan menggunakan fungsionalitas autentikasi ini.

LoadUser

```
import { useAuthStore } from '~/stores/useAuthStore';

export default defineNuxtPlugin(async (nuxtApp) => {
  const auth = useAuthStore()
  if (!auth.isLoggedIn) {
    await auth.fetchUser()
  }
})
```

loadUser.ts

Kode tersebut adalah implementasi dari Nuxt plugin yang bertujuan untuk memastikan bahwa pengguna sudah terautentikasi sebelum aplikasi Nuxt dimulai. Mari kita jelaskan bagian-bagian utamanya:

1. Import Statements:

   ```
   import { useAuthStore } from '~/stores/useAuthStore';
   ```

   Mengimpor store `useAuthStore` yang telah dibuat sebelumnya untuk manajemen autentikasi.
2. Define Nuxt Plugin:

   ```
   export default defineNuxtPlugin(async (nuxtApp) => {
   ```

   Menggunakan `defineNuxtPlugin` untuk mendefinisikan plugin Nuxt. Fungsi ini dijalankan ketika aplikasi Nuxt dimulai.
3. Retrieve Auth Store Instance:

   ```
   const auth = useAuthStore()
   ```

   Membuat instance dari store autentikasi (`useAuthStore`) menggunakan fungsi `useAuthStore()`.
4. Check Authentication Status:

   ```
   if (!auth.isLoggedIn) {
     await auth.fetchUser()
   }
   ```

   Memeriksa apakah pengguna belum terautentikasi (`!auth.isLoggedIn`). Jika belum terautentikasi, maka metode `fetchUser` dari store autentikasi dipanggil untuk mengambil data pengguna. Dengan kata lain, plugin ini memastikan bahwa data pengguna sudah diambil dan tersedia sebelum aplikasi Nuxt benar-benar dimulai.
   a. `auth.isLoggedIn`: Menggunakan computed property dari store autentikasi untuk memeriksa status login pengguna.
   b. `auth.fetchUser()`: Metode dari store autentikasi untuk mengambil data pengguna. Sebelumnya, kita telah melihat implementasi metode ini di store autentikasi.
   c. `await`: Menunggu hingga metode `fetchUser` selesai dijalankan sebelum melanjutkan eksekusi.

Dengan cara ini, plugin ini memastikan bahwa sebelum aplikasi dimulai, data pengguna telah diambil (jika pengguna sudah login), sehingga komponen atau halaman yang memerlukan informasi pengguna dapat bekerja dengan benar tanpa harus menunggu proses autentikasi.

useApiFetch.ts

```
import type { UseFetchOptions } from 'nuxt/app'

export function useApiFetch<T>(path: string | (() => string), options: UseFetchOptions<T> = {}) {
  let headers: any = {}
  useFetch('http://103.168.147.245/sanctum/csrf-cookie')
  const token = useCookie('XSRF-TOKEN')
```

```
  if (token.value) {
    headers['X-XSRF-TOKEN'] = token.value as string
  }

  // if (process.server) {
  //   headers = {
  //     ...headers,
  //     ...useRequestHeaders(["referer", "cookie"])
  //   }
  // }

  return useFetch("http://103.168.147.245/api" + path, {
    credentials: 'include',
    watch: false,
    ...options,
    headers: {
      ...headers,
      ...options?.headers
    }
  })
}
```

Kode tersebut merupakan implementasi dari fungsi `useApiFetch` yang digunakan untuk melakukan permintaan ke API dalam aplikasi Nuxt menggunakan Fetch API. Mari kita jelaskan setiap bagian kodenya:

1. Import Statements:

```
import type { UseFetchOptions } from 'nuxt/app'
```

Mengimpor tipe `UseFetchOptions` dari modul `nuxt/app`. Tipe ini menyediakan opsi yang dapat digunakan dalam penggunaan Fetch API pada aplikasi Nuxt.

2. Fungsi `useApiFetch`:

```
export function useApiFetch<T> (path: string | (() => string), options: UseFetchOptions<T> = {}) {
```

Membuat fungsi `useApiFetch` yang mengambil dua parameter, yaitu `path` (string atau fungsi yang mengembalikan string) dan `options` (opsi Fetch API). Tipe data generik `T` digunakan untuk menentukan tipe data dari respons API.

3. Mendapatkan CSRF Token:

```
  let headers: any = {}
  useFetch('http://103.168.147.245/sanctum/csrf-cookie')
  const token = useCookie('XSRF-TOKEN')

  if (token.value) {
    headers['X-XSRF-TOKEN'] = token.value as string
  }
```

- Membuat variabel `headers` yang digunakan untuk menyimpan header permintaan.
- Menggunakan `useFetch` untuk melakukan permintaan CSRF cookie ke endpoint Sanctum (sanctum/csrf-cookie).
- Menggunakan `useCookie` untuk mendapatkan nilai CSRF Token ('XSRF-TOKEN') dari cookie.
- Jika token tersedia, menambahkan header 'X-XSRF-TOKEN' dengan nilai token ke dalam objek `headers`.

4. Pemanggilan useFetch:

```
return useFetch("http://103.168.147.245/api" + path, {
  credentials: 'include',
  watch: false,
  ...options,
  headers: {
    ...headers,
    ...options?.headers
  }
})
```

- Memanggil `useFetch` dengan URL API yang terbentuk dari `path`.
- Menggunakan opsi `credentials: 'include'` untuk menyertakan kredensial dalam permintaan (menggunakan cookie).
- Menyertakan opsi tambahan seperti `watch: false` dan menggabungkannya dengan opsi yang diberikan.
- Menyertakan header yang telah didefinisikan sebelumnya dan menggabungkannya dengan header yang mungkin diberikan dalam opsi.

Dengan menggunakan fungsi `useApiFetch`, aplikasi dapat dengan mudah melakukan permintaan ke API dengan memperhatikan CSRF Token dan kredensial yang diperlukan.

4.1.1.5 Tampilan Menu Admin

Dalam tampilan menu admin berfungsi sebagai pengubah daftar menu makanan dan minuman yang dilakukan oleh admin restoran, seperti menambah menu, mengedit menu, membaca berbagai menu dan menghapus menu (CURD).

