Tugas Besar II3160 Teknologi Sistem Terintegrasi

Website-Based Pet Hotel Service - PawMates

Disusun Oleh:

Naomi Pricilla Agustine 18222065

Program Studi Sistem dan Teknologi Informasi

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

DAFTAR ISI

I. Business Capability Map 3

II. Business Core Capability 4

III. Sequence Diagram 6

IV. Software Architecture 7

V. Proposed Technology 8

VI. Development Plan 11

VII. Implementation Plan 12

VIII. Service Implementation 14

1. Authentication 14

2. Search and Recommendation Engine 16

3. Integration With Other Service (Customer Service and Support) 20

IX. Specification Implementation 21

1. Docker 21

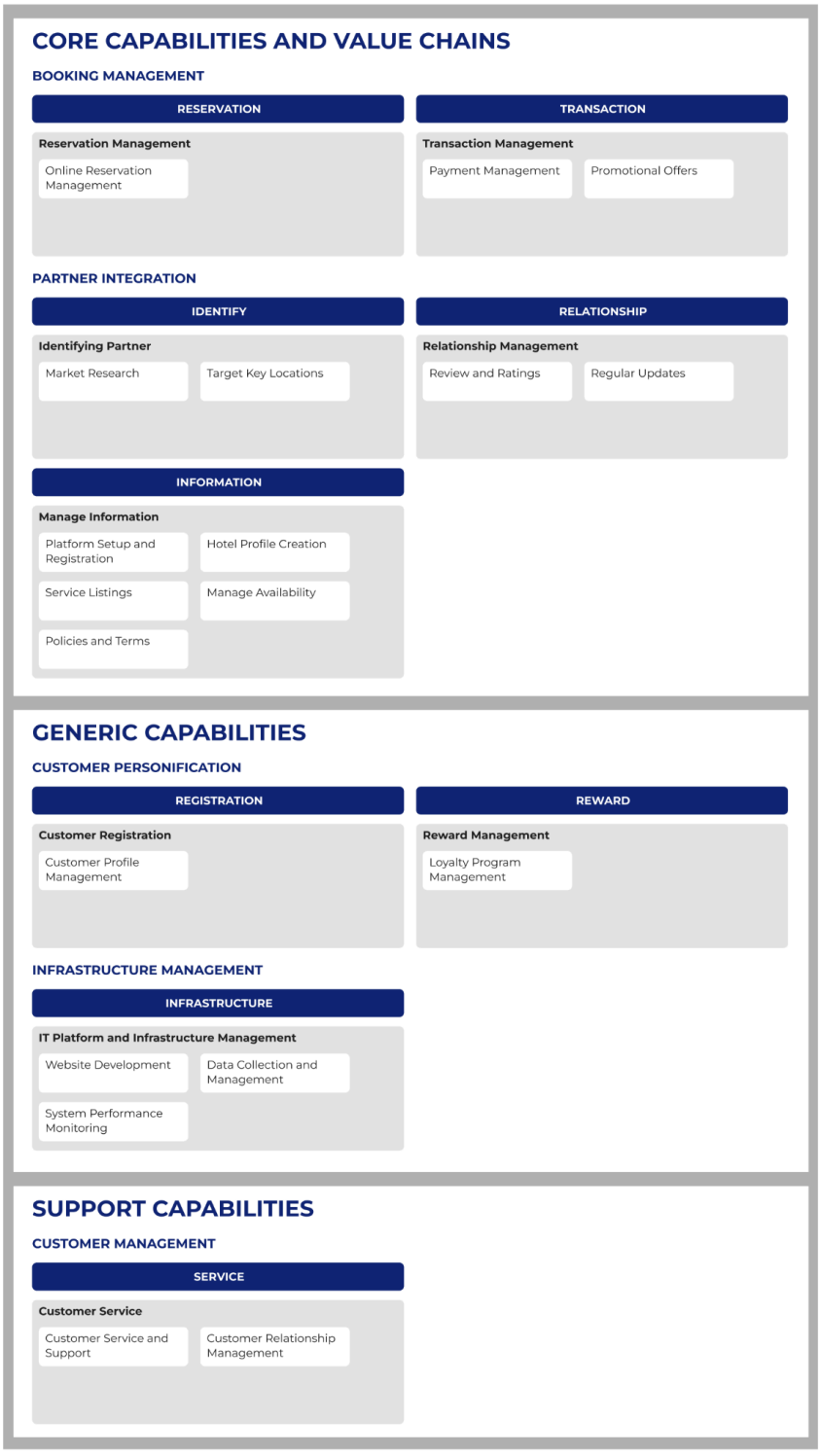
2. HTTPS 22

3. Integrasi Antar Service Menggunakan API Key Authentication 22

LAMPIRAN 24

I. Business Capability Map

Berikut merupakan *Business Capability Map* atau BCM yang dimiliki oleh *Website-Based Pet Hotel Service* - PawMates.



II. Business Core Capability

Online Reservation Management adalah domain yang termasuk **core capability.** Keunikan yang dimiliki oleh domain ini adalah memungkinkan **personalisasi rekomendasi berdasarkan data yang dimasukkan pengguna saat pencarian hotel** secara online. Data ini mencakup tanggal, lokasi, ukuran hewan peliharaan, preferensi layanan, rentang harga, dan ulasan dari pengguna lain. Data-data tersebut akan **diproses dan dianalisis secara personal dan dinamis** lalu disesuaikan dengan ketersediaan dan penawaran khusus untuk memberikan preferensi kepada pengguna yang lebih sesuai dan efisien dalam menemukan layanan terbaik mereka.

