SantaiRumah

“Santai, Kami yang Kerja Bersihkan Rumah Anda”

Habib Akhmad Al Farisi 18222029

Daftar Link

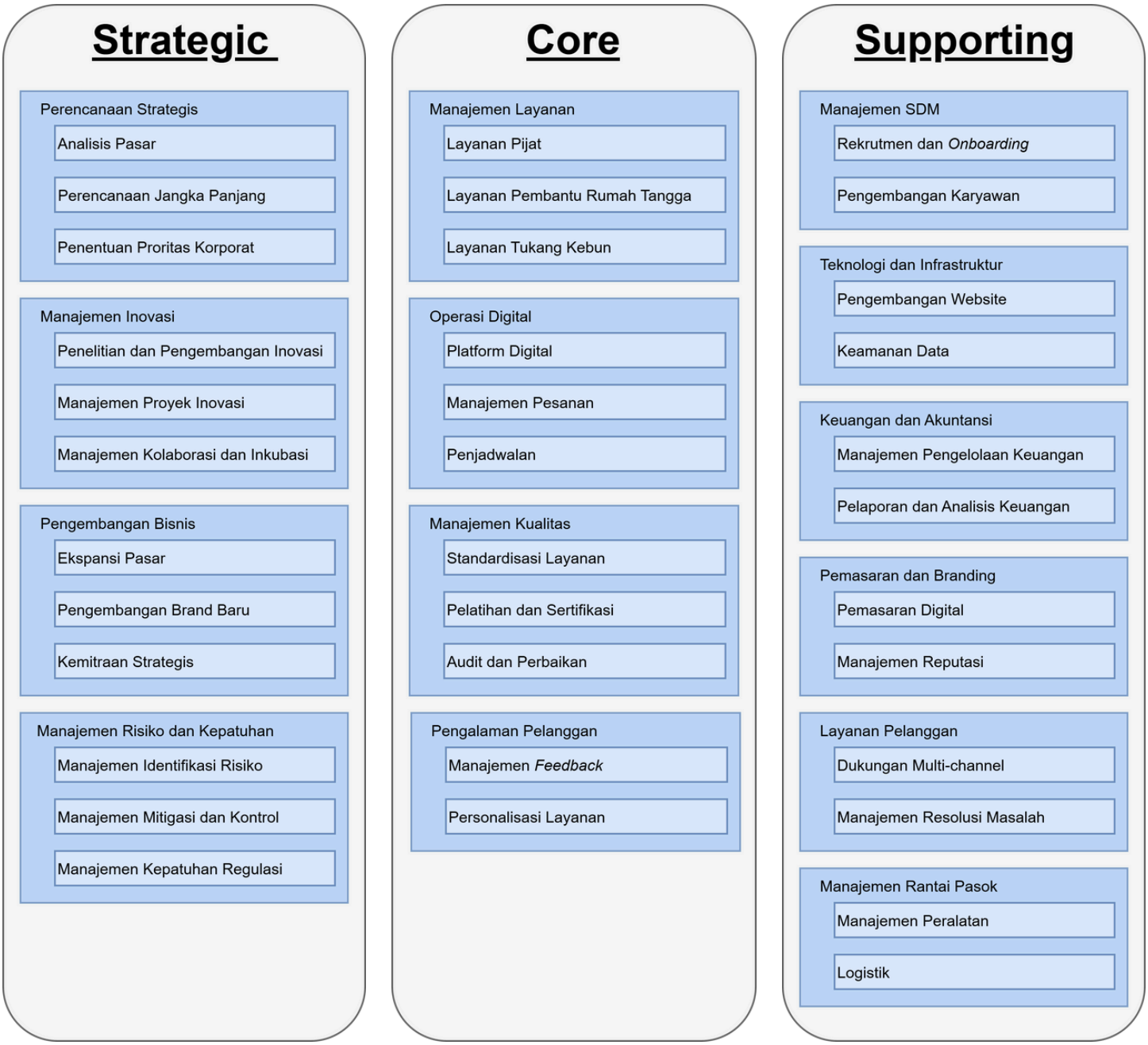
Nama Link	Link
Github	https://github.com/stdFaris/uasMatkulTST
Dokumentasi API	https://github.com/stdFaris/uasMatkulTST/tree/main/backend
Website	https://santairumah.netlify.app

catatan: entah kenapa saat membuka website melalui chrome dengan email std backend tidak bekerja, tapi bila menggunakan chrome dengan email lain, backend berfungsi dengan baik.