[Id].vue

```
<script setup>
import { reactive, onMounted } from 'vue'

const { id } = useRoute().params
const token = useCookie('token')
const menu = reactive({
  name: '',
  price: '',
  type: '',
  desc: '',
  image: null
})

onMounted(async () => {
  const { data:res, error } = await useApiFetch(`/admin/menu/${id}`, {
    headers:{
      'Authorization': 'Bearer '+token.value,
    }
  });

  menu.name = res.value.name
  menu.price = res.value.price
  menu.type = res.value.type
  menu.desc = res.value.desc
})

function onChangeFile(event) {
```

```
  const file = event.target.files[0]
  menu.image = file
}

async function update() {
  let formData = new FormData();
  formData.append('name', menu.name)
  formData.append('price', menu.price)
  formData.append('type', menu.type)
  formData.append('desc', menu.desc)
  if (menu.image != null) {
    formData.append('image', menu.image)
  }
  formData.append('_method', "PUT")

  const {data, error} = await useApiFetch(`/admin/menu/${id}`, {
    headers:{
      'Authorization': 'Bearer '+token.value
    },
    method: 'POST',
    body: formData,
  })

  await navigateTo('/admin/menu')
}

definePageMeta({
  layout: 'admin'
})
</script>

<template>
  <div>
    <div class="card">
      <div class="card-header">
        <div class="card-title">Edit Menu Form</div>
      </div>
      <div class="card-body">
        <form @submit.prevent="update" method="post" enctype="multipart/form-data">
          <div class="mb-3">
            <label for="name" class="form-label">Menu Name</label>
            <input type="text" class="form-control" id="name" v-model="menu.name">
           <!-- <div id="emailHelp" class="form-text">We'll never share your email with
anyone else.</div> -->
          </div>
          <div class="mb-3">
            <label for="price" class="form-label">Price</label>
            <input type="number" class="form-control" id="price" v-model="menu.price">
          </div>
          <div class="mb-3">
```

```
            <label for="type" class="form-label">Menu Type</label>
            <select class="form-select" v-model="menu.type">
              <option selected disabled>Choose Menu Type</option>
              <option value="minuman">Minuman</option>
              <option value="makanan">Makanan</option>
            </select>
          </div>
          <div class="form-floating mb-3">
            <textarea class="form-control" placeholder="Leave a desc menu" id="desc" v-
model="menu.desc"></textarea>
            <label for="desc">Menu Desc</label>
          </div>
          <div class="mb-3">
            <label for="image" class="form-label">Image Menu</label>
            <input type="file" class="form-control" id="image" @change="onChangeFile">
          </div>
          <button type="submit" class="btn btn-primary">Submit</button>
        </form>
      </div>
    </div>
  </div>
</template>

<style scoped></style>
```

Penjelasan source code diatas adalah :

1. Import Library dan Fungsi:
   - Menggunakan reactive dan onMounted dari Vue untuk membuat reaktif state dan mengeksekusi suatu fungsi setelah komponen dipasang.
   - Menggunakan fungsi dari Vue Router (useRoute) dan Cookies (useCookie) untuk mendapatkan parameter dari route dan token dari cookie.
   - Menggunakan useApiFetch untuk berinteraksi dengan API.
   - Membuat objek reaktif menu untuk menyimpan data menu yang akan diedit.
   - javascript
   - Copy code

```
<script setup>
import { reactive, onMounted } from 'vue'

const { id } = useRoute().params
const token = useCookie('token')
const menu = reactive({
  name: '',
  price: '',
  type: '',
  desc: '',
  image: null
})
// ...
</script>
```

2. Fungsi onMounted:
   - Menjalankan fungsi setelah komponen dipasang.
   - Menggunakan useApiFetch untuk mendapatkan data menu dari API berdasarkan id yang diperoleh dari parameter route.
   - Mengisi nilai menu dengan data menu yang diperoleh dari API.

```
// ...
onMounted(async () => {
  const { data: res, error } = await useApiFetch(`/admin/menu/${id}`, {
    headers: {
      'Authorization': 'Bearer ' + token.value,
    }
  });

  menu.name = res.value.name
  menu.price = res.value.price
  menu.type = res.value.type
  menu.desc = res.value.desc
})
// ...
```

3. Fungsi onChangeFile:
   - Dipanggil saat terjadi perubahan pada input file.
   - Mengambil file yang dipilih oleh pengguna dan menyimpannya dalam properti image pada objek menu.
   - Javascript

```
// ...
function onChangeFile(event) {
  const file = event.target.files[0]
  menu.image = file
}
// ...
```

4. Fungsi update:
   - Dipanggil saat formulir disubmit.
   - Membuat objek FormData untuk mengirim data formulir ke server.
   - Menambahkan properti name, price, type, desc, dan image ke objek FormData.
   - Menggunakan metode POST untuk mengirim formulir ke endpoint /admin/menu/${id}.
   - Jika berhasil, mengarahkan pengguna ke halaman admin menu.
   - javascript

```
// ...
async function update() {
  let formData = new FormData();
  formData.append('name', menu.name)
  formData.append('price', menu.price)
  formData.append('type', menu.type)
  formData.append('desc', menu.desc)
  if (menu.image != null) {
    formData.append('image', menu.image)
  }
  formData.append('_method', "PUT")

  const { data, error } = await useApiFetch(`/admin/menu/${id}`, {
    headers: {
      'Authorization': 'Bearer ' + token.value
    },
    method: 'POST',
    body: formData,
  })

  await navigateTo('/admin/menu')
}
// ...
```

5. Fungsi definePageMeta:
   - Mengatur layout halaman menjadi 'admin'.

```
// ...
definePageMeta({
  layout: 'admin'
})
// ...
```

6. Template:
   - Menampilkan formulir untuk mengedit menu dengan input untuk nama, harga, jenis, deskripsi, dan gambar.
   - Menggunakan @change untuk memanggil fungsi onChangeFile saat input gambar berubah.
   - Menggunakan @submit.prevent untuk mencegah perilaku bawaan formulir dan memanggil fungsi update saat formulir disubmit.

```
<template>
  <div>
    <!-- ... -->
    <form @submit.prevent="update" method="post" enctype="multipart/for
      <!-- ... -->
    </form>
    <!-- ... -->
  </div>
</template>
```

