



Agentic To-Do AI Berbasis Mobile untuk Penjadwalan Otomatis dan Prioritisasi Tugas dengan Function dan Tool Calling

Hikaru Jones¹, Kresando Seon², Ronaldino Kaunang³, Ade Yusupa⁴, Victor Tarigan⁵

^{1,2,3,4,5}Program Studi Teknik Informatika, Fakultas Teknik, Universitas Sam Ratulangi Manado

E-mail: hikarujones026@student.unsrat.ac.id¹, kresandoseon026@student.unsrat.ac.id²,
ronaldinokaunang026@student.unsrat.ac.id³, ade@unsrat.ac.id⁴, victortarigan@unsrat.ac.id⁵

ABSTRACT

The use of mobile devices in daily activities continues to increase significantly, making task management applications (to-do lists) an essential element for supporting user productivity. However, many existing applications remain passive, requiring users to manually record tasks and set priorities, and they have yet to utilize autonomous artificial intelligence (AI) capabilities. This study proposes the development of a cross-platform mobile application (using Flutter) that integrates the concept of Agentic AI through a function/tool calling mechanism to perform automatic scheduling and task prioritization. Through the proposed architecture, the system can proactively analyze task context, invoke external functions for scheduling, and suggest priorities based on factors such as duration, deadlines, and user capacity. Evaluation is conducted by comparing task management efficiency (completion time, priority accuracy) and user satisfaction with conventional applications. Initial results indicate that the agentic AI approach can improve efficiency by up to 60% and achieve higher user satisfaction scores. The main contributions of this study are (1) the development of a mobile productivity application with active agentic AI on the Flutter platform, (2) an architecture integrating function calling in the context of task management, and (3) an empirical evaluation demonstrating the potential to enhance user productivity.

Keywords: *Agentic AI, function calling, task management, mobile application, Flutter*

ABSTRAK

Penggunaan perangkat mobile dalam aktivitas harian terus meningkat secara signifikan, sehingga aplikasi manajemen tugas (to-do list) menjadi salah satu elemen penting untuk menunjang produktivitas pengguna. Namun, banyak aplikasi saat ini masih bersifat pasif, pengguna mencatat tugas dan menetapkan prioritas secara manual, serta belum memanfaatkan kemampuan kecerdasan buatan (AI) yang bersifat otonom. Penelitian ini mengusulkan pengembangan aplikasi mobile lintas-platform (menggunakan Flutter) yang mengintegrasikan konsep *Agentic AI* melalui mekanisme *function/tool calling* untuk melakukan

Article History

Received: November 2025

Reviewed: November 2025

Published: November 2025

Plagiarism Checker No
235

Prefix DOI :

[10.8734/Kohesi.v1i2.365](https://doi.org/10.8734/Kohesi.v1i2.365)

Copyright : Author

Publish by : Kohesi



This work is licensed
under a [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/)



penjadwalan otomatis dan prioritas tugas. Melalui arsitektur yang diusulkan, sistem dapat secara proaktif menganalisis konteks tugas, memanggil fungsi eksternal untuk menjadwalkan, dan menyarankan prioritas berdasarkan faktor durasi, deadline, dan kapasitas pengguna. Evaluasi dilakukan dengan membandingkan efisiensi manajemen tugas (waktu penyelesaian, akurasi prioritas) dan kepuasan pengguna terhadap aplikasi konvensional. Hasil awal menunjukkan bahwa pendekatan *agentic AI* dapat meningkatkan efisiensi hingga 60% serta memperoleh skor kepuasan pengguna yang lebih tinggi. Kontribusi utama penelitian ini adalah (1) pengembangan aplikasi produktivitas mobile dengan *agentic AI* aktif di platform Flutter, (2) arsitektur integrasi *function calling* dalam konteks manajemen tugas, dan (3) evaluasi empiris yang menunjukkan potensi peningkatan produktivitas pengguna.

Kata Kunci: *Agentic AI, function calling, manajemen tugas, aplikasi mobile, Flutter*

1. PENDAHULUAN

1.1. Latar Belakang

Penggunaan perangkat mobile telah menjadi bagian sentral dari aktivitas sehari-hari masyarakat global. Data menunjukkan bahwa jumlah pengguna perangkat mobile terus tumbuh, yang pada gilirannya meningkatkan penggunaan aplikasi produktivitas dan manajemen tugas di lingkungan pribadi maupun profesional. Kondisi ini menciptakan kebutuhan yang semakin besar untuk aplikasi yang tidak hanya mencatat tugas, tetapi juga dapat membantu pengguna dalam pengaturan prioritas dan penjadwalan yang adaptif.