Layanan Inti

- fokus pada penawaran **pembantu**, **tukang kebun, dan pijat**
- Memberikan penawaran waktu, bisa per jam hingga per bulan
- Menawarkan partner terbaik dalam satu kecamatan sehingga mempermudah customers

Business Capability Map (BCM)



Domain Dipilih: Domain Penjadwalan

Penjelasan sederhana domain penjadwalan.

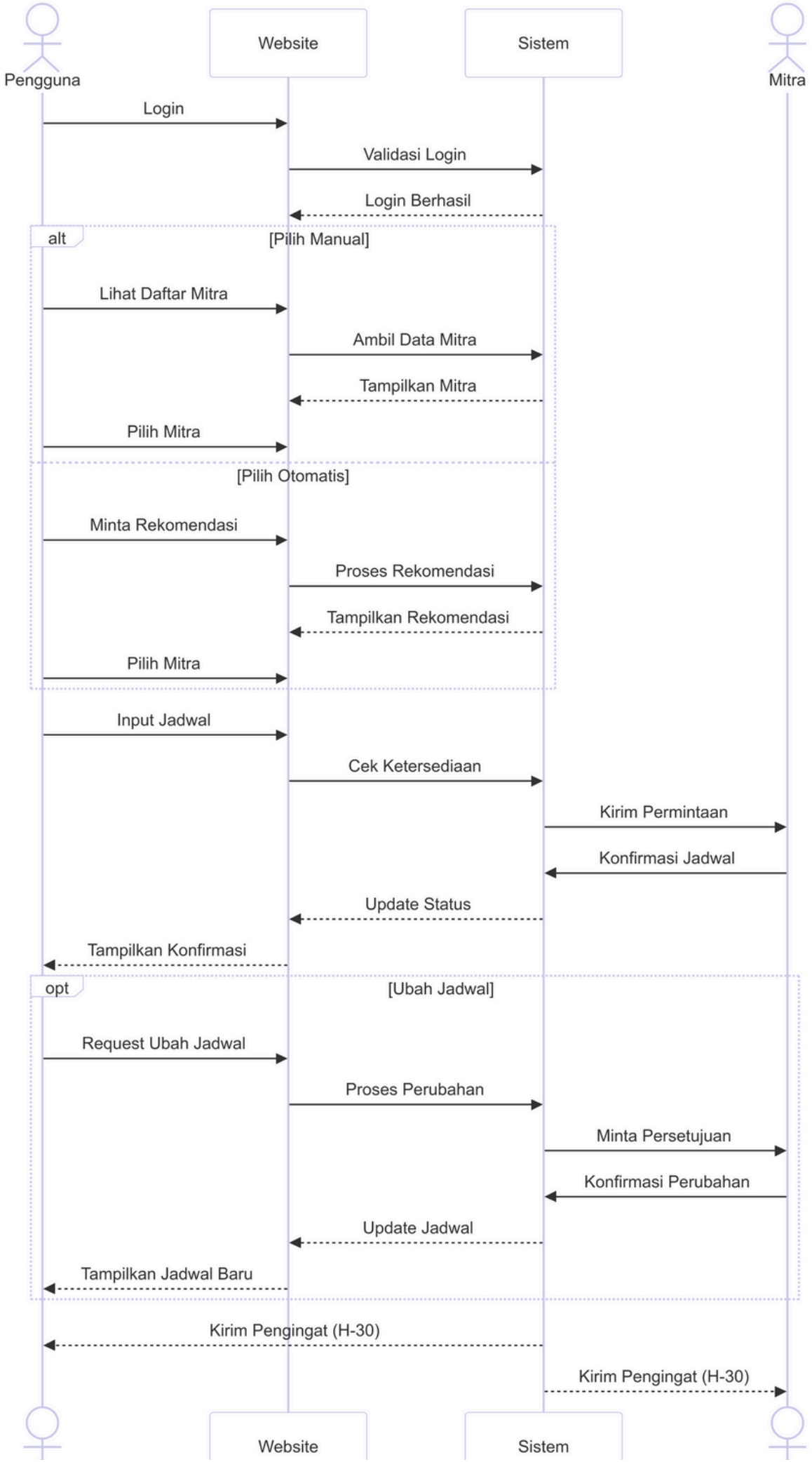
1. **Algoritma Penjadwalan**: Fokus Pada pengaturan jadwal sehingga tidak ada bentrok
2. **Notifikasi & Konfirmasi**: mengirimkan notifikasi dan meminta persetujuan partner dalam 3 menit (Namun penerapan tidak disediakan karena hanya fokus ke sisi customers)
3. **Fleksibilitas Waktu**: Menyediakan layanan dengan durasi beragam, 1 jam hingga bulanan

