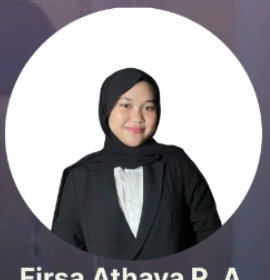
II3160

Integrated Systems Technology

Pica's Coworking Space

Platform Reservasi Tempat Duduk Terautentikasi



Firsa Athaya R. A.
18222051

Daftar Isi

Business Capability Model

18222051 – Firsa Athaya

CORE CAPABILITIES AND VALUE CHAIN

VENDOR AND PARTNER MANAGEMENT

SELECTION AND ENROLLMENT

Supplier Management

- Supplier Onboarding
- Contract Management
- Performance & Compliance Monitoring

PRODUCT AND SERVICE MANAGEMENT

SELECTION

Operations Management

- Service Customization
- Supplier and Inventory Management
- Pricing Management

MEMBERSHIP MANAGEMENT

ENROLLMENT AND MANAGEMENT

Membership

- Account Creation
- Membership Renewals
- Member Benefits

BOOKING AND FULFILLMENT

PROCESSING

Transaction Management

- Booking Management
- Payment Processing

CUSTOMER EXPERIENCE MANAGEMENT

ENGAGEMENT

- Personalized Recommendations

GENERIC CAPABILITIES

TECHNOLOGY AND PLATFORM MANAGEMENT

IT Infrastructure

- IT Specialist
- Software Development
- Data Analytics

Product Management

- User Interface
- API Integration

Transaction Management

- Fraud Prevention

Brand Explansion

- Brand Management
- Market Expansion

SUPPORTING CAPABILITIES

Human Capital Management

- HR Manager & Performance Management
- Training Specialist

Customer Support Roles

- Customer Success Manager
- Partnership Support Specialist

Compliance & Risk Management

- Legal Compliance
- Risk Assessment & Mitigation
- Insurance Coverage

EXECUTIVE SUMMARY

Tugas Teknologi Sistem Terintegrasi ini bertujuan untuk membangun sebuah sistem bernama Pica's Coworking Space, sebuah platform berbasis web yang memungkinkan pengguna untuk melakukan reservasi tempat duduk di coworking space. Platform ini memiliki fitur utama reservasi tempat duduk yang terintegrasi dengan sistem autentikasi login dan sign-up. Selain itu, terdapat fitur tambahan berupa rekomendasi obat, yang **diintegrasikan dengan layanan milik Ricky Wijaya (18222043), yaitu MediMatch.** MediMatch memberikan rekomendasi obat berdasarkan data yang dikirim dari sistem ini.

Proyek ini menggunakan:

- **Github**: https://github.com/firsaaa/Tugas-TST
- Frontend: https://www.picacoworkingspace.site/
- Backend: https://api.picacoworkingspace.site/
- Database: Supabase untuk menyimpan data pengguna, reservasi, dan integrasi dengan MediMatch.
- Autentikasi API: Endpoint dibagi menjadi public dan secure (login, register, reservasi, rekomendasi obat) dengan validasi API key.

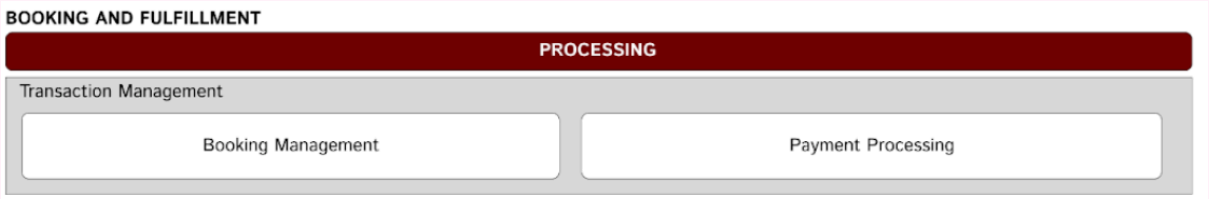
Proyek ini menggunakan metode integrasi yang menjaga keamanan data, memastikan akses ke layanan MediMatch hanya dapat dilakukan dengan API key yang valid.

