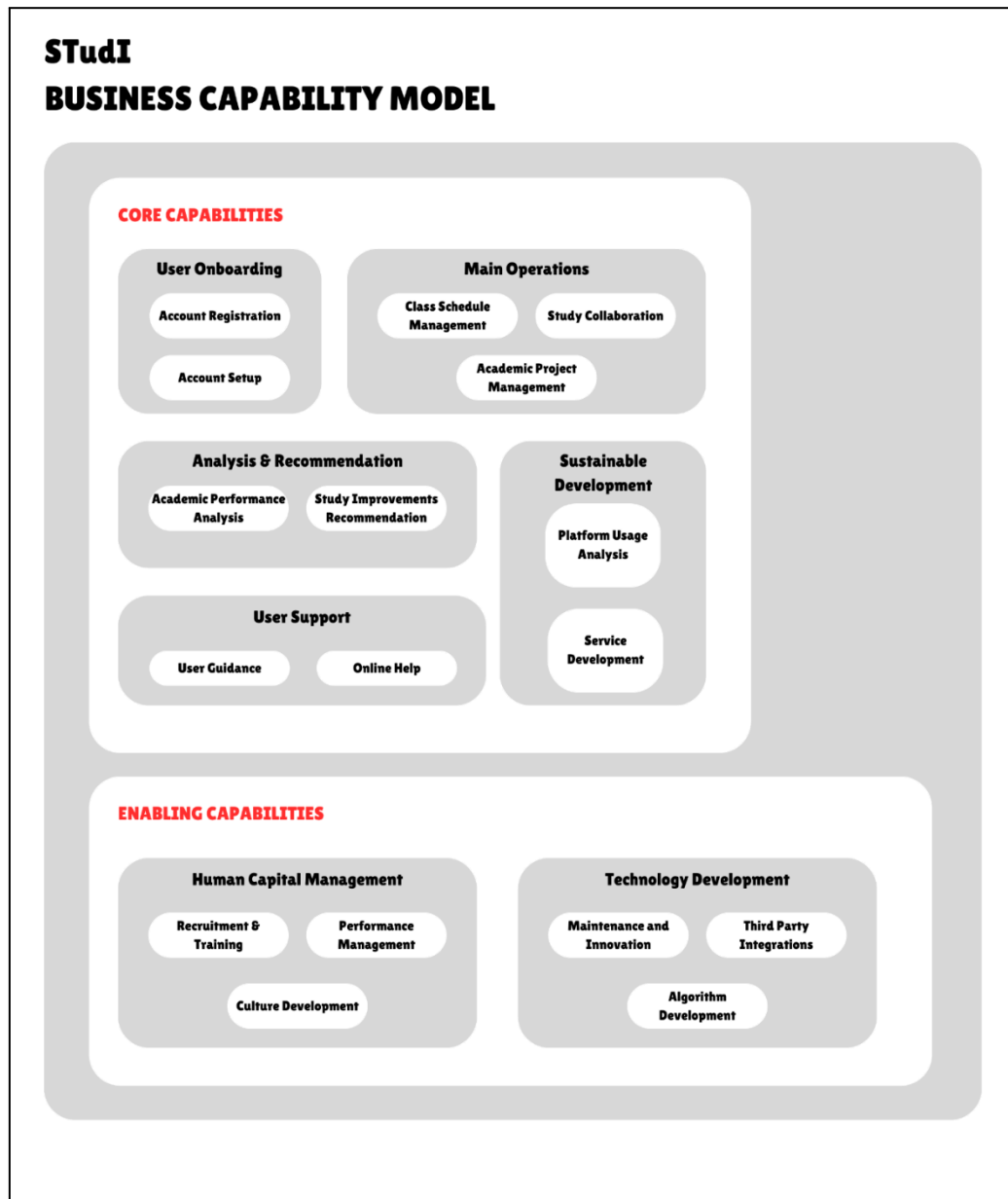


# Proyek Tugas Besar

## II3160 – Teknologi Sistem Terintegrasi

Oleh: Dama Dhananjaya Daliman (18222047)

### 1. Business Capability Model

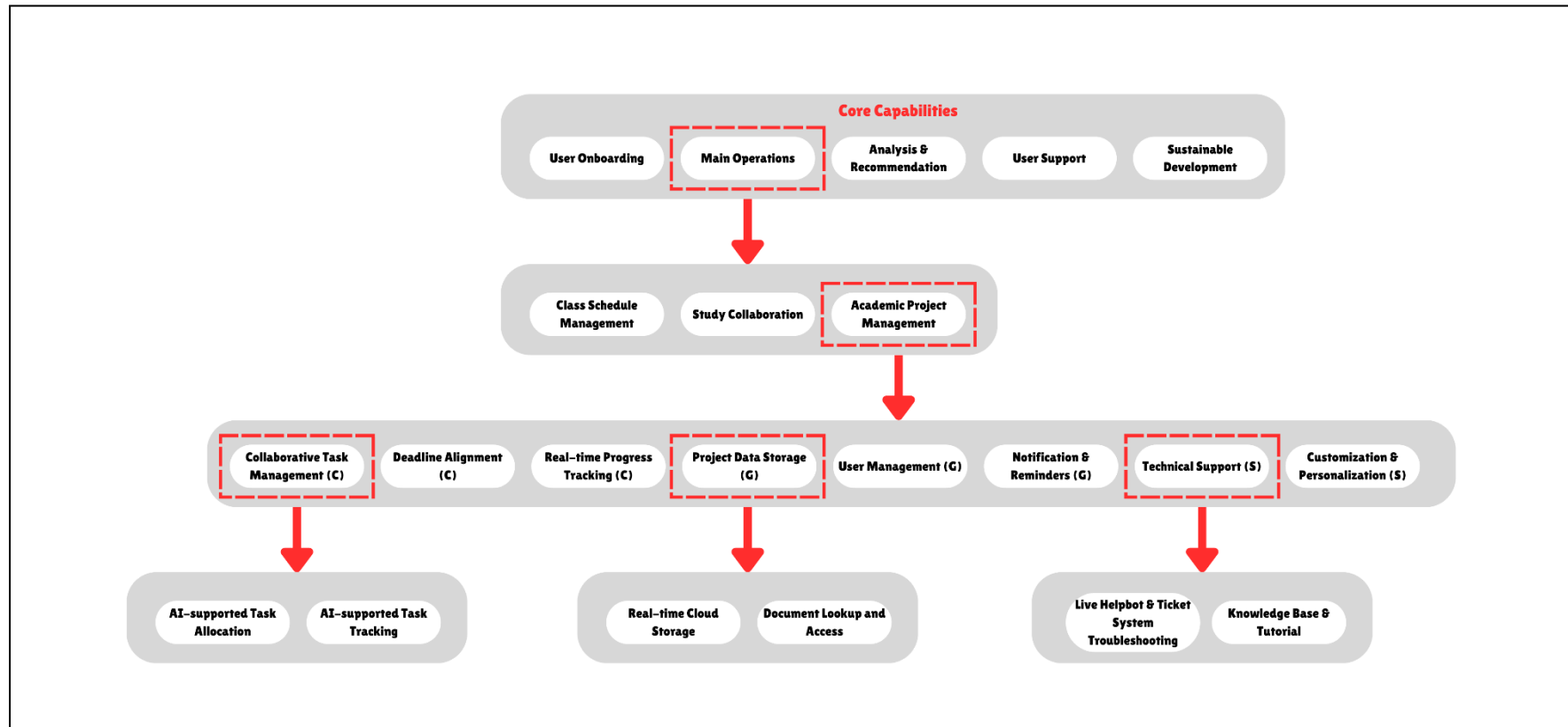


Gambar 1.1. Business Capability Model dari STudI

Ini adalah pemodelan bisnis STudI yaitu sebuah bisnis penyedia layanan dukungan akademik yang berfokus untuk mahasiswa STI. Pada business capability model (BCM) ini terlihat berbagai kemampuan yang ditawarkan bisnis ini, mulai dari awal ketika user mendaftar untuk layanan ini, layanan yang diberikan: manajemen jadwal, kelompok belajar, manajemen proyek, analisis dan rekomendasi studi, sampai ke kapabilitas pendukungnya.