Create.vue

```
<script setup>
import { reactive } from 'vue'

const token = useCookie('token')
const menu = reactive({
  name: '',
  price: '',
  type: '',
  desc: '',
  image: null
```

```
})

function onChangeFile(event) {
  const file = event.target.files[0]
  menu.image = file
}

async function store() {
  const formData = new FormData();
  formData.append('name', menu.name)
  formData.append('price', menu.price)
  formData.append('type', menu.type)
  formData.append('desc', menu.desc)
  formData.append('image', menu.image)
  const {data, error} = await useApiFetch('/admin/menu', {
    headers:{
      'Authorization': 'Bearer '+token.value,
    },
    method: 'POST',
    body: formData,
  })

  await navigateTo('/admin/menu')
}

definePageMeta({
  layout: 'admin'
})
</script>

<template>
  <div>
    <div class="card">
      <div class="card-header">
        <div class="card-title">Create Menu Form</div>
      </div>
      <div class="card-body">
        <form @submit.prevent="store" method="post" enctype="multipart/form-data">
          <div class="mb-3">
            <label for="name" class="form-label">Menu Name</label>
            <input type="text" class="form-control" id="name" v-model="menu.name">
           <!-- <div id="emailHelp" class="form-text">We'll never share your email with
anyone else.</div> -->
          </div>
          <div class="mb-3">
            <label for="price" class="form-label">Price</label>
            <input type="number" class="form-control" id="price" v-model="menu.price">
          </div>
          <div class="mb-3">
            <label for="type" class="form-label">Menu Type</label>
```

```
            <select class="form-select" v-model="menu.type">
              <option selected>Choose Menu Type</option>
              <option value="minuman">Minuman</option>
              <option value="makanan">Makanan</option>
            </select>
          </div>
          <div class="form-floating mb-3">
            <textarea class="form-control" placeholder="Leave a desc menu" id="desc" v-
model="menu.desc"></textarea>
            <label for="desc">Menu Desc</label>
          </div>
          <div class="mb-3">
            <label for="image" class="form-label">Image Menu</label>
            <input type="file" class="form-control" id="image" @change="onChangeFile">
          </div>
          <button type="submit" class="btn btn-primary">Submit</button>
        </form>
      </div>
    </div>
  </div>
</template>

<style scoped></style>
```

Penjelasan source code diatas adalah :

1. Import Library dan Fungsi:
   - Menggunakan reactive dari Vue untuk membuat reaktif state.
   - Menggunakan fungsi dari Vue Cookies (useCookie) untuk mendapatkan token dari cookie.
   - Menggunakan useApiFetch untuk berinteraksi dengan API.

```
<script setup>
import { reactive } from 'vue'

const token = useCookie('token')
const menu = reactive({
  name: '',
  price: '',
  type: '',
  desc: '',
  image: null
})
// ...
</script>
```

2. Fungsi onChangeFile:
   - Dipanggil saat terjadi perubahan pada input file.
   - Mengambil file yang dipilih oleh pengguna dan menyimpannya dalam properti image pada objek menu.

```
// ...
function onChangeFile(event) {
  const file = event.target.files[0]
  menu.image = file
}
// ...
```

3. Fungsi store:
   - Dipanggil saat formulir disubmit.
   - Membuat objek FormData untuk mengirim data formulir ke server.
   - Menambahkan properti name, price, type, desc, dan image ke objek FormData.
   - Menggunakan metode POST untuk mengirim formulir ke endpoint /admin/menu.
   - Jika berhasil, mengarahkan pengguna ke halaman admin menu.

```
// ...
async function store() {
  const formData = new FormData();
  formData.append('name', menu.name)
  formData.append('price', menu.price)
  formData.append('type', menu.type)
  formData.append('desc', menu.desc)
  formData.append('image', menu.image)
  const { data, error } = await useApiFetch('/admin/menu', {
    headers: {
      'Authorization': 'Bearer ' + token.value,
    },
    method: 'POST',
    body: formData,
  })

  await navigateTo('/admin/menu')
}
// ...
```

4. Fungsi definePageMeta:
   - Mengatur layout halaman menjadi 'admin'.

```
// ...
definePageMeta({
  layout: 'admin'
})
// ...
```

5. Template:
   - Menampilkan formulir untuk membuat menu baru dengan input untuk nama, harga, jenis, deskripsi, dan gambar.
   - Menggunakan @change untuk memanggil fungsi onChangeFile saat input gambar berubah.
   - Menggunakan @submit.prevent untuk mencegah perilaku bawaan formulir dan memanggil fungsi store saat formulir disubmit.

```
<template>
  <div>
    <!-- ... -->
    <form @submit.prevent="store" method="post" enctype="multipart/for
      <!-- ... -->
    </form>
    <!-- ... -->
  </div>
</template>
```

Index.vue

```
<script setup>
const token = useCookie('token')

const { data:res } = await useApiFetch('/admin/menu', {
    headers: {
      'Authorization': 'Bearer '+token.value
    }
  })

const menus = res.value

async function deleteMenu(event, id) {
  if (confirm('Are You Sure Want To Delete This ?')) {
    await useApiFetch(`/admin/menu/${id}`, {
      method: 'DELETE',
      headers: {
        'Authorization': 'Bearer '+token.value
      }
    })

    location.reload()
  }
}

definePageMeta({
  layout: 'admin'
})
</script>

<template>
  <div>
    <div class="card">
      <div class="card-header">
        <div class="card-title">
          <div class="row">
            <div class="col-md-6">
              Menu List
            </div>
            <div class="col-md-6">
```

```
                                <NuxtLink    class="btn    btn-primary    float-end"
to="/admin/menu/create">Create</NuxtLink>
            </div>
          </div>
        </div>
      </div>
      <div class="card-body">
        <div class="table-responsive">
          <table class="table">
            <thead>
              <tr>
                <th scope="col">Image</th>
                <th scope="col">Name</th>
                <th scope="col">Desc</th>
                <th scope="col">Price</th>
                <th scope="col">Type</th>
                <th scope="col">Action</th>
              </tr>
            </thead>
            <tbody>
              <tr v-for="(menu, index) in menus" :key="'menu-'+index">
                <td>
                  <img :src="menu.image" alt="image-menu" height="100px">
                </td>
                <td>{{ menu.name }}</td>
                <td>{{ menu.desc }}</td>
                <td>
                  <Rupiah :angka="menu.price"/>
                </td>
                <td>{{ menu.type }}</td>
                <td>
                                    <NuxtLink   class="btn   btn-warning   text-white"
:to="'/admin/menu/'+menu._id">Edit</NuxtLink>
                                <button   type="button"   class="btn   btn-danger   mx-3"
@click="deleteMenu($event, menu._id)">Delete</button>
                </td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </div>
  </div>
</template>

<style scoped></style>
```

Penjelasan source code diatas adalah:

a. Setup Script:
   - Menggunakan useCookie untuk mendapatkan nilai token dari cookie.