Platform ini tidak hanya membantu pengguna dalam manajemen reservasi tempat duduk, tetapi juga memberikan nilai tambah melalui fitur rekomendasi obat untuk meningkatkan produktivitas pengguna selama berada di coworking space.

CORE CAPABILITY

Deskripsi Bisnis:
Berikut adalah Business Capability Model dari bisnis kombinasi co-working space, perpustakaan, dan café, di mana pelanggan bisa memesan tempat dan membeli makanan/minuman melalui sistem reservasi *online*. Selain itu, perpustakaan juga menyediakan *coworking space* dimana *customer* dapat melakukan reservasi terlebih dahulu sebelum menggunakan *coworking space*.

Domain yang dipilih:

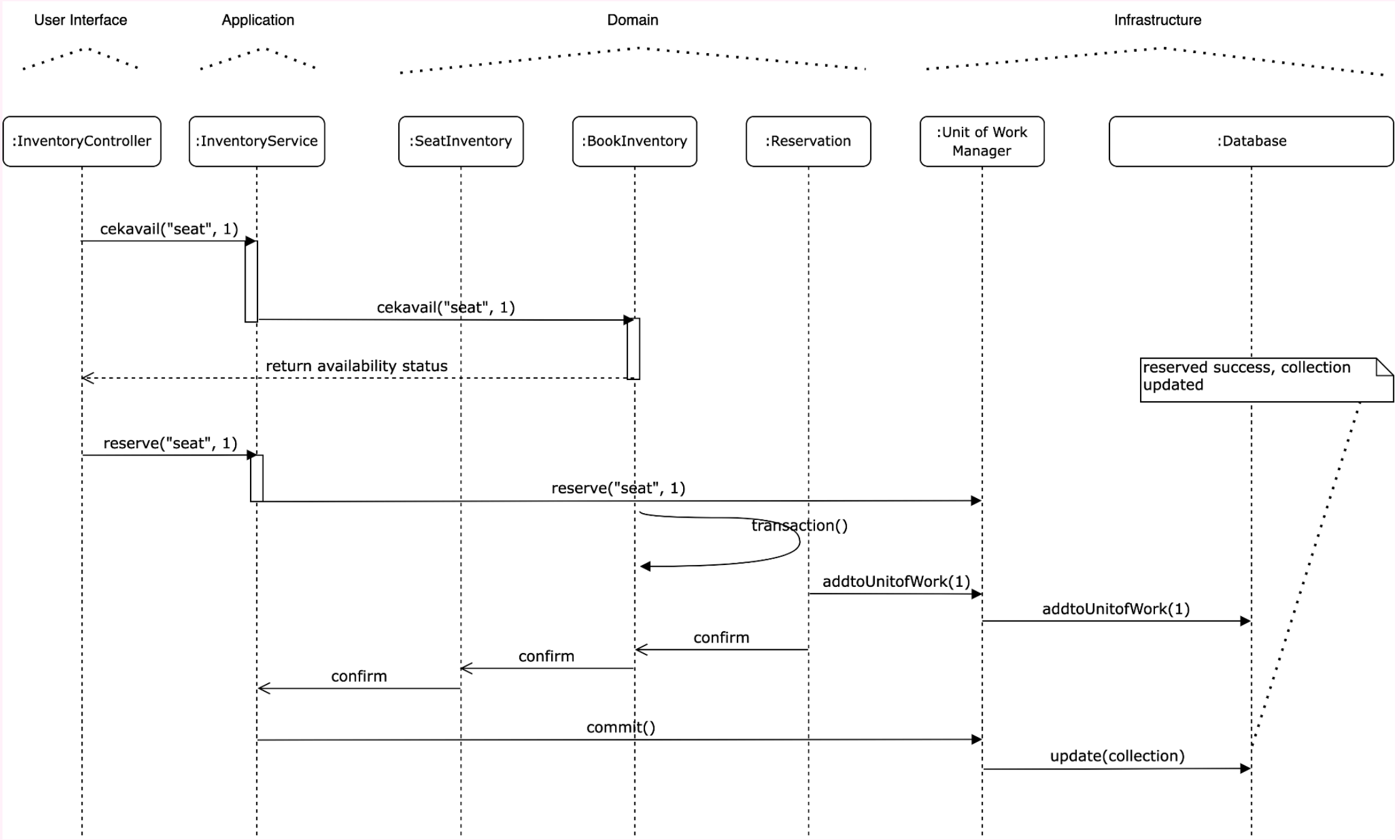


Domain *booking and fullfillment* memberikan transparansi kepada pelanggan mengenai jumlah stok buku dan ketersediaan tempat duduk.