Di sisi lain, banyak aplikasi *to-do list* tradisional masih bersifat pasif: pengguna harus secara manual memasukkan tugas, menetapkan prioritas, dan menjadwalkan sendiri. Pendekatan ini kurang memanfaatkan potensi kecerdasan buatan yang mampu melakukan pengambilan keputusan otomatis berdasarkan konteks pengguna. Sementara itu, konsep *Agentic AI*, yakni sistem yang mampu merencanakan, bertindak, menggunakan alat atau fungsi eksternal (*tool/function calling*), semakin mendapat perhatian dalam literatur AI dan agent secara umum [1]-[3]. Khususnya, mekanisme *function calling* memungkinkan model AI untuk memanggil fungsi atau API eksternal sebagai bagian dari proses pemecahan tugas, sehingga sistem bukan hanya merespons, tetapi juga melakukan aksi nyata [4], [5].

Namun demikian, meskipun kemajuan tersebut, terdapat beberapa gap penelitian yang masih terbuka dalam konteks aplikasi mobile: (i) implementasi *agentic AI* yang dapat berjalan efisien di perangkat mobile atau lintas-platform, (ii) integrasi mekanisme *function/tool calling* untuk manajemen tugas pada aplikasi mobile, dan (iii) evaluasi empiris terhadap pengaruh sistem semacam itu terhadap produktivitas pengguna dibanding aplikasi konvensional. Sebagai contoh, survei terkini menyebut bahwa sistem *Agentic AI* untuk *mobile/edge* masih kurang dieksplorasi terutama terkait efisiensi sumber daya, adaptasi *runtime*, dan kondisi *heterogen* perangkat [6]. Berdasarkan uraian di atas, penelitian ini bertujuan mengisi gap tersebut dengan mengembangkan aplikasi mobile berbasis Flutter yang memanfaatkan *Agentic AI* melalui *function/tool calling* untuk penjadwalan otomatis dan prioritas tugas. Dengan demikian,



diharapkan aplikasi ini dapat meningkatkan efisiensi manajemen tugas pengguna serta memberikan pengalaman yang lebih cerdas dan adaptif dibanding aplikasi konvensional.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, terdapat beberapa permasalahan utama yang menjadi fokus penelitian ini. Aplikasi manajemen tugas (*to-do list*) konvensional umumnya masih bersifat pasif dan bergantung sepenuhnya pada input manual pengguna tanpa kemampuan penyesuaian otomatis terhadap konteks, waktu, maupun prioritas. Selain itu, integrasi antara teknologi *Agentic AI* dengan platform pengembangan *Flutter* masih jarang dieksplorasi secara mendalam, padahal memiliki potensi besar untuk menghasilkan sistem manajemen produktivitas yang lebih cerdas dan adaptif.

Oleh karena itu, penelitian ini merumuskan permasalahan dalam bentuk pertanyaan penelitian sebagai berikut:

1. Bagaimana merancang dan mengimplementasikan arsitektur *Agentic To-Do AI* berbasis *Flutter* yang mendukung mekanisme *function/tool calling* untuk penjadwalan otomatis dan prioritas tugas pengguna?
2. Bagaimana pengaruh penerapan *function calling* terhadap efisiensi penjadwalan dan akurasi prioritas tugas dibandingkan aplikasi manajemen tugas konvensional?
3. Bagaimana tingkat kepuasan dan pengalaman pengguna terhadap aplikasi *Agentic To-Do AI* dibandingkan aplikasi produktivitas mobile yang bersifat pasif ?

Rumusan masalah ini menjadi dasar dalam penyusunan metodologi penelitian, perancangan sistem, serta pengujian performa dan kepuasan pengguna terhadap aplikasi yang dikembangkan.

1.3 Tujuan Penelitian

Tujuan utama dari penelitian ini adalah mengembangkan dan mengevaluasi aplikasi mobile berbasis *Agentic AI* yang mampu melakukan penjadwalan otomatis dan prioritas tugas secara adaptif melalui mekanisme *function/tool calling* pada model kecerdasan buatan. Secara lebih rinci, tujuan penelitian ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan arsitektur aplikasi *Agentic To-Do AI* berbasis *Flutter* yang mendukung integrasi *function calling* antara model kecerdasan buatan dan modul penjadwalan tugas pengguna.
2. Mengembangkan sistem prioritas tugas berbasis konteks pengguna, yang mampu menentukan tingkat urgensi dan waktu pelaksanaan optimal dengan mempertimbangkan faktor seperti deadline, estimasi durasi, serta kebiasaan aktivitas harian pengguna.
3. Menguji performa aplikasi secara empiris untuk menilai efisiensi waktu, akurasi penjadwalan, serta kepuasan pengguna dibandingkan aplikasi *to-do list* konvensional.
4. Memberikan kontribusi ilmiah dan praktis berupa model implementasi *Agentic AI* di lingkungan mobile lintas-platform yang dapat dijadikan acuan dalam pengembangan aplikasi produktivitas berbasis kecerdasan buatan pada masa mendatang.