Subdomain dan Deskripsi

Penjelasan Subdomain

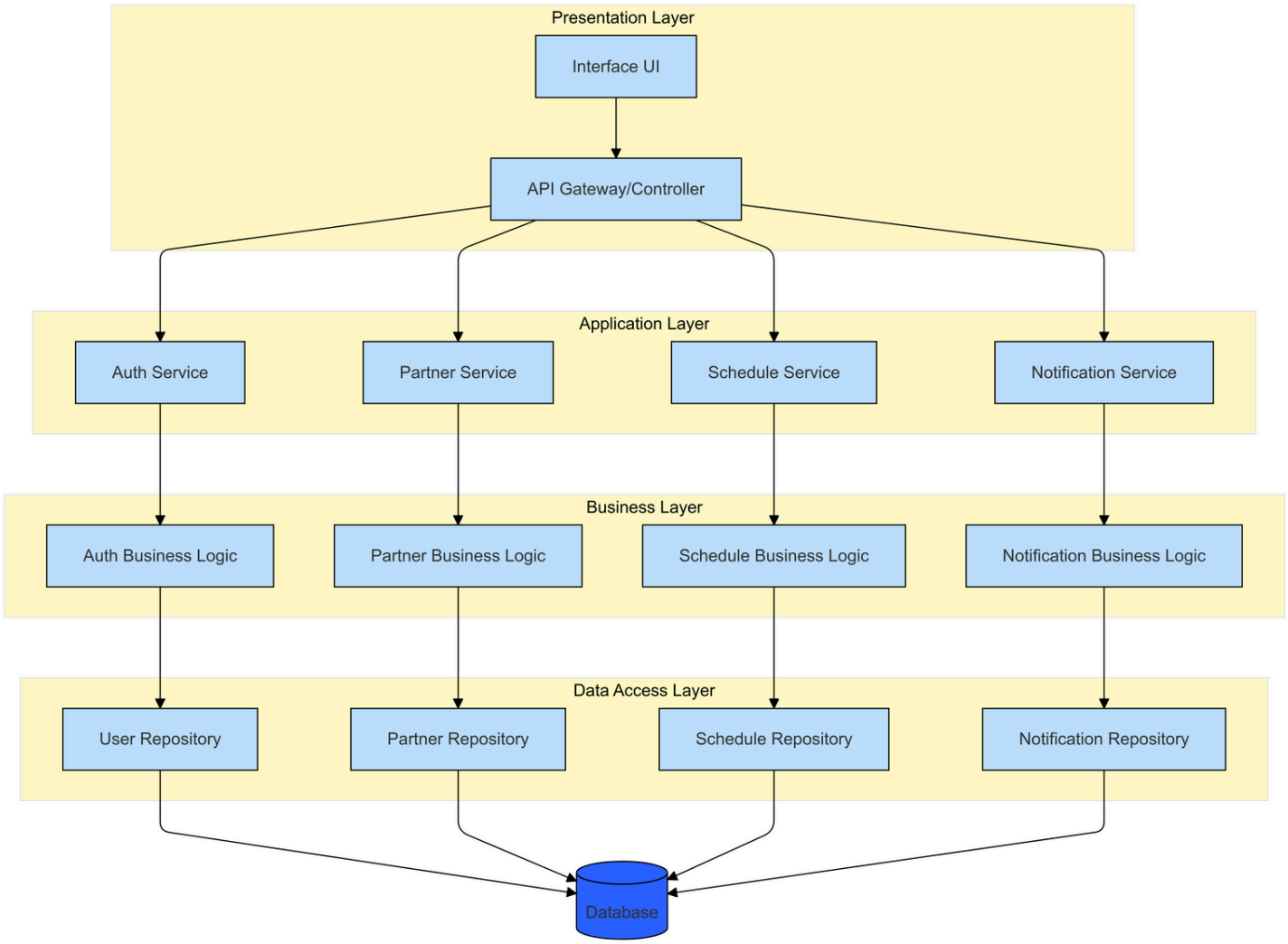
1. **Algoritma Pencocokan partner(Core)**: Memberikan daftar rekomendasi partner berdasarkan beberapa perhitungan sederhana, harus dalam satu daerah
2. **manajemen Jadwal Dinamis(Core)**: Jadwal yang disediakan beragam, bisa 1 jam hingga bulanan. Jadwal yang dimiliki partner tidak akan bentrok antar jadwal
3. **Penyesuaian Jadwal Cerdas(Generic)**: Mengelola perubahan jadwal otomatis, seperti pembatalan atau penundaan, dengan optimasi ulang yang transparan dan disetujui mitra.
4. **Sistem Notifikasi Komprehensif(Supporting)**: Memberikan notifikasi pengingat jadwal pada 30 menit sebelum pelayanan dimulai sebagai bentuk komunikasi yang efektif.