Subdomain: Booking Management
Deskripsi Fungsi: Sistem ini akan mengelola dan menampilkan ketersediaan tempat duduk di *co-working space secara real-time*. Pelanggan bisa memantau jumlah kursi kosong di area *co-working space* dan *café* sebelum melakukan reservasi.

Komponen Utama:

1. **Real-time Monitoring:** Mengupdate jumlah ketersediaan kursi secara langsung.
2. **Automated Notifications:** Mengirimkan notifikasi otomatis saat kursi tersedia atau tidak tersedia.
3. **User Interface:** Tampilan visual yang memudahkan pelanggan untuk melihat ketersediaan langsung di website atau aplikasi.
4. **Integrasi dengan Sistem Pemesanan:** Menghubungkan ketersediaan kursi dengan sistem pemesanan.


User Interface
Application
Domain
Infrastructure
:InventoryController
:InventoryService
:SeatInventory
:BookInventory
:Reservation
:Unit of Work Manager
:Database
cekavail("seat", 1)
cekavail("seat", 1)
return availability status
reserved success, collection updated
reserve("seat", 1)
reserve("seat", 1)
transaction()
addtoUnitofWork(1)
addtoUnitofWork(1)
confirm
confirm
confirm
commit()
update(collection)

SOFTWARE ARCHITECTURE

Arsitektur Sistem dari sistem *Management Inventory*:

1. **User Interface**
   InventoryController: Menerima permintaan dari *user* untuk mengecek ketersediaan kursi dan stok buku secara *real-time*.

2. **Application**
   InventoryService: Berkomunikasi dengan lapisan domain untuk mendapatkan dan memperbarui data terkait kursi dan buku agar dapat di-*update* secara *real-time*.

3. **Domain**
   SeatInventory: Mengelola ketersediaan kursi di *co-working space* dan *café* dengan melacak status dari setiap kursi (kosong, dipesan, atau sedang dipakai).
   BookInventory: Mengelola stok buku, baik yang baru maupun *pre-loved*.
   Reservation: Mengatur proses pemesanan kursi, termasuk memperbarui status kursi di inventori.

4. **Infrastructure**
   Unit of Work Manager: Mengelola seluruh proses transaksi, memastikan bahwa jika terjadi kegagalan pada saat reservasi atau pembaruan inventori, seluruh proses akan dibatalkan.
   Database: Penyimpanan untuk data inventori, meliputi jumlah kursi yang tersedia dan stok buku.

TEKNOLOGI SOFTWARE ARCHITECTURE

Berikut adalah teknologi yang digunakan pada *Software Architecture* beserta alasannya:

1. **Frontend (UI Layer)**
   Teknologi **:** HTML, CSS , JavaScript
   Alasan **:** Interaktif dan responsif untuk pengguna.

