**2. Pemilihan Arsitektur Aplikasi**

**2.1. Tujuan Pemilihan Arsitektur**

* Stabilitas: Memberikan struktur yang dapat diandalkan untuk memastikan aplikasi berjalan dengan lancar.
* Skalabilitas: Memungkinkan aplikasi untuk berkembang sesuai kebutuhan, baik dari sisi pengguna maupun fitur.
* Kesesuaian Kebutuhan: Memastikan arsitektur dapat mengakomodasi semua fitur fungsional (seperti CRUD, login multi-role) dan non-fungsional (seperti performa dan keamanan).

**2.2. Pilihan Arsitektur**

**2.2.1. Model-View-Controller (MVC)**

* **Deskripsi**: Arsitektur MVC membagi aplikasi menjadi tiga komponen utama: Model (mengelola data dan logika bisnis), View (menyajikan data kepada pengguna), dan Controller (menangani input dan memperbarui model).
* **Keuntungan**: Pemisahan yang jelas antara logika bisnis, tampilan, dan kontrol. Memudahkan pemeliharaan dan pengembangan fitur baru.
* **Kekurangan**:
  + Kompleksitas: Struktur aplikasi bisa menjadi rumit seiring bertambahnya fitur jika tidak diorganisasi dengan baik.
  + Kurva Belajar: Pemula mungkin perlu waktu untuk memahami pola MVC sepenuhnya.

**2.2.2. Microservices**

* **Deskripsi**: Pendekatan arsitektur di mana aplikasi dibagi menjadi serangkaian layanan kecil dan independen yang berkomunikasi melalui API.
* **Keuntungan**: Skalabilitas tinggi, memungkinkan pengembangan dan penyebaran terpisah, fleksibel dalam pemilihan teknologi.
* **Kekurangan**: Terlalu kompleks untuk proyek dengan tim kecil atau fitur terbatas.

**2.2.3. Single Page Application (SPA)**

* **Deskripsi**: Aplikasi web yang memuat satu halaman HTML dan mengupdate konten secara dinamis dengan JavaScript dan terhubung ke database.
* **Keuntungan**: Pengalaman pengguna yang lebih mulus dan interaktif, pengurangan waktu muat halaman.
* **Kekurangan**: SEO bisa menjadi tantangan, lebih kompleks dalam hal pengelolaan status dan rute.

**2.3. Rekomendasi Arsitektur**

* **Rekomendasi**: Berdasarkan analisis kebutuhan, proyek ini direkomendasikan untuk menggunakan arsitektur Model-View-Controller (MVC).
  + Alasan Spesifik:
    - Laravel, framework yang digunakan dalam proyek ini, dibangun berdasarkan pola MVC, sehingga mendukung proses pengembangan yang efisien.
    - Arsitektur ini memastikan pemisahan tanggung jawab antara data (Model), logika aplikasi (Controller), dan tampilan (View), membuat sistem lebih mudah dipelihara dan dikembangkan.
    - Dukungan komunitas Laravel untuk MVC memungkinkan pengembangan dan penyelesaian masalah secara cepat.

**2. Desain Struktur Database Awal**

**2.1. Tujuan Desain Struktur Database**

* Memastikan penyimpanan data yang efisien dan terorganisir untuk mendukung kebutuhan aplikasi.
* Mempermudah akses dan pengelolaan data untuk operasional aplikasi.
* Memberikan fleksibilitas untuk penambahan fitur di masa depan tanpa perlu perubahan besar pada struktur data.

**2.2. Desain Struktur Database**

**2.2.1. Diagram Entity-Relationship (ER)**

* **Deskripsi**: Diagram ER menggambarkan entitas dalam sistem, atribut mereka, dan hubungan antar entitas.
* **Diagram**: [Sertakan diagram ER yang relevan]

**2.2.2. Tabel-Tabel Utama**

* **Tabel Pengguna**
  + **Kolom**: UserID (PK), Username, PasswordHash, Email, CreatedAt, UpdatedAt
* **Tabel Produk**
  + **Kolom**: ProductID (PK), ProductName, Description, Price, StockQuantity, CreatedAt, UpdatedAt
* **Tabel Pesanan**
  + **Kolom**: OrderID (PK), UserID (FK), OrderDate, TotalAmount, Status
* **Tabel DetailPesanan**
  + **Kolom**: OrderDetailID (PK), OrderID (FK), ProductID (FK), Quantity, Price

**2.3. Rekomendasi Struktur Database**

* **Rekomendasi**: Struktur database yang diusulkan ini dirancang untuk memastikan integritas data dan efisiensi operasi. Tabel-tabel utama telah diidentifikasi dan hubungan antar tabel diatur untuk mendukung fungsionalitas aplikasi yang diinginkan.

**3. Pemilihan Teknologi Frontend dan Backend**

**3.1. Tujuan Pemilihan Teknologi**

* Memastikan teknologi yang dipilih dapat memenuhi kebutuhan fungsional dan teknis aplikasi.
* Mempertimbangkan aspek performa, skalabilitas, dan kemudahan pemeliharaan.

**3.2. Pemilihan Teknologi Frontend**

**3.2.1. Framework/Library**

* **Pilihan:** Blade (Laravel Templating Engine)
* **Deskripsi:** Laravel Blade digunakan untuk membangun antarmuka pengguna sederhana dan terintegrasi langsung dengan backend.
* **Kriteria Pemilihan:** Kemudahan integrasi, sederhana untuk desain dinamis

**3.2.2. Teknologi Tambahan**

* **CSS Framework**: Bootstrap, Tailwind CSS (digunakan secara bawaan oleh Laravel Breeze)
* **Tooling**: -

**3.3. Pemilihan Teknologi Backend**

**3.3.1. Bahasa Pemrograman dan Framework**

* **Pilihan**: HTML, Java sctipt, dan Laravel untuk framework
* **Deskripsi**: : Laravel digunakan untuk mengelola logika aplikasi, API, dan fitur backend lainnya dan Laravel Breeze digunakan untuk login multi-role.
* **Kriteria Pemilihan**: Framework populer dengan komunitas besar, performa tinggi, mudah digunakan, serta menyediakan berbagai fitur bawaan untuk pengembangan aplikasi modern.

**3.3.2. Basis Data**

* **Pilihan**: MySQL
* **Deskripsi**: Sistem manajemen basis data yang digunakan untuk penyimpanan data.
* **Kriteria Pemilihan**: Mendukung model relasional, performa tinggi, dan kompatibilitas penuh dengan Laravel.

**3.4. Rekomendasi Teknologi**

* **Frontend**: Laravel Blade dengan Tailwind CSS karena integrasi langsung dengan Laravel Breeze dan kemudahan pengelolaan tampilan.
* **Backend**: Laravel, Laravel memiliki ekosistem yang lengkap dengan fitur bawaan seperti autentikasi, pengelolaan database dan pengelolaan API.