- Menggunakan useApiFetch untuk mendapatkan data menu dari endpoint /admin/menu.

```
<script setup>
const token = useCookie('token')

const { data:res } = await useApiFetch('/admin/menu', {
    headers: {
      'Authorization': 'Bearer '+token.value
    }
  })

const menus = res.value

async function deleteMenu(event, id) {
  if (confirm('Are You Sure Want To Delete This ?')) {
    await useApiFetch(`/admin/menu/${id}`, {
      method: 'DELETE',
      headers: {
        'Authorization': 'Bearer '+token.value
      }
    })

    location.reload()
  }
}

definePageMeta({
  layout: 'admin'
})
</script>
```

b. Template:
   - Menampilkan daftar menu dalam bentuk tabel dengan kolom-kolom seperti gambar, nama, deskripsi, harga, jenis, dan aksi.
   - Menggunakan v-for untuk mengulang setiap menu dalam array menus.
   - Setiap baris tabel memiliki tombol "Edit" yang mengarahkan ke halaman edit menu dan tombol "Delete" untuk menghapus menu tersebut.
   - Ketika tombol "Delete" diklik, fungsi deleteMenu dipanggil, yang mengkonfirmasi pengguna terlebih dahulu dan kemudian menghapus menu jika pengguna setuju.

```
<template>
  <div>
    <div class="card">
      <div class="card-header">
        <div class="card-title">
          <div class="row">
            <div class="col-md-6">
              Menu List
            </div>
            <div class="col-md-6">
              <NuxtLink class="btn btn-primary float-end" to="/admin/me
            </div>
          </div>
        </div>
      </div>
```

```
    </div>
    <div class="card-body">
      <div class="table-responsive">
        <table class="table">
          <thead>
            <tr>
              <th scope="col">Image</th>
              <th scope="col">Name</th>
              <th scope="col">Desc</th>
              <th scope="col">Price</th>
              <th scope="col">Type</th>
              <th scope="col">Action</th>
            </tr>
          </thead>
          <tbody>
```

```
              <tr v-for="(menu, index) in menus" :key="'menu-'+index">
                <!-- ... -->
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </div>
  </div>
</template>
```

c. Style:
Tidak ada gaya khusus yang ditambahkan dalam konteks ini.

```
<style scoped></style>
```

4.1.1.6 Tampilan Order Today

Dalam tampilan ordey today pada admin menampilkan hasil dari pesanan cutomer yang di pesan melalui tampilan order cutomer.

[id].vue

```
<script setup>
const { id } = useRoute().params
const token = useCookie('token')
const {data:res} = await useApiFetch(`/admin/order/${id}`, {
    headers: {
      'Authorization': 'Bearer '+token.value
    }
  })

const transaction = res.value

definePageMeta({
  layout: 'admin'
})
</script>
```

```
<template>
  <div>
    <div class="row">
      <div class="col-md-3">
        <div class="card" style="width: 18rem;">
          <div class="card-header">
            <div class="card-title">
                Pesananan No Meja <span class="float-end">{{ transaction.table_number
}}</span>
            </div>
          </div>
          <ul class="list-group list-group-flush">
            <li class="list-group-item" v-for="(detail, index) in transaction.details"
:key="index">{{ detail.menu.name }} <span class="float-end"><strong>{{ detail.qty
}}</strong></span></li>
          </ul>
          <div class="card-footer">
            <NuxtLink class="btn btn-danger" to="/admin/order/today">Back</NuxtLink>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

<style scoped></style>
```

Penjelasan source code diatas adalah:

a. Setup Script:
   - Menggunakan useRoute().params untuk mendapatkan parameter dari URL, khususnya id dari pesanan yang akan ditampilkan.
   - Menggunakan useCookie('token') untuk mendapatkan nilai token dari cookie.
   - Menggunakan useApiFetch untuk mendapatkan data pesanan dengan mengirimkan request ke endpoint /admin/order/${id}, yang membutuhkan token untuk otorisasi.

```
<script setup>
const { id } = useRoute().params
const token = useCookie('token')
const {data:res} = await useApiFetch(`/admin/order/${id}`, {
    headers: {
      'Authorization': 'Bearer '+token.value
    }
  })

const transaction = res.value

definePageMeta({
  layout: 'admin'
})
</script>
```

b. Template:
   - Menampilkan informasi pesanan dalam sebuah kartu (card) dengan menggunakan Bootstrap CSS.
   - Bagian header kartu menampilkan nomor meja pesanan.