## 2. Dekomposisi Subdomain

Dari BCM yang dibuat, dipilih domain dari main operations yang dirasa merupakan paling unik dan kapabilitas utama dari bisnisnya yaitu, Academic Project Management.

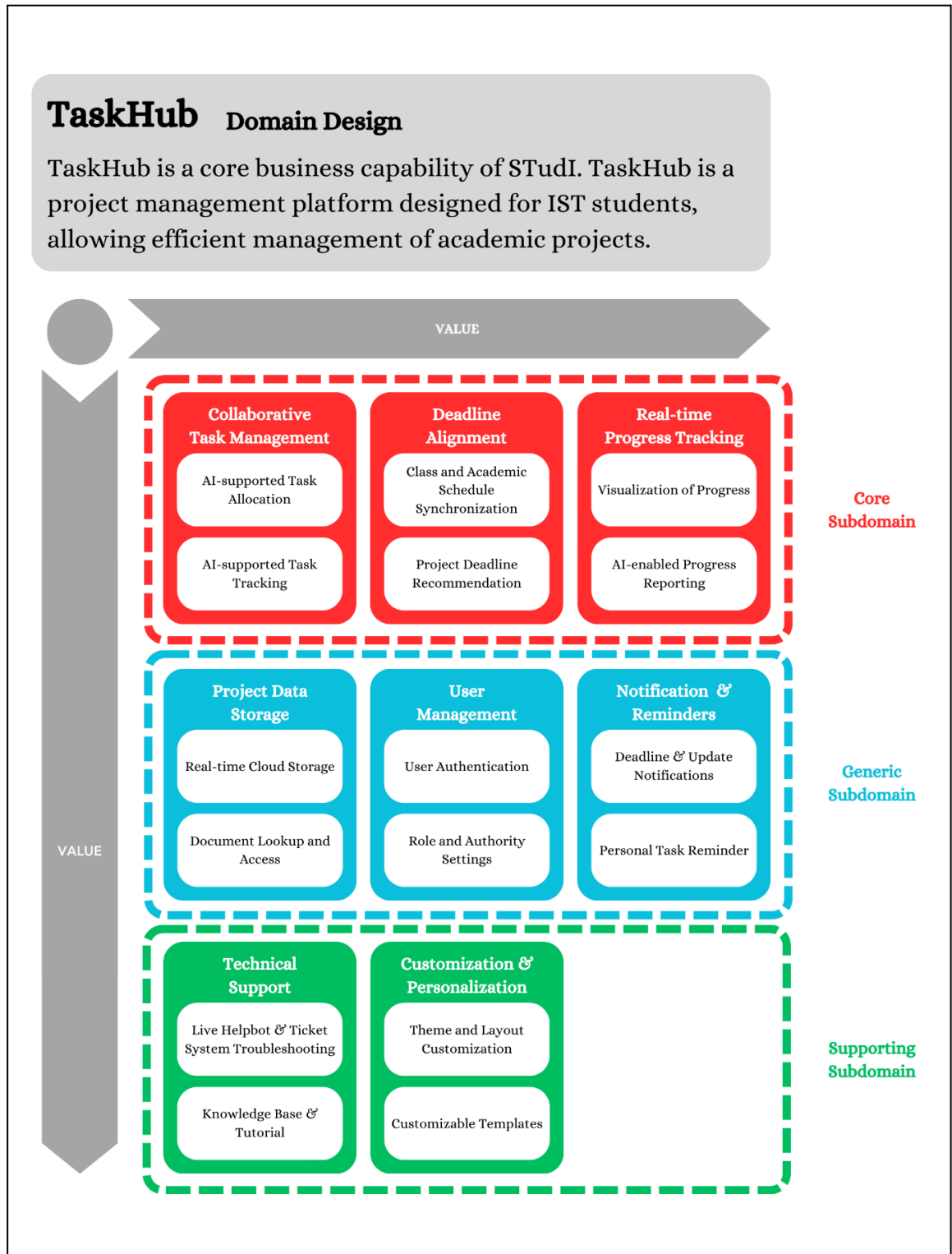


Gambar 2.1. Dekomposisi Capability menjadi Subdomain

Domain ini unik karena jarang ada kakas pendukung akademik yang berfokus pada manajemen proyek apalagi untuk proyek di STI. Selain itu, dengan bantuan AI dirasa bisa ada inovasi yang baru yang membantu proyek-proyek di STI.

### 3. Fungsi Subdomain

Dari domain yang dipilih, dibuatlah pemetaan terhadap subdomain, value, dan fungsi yang terkait untuk setiap subdomainnya, domain ini diberikan suatu nama layanan: TaskHub yang pemetaannya sebagai berikut,



Gambar 3.1. Pemetaan fungsi-fungsi subdomain

dari peta di atas, value tertinggi ada pada subdomain-subdomain core dari kiri ke kanan. Jadi, core subdomain dengan value tertinggi adalah pada bagian collaborative task management. Di sini ditawarkan AI-supported Task Allocation serta AI-supported Task Tracking. Ini memiliki potensi paling tinggi untuk menjadi kapabilitas unik dari layanan STudI dan TaskHub karena menyasar untuk membantu proyek-proyek akademik, terutama di prodi STI dengan bantuan AI.

Penjelasan setiap fungsi subdomain:

a. Core Subdomain:

- i. AI-supported Task Allocation: Sistem memfasilitasi pengguna untuk bisa mengalokasikan tugas yang didistribusikan dalam kelompok. Alokasi ini akan memanfaatkan AI untuk bisa merekomendasikan distribusi tugas yang cocok.
- ii. AI-supported Task Tracking: Sistem memfasilitasi pengguna untuk bisa memantau tugas yang sudah dialokasikan. Fungsi ini memanfaatkan AI untuk membantu memantau pemenuhan tugas berdasarkan kriteria yang ditentukan pengguna.
- iii. Class and Academic Schedule Synchronization: Sistem melakukan sinkronisasi dengan jadwal kelas dan kalender akademik untuk membantu sistem dan pengguna untuk menentukan tenggat waktu.
- iv. Project Deadline Recommendation: Sistem merekomendasikan tenggat waktu tugas sesuai dengan bobot alokasi tugas dan jadwal yang sudah disinkronisasi, pengguna bisa memilih untuk menggunakan tenggat yang direkomendasikan atau menetapkan tenggat sendiri.
- v. Visualization of Progress: Sistem memfasilitasi visualisasi dari kemajuan tugas via grafik, kemajuan disajikan dalam 3 (tiga) perspektif: kemajuan keseluruhan, kemajuan per anggota, kemajuan per tahapan tugas (jika proyek terbagi dalam beberapa tahapan).
- vi. AI-enabled Progress Reporting: Sistem memfasilitasi laporan kemajuan dengan memanfaatkan AI. Laporan yang dibuat dengan AI ditujukan untuk konsumsi tim sendiri.