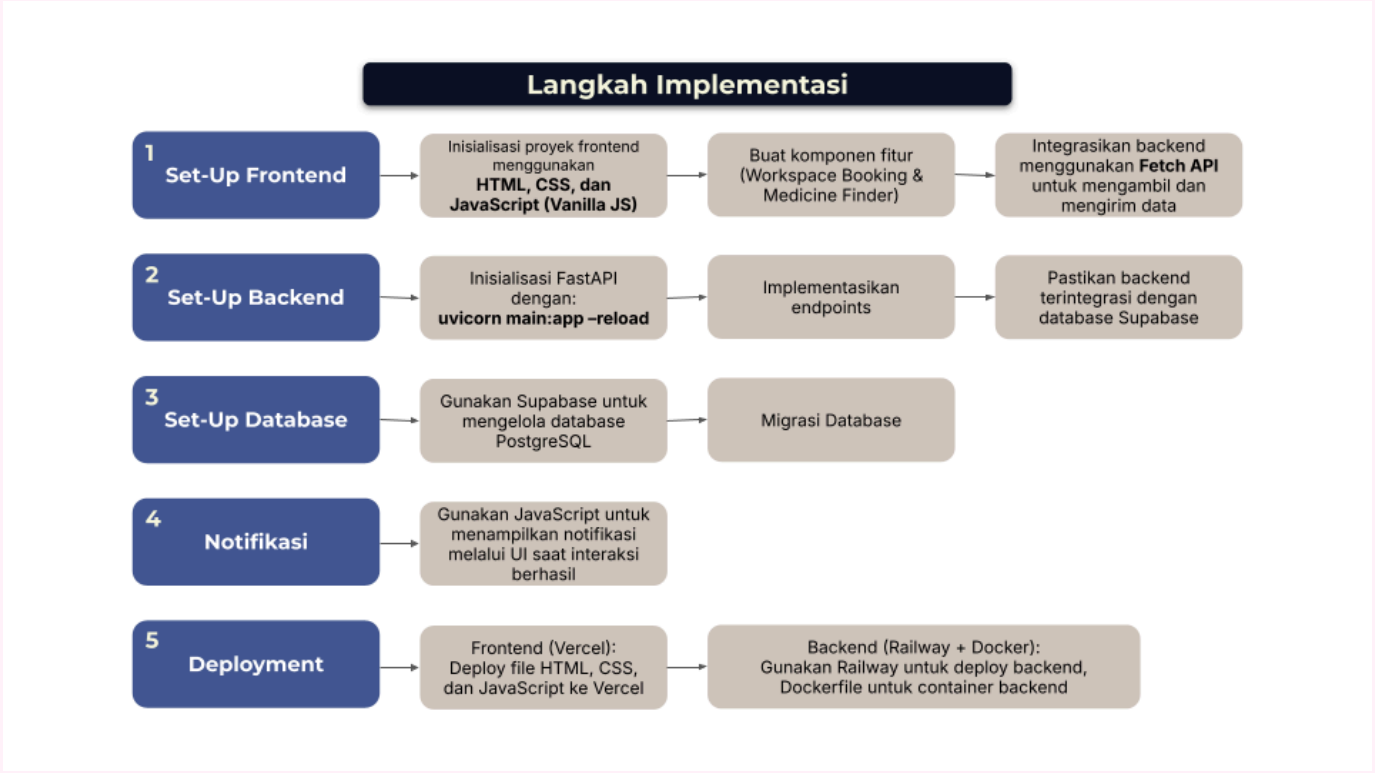
2. **Backend (Application Layer)**
   Teknologi **:** FastAPI, Railway
   Alasan **:** FastAPI cepat untuk membangun API dan mendukung documentation otomatis berbasis OpenAPI, yang memudahkan pengembang lain untuk berinteraksi dengan API melalui dokumentasi yang sudah disediakan secara otomatis.

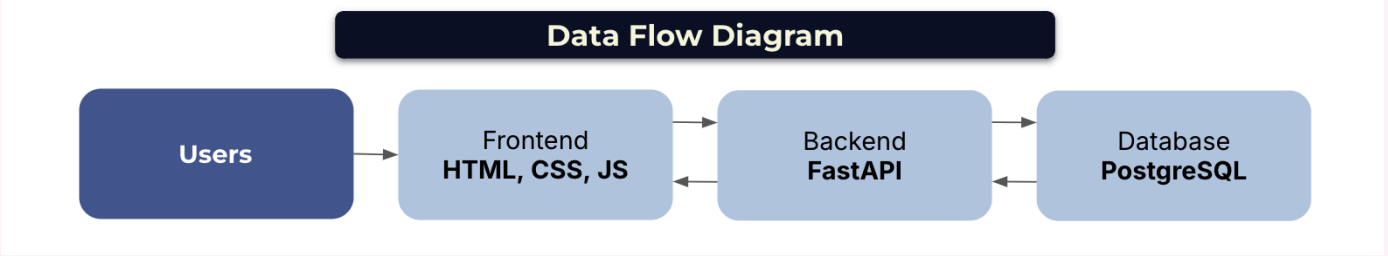
3. **Database Layer**
   Teknologi **:** PostgreSQLmelalui Supabase
   Alasan **:** Supabase mempermudah pengelolaan database tanpa memikirkan infrastruktur dan *real-time capability.*

4. **Hosting and Deployment**
   Teknologi **:** Vercel (Frontend), Railway+Docker (Backend)
   Alasan **:** Vercel cocok untuk deployment karena mendukung integrasi cepat dengan React.js atau framework frontend lainnya. Railway mempermudah deploy dengan membaca Dockerfile untuk membangun container aplikasi.