- List item di dalam kartu menampilkan detail setiap menu yang ada dalam pesanan beserta jumlah pesanan (quantity).
- Bagian footer kartu memiliki tombol "Back" yang mengarahkan kembali ke halaman daftar pesanan hari ini.

```
<template>
  <div>
    <div class="row">
      <div class="col-md-3">
        <div class="card" style="width: 18rem;">
          <div class="card-header">
            <div class="card-title">
              Pesananan No Meja <span class="float-end">{{ transaction.
            </div>
          </div>
          <ul class="list-group list-group-flush">
            <li class="list-group-item" v-for="(detail, index) in trans
          </ul>
          <div class="card-footer">
            <NuxtLink class="btn btn-danger" to="/admin/order/today">Ba
          </div>
        </div>
      </div>
    </div>
  </div>
</template>
```

c. Style:

Tidak ada gaya khusus yang ditambahkan dalam konteks ini.

```
<style scoped></style>
```

Index.vue

```
<script setup>
import { ref, watch } from 'vue';
const token = useCookie('token')
const { data:res } = await useApiFetch('/admin/order', {
  headers: {
    'Authorization': 'Bearer '+token.value
  }
})


const orders = res.value

const btnLabel = ref('Proces')
watch(btnLabel, (newStatus) => {
  btnLabel.value = newStatus
})

async function procesOrder(event, id) {
  const {data:res} = await useApiFetch(`/admin/order/${id}`, {
    headers: {
      'Authorization': 'Bearer '+token.value
    }
  })

const transaction = res.value
```

```
const status = changeStatus(transaction.status)
btnLabel.value = status

const {data:resUpdate} = await useApiFetch(`/admin/order/${id}`, {
    method: 'PUT',
    body: {
      status
    },
    headers: {
      'Authorization': 'Bearer '+token.value
    }
  })

  location.reload()
}

function changeStatus(currentStatus) {
  let newStatus = ''
  switch (currentStatus) {
    case 'waiting':
      newStatus = 'proses'
      break;
    case 'proses':
      newStatus = 'payment'
      break;

    default:
    newStatus = 'finish'
      break;
  }

  return newStatus
}

definePageMeta({
  layout: 'admin'
})
</script>

<template>
  <div>
    <div class="row">
      <div class="col-md-3 me-3" v-for="(order, index) in orders" :key="index">
        <div class="card" style="width: 18rem;">
          <div class="card-body">
            <ul class="list-group list-group-flush">
             <li class="list-group-item">Table Number <span class="float-end"><strong>{{
order.table_number }}</strong></span></li>
```

```
            <li class="list-group-item">Order Name <span class="float-end"><strong>{{
order.order_name }}</strong></span></li>
                <li class="list-group-item">Status <span class="float-end"><strong>{{
order.status }}</strong></span></li>
                    <li class="list-group-item">Total Price <span class="float-
end"><strong><Rupiah :angka="order.grand_total"/></strong></span></li>
          </ul>
        </div>
        <div class="card-footer">
          <button type="button" class="btn btn-success" @click="procesOrder($event,
order._id)" v-if="order.status != 'finish'">{{ btnLabel }}</button>
                        <NuxtLink class="btn btn-info text-white float-end"
:to="'/admin/order/today/'+order._id">Detail</NuxtLink>
        </div>
      </div>
    </div>
  </div>
</template>

<style scoped></style>
```

Penjelasan source code diatas adalah:

1. Setup Script:
   - Menggunakan useCookie('token') untuk mendapatkan nilai token dari cookie.
   - Menggunakan useApiFetch untuk mendapatkan data pesanan dengan mengirimkan request ke endpoint /admin/order dengan token untuk otorisasi.
   - Membuat variabel orders untuk menyimpan daftar pesanan.
   - Membuat variabel btnLabel sebagai referensi (ref) untuk label tombol dan menggunakan watch untuk memantau perubahan pada label tombol.

```
<script setup>
import { ref, watch } from 'vue';
const token = useCookie('token')
const { data:res } = await useApiFetch('/admin/order', {
  headers: {
    'Authorization': 'Bearer '+token.value
  }
})

const orders = res.value

const btnLabel = ref('Proces')
watch(btnLabel, (newStatus) => {
  btnLabel.value = newStatus
})

// ... (fungsi-fungsi lainnya)
</script>
```

2. Fungsi procesOrder:
   - Memproses pesanan berdasarkan ID pesanan tertentu.
   - Mengirimkan request ke endpoint /admin/order/${id} untuk mendapatkan data pesanan terkini.
   - Mengubah status pesanan menggunakan fungsi changeStatus.
   - Mengirimkan request ke endpoint /admin/order/${id} dengan metode PUT untuk mengupdate status pesanan.
   - Melakukan reload halaman setelah proses selesai.

```
async function procesOrder(event, id) {
  const {data:res} = await useApiFetch(`/admin/order/${id}`, {
    headers: {
      'Authorization': 'Bearer '+token.value
    }
  })

  const transaction = res.value

  const status = changeStatus(transaction.status)
  btnLabel.value = status

  const {data:resUpdate} = await useApiFetch(`/admin/order/${id}`, {
    method: 'PUT',
    body: {
      status
    },
    headers: {
      'Authorization': 'Bearer '+token.value
    }
  })

  location.reload()
}
```

3. Fungsi changeStatus:
   Menerima status pesanan saat ini dan mengembalikan status berikutnya berdasarkan logika pengubahan status (waiting → proses → payment → finish).

```
function changeStatus(currentStatus) {
  let newStatus = ''
  switch (currentStatus) {
    case 'waiting':
      newStatus = 'proses'
      break;
    case 'proses':
      newStatus = 'payment'
      break;
    default:
      newStatus = 'finish'
      break;
  }

  return newStatus
}
```

4. Template:
   • Menampilkan setiap pesanan dalam bentuk kartu (card) menggunakan Bootstrap CSS.
   • Informasi-informasi pesanan seperti nomor meja, nama pemesan, status, dan total harga ditampilkan.
   • Terdapat tombol "Proces" (dengan label dinamis) untuk mengubah status pesanan.
   • Tombol "Detail" untuk melihat detail pesanan lebih lanjut.

```
<template>
  <div>
    <div class="row">
      <div class="col-md-3 me-3" v-for="(order, index) in orders" :key=
        <div class="card" style="width: 18rem;">
          <div class="card-body">
            <ul class="list-group list-group-flush">
              <li class="list-group-item">Table Number <span class="fl
              <li class="list-group-item">Order Name <span class="float
              <li class="list-group-item">Status <span class="float-end
              <li class="list-group-item">Total Price <span class="flo
            </ul>
          </div>
          <div class="card-footer">
            <button type="button" class="btn btn-success" @click="proce
            <NuxtLink class="btn btn-info text-white float-end" :to="'/
          </div>
        </div>
      </div>
    </div>
  </div>
</template>
```

5. Style:

```
<style scoped></style>
```

4.1.1.7 Tampilan Order History

Dalam tampilan order history pada admin menampilkan hasil riwayat pembelian yang dilakukan oleh cutomer.