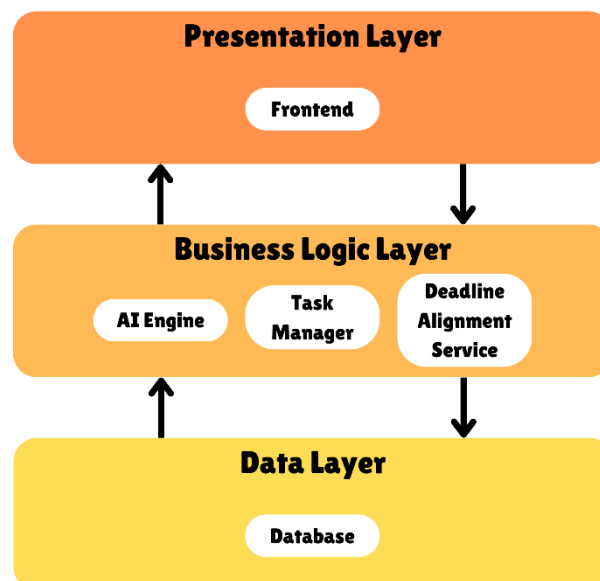
b. Generic Subdomain:

- i. Real-time Cloud Storage: Sistem memfasilitasi penyimpanan berbasis *cloud* untuk dokumen-dokumen yang relevan dengan proyek dan ingin dipantau.
- ii. Document Lookup and Access: Setelah sistem memfasilitasi penyimpanan dokumen, maka sistem juga harus bisa memberikan kemampuan mencari dan mengakses dokumen yang telah disimpan.
- iii. User Authentication: Sistem mewajibkan autentikasi dari semua user yang mengakses sistem.
- iv. Role and Authority Settings: Sistem memfasilitasi pemilik atau ketua proyek untuk memberikan peran (*role*) dan memberikan wewenang (*authority*) pada setiap peran sesuai yang tersedia di sistem.
- v. Deadline & Update Notification: Sistem memberikan notifikasi untuk tenggat waktu yang sudah ditetapkan dan untuk aktivitas-aktivitas yang perlu dipantau (berdasarkan keinginan pengguna).

- vi. Personal Task Reminder: Sistem memberikan pengingat untuk tugas-tugas setiap anggota yang sudah dialokasikan.
- c. Supporting Subdomain:
  - i. Live Helpbot & Ticket System Troubleshooting: Untuk mempermudah pengguna menggunakan sistem TaskHub, maka disediakan live helpbot dan troubleshooting yang berbasis tiket. Live helpbot adalah chatbot yang live untuk membantu pengguna secara real-time dan untuk permasalahan sederhana yang sering muncul. Sedangkan ticket system troubleshooting adalah sistem untuk membantu pengguna mengatasi permasalahan yang lebih kompleks oleh tim support yang ahli.
  - ii. Knowledge Base & Tutorial: Fungsi ini mendukung pengguna untuk memahami apa saja fitur yang ada di sistem, terutama fungsi tutorial yang bisa dimanfaatkan oleh pengguna baru untuk bisa familiar dengan sistem.
  - iii. Theme and Layout Customization: Fungsi ini mendukung pengguna untuk bisa menyesuaikan tema dan susunan tampilan sistem sesuai keinginan pengguna.
  - iv. Customizable Templates: Fungsi ini mendukung pengguna untuk membuat dokumen-dokumen umum dengan menyediakan template yang bisa dikustomisasi.

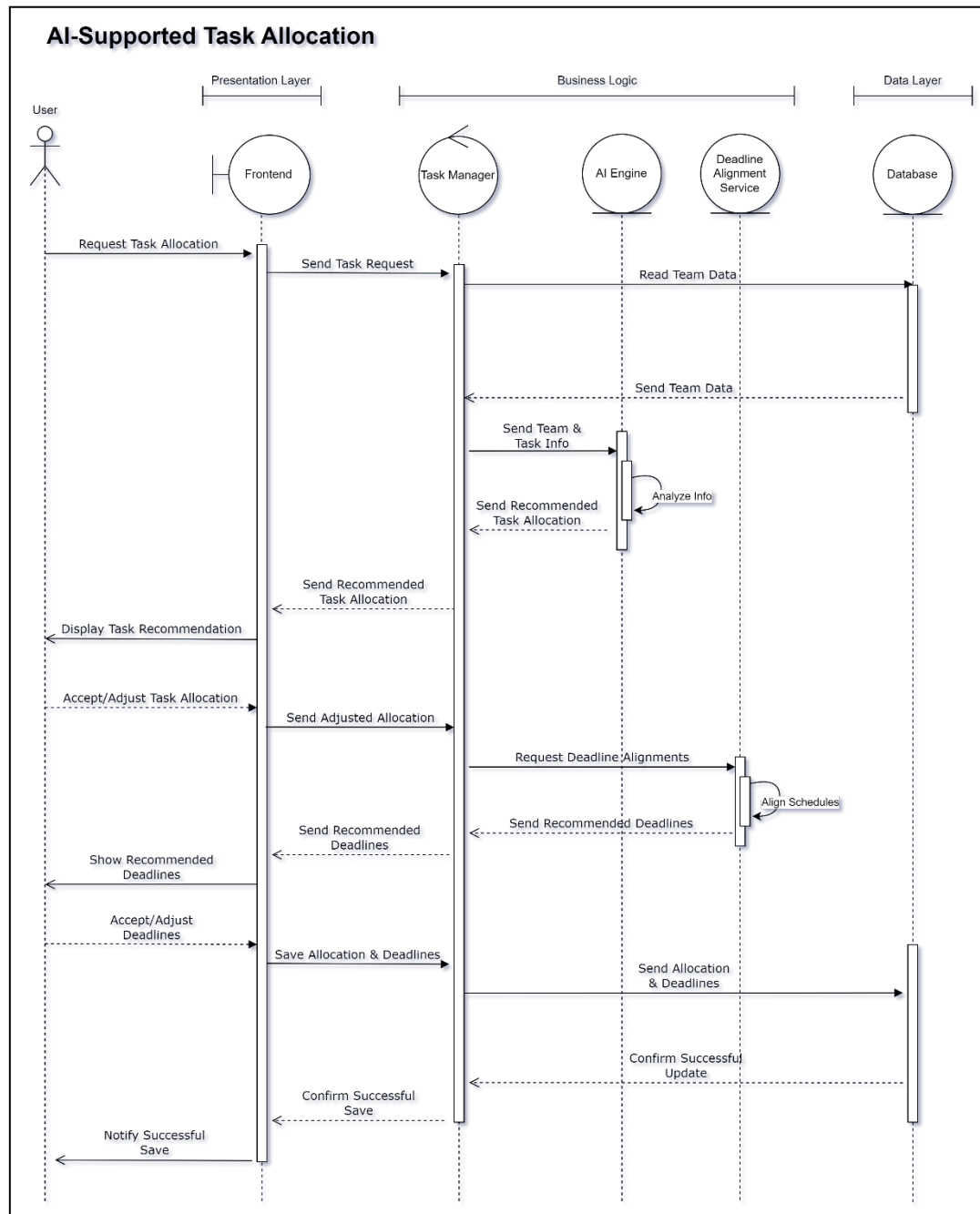
#### 4. Arsitektur yang Diajukan dan Model Prosesnya

Dari semua fungsi subdomain yang dijabarkan, untuk tugas ini dipilih untuk mengimplementasikan enam fungsi, yaitu empat fungsi dari core subdomain Collaborative Task Management dan Deadline Alignment serta dua fungsi dari generic subdomain Project Data Storage. Fungsi-fungsi ini akan diimplementasikan dalam bentuk aplikasi yang arsitekturnya berbentuk lapisan-lapisan (*layered*) yang umumnya dibagi menjadi 3 (tiga) yaitu, infrastruktur, logika bisnis, dan presentasi.