Berikut merupakan penjelasan dari setiap sub domain pada *core capability* dari *Website-Based Pet Hotel Service* - PawMates.

1. Hotel Recommender

Sub domain ini berfungsi untuk memberikan rekomendasi kepada pengguna berdasarkan preferensi pelanggan seperti tanggal dan lokasi reservasi, ukuran hewan peliharaan, dan rentang harga untuk menawarkan hasil yang paling relevan bagi setiap pengguna.

2. ***Booking Management***

   Sub domain ini berfungsi untuk mengelola proses pemesanan hotel seperti pencarian hotel, pengecekan ketersediaan, pemilihan kamar, dan memverifikasi informasi yang relevan.

3. ***Booking Confirmation***

   Sub domain ini berfungsi untuk memastikan bahwa pemesanan yang telah dilakukan pengguna dicatat dengan benar dan memberikan informasi kepada pengguna mengenai detail pemesanan mereka. Pengguna akan menerima konfirmasi melalui notifikasi pada aplikasi.

4. ***Booking Modification***

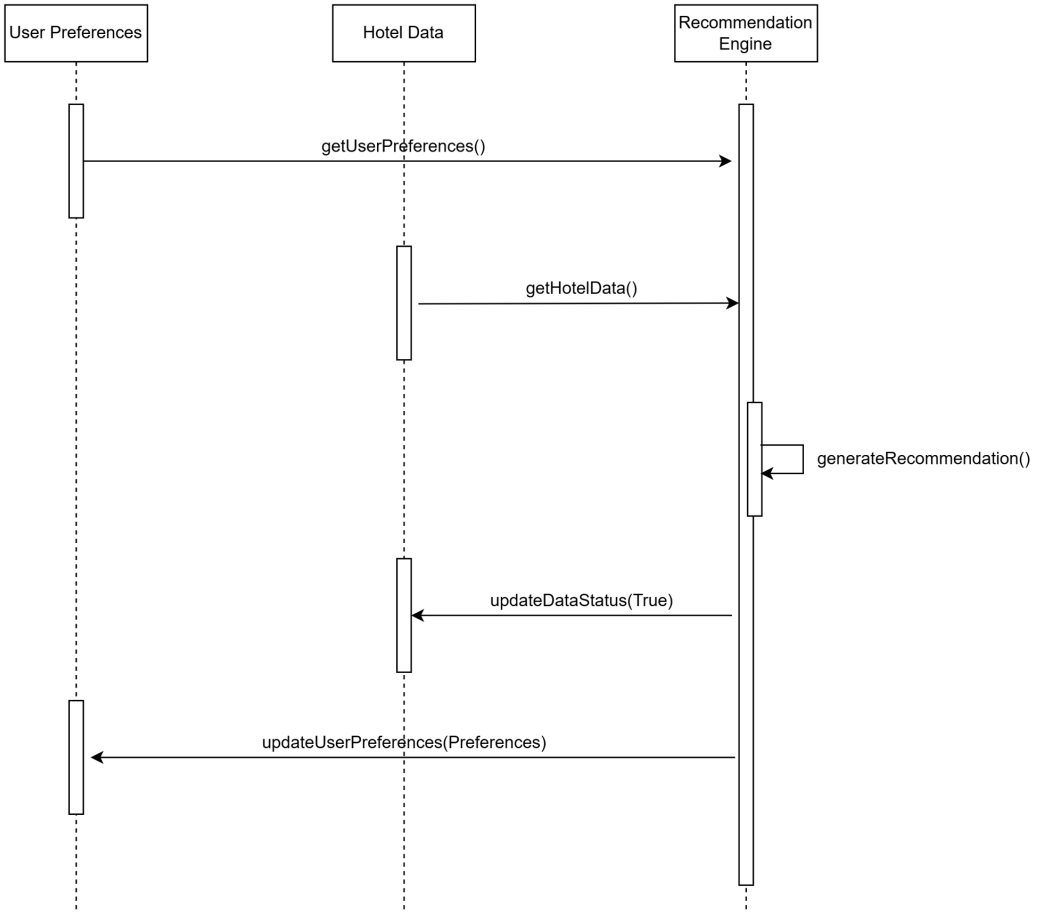
   Sub domain ini berfungsi untuk mengubah detail pemesanan yang sudah dikonfirmasi, tetapi harus memenuhi syarat dan ketentuan yang ditetapkan hotel yang bersangkutan.

5. ***Communication and Notification System***

   Sub domain ini berfungsi untuk mengatur semua komunikasi antara platform dengan pengguna terkait pengiriman notifikasi untuk status pemesanan, pengingat check-in, penawaran khusus, dan lainnya. Komunikasi ini dapat dilakukan melalui aplikasi.

Berdasarkan beberapa sub domain tersebut, sub domain dari *core capability* yang akan saya terapkan adalah pada ***Hotel Recommender.*** Selain dari *core capability*, saya juga menerapkan sub domain dari *support capability* yaitu ***Customer Service and Support.***

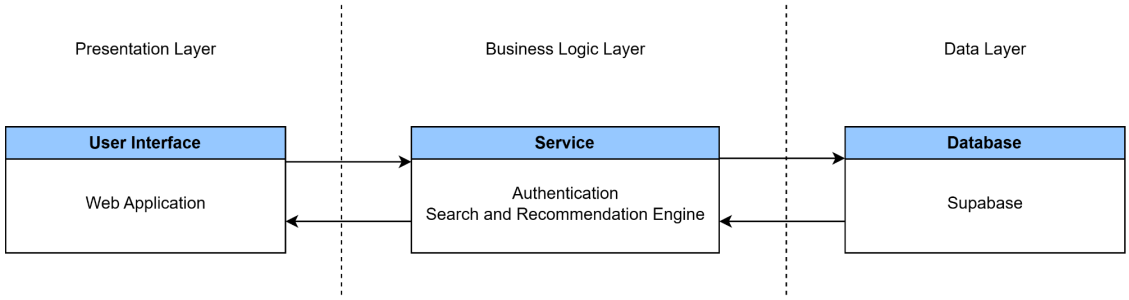
III. Sequence Diagram



Berikut merupakan alur dari *sequence diagram* dari *Website-Based Pet Hotel Service* - PawMates yaitu sebagai berikut.

