**2. Pemilihan Arsitektur Aplikasi**

**2.1. Tujuan Pemilihan Arsitektur**

* Menyediakan dasar yang solid untuk pengembangan, pemeliharaan, dan skalabilitas aplikasi.
* Memastikan bahwa arsitektur yang dipilih dapat memenuhi kebutuhan fungsional dan non-fungsional proyek.

**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 (menampilkan data kepada pengguna), dan Controller (menghubungkan input pengguna ke model dan view).
* **Keuntungan**: Memudahkan pengembangan dan pemeliharaan karena adanya pemisahan tanggung jawab, Cocok untuk aplikasi yang membutuhkan pengelolaan data sederhana seperti menu toko roti.
* **Kekurangan**: Jika tidak dikelola dengan baik, arsitektur ini dapat menjadi kompleks.

**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**: Kompleksitas tinggi dalam hal pengelolaan layanan dan komunikasi antar layanan, memerlukan manajemen distribusi yang cermat.

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

* **Deskripsi**: SPA memungkinkan aplikasi web untuk memuat satu halaman HTML dan memperbarui konten secara dinamis tanpa memuat ulang seluruh halaman.
* **Keuntungan**: Pengalaman pengguna yang lebih cepat dan interaktif, Cocok untuk aplikasi kecil dengan antarmuka sederhana
* **Kekurangan**: SEO menjadi lebih rumit, Membutuhkan pengelolaan status yang lebih kompleks.

**2.3. Rekomendasi Arsitektur**

* **Rekomendasi**: Arsitektur **Model-View-Controller (MVC)** dipilih karena:
* Cocok dengan kebutuhan fungsional aplikasi sederhana untuk menampilkan menu roti.
* Laravel sebagai framework backend mendukung MVC dengan baik, memungkinkan pemisahan yang jelas antara komponen data, logika, dan tampilan.