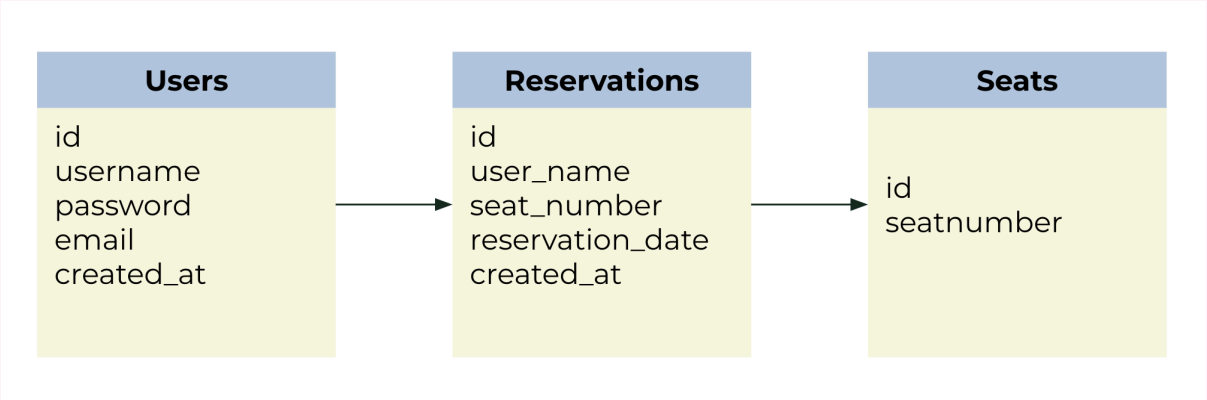
RENCANA IMPLEMENTASI

Berikut adalah langkah-langkah implementasi yang dilakukan dalam pengembangan proyek:





Rincian Setup Database



Berikut adalah timeline pengerjaan proyek:

Timeline

Action	November				Desember			
	W1	W2	W3	W4	W1	W2	W3	W4
Persiapan dan Set-Up Proyek	■							
Pengembangan Frontend Dasar		■						
Pengembangan Backend			■					
Integrasi Frontend dan Backend				■				
Pengujian, Debugging, dan Validasi					■			
Deployment dan Penyesuaian Akhir						■		

Features

1. Register dan login pengguna
    - Deskripsi: Pengguna dapat membuat akun baru melalui fitur registrasi dan masuk ke sistem menggunakan fitur login.
    - Tujuan: Memastikan hanya pengguna yang terdaftar yang dapat mengakses fitur utama sistem.
    - Teknologi yang Digunakan:
        - Backend: Endpoint FastAPI dengan validasi username dan password.
        - Keamanan: Hashing password menggunakan bcrypt.
    - Endpoint:
        - POST /api/public/register: Untuk registrasi pengguna baru.
        - POST /api/public/login: Untuk autentikasi pengguna.
2. Reservasi tempat duduk
    - Deskripsi: Pengguna dapat memesan tempat duduk di coworking space melalui sistem reservasi.
    - Tujuan: Memudahkan pengelolaan tempat duduk sehingga pengguna mendapatkan kepastian ketersediaan sebelum datang.
    - Teknologi yang Digunakan:
        - Backend: FastAPI untuk manajemen reservasi.
        - Frontend: Formulir di antarmuka pengguna untuk input data reservasi.
        - Database: Penyimpanan data reservasi menggunakan Supabase.
    - Fitur Tambahan:
        - Validasi tanggal reservasi.

        - Cancel/Complete Reservation: Sistem otomatis mengubah status reservasi jika melewati tanggal yang sudah dipesan.
    - Endpoint:
        - POST /api/secure/reservations: Membuat reservasi baru.
        - GET /api/secure/my-reservations: Melihat daftar reservasi pengguna.
        - DELETE /api/secure/reservations/{id}: Membatalkan reservasi.