1. Mengambil preferensi pengguna berdasarkan kategori hotel pilihan mereka atau kategori hotel yang sebelumnya sudah mereka cari dengan fungsi ***getpreferences()*** dari User *Preferences entity* untuk digunakan dalam *recommendation engine*.
2. Mengambil data hotel yang ada dengan fungsi ***getHotelData()*** dari database hotel untuk digunakan juga dalam *recommendation engine*.
3. Mengolah data preferensi pengguna dengan data hotel yang sesuai dalam *recommendation engine* dengan fungsi ***generateRecommendation().***
4. Mengirim status ke basis data bahwa data yang dikirim benar dengan fungsi ***updateDataStatus().***
5. Mengirim hasil rekomendasi kepada pengguna dengan fungsi ***updateUserPreferences().***

IV. Software Architecture



Software Architecture ini merupakan arsitektur berbasis *layered design* atau *three-tier architecture*.

1. **Presentation Layer (User Interface)**

   Layer ini berguna untuk menangani interaksi langsung dengan pengguna yang terdiri dari elemen *front-end* seperti HTML, CSS, dan JavaScript tanpa menggunakan *framework* untuk menyediakan antarmuka, mengirimkan permintaan ke lapisan *service* melalui API, dan menampilkan data yang diterima.

2. **Business Logic Layer (Service)**

   Layer ini berguna sebagai penghubung antar *presentation layer* dan *data layer* yang diimplementasikan pada *back-end* menggunakan *framework* FastAPI yang menerima permintaan melalui *endpoint* API, memproses *business logic,* dan komunikasi dengan basis data untuk mendapatkan atau memperbarui data.

3. **Data Layer (Database)**

   Layer ini berguna untuk menyimpan data yang dikelola dengan menggunakan Supabase untuk menyimpan data yang dibutuhkan aplikasi dan melayani permintaan data.

V. Proposed Technology

Berikut merupakan teknologi yang akan saya gunakan pada *Website-Based Pet Hotel Service* - PawMates.

1. Database Layer

Teknologi yang digunakan untuk menangani data pelanggan, hotel, riwayat pencarian, dan ulasan pengguna adalah **Supabase** dengan alasan sebagai berikut.

- Supabase merupakan platform *open-source* yang berbasis **PostgreSQL.**
- Database relasional yang cocok untuk data yang terstruktur dengan baik.
- Mampu menangani data dalam skala yang besar dengan tambahan fitur-fitur yang memudahkan pengelolaan dan akses data.
- Fleksibel dalam menangani data semi-terstruktur seperti tipe data JSON.
- Dilengkapi dengan **API *real-time*** untuk memudahkan sinkronisasi data secara langsung dengan *front-end*.
- Memiliki tambahan fitur berbasis ***cloud*** yang dapat meningkatkan produktivitas dalam pengembangan aplikasi.

2. Front-end Layer

Front-end dirancang untuk memberikan antarmuka yang responsif, menarik, dan interaktif menggunakan kombinasi teknologi sebagai berikut.

1) **HTML5:**
   - Digunakan untuk membangun struktur dasar halaman web.
   - **Implementasi:**
     - Elemen semantik seperti <nav> untuk navigasi, <header> untuk bagian atas halaman, dan untuk konten utama
     - Link navigasi menggunakan <a href="/login"> untuk routing halaman.
     - Struktur konten yang terorganisir dengan class seperti text-content dan image-content
     - Integrasi font dari Google Fonts menggunakan <link> tag
2) **CSS3:**
   - Digunakan untuk menciptakan desain yang menarik dan responsif.

- **Implementasi:**
    - ***Flexbox and Grid Layout***
    - **Animasi dan Transisi**
    - **Warna dan tipografi** yang konsisten menggunakan Google Fonts (seperti Montserrat) untuk estetika yang modern.

3) **JavaScript:**
    - Digunakan untuk interaktivitas dan manajemen *state*.
    - **Implementasi:**
        - ***Event Handling***
        - ***Local Storage* :** Untuk menyimpan data *user*
        - ***Fetch* API :** Untuk komunikasi dengan *back-end*

Untuk UI/UX design akan menggunakan **Figma** yang memudahkan desain yang responsif. ***Hosting*** untuk *front-end layer* menggunakan **Vercel** dan **Railway.**

3. Back-end Layer

Teknologi yang digunakan untuk ***back-end layer*** adalah **FastAPI** dengan alasan sebagai berikut.

- Cocok untuk aplikasi yang butuh performa tinggi dan kecepatan untuk menangani *request* dalam jumlah banyak secara *real-time*, seperti sistem rekomendasi.
- Dapat menangani permintaan asinkron dengan baik sehingga dapat memproses data secara efisien tanpa mengganggu performa.
- Memiliki dokumentasi API otomatis menggunakan Swagger UI dan ReDoc untuk memudahkan pengembangan dan integrasi API.
- Menggunakan Python sehingga fleksibel untuk diintegrasikan dengan *machine learning*.

Hosting yang akan digunakan untuk *back-end layer* menggunakan **Vercel** dan **Railway.**

4. IDE

Visual Studio Code akan digunakan untuk pengembangan sistem dengan alasan sebagai berikut.