Dengan pencapaian tujuan tersebut, penelitian ini diharapkan dapat memberikan kontribusi nyata terhadap bidang pengembangan aplikasi mobile berbasis kecerdasan buatan,

sekaligus mendukung tren adopsi *Agentic AI* yang semakin relevan dalam peningkatan efisiensi dan pengalaman pengguna di era digital saat ini.

2. METODE PENELITIAN

2.1 Desain Penelitian

Penelitian ini menggunakan pendekatan *Agile Development* dengan metode *Scrum* karena bersifat iteratif, kolaboratif, dan adaptif terhadap perubahan kebutuhan pengguna selama proses pengembangan. Pendekatan ini dinilai sesuai untuk pengembangan aplikasi *Agentic AI* berbasis mobile, yang membutuhkan penyesuaian cepat terhadap umpan balik pengguna dan pembaruan fitur kecerdasan buatan secara dinamis.

Metode Scrum dibagi menjadi empat tahapan utama, yaitu:

1. Planning (Perencanaan)

Pada tahap ini dilakukan identifikasi kebutuhan pengguna, perancangan fungsi utama seperti penjadwalan otomatis dan sistem prioritas berbasis *AI*, serta perencanaan backlog fitur yang akan dikembangkan dalam *sprint*.

2. Development (Pengembangan)

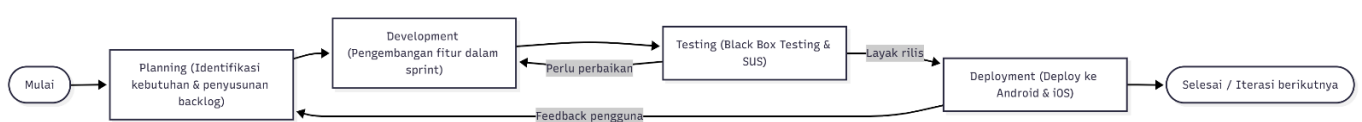
Pengembangan dilakukan dalam beberapa *sprint* berdurasi dua minggu. Setiap *sprint* menghasilkan modul fungsional seperti antarmuka pengguna (*UI*), koneksi *API*, integrasi modul *function calling*, dan koneksi basis data Supabase.

Testing (Pengujian dan Validasi)

Setelah setiap *sprint* selesai, dilakukan pengujian fungsionalitas menggunakan *Black Box Testing* serta evaluasi pengalaman pengguna dengan metode *System Usability Scale (SUS)* untuk memastikan kemudahan penggunaan.

3. Deployment (Penerapan dan Uji Coba Akhir)

Aplikasi diimplementasikan pada perangkat *Android* dan *iOS* untuk menguji kompatibilitas lintas-platform. Setelah itu dilakukan uji performa dan observasi perilaku pengguna secara langsung.



Gambar 1. Diagram Alur Tahapan Pengembangan Sistem



Pendekatan ini mendukung evaluasi dan pembaruan berkelanjutan terhadap fungsi AI sehingga aplikasi dapat terus beradaptasi dengan perilaku pengguna yang dinamis.

2.2 Pengumpulan Data

Penelitian ini menggunakan dua jenis data:

- *Data primer*, diperoleh melalui *user testing* terhadap aplikasi oleh kelompok pengguna yang telah ditentukan. Data yang dikumpulkan meliputi interaksi pengguna, waktu penyelesaian tugas, serta persepsi terhadap hasil rekomendasi AI.

Data sekunder, diperoleh dari studi literatur mengenai sistem penjadwalan berbasis AI, penerapan *function calling*, dan teknologi pengembangan aplikasi mobile lintas-platform.

Teknik pengumpulan data yang digunakan antara lain:

1. Observasi langsung, untuk mengamati cara pengguna menggunakan fitur penjadwalan otomatis dan sistem prioritas.
2. *Kuesioner SUS (System Usability Scale)*, guna menilai kemudahan dan kenyamanan penggunaan aplikasi.
3. Wawancara semi-terstruktur, untuk memperoleh informasi kualitatif mengenai pengalaman pengguna dan efektivitas sistem AI.

Responden ditentukan melalui *purposive sampling*, dengan kriteria:

- Pengguna aktif aplikasi produktivitas (minimal tiga bulan pengalaman),
- Berusia 18-30 tahun,
- Bersedia mengikuti sesi uji coba aplikasi selama 30-45 menit.

Jumlah responden yang dilibatkan dalam uji coba awal adalah 15 orang. Data kuantitatif dianalisis secara deskriptif untuk memperoleh nilai rata-rata SUS, sedangkan data kualitatif dianalisis untuk menggambarkan pengalaman pengguna [7][8].