[id].vue

```
    <script setup>

const { id } = useRoute().params
const token = useCookie('token')
const {data:res} = await useApiFetch(`/admin/order/${id}`, {
    headers: {
      'Authorization': 'Bearer '+token.value
    }
  })

const transaction = res.value

definePageMeta({
  layout: 'admin'
})
</script>

<template>
  <div>
    <div class="row">
```

```
      <div class="col-md-3">
        <div class="card" style="width: 18rem;">
          <div class="card-header">
            <div class="card-title">
                Pesananan No Meja <span class="float-end">{{ transaction.table_number
}}</span>
            </div>
          </div>
          <ul class="list-group list-group-flush">
            <li class="list-group-item" v-for="(detail, index) in transaction.details"
:key="index">{{ detail.menu.name }} <span class="float-end"><strong>{{ detail.qty
}}</strong></span></li>
          </ul>
          <div class="card-footer">
            <NuxtLink class="btn btn-danger" to="/admin/order/history">Back</NuxtLink>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

<style scoped></style>
```

Penjelasan source code diatas adalah:

1. Setup Script:
   - Menggunakan useRoute().params untuk mendapatkan parameter dari URL, khususnya nilai id.
   - Menggunakan useCookie('token') untuk mendapatkan nilai token dari cookie.
   - Menggunakan useApiFetch untuk mengambil data pesanan dari endpoint /admin/order/${id} dengan menyertakan token untuk otorisasi.
   - Menyimpan data pesanan dalam variabel transaction.

```
<script setup>
const { id } = useRoute().params
const token = useCookie('token')
const {data:res} = await useApiFetch(`/admin/order/${id}`, {
    headers: {
      'Authorization': 'Bearer '+token.value
    }
  })

const transaction = res.value

definePageMeta({
  layout: 'admin'
})
</script>
```

2. Template:
   - Menampilkan informasi pesanan dalam bentuk kartu (card) menggunakan Bootstrap CSS.
   - Menampilkan nomor meja pesanan (transaction.table_number) dalam judul kartu.
   - Menampilkan daftar detail pesanan (menu dan jumlah) dalam bentuk daftar (ul).
   - Menambahkan tombol "Back" dengan menggunakan NuxtLink yang mengarahkan ke halaman /admin/order/history.

```
<template>
  <div>
    <div class="row">
      <div class="col-md-3">
        <div class="card" style="width: 18rem;">
          <div class="card-header">
            <div class="card-title">
              Pesananan No Meja <span class="float-end">{{ transaction.
            </div>
          </div>
          <ul class="list-group list-group-flush">
            <li class="list-group-item" v-for="(detail, index) in trans
          </ul>
          <div class="card-footer">
            <NuxtLink class="btn btn-danger" to="/admin/order/history">
          </div>
        </div>
      </div>
    </div>
  </div>
</template>
```

3. Style:

```
<style scoped></style>
```

Index.vue

```
<script setup>
const token = useCookie('token')
const { data:res } = await useApiFetch('/admin/order/history', {
  headers: {
    'Authorization': 'Bearer '+token.value
  }
})

const orders = res.value

definePageMeta({
  layout: 'admin'
})
</script>

<template>
  <div>
    <div class="row">
      <div class="col-md-3 me-3" v-for="(order, index) in orders" :key="index">
        <div class="card mb-3" style="width: 18rem;">
          <div class="card-body">
            <ul class="list-group list-group-flush">
                        <li  class="list-group-item">Order  Date  <span  class="float-
end"><strong><DateFormat :angka="order.created_at"/></strong></span></li>
              <li class="list-group-item">Table Number <span class="float-end"><strong>{{
order.table_number }}</strong></span></li>
                <li class="list-group-item">Order Name <span class="float-end"><strong>{{
order.order_name }}</strong></span></li>
```

```
                <li class="list-group-item">Status <span class="float-end"><strong>{{
order.status }}</strong></span></li>
                    <li class="list-group-item">Total Price <span class="float-
end"><strong><Rupiah :angka="order.grand_total"/></strong></span></li>
            </ul>
          </div>
          <div class="card-footer">
                        <NuxtLink class="btn btn-info text-white float-end"
:to="'/admin/order/history/'+order._id">Detail</NuxtLink>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

<style scoped></style>
```

Penjelasan source code diatas adalah:

a. Setup Script:

- useCookie('token'): Mendapatkan nilai token dari cookie untuk autentikasi.
- useApiFetch('/admin/order/history', ...): Mengambil data riwayat pesanan dari endpoint /admin/order/history dengan menyertakan token untuk otorisasi.
- Data pesanan disimpan dalam variabel orders.
- definePageMeta: Mengatur meta informasi halaman, dalam hal ini, layout halaman diatur sebagai 'admin'.

```
<script setup>
const token = useCookie('token')
const { data:res } = await useApiFetch('/admin/order/history', {
  headers: {
    'Authorization': 'Bearer '+token.value
  }
})

const orders = res.value

definePageMeta({
  layout: 'admin'
})
</script>
```

b. Template:

- Menampilkan setiap pesanan dalam bentuk kartu (card) menggunakan Bootstrap CSS.
- Menampilkan informasi seperti tanggal pesanan (order.created_at), nomor meja, nama pesanan, status pesanan, dan total harga dalam bentuk daftar (ul).
- Menambahkan tombol "Detail" dengan menggunakan NuxtLink yang mengarahkan ke halaman detail pesanan.

```
<template>
  <div>
    <div class="row">
      <div class="col-md-3 me-3" v-for="(order, index) in orders" :key="index">
        <div class="card mb-3" style="width: 18rem;">
          <div class="card-body">
            <ul class="list-group list-group-flush">
```

```
                    <li class="list-group-item">Order Date <span class="float-
end"><strong><DateFormat :angka="order.created_at"/></strong></span></li>
            <li class="list-group-item">Table Number <span class="float-end"><strong>{{
order.table_number }}</strong></span></li>
             <li class="list-group-item">Order Name <span class="float-end"><strong>{{
order.order_name }}</strong></span></li>
                <li class="list-group-item">Status <span class="float-end"><strong>{{
order.status }}</strong></span></li>
                    <li class="list-group-item">Total Price <span class="float-
end"><strong><Rupiah :angka="order.grand_total"/></strong></span></li>
            </ul>
          </div>
          <div class="card-footer">
                        <NuxtLink class="btn btn-info text-white float-end"
:to="'/admin/order/history/'+order._id">Detail</NuxtLink>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>
```