Gambar 4.1. Diagram Arsitektur Sistem

Selanjutnya penjelasan proses dari setiap fungsi akan digambarkan dengan dua sequence diagram berikut ini,

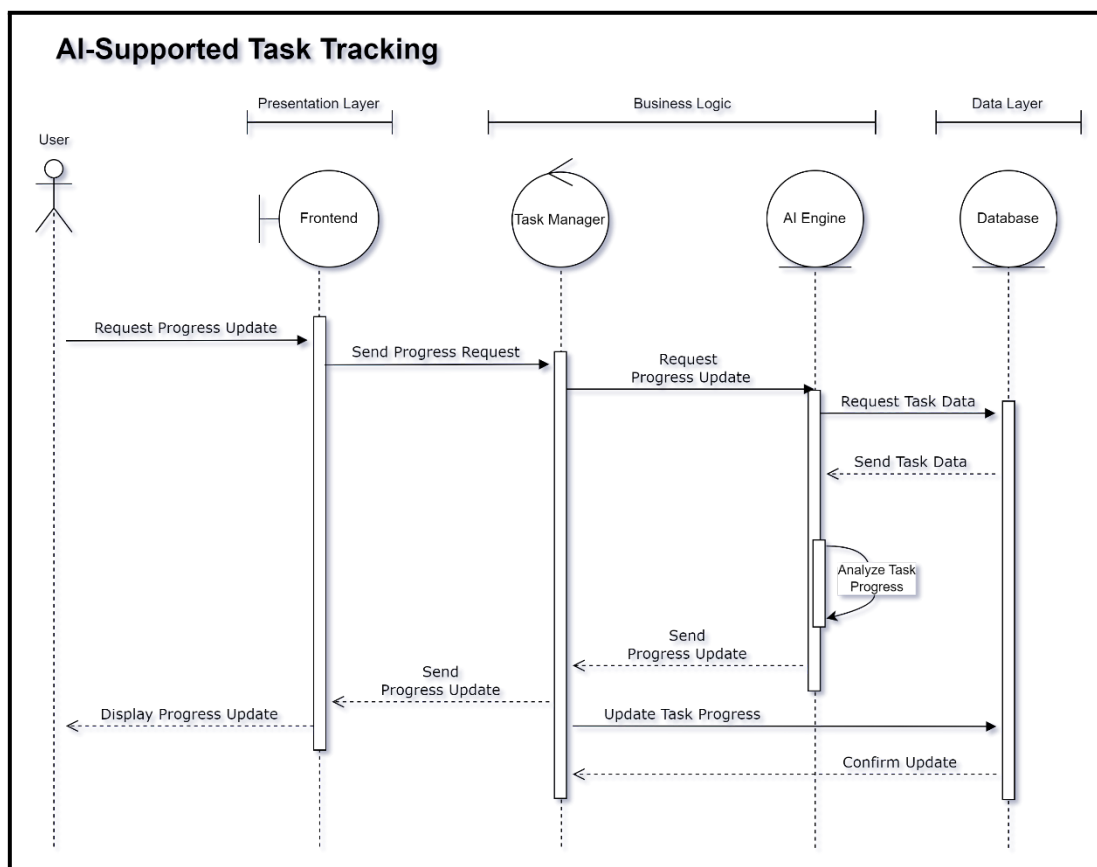


Gambar 4.2. Proses Fungsi AI-Supported Task Allocation

Secara garis besar, proses dalam AI-supported Task Allocation ini sebagai berikut,

- Setelah pengguna membuat tugas, pengguna bisa melakukan request untuk alokasi tugas dengan AI
- Task manager akan meminta informasi tim dari database dan informasi ini akan digabung dengan task yang baru dibuat untuk dikirimkan ke AI engine.
- AI engine akan menganalisa informasi dari task manager untuk membuat rekomendasi alokasi tugas.

- d. Rekomendasi dikirimkan ke pengguna dan pengguna bisa menerima atau mengubahnya. Hasil dari penerimaan / perubahan ini dikirim ke task manager.
- e. Task manager akan meminta deadline alignment service untuk menentukan tenggat waktu yang cocok.
- f. Deadline alignment service akan menentukan tenggat waktu yang cocok berdasarkan alokasi tugas dan diselaraskan dengan jadwal tim (yang ditetapkan saat pertama kali login ke aplikasi)
- g. Hasil rekomendasi tenggat waktu dikirimkan ke pengguna untuk disesuaikan. Hasil penyesuaian ini dikirimkan lagi ke task manager.
- h. Task manager akan menetapkan alokasi dan tenggat waktu tugas ke dalam database.



Gambar 4.3. Proses Fungsi AI-Supported Task Tracking

Selanjutnya ada AI-supported Task Tracking, proses dari fungsi ini secara garis besar sebagai berikut,

- a. Pengguna melakukan permintaan untuk pemantauan kemajuan tugas.
- b. Task manager akan meminta AI engine untuk menentukan kemajuan tugasnya.
- c. AI engine akan meminta informasi keadaan tugas ke database dan menganalisisnya.
- d. Hasil analisisnya adalah kemajuan tugas yang akan dikirimkan oleh AI engine ke Task Manager.
- e. Task Manager melalui Frontend akan menampilkan kemajuannya pada pengguna.

## 5. Teknologi yang Diajukan

Implementasi sistem yang dijelaskan di atas akan menggunakan beberapa teknologi yang dirasa sesuai dengan spesifikasi yang diharapkan. Setiap teknologi yang diajukan akan dijelaskan di bawah,

a. Frontend: Vanilla HTML, CSS, dan JavaScript

Teknologi frontend klasik dan fundamental untuk membangun antarmuka pengguna yang sederhana namun tetap interaktif. Kombinasi ini cocok untuk aplikasi dengan kebutuhan frontend yang ringan, seperti menampilkan data dari API backend ataupun form.

b. Task Manager: Python dan FastAPI

FastAPI, framework mikro berbasis Python, menawarkan fleksibilitas tinggi untuk membangun backend yang kuat namun tetap ringan. FastAPI cocok untuk membuat aplikasi berbasis RESTful API yang mendukung kebutuhan dalam sistem ini seperti, alokasi tugas, pelacakan progress, dan penyimpanan data proyek.

c. AI Engine: Fuzzy Rule Based System dengan Numpy

Untuk implementasi AI-supported task allocation dan tracking, akan digunakan Fuzzy Rule Based System yang dibangun sendiri menggunakan Numpy. Nantinya sistem ini akan dijadikan layanan yang terintegrasi ke endpoint yang ada di task manager.

d. Deadline Alignment Service: Python dan FastAPI

Deadline Alignment Service akan diimplementasikan menggunakan FastAPI sebagai modul tambahan dalam backend utama. Fungsinya adalah mengelola sinkronisasi kalender dan jadwal kelas untuk menentukan tenggat waktu tugas secara otomatis.