3. Rekomendasi Obat (Integrasi dengan Medimatch)

Integrasi dengan Layanan MediMatch

Dalam proyek ini, salah satu fitur yang ditawarkan adalah rekomendasi obat untuk pengguna. Fitur ini diimplementasikan dengan memanfaatkan **API Medimatch**, sebuah layanan milik Ricky Wijaya (NIM: 18222043). Medimatch dirancang untuk memberikan rekomendasi obat berdasarkan nama obat yang dimasukkan pengguna. Berikut adalah rincian integrasi:

Proses Integrasi:

1. **Endpoint pada Backend**
    - Endpoint /api/secure/recommend-drugs dibuat di backend Anda untuk menerima input berupa nama obat (drug_name) dan jumlah rekomendasi (top_n).
    - Input tersebut kemudian diteruskan ke API Medimatch untuk mendapatkan data rekomendasi obat.
    - API Medimatch mengharuskan adanya **API Key** yang valid untuk memproses permintaan. API Key ini disimpan secara aman di backend Anda dan hanya dikirimkan pada saat mengakses layanan Medimatch.
2. **Flow Permintaan**
    - Pengguna memasukkan nama obat dan jumlah rekomendasi melalui antarmuka frontend.
    - Data yang dimasukkan pengguna diteruskan ke backend.
    - Backend memvalidasi permintaan, mengakses API Medimatch dengan API Key, dan mengembalikan hasil rekomendasi obat kepada pengguna.
3. **Dokumentasi API**
    - Endpoint /api/secure/recommend-drugs:
        - **Method**: POST

          Request Body:
          ```
          {
            "drug_name": "string",
            "top_n": 5
          }
          ```

        - **Headers**:

api-key: API Key untuk otorisasi.

Response: Rekomendasi obat dalam format JSON.


```
{
  "success": true,
  "data": {
    "name": {
      "1278": "Acetyl Salicylic Acid 50mg Tablet",
      "1627": "Aspin 100mg Tablet"
    },
    "combined_composition": {
      "1278": "Aspirin (50mg)",
      "1627": "Aspirin (100mg)"
    }
  }
}
```


Alasan Integrasi

Integrasi ini memberikan nilai tambah pada proyek **Pica's Coworking Space** dengan menyediakan fitur yang relevan dan bermanfaat bagi pengguna. Pengguna tidak hanya dapat melakukan reservasi tempat duduk, tetapi juga mendapatkan rekomendasi obat yang sesuai melalui integrasi layanan pihak ketiga.

Hasil Implementasi

- **Keamanan**: Validasi API Key memastikan hanya backend yang dapat mengakses layanan Medimatch.
- **Kemudahan Penggunaan**: Proses integrasi berjalan di latar belakang sehingga pengguna mendapatkan pengalaman yang mulus.
- **Keandalan**: API Medimatch memberikan hasil yang akurat berdasarkan database obat yang dimilikinya.

Tantangan

- Penyesuaian format input dan output antara layanan backend Anda dengan API Medimatch.
- Pengelolaan API Key secara aman untuk mencegah penyalahgunaan.

API AUTHENTICATION

Dalam proyek ini, API Authentication digunakan untuk melindungi layanan penting seperti reservasi tempat dan integrasi rekomendasi obat dari layanan Medimatch. Jika endpoint ini tidak diamankan, siapa pun dapat mengaksesnya, yang dapat menyebabkan kebocoran data atau penyalahgunaan layanan. Oleh karena itu, API key authentication diterapkan untuk memberikan lapisan keamanan tambahan.

Pada proyek ini, API *key authentication* diterapkan dengan membagi *endpoint* menjadi 2 kategori, yaitu:

Public Endpoint: Dapat diakses tanpa API *key*.

Secure Endpoint: Hanya dapat diakses dengan API *key* yang valid.