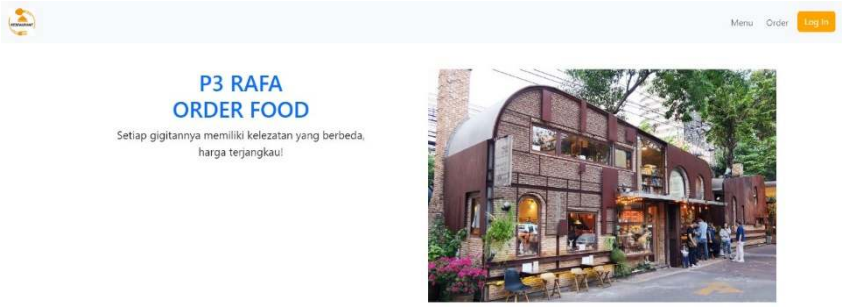
c. Style:

```
<style scoped></style>
```

CHAPTER 5
DEBUGGING APLIKASI PEMESANAN MAKANAN

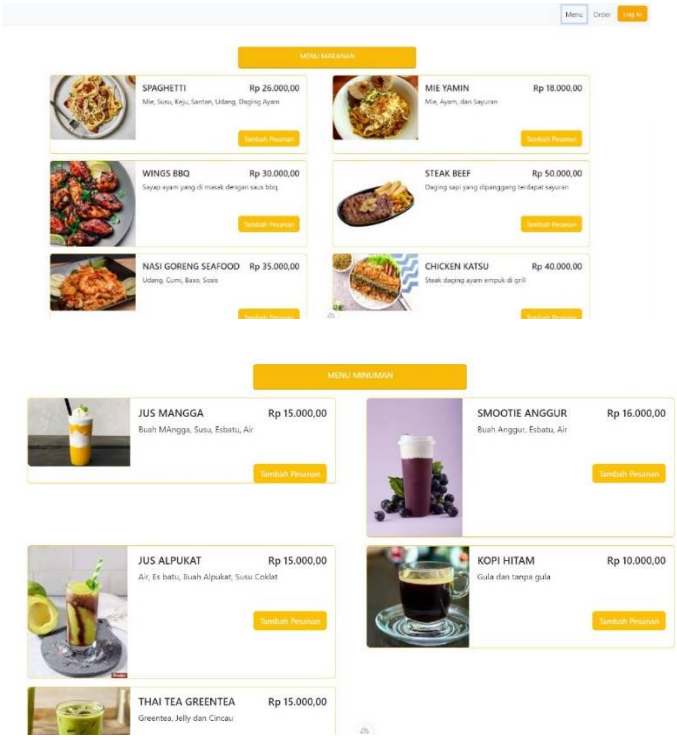
5.1 Hasil Tampilan Dashboard

Terlihat tampilan dari dashboard restoran yaitu tampilan utama dari restoran P3 Rafa Order Food, memiliki tiga fitur yang terletak pada bagian kanan atas diantaranya menu, order dan login. Pada fitur menu dan order digunakan oleh customer untuk memilih dan memesan makanan yang diinginkan, sedangkan fitur login hanya di gunakan untuk admin dari pihak restoran saja.



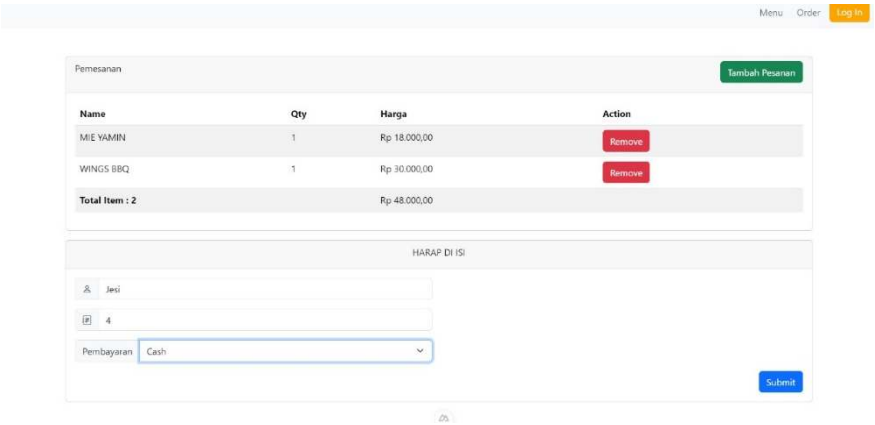
5.2 Hasil Tampilan Menu Customer

Terlihat tampilan dari halaman menu customer yaitu terdapat berbagai pilihan makanan dan minuman. Dimana customer dapat memilih makanan dan minuman dengan mengklik button tambah pesanan pada item menu tersebut yang akan masuk ke tampilan order customer selanjutnya.



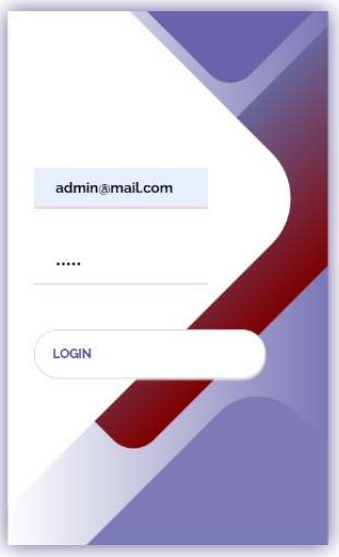
5.3 Hasil Tampilan Order Customer

Terlihat tampilan dari halaman order customer yaitu terdapat fitur list pesanan makanan dan minuman miliki customer. Dimana customer setelah memilih menu yang diinginkan selanjutnya customer mengunjungi fitur order yang isinya menu yang di pilih customer dan di bawah nya terdapat beberapa data seperti nama pembeli dan nomor meja dan milih salah satu metode pembayaran yang harus di lengkapi oleh cutomer.