System Modelling (Sequence Diagram)


Pengguna
Website
Sistem
Mitra
Login
Validasi Login
Login Berhasil
alt
[Pilih Manual]
Lihat Daftar Mitra
Ambil Data Mitra
Tampilkan Mitra
Pilih Mitra
[Pilih Otomatis]
Minta Rekomendasi
Proses Rekomendasi
Tampilkan Rekomendasi
Pilih Mitra
Input Jadwal
Cek Ketersediaan
Kirim Permintaan
Konfirmasi Jadwal
Update Status
Tampilkan Konfirmasi
opt
[Ubah Jadwal]
Request Ubah Jadwal
Proses Perubahan
Minta Persetujuan
Konfirmasi Perubahan
Update Jadwal
Tampilkan Jadwal Baru
Kirim Pengingat (H-30)
Kirim Pengingat (H-30)
Website
Sistem


Pengguna
Mitra

Software Architecture



Alasan Dipilih Layered Architecture

Layered architecture dipilih karena memberikan pemisahan yang jelas antara komponen-komponen aplikasi. Setiap layer memiliki tanggung jawab spesifik, seperti misalnya business layer untuk logika business. Pemisahan ini memudahkan proses development karena setiap layer bisa dikembangkan dan diuji secara terpisah.

Untuk pengembangan pribadi, architecture ini cocok karena strukturnya yang langsung ke tujuan (straightforward) dalam artian setiap layer hanya berkomunikasi dengan layer di atas atau di bawahnya secara langsung. Ini membuat alur data dan proses dalam aplikasi menjadi lebih mudah diikuti dan di-debug jika terjadi masalah.

Penjelasan Masing-Masing Layer

Presentation Layer:

- Interface UI: Interface untuk interaksi user
- API Gateway/Controller: Menangani request dan response dari UI

Application Layer:

- Auth Service: Layanan untuk autentikasi dan autorisasi
- Partner Service: Manajemen partner/mitra
- Schedule Service: Mengelola penjadwalan
- Notification Service: Menangani notifikasi