e. Database: PostgreSQL + Supabase

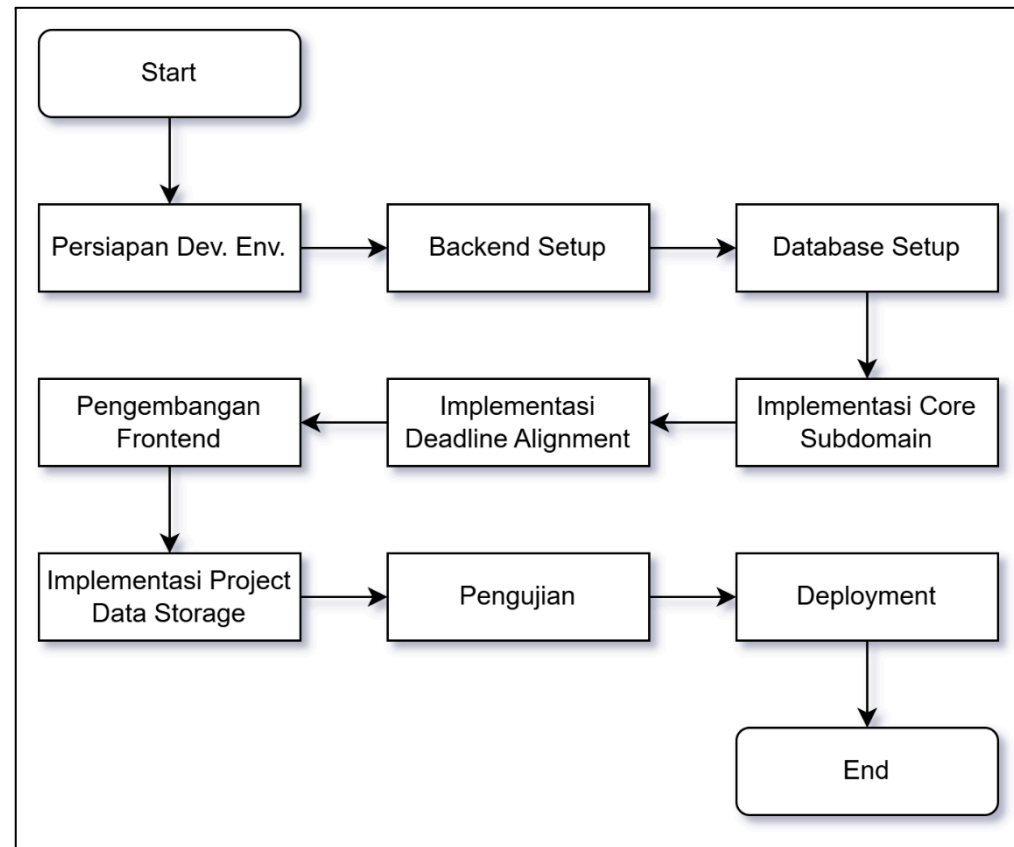
PostgreSQL ini cocok untuk aplikasi yang memerlukan manajemen data relasional yang kompleks dan skalabel. Skalabilitas yang baik memudahkan apabila aplikasi perlu dikembangkan lebih lanjut. Untuk men-*deploy* databasenya akan menggunakan Supabase karena bersifat *real-time* dan memanfaatkan Row Level Security dari PostgreSQL.

f. Autentikasi: Supabase

Supabase cocok untuk aplikasi ini karena melayani anonymous sign-in, social OAuth, dan basic multi-factor Auth. Ini bisa melayani berbagai jenis user dari yang tidak mau login (*guest / anon* login) sampai yang mau login dengan social OAuth. Autentikasi dari supabase juga menggunakan JWT dan mendukung 50.000 MAU sehingga memadai untuk pengembangan awal.



## 6. Flow dan Timeline Pengembangan



Gambar 6.1. Diagram Flow Pengembangan

- a. Pengembangan diawali dengan persiapan lingkungan pengembangan yaitu virtual environment dengan python, FastAPI, dan supabase. Tahap ini akan menggunakan dua kakas utama yaitu VS Code dan Supabase Dashboard.

```
# setup virtual environment dengan asumsi python sudah diinstal secara global di mesin
> py -m venv venv

# instalasi FastAPI dan library-library yang sekiranya diperlukan
(project-env)> py -m pip install fastapi uvicorn jinja2

# instalasi Supabase
(project-env) > py -m pip install supabase
```

- b. Selanjutnya setup backend dilakukan dengan FastAPI dan Supabase. Struktur proyeknya kemungkinan akan seperti ini:

```
TUBES-TST-2024/
├── .gitignore
├── LICENSE
├── README.md
├── backend
│   ├── .dockerignore
│   ├── app
│   │   ├── __init__.py
│   │   ├── main.py
│   │   ├── models.py
│   │   └── services
│   │       ├── rule_based_fuzzy_logic.py
│   │       ├── simple_rbfl_system.py
│   │       ├── supabase.py
│   │       └── task_manager.py
│   ├── docker-compose.yaml
│   ├── dockerfile
│   └── railway.json
```

```
├── requirements.txt
├── run.bat
├── directory_tree.txt
├── frontend
│   ├── css
│   │   └── styles.css
│   ├── home.html
│   ├── index.html
│   ├── js
│   │   ├── home.js
│   │   ├── index.js
│   │   ├── register.js
│   │   └── sandbox.js
│   ├── register.html
│   └── sandbox.html
```

- c. Kemudian dilakukan juga setup database, yaitu menghubungkan FastAPI dengan PostgreSQL pada Supabase. Pada tahap ini juga dibuat skema database serta implementasinya di Supabase.

Gunakan **Supabase Dashboard** untuk membuat project baru

Salin connection URL serta API Key, kemudian simpan dalam file .env

Setup routing dan fungsi-fungsi yang berkaitan dengan Supabase di file services/supabase.py, termasuk untuk autentikasi

- d. Implementasi core subdomain adalah tahap selanjutnya, di sini adalah tahap paling intensif dan penting. Pada tahap ini akan diimplementasikan AI Engine untuk Task Allocation dan Task Tracking. Kakas yang digunakan adalah FastAPI dan Tensorflow.
- e. Selanjutnya Deadline Alignment, service ini akan diimplementasikan menggunakan algoritma sederhana, integrasi dengan GCal akan ditambahkan bila memungkinkan
- f. Setelah backend siap, maka dilanjutkan dengan pengembangan frontend. Perkakas yang akan banyak berperan di sini adalah HTML, CSS, dan JS.
- g. Terakhir adalah pengujian lokal sebelum deployment. Tahap pengujian ini akan menguji fungsionalitas sistem secara end-to-end, sebelum diluncurkan.

- h. Deployment atau peluncuran adalah tahapan terakhir. Ketika keseluruhan aplikasi sudah siap dan telah diuji, maka bisa dilakukan deployment. Deployment ini akan dilakukan dengan memanfaatkan Vercel Serverless Functions karena Vercel Serverless Functions memfasilitasi deployment FastAPI sebagai layanan API.