5.4 Hasil Tampilan Login Admin Restoran

Terlihat tampilan dari login yaitu dimana admin restoran yang hanya memiliki akun tersebut dan dapat memasukan akun tersebut dan masuk kehalaman admin yang berikan menu dan order.

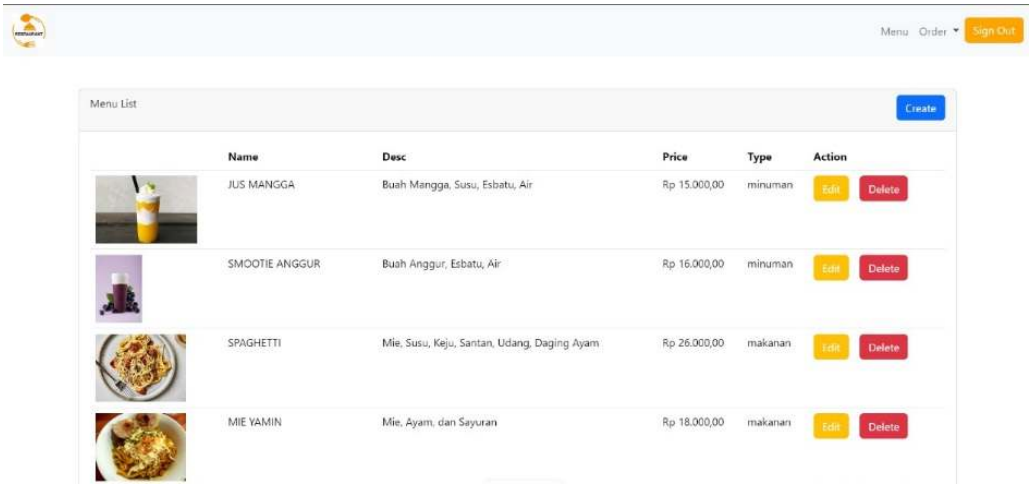


5.5 Hasil Tampilan Dashboard Admin Restoran

Terlihat tampilan dari halaman dashboard admin yaitu terdapat tampilan menu, today order dan all order. Pada menu terlihat banyak Jumlah menu yang tersedia yaitu 12 item, untuk today order berisikan Jumlah total pesanan cutomer hari ini, sedangkan all order yaitu Jumlah semua riwayat pembelian customer yang memesan menu di restoran.

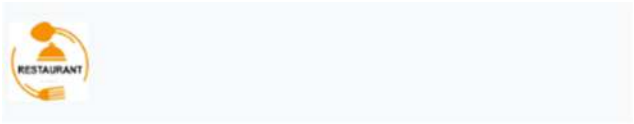
5.6 Hasil Tampilan Menu Admin

Terlihat tampilan dari halaman menu admin yaitu terdapat tampilan menu. Dimana admin restoran dapat melakukan perubahan menu seperti menambah menu, mengedit menu dan menghapus menu (system CRUD).



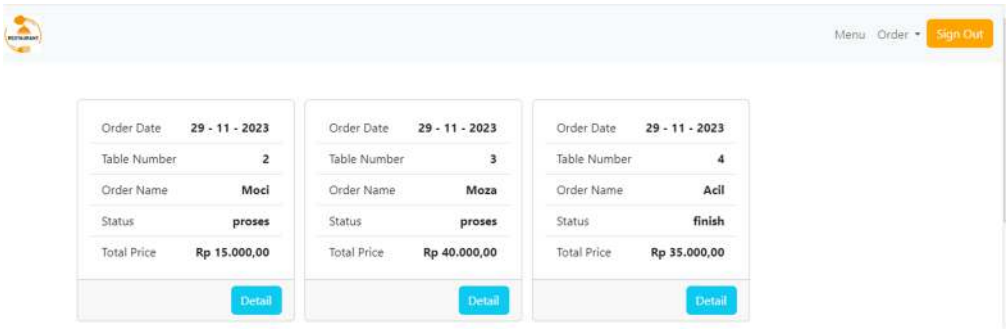
5.7 Hasil Tampilan Order Admin Bagian Today

Terlihat tampilan dari halaman order admin yaitu terdapat tampilan order today. Dimana admin restoran dapat melihat pesanan cutomer yang masuk pada hari ini. Pada button process jika di tekan akan berlanjut terus hingga step berikutnya yaitu process, waiting, payment dan terakhir finish.



5.8 Hasil Tampilan Order Admin Bagian History

Terlihat tampilan dari halaman admin order yaitu terdapat tampilan Hitory . Dimana admin restoran dapat melihat keselurusan riwayat pesanan menu yang dilakukan oleh cutomer.



SISTEM OPERASIONAL PEMESANAN MAKANAN DENGAN QR *WAITING LINE METHOD*

Restoran saat ini sudah mulai mengikuti perkembangan teknologi. Salah satu nya dalam proses pemesanan makanan yang awalnya masih menggunakan buku menu sebagai media pemilihan serta pemesanan makanan. Penulis membangun aplikasi pemesanan makanan berbasis website bertujuan untuk mempermudah sistem pelayanan pemesanan makanan. Dengan adanya sistem ini customer tidak lagi menunggu lama buku menu dan apabila restoran sedang ramai dari pihak restoran tidak bingung dalam sistem antrian dari pelanggan atau customer. Dengan adanya sistem ini pihak restoran lebih efisien dalam menyediakan menu dan apabila ada perubahan pada menu tidak lagi harus mencetak buku menu sebanyak mungkin.

Sistem ini memiliki beberapa keunggulan yaitu:

1. Dapat menghemat waktu dalam melayani customer.
2. Data pelanggan lebih rapih, jelas dan terstruktur dari banyaknya pesanan customer yang masuk.
3. Pemilik restoran dapat melihat langsung bagaimana pemasukan pelanggan setiap harinya.

Aplikasi ini masih memiliki kekurangan namun dapat membawa perubahan dan membantu dalam pemesanan menu makanan yang ada pada restoran.

Ratu Sukmakomala
Isnaenti Nur Latifah
Rd. Nuraini Siti Fathonah

bukupedia