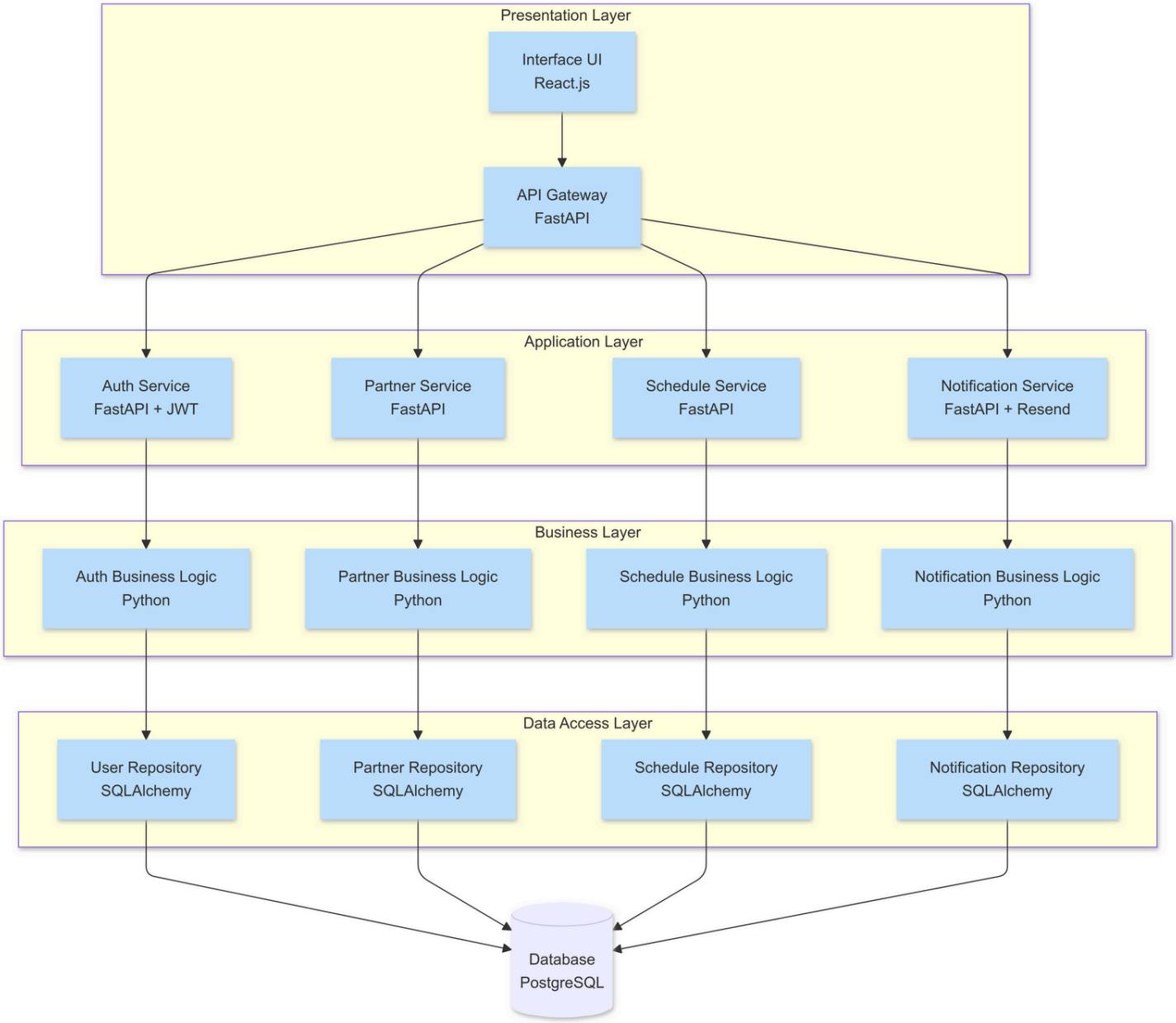
Business Layer:

- Auth Business Logic: Aturan bisnis terkait autentikasi
- Partner Business Logic: Aturan bisnis terkait partner
- Schedule Business Logic: Aturan bisnis terkait penjadwalan
- Notification Business Logic: Aturan bisnis terkait notifikasi

Data Access Layer:

- User Repository: Akses data pengguna
- Partner Repository: Akses data partner
- Schedule Repository: Akses data jadwal
- Notification Repository: Akses data notifikasi
- Database: Penyimpanan seluruh data

Technology



Presentation Layer

1. **Interface UI:**
   - **Teknologi:** *React.js*
   - Penjelasan: React adalah library JavaScript populer untuk membangun antarmuka pengguna. React digunakan untuk membuat komponen UI yang dapat digunakan kembali, memberikan pengalaman interaktif kepada pengguna dengan pembaruan DOM secara efisien melalui konsep Virtual DOM.
2. **API Gateway/Controller:**
   - **Teknologi:** *FastAPI*

- Penjelasan: FastAPI digunakan untuk menangani komunikasi antara antarmuka pengguna dan lapisan aplikasi. Ini menyediakan endpoint HTTP untuk menerima permintaan dari UI dan mengirimkan respons yang sesuai. Dengan fitur asinkron bawaan, FastAPI mampu menangani API dengan performa tinggi.