- Visual Studio Code bersifat gratis dan *open source*.

- Sangat fleksibel untuk pengembangan lintas bahasa.
- Memiliki kinerja yang cepat dan ringan.
- Memiliki *built-in integration* dengan Git yang mudah digunakan langsung dari dalam *editor*.

5. Version Control

Git dengan repositori di platform Github akan digunakan untuk pengembangan sistem dengan alasan sebagai berikut.

- Memungkinkan pengembangan secara *offline* dan melakukan commit tanpa koneksi internet.
- Memungkinkan pengelolaan *branch* dengan mudah untuk kolaborasi tim, pengembangan paralel, dan *testing* yang terisolasi.

VI. Development Plan

Berikut merupakan langkah-langkah rencana pengembangan dari *Website-Based Pet Hotel Service* - PawMates.

VII. Implementation Plan

Berikut merupakan langkah-langkah rencana implementasi dari *Website-Based Pet Hotel Service* - PawMates.

Actions	4-10 Nov W1	11-17 Nov W2	18-24 Nov W3	25 Nov-1 Des W4	2-8 Des W5	9-15 Des W6	16-22 Des W7	23-29 Des W8
Persiapan dan Setup Lingkungan								
Pengembangan Skema Database								
Pengembangan Frontend Dasar								
Pengembangan Backend dan API Dasar								
Integrasi dan Testing								
Setup dan Konfigurasi Deployment								

Minggu 1: Persiapan dan Setup Lingkungan (4-10 November 2024)

- Install alat pengembangan yang dibutuhkan dalam pembuatan proyek.
- Buat repositori Git dan setup kontrol versi di GitHub.
- Setup IDE VS Code dan workspace untuk pembuatan proyek.
- Inisialisasi proyek frontend dengan HTML, CSS, dan JavaScript dan backend dengan FastAPI.

Minggu 2: Pengembangan Skema Database (11-17 November 2024)

- Buat skema database di Supabase dengan tabel seperti users, hotels, dan search_history.
- Uji integritas skema database untuk memastikan tidak ada error struktural.

Minggu 3: Pengembangan Frontend Dasar (18 November-1 Desember 2024)

- Inisialisasi proyek frontend
- Desain UI sesuai prototipe dari Figma.
- Implementasi routing dasar dan navigasi di aplikasi.
- Testing awal komponen frontend.

Minggu 4: Pengembangan Backend dan API Dasar (2-15 Desember 2024)

- Setup proyek backend dengan struktur dan library yang diperlukan.
- Buat API untuk fitur CRUD (Create, Read, Update, Delete) pada data pengguna, data hotel, dan data riwayat pencarian pengguna.
- Integrasi database Supabase dengan backend.

- Implementasi dasar modul rekomendasi untuk fitur rekomendasi hotel.

Minggu 5: Integrasi dan Testing (16-22 Desember 2024)

- Hubungkan frontend dengan API di backend.
- Uji setiap endpoint dan integrasi data dari backend ke frontend.
- Debugging.
- Testing alur end-to-end.

Minggu 6: Setup dan Konfigurasi Deployment (23-29 Desember 2024)

- Setup akun dan proyek di Vercel dan Railway.
- Konfigurasi environment variables.
- Deployment aplikasi di Vercel dan Railway.
- Uji aplikasi untuk memastikan semua fitur berfungsi dengan baik.
- Lakukan dokumentasi akhir dan maintenance.

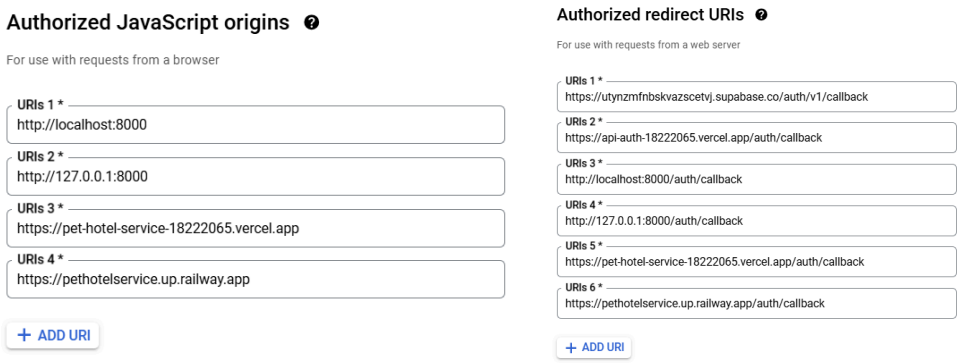
VIII. Service Implementation

Berikut merupakan penjelasan dari setiap *service* yang diimplementasikan pada *Website-Based Pet Hotel Service* - PawMates.

1. Authentication

Penerapan autentikasi menggunakan **OAuth 2.0 dengan Google** sebagai mekanisme autentikasi dengan memanfaatkan layanan Google dengan tahapan sebagai berikut.