Dengan implementasi sebagai berikut:

a. **Menambahkan Dependency API Key Validation**

File auth.py dibuat untuk mendefinisikan fungsi validasi API key:

```
from passlib.context import CryptContext
import os
from datetime import datetime, timedelta
from jose import JWTError, jwt
from dotenv import load_dotenv

load_dotenv()

pwd_context = CryptContext(schemes=["bcrypt"],
deprecated="auto")
SECRET_KEY = os.getenv("SECRET_KEY")
ALGORITHM = "HS256"
ACCESS_TOKEN_EXPIRE_MINUTES = 30

def verify_password(plain_password, hashed_password):
    return pwd_context.verify(plain_password,
hashed_password)

def get_password_hash(password):
    return pwd_context.hash(password)

def create_access_token(data: dict):
    to_encode = data.copy()
    expire = datetime.utcnow() +
timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES)
    to_encode.update({"exp": expire})
    encoded_jwt = jwt.encode(to_encode, SECRET_KEY,
algorithm=ALGORITHM)
    return encoded_jwt
```

b. **Menambahkan Public dan Secure Endpoint**

Router public.py dan secure.py dibuat untuk masing-masing jenis endpoint:

- public.py: Endpoint ini dapat diakses tanpa API key.

```
from fastapi import APIRouter
import os

router = APIRouter()

@router.get("/")
def read_public():
    return {"message": "Welcome to Coworking Space
API"}
```

- secure.py: Endpoint ini membutuhkan API key yang valid.

```
from fastapi import APIRouter, Depends
from auth import get_user

router = APIRouter()

@router.get("/")
async def get_testroute(user: dict =
Depends(get_user)):
    return user
```

c. Mengintegrasikan Router ke main.py

File main.py untuk menggabungkan router dari public.py dan secure.py:

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from app.routes import public, secure
from app.database import Base, engine
from dotenv import load_dotenv
import os

load_dotenv()

app = FastAPI()
firebase_api_key = os.getenv("FIREBASE_API_KEY")

# Handle database creation
try:
    Base.metadata.create_all(bind=engine)
except Exception as e:
    print(f"Database initialization error: {e}")

# Add CORS middleware
app.add_middleware(
    CORSMiddleware,
    allow_origins=[
        "http://127.0.0.1:5500",
        "https://coworkingspace-six.vercel.app",
        "https://coworkingspace-backend.vercel.app",
```

```
        "https://coworkingspace.up.railway.app"  \
    ],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Root route
@app.get("/")
def read_root():
    return {"message": "Welcome to the Coworking Space
API!"}

# Tambahkan route lainnya
app.include_router(public.router, prefix="/api/public",
tags=["Public"])
app.include_router(secure.router, prefix="/api/secure",
tags=["Secure"])
app.include_router(public.router)
```

Setelah itu, pengujian dapat dilakukan menggunakan Thunderclient dengan skenario berikut:

- **Public Endpoint** (/api/v1/public/)
  Request tanpa API key: 200 OK dengan response "OK".
  Public endpoint dapat diakses oleh semua klien.
- **Secure Endpoint** (/api/v1/secure/)
  - Tanpa API Key:
    Status: 403 Forbidden.
    Response: {"detail": "Not authenticated"}.
  - API Key Invalid:
    Status: 401 Unauthorized.
    Response: {"detail": "Missing or invalid API key"}.
  - API Key Valid:
    Status: 200 OK.
    Response:
    {
      "message": "Secure endpoint accessed successfully!",
      "user": {"message": "Valid API key"}
    }

KESIMPULAN

Proyek Pica's Coworking Space merupakan pengalaman yang penuh tantangan sekaligus menyenangkan. Pengembangan sistem berhasil mencakup fitur utama seperti **reservasi tempat duduk** dan **rekomendasi obat dari Medimatch,** serta penerapan sistem keamanan berbasis API Key.

Tantangan utama yang dihadapi meliputi **pengelolaan token API yang cukup kompleks, debugging selama integrasi API Medimatch, dan konfigurasi deployment menggunakan Docker di Railway.** Walaupun memerlukan banyak percobaan, semua tantangan ini memberikan pemahaman lebih mendalam tentang pengelolaan layanan terintegrasi dan keamanan API.

Pengalaman selama pengerjaan ini menunjukkan pentingnya kesabaran, ketelitian, dan dokumentasi yang baik dalam memastikan semua komponen sistem dapat berjalan sesuai harapan.

REFERENCES

Berry, T. (n.d.). *FastAPI with API keys*. Retrieved November 19, 2024, from https://timberry.dev/fastapi-with-apikeys