Application Layer

1. **Auth Service:**
   - **Teknologi:** FastAPI + JWT
   - Penjelasan: Layanan ini menangani autentikasi dan otorisasi menggunakan protokol JSON Web Token (JWT), memastikan akses hanya diberikan kepada pengguna yang sah.
2. **Partner Service:**
   - **Teknologi:** FastAPI
   - Penjelasan: Mengelola operasi CRUD terkait data mitra/partner. Dapat berinteraksi dengan database atau layanan pihak ketiga jika diperlukan.
3. **Schedule Service:**
   - **Teknologi:** FastAPI
   - Penjelasan: Layanan ini mengelola penjadwalan, termasuk penjadwalan pelayanan tiap penyedia jasa, dalam hal ini semisal dalam waktu tertentu partner melakukan pengerjaan maka tidak bisa dipesan hingga waktu tertentu berdasarkan business logic nya
4. **Notification Service:**
   - **Teknologi:** FastAPI + layanan notifikasi eksternal dalam hal ini saya menggunakan resend melalui email
   - Penjelasan: Dipilih dikarenakan dengan email ini bisa menjadi lebih mudah dan lebih memberikan pengertian yang sederhana

Business Layer

1. **Auth Business Logic:**
   - **Teknologi:** Python (FastAPI untuk implementasi)
   - Penjelasan: Mengimplementasikan aturan bisnis terkait autentikasi, seperti validasi token, pengelolaan peran (role-based access control), dan pengaturan sesi pengguna.
2. **Partner Business Logic:**

- **Teknologi:** Python
- Penjelasan: Mengatur logika yang lebih kompleks terkait partner, seperti validasi data atau aturan kolaborasi antar mitra.

3. **Schedule Business Logic:**
   - **Teknologi:** Python
   - Penjelasan: Menerapkan aturan bisnis untuk penjadwalan, termasuk validasi waktu dan konflik jadwal.
4. **Notification Business Logic:**
   - **Teknologi:** Python
   - Penjelasan: Mengelola aturan pengiriman notifikasi, termasuk personalisasi pesan dan prioritas pengiriman.

Data Access Layer

1. **User Repository:**
   - **Teknologi:** SQLAlchemy (untuk ORM) + PostgreSQL
   - Penjelasan: Mengelola akses data pengguna di database dengan fitur ORM dari SQLAlchemy, memudahkan interaksi antara Python dan database relasional.
2. **Partner Repository:**
   - **Teknologi:** SQLAlchemy + PostgreSQL
   - Penjelasan: Mengatur operasi data partner/mitra dalam database.
3. **Schedule Repository:**
   - **Teknologi:** SQLAlchemy + PostgreSQL
   - Penjelasan: Mengelola data terkait penjadwalan, termasuk pembacaan dan pembaruan jadwal.
4. **Notification Repository:**
   - **Teknologi:** SQLAlchemy + Redis (untuk notifikasi sementara)
   - Penjelasan: Mengakses data terkait notifikasi, termasuk status pengiriman. Redis dapat digunakan untuk menyimpan data sementara seperti antrian notifikasi.

Database

- **Teknologi:** PostgreSQL

- Penjelasan: Digunakan sebagai database relasional utama karena mendukung fitur-fitur canggih seperti transaksi, indexing, dan query yang kompleks. Alternatif seperti Redis dapat digunakan untuk caching data yang sering diakses atau penyimpanan sementara.

Development Environment

Version Control	**VS Code**
Code Editor	**Git**
Frontend	**React Typescript Vite + shdcn ui + Tailwind CSS**
Backend	**FastAPI**
Database	**PostgreSQL**
Deployment	**Netlify (Frontend) + Backend (Railway)**

Development Step

Persiapan Project	• **Setup react typescript vite + tailwind css + shdcn UI** • **Setup PostgreSQL** • **Setup FastAPI + venv** • **Membuat github repositori**
Backend Development	• **Membuat schemas dan models** • **Membuat services** • **Membuat routes**
Frontend Development	• **Implementasi UI** • **Responsive**
Testing	• **Manual Testing fitur**
Deployment	• **Setup Docker** • **Build Docker** • **Deploy**