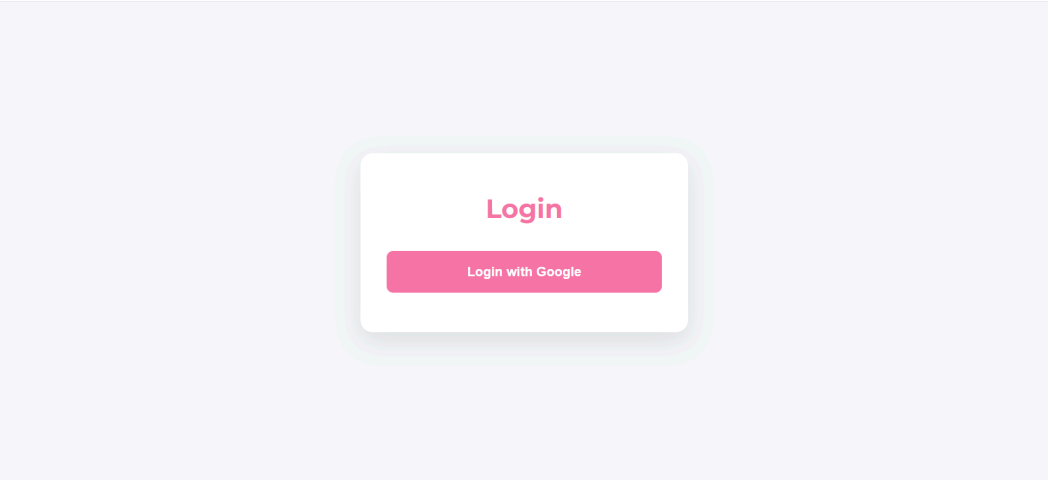
1) **Konfigurasi Google OAuth** dengan mendefinisikan GOOGLE_CLIENT_ID, GOOGLE_CLIENT_SECRET, dan CALLBACK_URL pada file .env. **Hubungkan dengan Supabase** untuk menyimpan dan mengelola data pengguna.

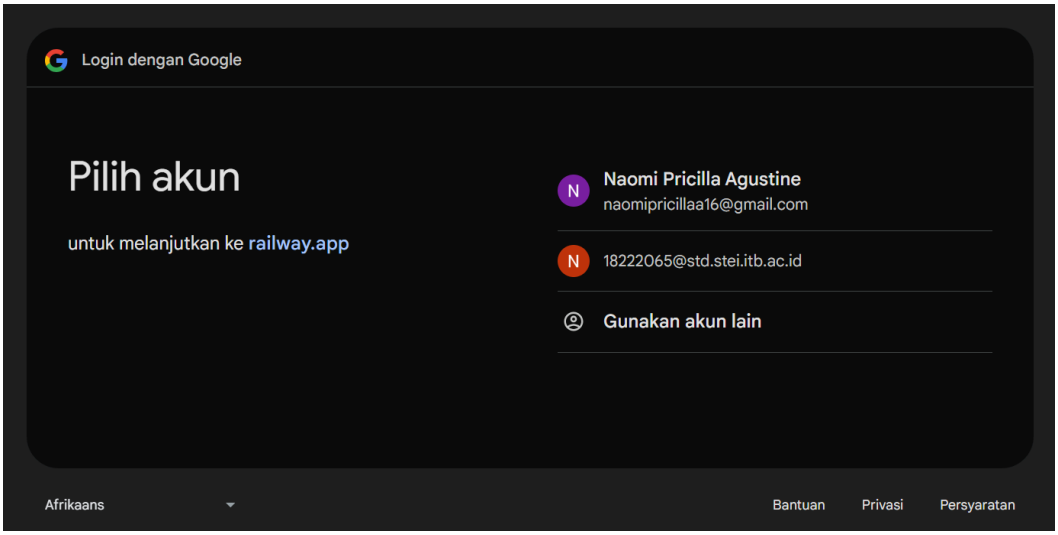


File .env

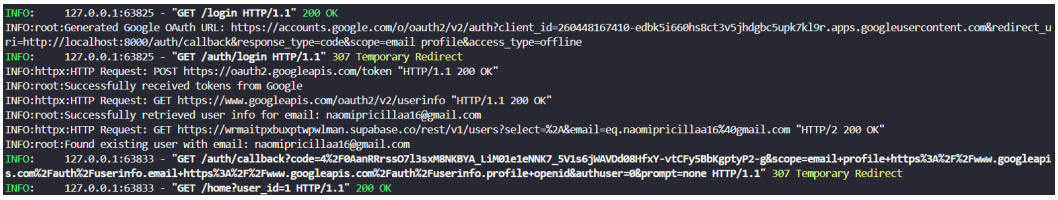
```
GOOGLE_CLIENT_ID=260448167410-edbk5i660hs8ct3v5jhdgbc5upk7kl9r
.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=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
CALLBACK_URL=http://localhost:8000/auth/callback
SUPABASE_URL=https://wrmaitpxbuxptwpwlman.supabase.co
SUPABASE_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJz
dXBhYmFzZSIsInJlZiI6IndybWFpdHB4YnV4cHR3cHdsbWFuIiwicm9sZSI6In
NlcnZpY2Vfcm9sZSIsImlhdCI6MTczNTU0MTYxNCwiZXhwIjoyMDUxMTE3NjE0
fQ.zoVB3N2nBVLZYh2JtRgxEtaaHBDosvm8BaS_JPeSZEA
```

2) ***Login* dengan Google dengan *endpoint* /auth/login** untuk menghasilkan URL otorisasi Google dan pengguna akan diarahkan ke halaman tersebut.

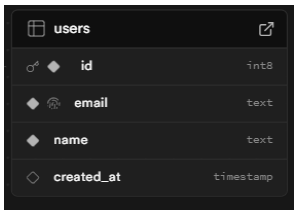




3) Setelah pengguna menyelesaikan otorisasi, **Google mengembalikan kode otorisasi ke *endpoint* /auth/callback** yang nantinya akan **ditukar dengan token akses melalui *endpoint* Google Token** untuk mendapatkan informasi pengguna.