Untuk timeline pengembangannya secara lebih rinci, bisa dilihat pada gambar 6.2 di bawah ini,

[illegible]

### Gambar 6.2. Timeline Pengembangan

## 7. Fitur Utama: Fuzzy Rule Based System

Pada aplikasi yang dibuat, diimplementasikan suatu fitur utama yaitu Fuzzy Rule Based System (FRBS), jadi sistem ini memanfaatkan Fuzzy Logic untuk menyusun suatu Rule Based System. Dengan referensi dari wikipedia: [https://en.wikipedia.org/wiki/Fuzzy\\_logic](https://en.wikipedia.org/wiki/Fuzzy_logic), yang diimplementasikan di sini juga memanfaatkan [fuzzy set](#) yang didefinisikan dengan kurva-kurva trapezoid untuk proses fuzzificationnya dan memanfaatkan algoritma defuzzification yang dijelaskan juga di wikipedia: [https://en.wikipedia.org/wiki/Fuzzy\\_logic#:~:text=the%20OR%20operator-,Defuzzification,-%5Bedit%5D](https://en.wikipedia.org/wiki/Fuzzy_logic#:~:text=the%20OR%20operator-,Defuzzification,-%5Bedit%5D)

Implementasi kode yang dicantumkan di laporan akan dibagi-bagi menjadi 3 bagian utamanya saja: pembuatan rule set (atau fuzzy set), fuzzification, dan defuzzification. Lebih lengkapnya bisa langsung mengunjungi tautan repositori yang dicantumkan pada lampiran.

Rule Set
<pre># Fungsi pembuatan rules def add_rule(self, antecedents: List[Tuple[str, str]], consequent: Tuple[str, str]) -&gt; None:     """Add a rule to the system."""     self.rules.append(FuzzyRule(antecedents, consequent))  # Contoh penambahan rules pada system system.add_rule(     antecedents=[("workload", "low"), ("availability", "high")],     consequent=("suitability", "high") ) system.add_rule(     antecedents=[("workload", "medium"), ("availability", "high")],     consequent=("suitability", "high") ) system.add_rule(     antecedents=[("workload", "high"), ("availability", "high")],     consequent=("suitability", "medium") )</pre>

## Fuzzification

```
def get_membership(self, value: float) -> Dict[str, float]:
    """Calculate membership values for all sets given a crisp input."""
    memberships = {}
    for set_name, params in self.sets.items():
        memberships[set_name] = self._trapezoid(value, *params)
    return memberships

def _trapezoid(self, x: float, a: float, b: float, c: float, d: float) -> float:
    """Calculate trapezoid membership value."""
    if x < a or x > d:
        return 0
    elif a <= x <= b:
        return (x - a) / (b - a) if b != a else 1
    elif b <= x <= c:
        return 1
    else: # c < x <= d
        return (d - x) / (d - c) if d != c else 1
```

## Defuzzification

```
def evaluate(self, inputs: Dict[str, float]) -> float:
    """Evaluate the system for given inputs."""
    # Calculate memberships for all input variables
    variable_states = {}
    for var_name, value in inputs.items():
        variable_states[var_name] = self.input_variables[var_name].get_membership(value)
    print(variable_states)

    # Evaluate all rules
    rule_outputs: Dict[str, List[float]] = {}
    for rule in self.rules:
        rule_strength = rule.evaluate(variable_states)
        consequent_set = rule.consequent[1]
        if consequent_set not in rule_outputs:
```

```

        rule_outputs[consequent_set] = []

rule_outputs[consequent_set].append(rule_strength)
    print(rule_outputs)

    # Aggregate rule outputs using maximum
    aggregated_outputs = {
        set_name: max(strengths)
        for set_name, strengths in
rule_outputs.items()
    }

    # Defuzzify using center of gravity
    x = np.linspace(self.output_variable.range_min,
self.output_variable.range_max, 1000)

    output_curves = []
    for set_name, strength in
aggregated_outputs.items():
        set_params =
self.output_variable.sets[set_name]
        curve = np.minimum(
            strength,
np.array([self.output_variable._trapezoid(xi, *set_params) for
xi in x])
        )
        output_curves.append(curve)

    combined_curve = np.maximum.reduce(output_curves)
    if output_curves else np.zeros_like(x)

    numerator = np.sum(x * combined_curve)
    denominator = np.sum(combined_curve)

    return numerator / denominator if denominator != 0
else 0

```

## 8. Kontainerisasi

Pada aplikasi yang dikembangkan, diterapkan kontainerisasi, spesifiknya pada layanan *backend* karena kontainerisasi mempermudah proses *deployment*. Kontainerisasi menggunakan Docker, sebagai salah satu perkakas kontainerisasi yang



cukup populer dan didukung di Windows. Proses kontainerisasi dimulai dengan pembuatan Dockerfile:

Dockerfile
<pre># Use an official Python runtime as a parent image FROM python:3.11-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 application code COPY . .  # Expose port 8000 for FastAPI EXPOSE 8000  # Run the application CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]</pre>

Pertama-tama butuh mengimpor runtime Python sebagai parent image, di sini menggunakan 3.11-slim, kata kunci “slim” mengindikasikan kita menggunakan versi lebih ringan dari python 3.11, hal ini bermanfaat karena akan mempercepat proses pembangunan image docker. Selanjutnya menetapkan working directorynya di folder app dan menyalin requirements.txt. Penyalinan ini penting karena akan digunakan di tahap selanjutnya, yaitu pip install requirements.txt, ini menjamin semua requirements dari lokal dan kontainer sama persis. Terakhir semua file dalam folder app disalin ke container dan port 8000 dibuka untuk mempersiapkan menjalankan uvicorn di port tersebut. Sebenarnya ini sudah cukup untuk melakukan kontainerisasi, tapi ditambahkan dockerignore karena saat diuji coba, dockerfile ini memakan waktu yang sangat lama untuk build. Waktu yang lama ini ternyata disebabkan venv dan pycache di folder app masih diikutsertakan. Jadi di dockerignore ditambahkan ketentuan berikut,

.dockerignore
<pre># Ignore Python cache files</pre>

```

__pycache__/
*.pyc
*.pyo

# Ignore virtual environments
venv/

# Ignore Git repository
.git/
.gitignore

.vercel

```

## 9. Autentikasi Manusia

Pada aplikasi yang dikembangkan akan diterapkan autentikasi dengan memanfaatkan layanan autentikasi dari supabase yang bawaannya memanfaatkan JWT, tetapi juga menyediakan OAuth dengan bantuan pihak ketiga seperti Google, Apple, Github, dan sebagainya. Proses autentikasi yang sudah berhasil diterapkan adalah via password seperti sistem login tradisional dan via Github OAuth.

### Inisialisasi Client Supabase

```

# Supabase credentials
SUPABASE_URL = os.getenv("SUPABASE_URL")
SUPABASE_KEY = os.getenv("SUPABASE_KEY")

# Initialize Supabase client
client: Client = create_client(SUPABASE_URL, SUPABASE_KEY)

base_url = os.getenv("BASE_URL")
frontend_url = os.getenv("FRONTEND_URL")

```

### Request Pembuatan User

```

@supabase_router.post("/user-signup", summary="This is
used for user signup with classic email and password")
async def signup(request: Request):
    if validate_api_key(request):
        email = request.headers.get("email")
        password = request.headers.get("password")

```

```

        print(email, password)

        if not email or not password:
            raise HTTPException(status_code=400,
detail="Email and Password headers are required")

        try:
            # client = get_supabase_client()
            response = client.auth.sign_up({"email":
email, "password": password})
            return response
        except Exception as e:
            print(e)
            raise HTTPException(status_code=500,
detail=str(e))
        else:
            raise HTTPException(status_code=403,
detail="Invalid API key")

```

Pada pemanggilan route /user-signup HTTP Request perlu menyediakan atribut “email” dan “password” dalam headernya agar bisa diproses. Email dan password tersebut akan dikirimkan ke Supabase dan akan tercatat dalam Auth Databasenya.