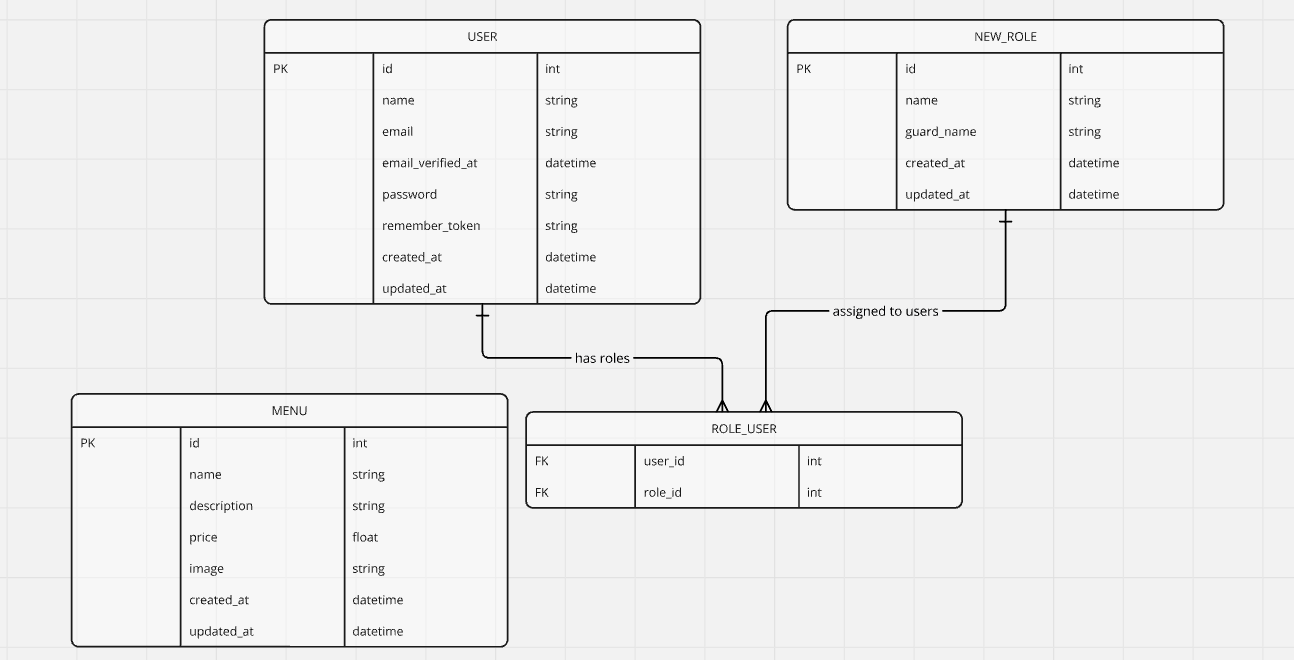
**3. Desain Struktur Database Awal**

**3.1. Tujuan Desain Struktur Database**

* Menjamin data tersimpan dengan efisien dan terstruktur untuk mendukung fungsionalitas aplikasi.
* Mempermudah pengelolaan autentikasi login dan operasi CRUD pada menu roti.
* Memberikan penyimpanan yang terstruktur untuk data pengguna, produk, dan fitur lainnya.

**3.2. Desain Struktur Database**

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

* **Deskripsi**: Diagram ER akan mencakup entitas utama seperti pengguna dan menu, dengan hubungan sederhana yang memungkinkan manajemen menu.
* **Diagram**: 

**3.2.2. Tabel-Tabel Utama**

* **Tabel Pengguna**
  + **Kolom**: UserID (PK), Username, PasswordHash, Role(Admin/User), CreatedAt, UpdatedAt
* **Tabel Produk**
  + **Kolom**: ProductID (PK), ProductName, Description, Price, StockStatus(ready/not ready, CategoryID(FK), CreatedAt, UpdatedAt
* **Tabel Kategori**
  + **Kolom**: CategoryID (PK), CategoryName , CreatedAt, UpdateAt.
* **Tabel LoginSession**
  + **Kolom**: SessionID (PK), UserID (FK), LoginTime, LogoutTime

**3.3. Rekomendasi Struktur Database**

* **Rekomendasi**: Database dirancang untuk mendukung autentikasi login dan pengelolaan produk dengan relasi yang efisien.

**4. Pemilihan Teknologi Frontend dan Backend**

**4.1. Tujuan Pemilihan Teknologi**

* Menggunakan teknologi yang mendukung fitur login, pengelolaan menu, dan kecepatan akses.
* Memastikan skalabilitas dan kemudahan pemeliharaan di masa depan

**4.2. Pemilihan Teknologi Frontend**

**4.2.1. Framework/Library**

* **Pilihan**: React,
* **Deskripsi**: Framework/library untuk membangun antarmuka pengguna yang interaktif.
* **Kriteria Pemilihan**: Komponen reusable mempermudah pembuatan UI seperti form login dan dashboard admin, Komunitas luas dan dukungan integrasi dengan backend API.

**4.2.2. Teknologi Tambahan**

* **CSS Framework**: Tailwind CSS untuk fleksibilitas desain responsif.
* **Tooling**: Webpack untuk pengelolaan modul dan Babel untuk kompatibilitas JavaScript.

**4.3. Pemilihan Teknologi Backend**

**4.3.1. Bahasa Pemrograman dan Framework**

* **Pilihan**: Laravel
* **Deskripsi**: Bahasa dan framework untuk pengembangan logika aplikasi dan layanan backend.
* **Kriteria Pemilihan**: Mendukung autentikasi bawaan untuk fitur login, Struktur MVC yang sesuai untuk mengelola data pengguna dan produk.

**4.3.2. Basis Data**

* **Pilihan**: MySQL,
* **Deskripsi**: Sistem manajemen basis data yang digunakan untuk penyimpanan data.
* **Kriteria Pemilihan**: Relasional dan cocok untuk pengelolaan data seperti pengguna, produk, dan kategori.

**3.4. Rekomendasi Teknologi**

* **Frontend**: **React** karena mendukung desain interaktif dan responsif dengan integrasi API yang mudah.
* **Backend**: **Laravel** karena kemampuannya dalam menangani autentikasi, skalabilitas, dan dukungan komunitas yang luas.
* **Database:** **MySQL** karena stabilitasnya untuk aplikasi berskala kecil hingga menengah.