4) Informasi pengguna akan **disimpan di tabel users pada Supabase.**

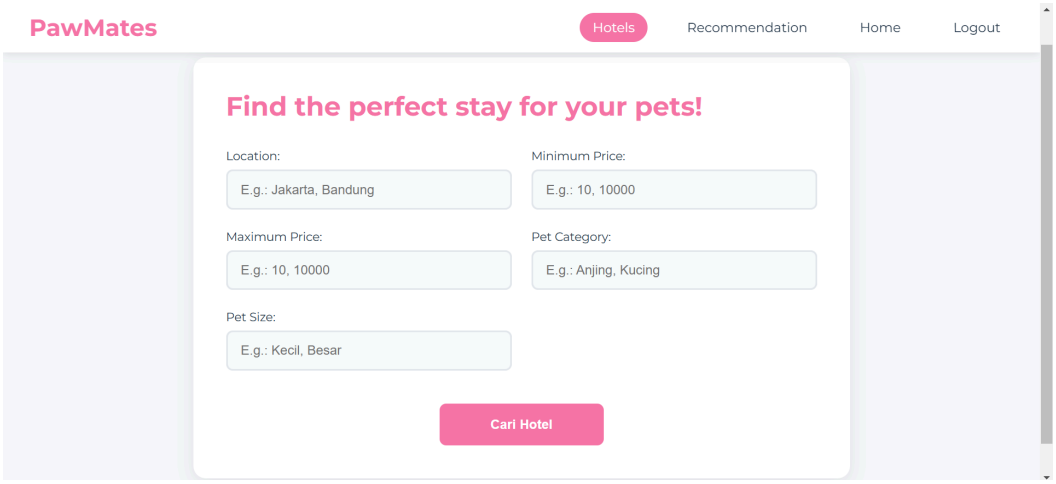


5) ***Endpoint logout*** juga tersedia untuk permintaan *logout*.

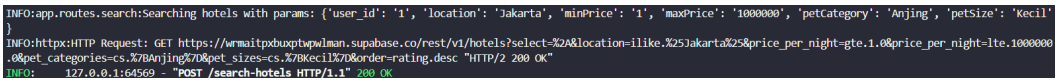
2. *Search and Recommendation Engine*

Fitur ***Search*** ditujukan agar pengguna dapat mencari hotel untuk hewan berdasarkan beberapa kriteria tertentu dengan langkah-langkah sebagai berikut.

1) Halaman *Search* dengan HTML menyediakan ***front-end*** **untuk menerima input dari pengguna**, seperti lokasi, harga minimum, harga maksimum, kategori, dan ukuran hewan. Input dari pengguna akan dikirimkan ke JavaScript melalui *event listener* untuk memvalidasi data dan mengirimkan ke *endpoint back-end*.

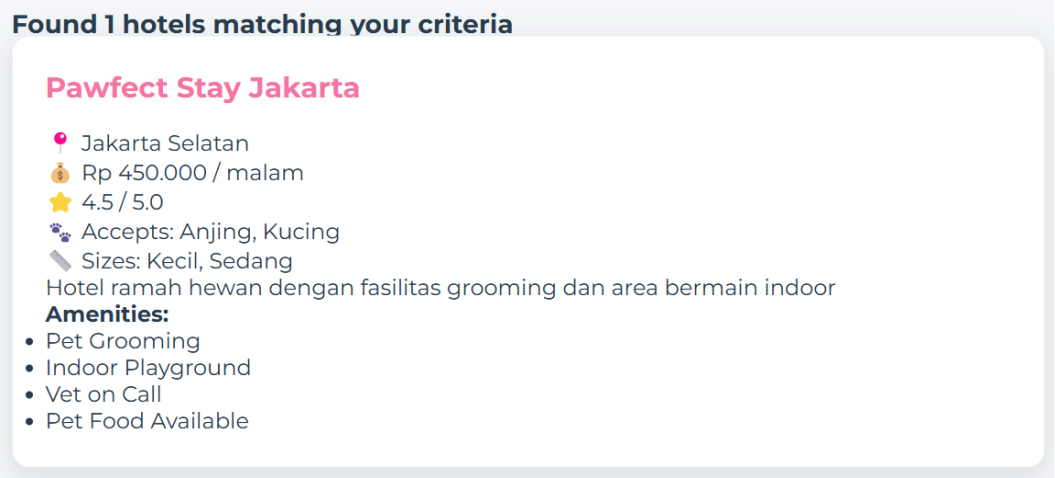


2) **Proses pencarian dilakukan pada *endpoint* /search-hotels** yang akan menggunakan filter untuk mencocokan database Supabase dengan filter dari pengguna.



3) **Proses penyimpanan riwayat pencarian dilakukan pada *endpoint* /save-search**.

4) Menampilkan **hasil pencarian**



Fitur ***Recommendation*** ditujukan untuk memberikan rekomendasi hotel berdasarkan riwayat pencarian pengguna dengan langkah-langkah sebagai berikut.

1) **Data yang dimasukkan pengguna diterima oleh *endpoint* /api/recommendations/{user_id}** untuk mendapatkan riwayat pencarian pengguna dari tabel search_history dalam 30 hari terakhir.

```
def get_user_search_history(user_id: str) -> List[Dict]:
    try:
            # Get search history from last 30 days for more relevant
recommendations
        thirty_days_ago = (datetime.now() - timedelta(days=30)).isoformat()
        result = supabase.table("search_history")\
            .select("*")\
            .eq("user_id", user_id)\
            .gte("created_at", thirty_days_ago)\
            .execute()
        return result.data
    except Exception as e:
        logger.error(f"Error fetching search history: {str(e)}")
        return []
```