### Request Melakukan Login

```

@supabase_router.post("/user-signin", summary="This is
used for user signin with classic email and password")
async def signin(
    request: Request
):
    if validate_api_key(request):
        email = request.headers.get("email")
        password = request.headers.get("password")

        if not email or not password:
            raise HTTPException(status_code=400,
detail="Email and Password headers are required")

```

```

        try:
            # client = get_supabase_client()
            response =
client.auth.sign_in_with_password({"email": email, "password":
password})

            token = response.session.access_token

            fastapi_response =
RedirectResponse(url=base_url + "/protected-home",
status_code=302)

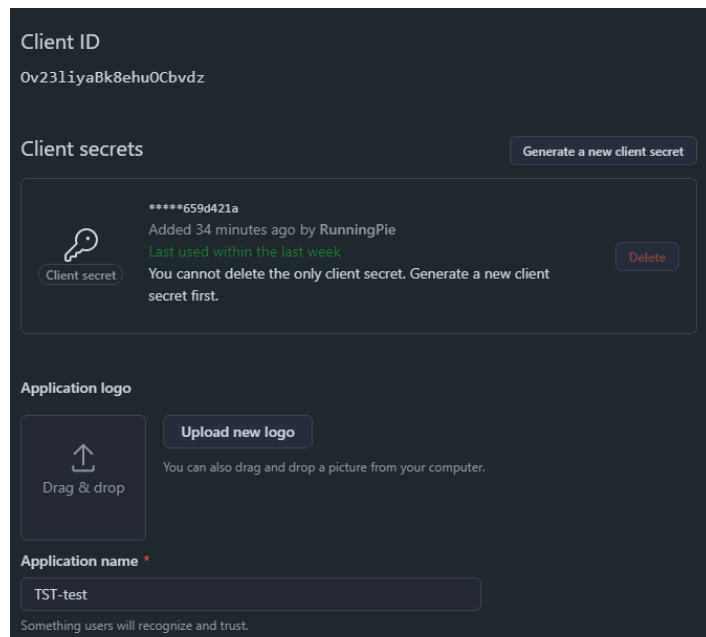
            fastapi_response.set_cookie(key="access_token", value=token,
httponly=True)

            return fastapi_response

        except Exception as e:
            raise HTTPException(status_code=500,
detail=str(e))
        else:
            raise HTTPException(status_code=403,
detail="Invalid API key")

```

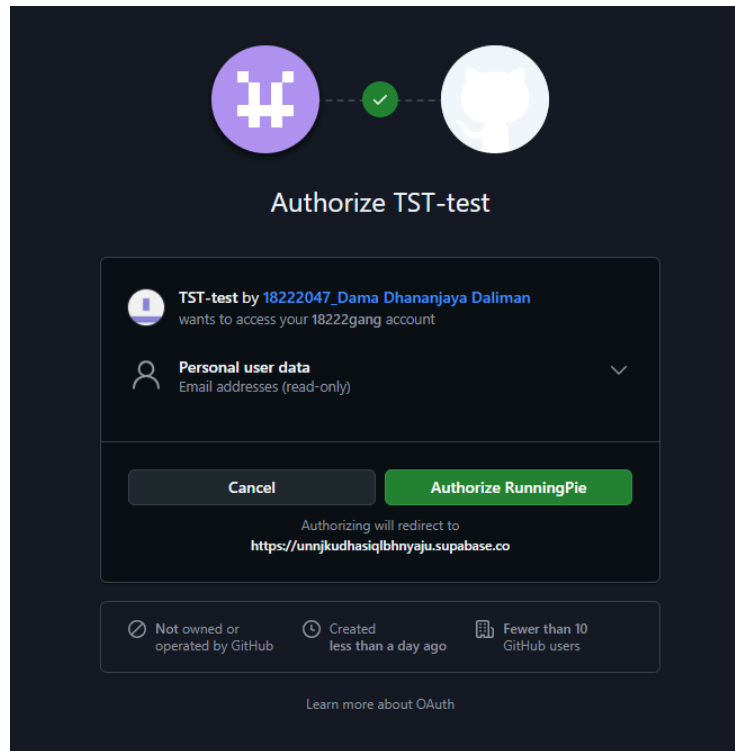
Pemanggilan route /user-signin juga perlu menyediakan "email" dan "password" pada header HTTP requestnya. Email dan password yang disediakan akan dijadikan parameter untuk memanggil fungsi sign in dari Supabase. Sedikit catatan, fungsi ini hanya bisa berjalan ketika pengguna yang melakukan sign up sudah mengonfirmasi emailnya.



### Request Melakukan Login dengan OAuth Github

```
@router_oauth.get("/github-login")
async def github_login():
    response = supabase.auth.sign_in_with_oauth(
        {"provider": 'github'})
    return response
```

Pemanggilan route `/github-login` memungkinkan pengguna untuk melakukan login dengan akun GitHub karena Supabase sign in with OAuth akan meneruskan permintaannya ke aplikasi OAuth yang dibuat di GitHub. Response dari request ini akan berupa dictionary / JSON yang menyatakan nama provider beserta URL redirect untuk mengalihkan pengguna ke laman otorisasi aplikasi dari GitHub, seperti gambar di bawah ini.



Live deployment dari implementasi di atas bisa diakses di <https://taskhub-tst.vercel.app/index.html> atau secara langsung ke *endpoint backend* di <https://tubes-tst-2024-production.up.railway.app/github-signin>

## 10. Autentikasi Layanan

Pada aplikasi yang dibuat, ada juga autentikasi yang dilakukan terhadap layanan lain apabila melakukan pemanggilan terhadap API Endpoints pada layanan sendiri. Fungsi utamanya ada di `supabase.py` dan diimplementasikan sebagai berikut,

### Fungsi Validasi API Key

```
def validate_api_key(request: Request):
    requester_domain_origin =
request.headers.get("Origin")

    # Lookup domain key in db
    try:
        lookup_response =
client.table("API_KEYS").select("*").eq("domain",
requester_domain_origin).execute().data[0]["key"]
        print(request.headers.get("API-Key"))
        print(lookup_response)
    except Exception as e:
        print(e)
```

```

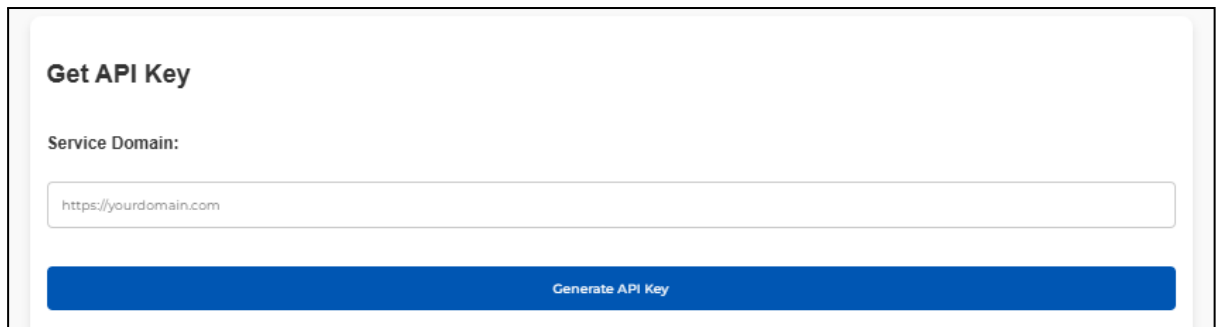
        raise HTTPException(status_code=404,
detail="Domain not found")

    return lookup_response ==
request.headers.get("API-Key")

```

Validasi API Key dilakukan dengan memeriksa origin dari requester kemudian mencari origin tersebut ke tabel API-Key pada database. Apabila domain tidak ditemukan maka kembalikan 404 Not Found. Sedangkan jika ditemukan, maka dibandingkan dengan API-Key yang disediakan di header request.