2) Proses **perhitungan rekomendasi** dilakukan dengan **mengumpulkan data dari tabel search_history dan hotels.** Setiap hotel akan **dinilai berdasarkan kecocokan dengan riwayat pencarian pengguna**, **bobot tambahan akan diberikan** untuk pencarian terbaru dan peringkat hotel, dan **skor normalisasi** digunakan untuk memastikan nilai di rentang 0 sampai 1.

```
for hotel in hotels:
        total_score = 0
```

```
        matches = 0

        for search, weight in zip(search_history, weights):
            score = 0

            # Location matching (exact match gets higher score)
            if search.get('location') and hotel.get('location'):
                                if  search['location'].lower()  ==
hotel['location'].lower():
                    score += 3 * weight
                                elif  search['location'].lower()  in
hotel['location'].lower():
                    score += 1 * weight

            # Price range matching
            if hotel.get('price_per_night'):
                min_price = search.get('min_price', 0)
                max_price = search.get('max_price', float('inf'))
                        if  min_price  <=  hotel['price_per_night']  <=
max_price:
                    score += 2 * weight

            # Pet category matching
                                if   search.get('pet_category')   and
hotel.get('pet_categories'):
                                    if   search['pet_category']   in
hotel['pet_categories']:
                    score += 2 * weight

            # Pet size matching
            if search.get('pet_size') and hotel.get('pet_sizes'):
                if search['pet_size'] in hotel['pet_sizes']:
                    score += 2 * weight

            if score > 0:
                matches += 1
                total_score += score

         # Adjust final score based on number of matches and hotel
rating
        if matches > 0:
            avg_score = total_score / matches
                rating_bonus  =  hotel.get('rating',  0)  *  0.2   # Add
rating bonus
```

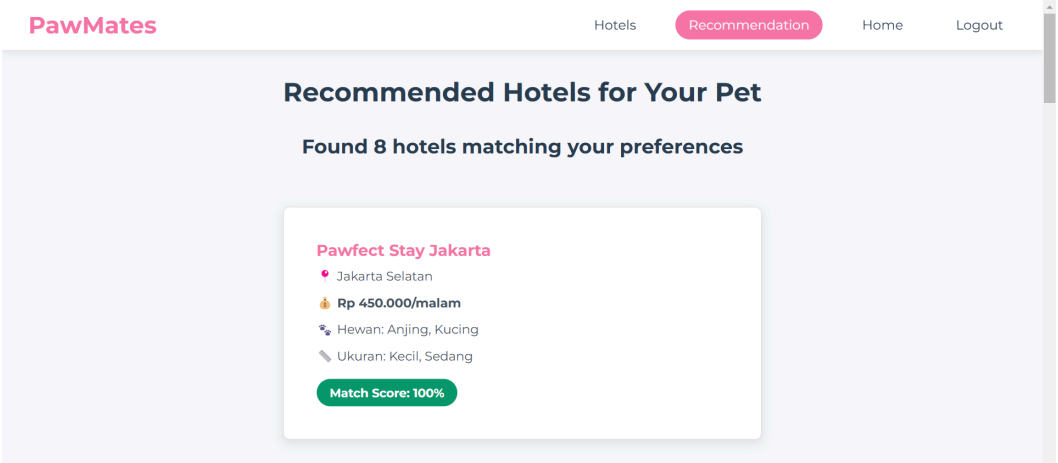
```
            final_score = avg_score + rating_bonus

            hotel_scores.append({
                **hotel,
                'recommendation_score': final_score
            })

    # Normalize scores between 0 and 1
    if hotel_scores:
        scores = [h['recommendation_score'] for h in hotel_scores]
        min_score, max_score = min(scores), max(scores)

        for hotel in hotel_scores:
            hotel['recommendation_score'] = normalize_score(
                hotel['recommendation_score'],
                min_score,
                max_score
            )

                return sorted(hotel_scores, key=lambda x:
x['recommendation_score'], reverse=True)
```

3) **Penyaringan dan pengembalian hasil** dilakukan dengan **memfilter hasil rekomendasi** untuk mencakup hotel dengan **skor kecocokan > 0.5 atau 50%** dan **10 hotel dengan skor tertinggi** akan ditampilkan pada pengguna.

```
# Calculate recommendations
        all_recommendations = calculate_hotel_scores(search_history, hotels)

        # Filter recommendations with score > 0.5 (50%)
        filtered_recommendations = [
            hotel for hotel in all_recommendations
            if hotel['recommendation_score'] > 0.5
        ]

        if not filtered_recommendations:
            return JSONResponse(
                status_code=200,
                content={
                    "recommendations": [],
                    "message": "No hotels match your preferences"
                }
            )

        # Return top 10 from filtered recommendations
        return JSONResponse(
            status_code=200,
```

```
            content={
                "recommendations": filtered_recommendations[:10],
                    "message": f"Found {len(filtered_recommendations)} hotels
matching your preferences"
            }
        )
```

4) Menampilkan **hasil rekomendasi**

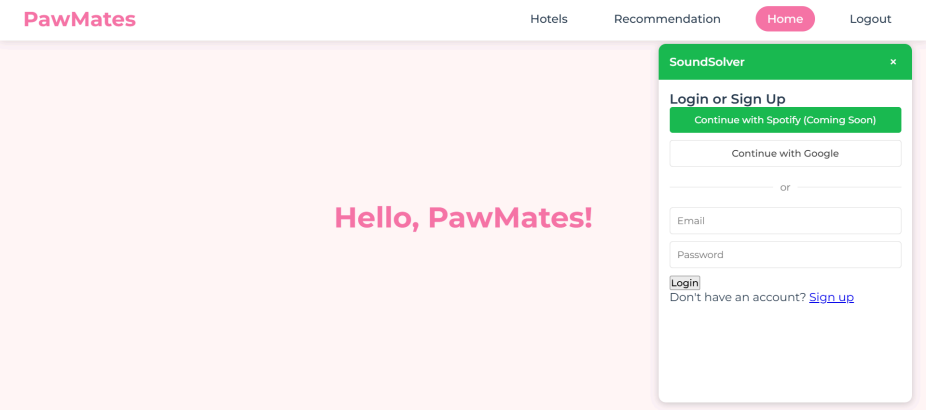


3. *Integration With Other Service (Customer Service and Support)*

Fitur ***Customer Service and Support*** ditujukan agar pengguna dapat meminta bantuan kepada ***Chatbot*** apabila menghadapi masalah. Penerapan *service* yang diintegrasikan dengan *service* lain adalah sebagai berikut.

```
<!-- Music Widget Integration -->
    <script>
        window.MUSICMATE_API_KEY = 'mk_JuwP4ZUPSnUaiqblKfyW7hDkDTYntZLMVMdn9he9yuI';
    </script>
                                                                    <script
src="https://spotify-bot.azurewebsites.net/static/js/widget-loader.js"></script>
```

Service ini hanya diterapkan pada halaman *home* dengan tampilan sebagai berikut.



IX. Specification Implementation

Berikut merupakan implementasi dari spesifikasi pada tugas besar ini.

1. Docker

Website-Based Pet Hotel Service - PawMates menggunakan Docker yang juga akan berguna untuk mengemas aplikasi ke dalam *container* sehingga memudahkan proses *deployment* dengan Railway. Pembangunan dan menjalankan *container* menggunakan docker-compose up –build.

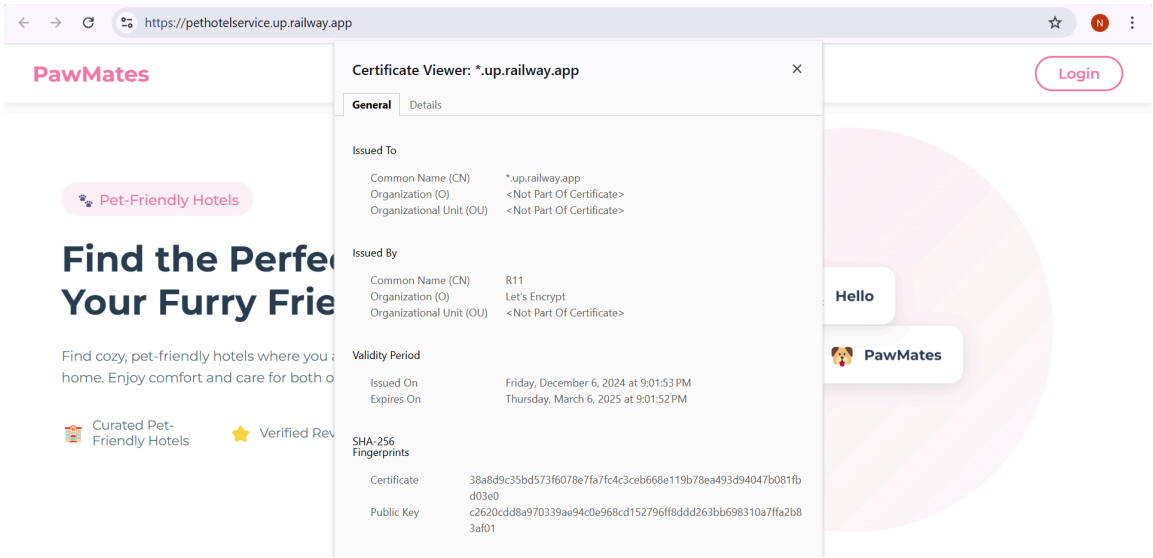
docker-compose.yml

```yaml
version: '3.8'

services:
  app:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8000:8000"
    volumes:
      - .:/app
    env_file:
      - .env
    command: ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

```dockerfile
# Use an official Python runtime as a parent image
FROM python:3.10-slim

# Set the working directory in the container
WORKDIR /app

# Copy the requirements file into the container
COPY requirements.txt .

# Install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code
COPY . .

# Expose the port the app runs on
EXPOSE 8000

# Command to run the application
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

2. HTTPS

Website-Based Pet Hotel Service - PawMates menggunakan HTTPS yang dihasilkan secara langsung setelah di-*deploy* dengan menggunakan Railway.



3. **Integrasi Antar *Service* Menggunakan *API Key Authentication***

Website-Based Pet Hotel Service - PawMates memiliki *middleware* API Key Authentication dan *endpoint Recommendations* dengan autentikasi. *Service client* perlu mendapatkan API *key* terlebih dahulu dengan memanggil *endpoint* generate-api-key dengan curl https://pethotelservice.up.railway.app/api/generate-api-key/[CLIENT] dan perlu menyertakan API *key* dalam *header* setiap *request* ke *endpoint Recommendations.* Berikut adalah contoh implementasi dengan menggunakan FastAPI di *service client*.

```python
from fastapi import FastAPI
import httpx

app = FastAPI()
API_KEY = "your_generated_api_key"
BASE_URL = "https://[URL-SERVICE-UTAMA]"

@app.get("/get-recommendations")
async def fetch_recommendations():
    headers = {
        "X-API-Key": API_KEY
    }
    async with httpx.AsyncClient() as client:
        response = await client.get(
            f"{BASE_URL}/api/recommendations",
            headers=headers
        )
        return response.json()
```

LAMPIRAN

Link GitHub : https://github.com/naomipricillaa/PetHotelService-18222065.git

Link Deployment : https://pethotelservice.up.railway.app/

Link API Docs : https://pethotelservice.up.railway.app/docs