API Key yang digunakan oleh service bisa didapatkan dari endpoint /request-api-key. Kalau di frontend bisa didapatkan pada home.html setelah login, di sini:



Gambar 10.1. Antarmuka frontend untuk mendapatkan API Key

## 11. Dokumentasi RESTful API Endpoints

Di bawah ini ada tabel yang menjelaskan semua endpoints yang tersedia dan fungsinya secara garis besar.

Tabel 11.1. Dokumentasi RESTful API yang tersedia dalam aplikasi

Category	Endpoint	Function
Documentation & API	/openapi.json	Menyediakan dokumentasi OpenAPI dalam format JSON
	/docs	Menampilkan dokumentasi API menggunakan Swagger UI
	/docs/oauth2-redirect	Endpoint redirect untuk autentikasi OAuth2
	/redoc	Menampilkan dokumentasi API menggunakan Redoc

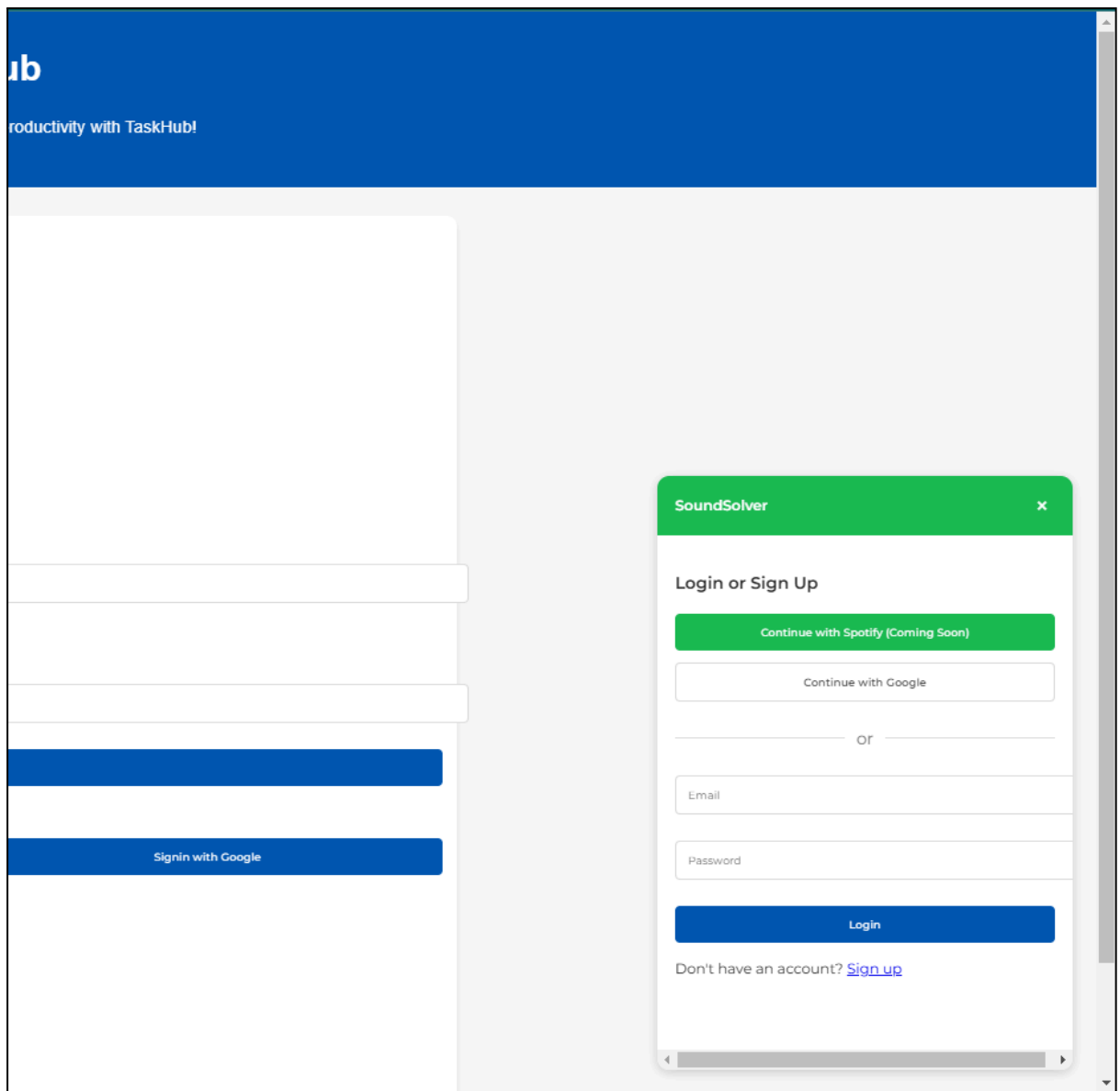
Authentication & User	/	Halaman utama aplikasi
	/protected-home	Halaman beranda yang memerlukan autentikasi (tidak jadi digunakan)
	/request-api-key	Endpoint untuk meminta API key
	/user-signup	Pendaftaran pengguna baru
	/user-signin	Login pengguna
	/github-signin	Login menggunakan akun GitHub
	/google-signin	Login menggunakan akun Google
	/user-signout	Logout pengguna
	/callback	Endpoint callback untuk autentikasi pihak ketiga
Team Management	/create-team	Membuat tim baru
	/add-team-member	Menambahkan anggota ke tim
	/add-members-availability	Menambahkan ketersediaan waktu anggota tim
	/add-team-task	Menambahkan tugas untuk tim
	/show-teams	Menampilkan daftar tim
	/show-team-members	Menampilkan daftar anggota tim
	/show-team-tasks	Menampilkan daftar tugas tim
	/remove-team-member	Menghapus anggota dari tim
	/view-availability	Melihat ketersediaan waktu anggota
	/remove-availability	Menghapus data ketersediaan waktu
	/remove-task	Menghapus tugas
Rule Based Fuzzy Logic	/create-rbfl	Membuat aturan fuzzy logic baru
	/rbfl-evaluate	Melakukan evaluasi menggunakan sistem fuzzy logic



Jika ingin mencoba lebih lanjut, beberapa endpoint di atas sudah disediakan dalam sandbox: <https://taskhub-tst.vercel.app/sandbox.html>, untuk diuji coba secara interaktif dan responsnya akan ditampilkan secara langsung.

## 12. Integrasi dengan Layanan Lain

Pada aplikasi ini, layanan lain yang diintegrasikan adalah layanan chatbot, dari tautan ini: <https://spotify-bot.azurewebsites.net/>, layanan ini memang utamanya untuk menyelesaikan masalah terkait musik, tapi memiliki pilihan untuk mendukung juga pertanyaan secara general. Layanan ini diintegrasikan berupa sebuah widget yang diletakkan di frontend dengan peran seperti widget customer service chatbot.



Gambar 12.1. Tangkapan layar index.html dengan chatbot

## 13. Lampiran

Tautan Repositori GitHub: <https://github.com/RunningPie/Tubes-TST-2024>

Tautan Deployment Frontend: <https://taskhub-tst.vercel.app/>

Tautan Deployment Backend: <https://tubes-tst-2024-production.up.railway.app>