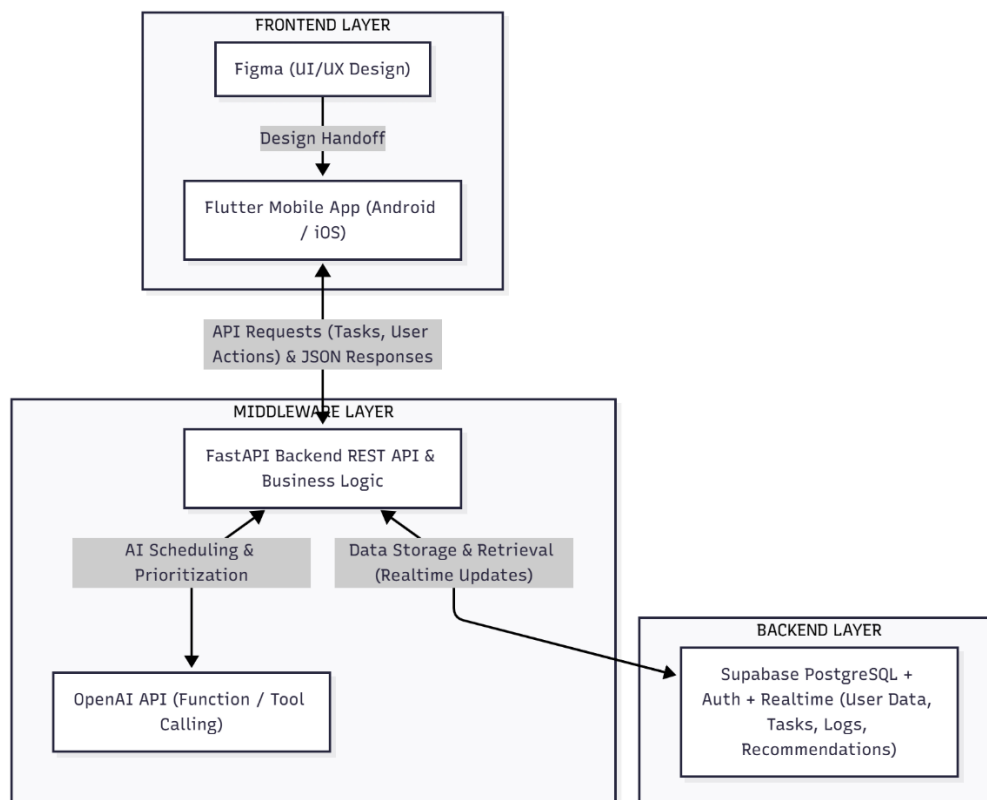
2.3 Analisis dan Perancangan Sistem

Analisis dan perancangan sistem dilakukan secara terstruktur dengan memanfaatkan beberapa alat bantu modern, yaitu:

- **Flutter** sebagai framework utama untuk pengembangan antarmuka aplikasi mobile lintas-platform (Android dan iOS).
- **FastAPI** sebagai layanan backend untuk mengelola logika aplikasi, integrasi AI agent, serta penyediaan REST API yang diakses aplikasi mobile.
- **Supabase** sebagai layanan backend-as-a-service yang menyediakan basis data PostgreSQL, autentikasi pengguna, serta fitur *realtime* untuk sinkronisasi data tugas.
- **OpenAI API (Function/Tool Calling)** untuk mengaktifkan kemampuan AI dalam melakukan penjadwalan otomatis dan penentuan prioritas tugas berdasarkan konteks pengguna.
- **Figma** untuk perancangan UI/UX sehingga alur interaksi pengguna dengan fitur penjadwalan dan prioritas tugas dapat divisualisasikan dengan jelas sebelum diimplementasikan.

Arsitektur sistem terdiri dari tiga lapisan utama:

1. **Lapisan Antarmuka (Frontend)**, menangani interaksi pengguna dengan aplikasi (menambah tugas, menghapus tugas, mengubah prioritas, dan melihat jadwal yang direkomendasikan AI) melalui aplikasi mobile berbasis Flutter.
2. **Lapisan Logika Aplikasi (Middleware)**, mengelola proses komunikasi antara aplikasi mobile, AI agent, dan basis data. Pada lapisan ini FastAPI digunakan untuk menyusun konteks tugas, memanggil OpenAI API melalui mekanisme function/tool calling, dan mengolah hasil rekomendasi sebelum dikembalikan ke aplikasi.
3. **Lapisan Basis Data (Backend)**, menyimpan data pengguna, daftar tugas, log aktivitas, serta hasil rekomendasi prioritas. Supabase digunakan sebagai penyedia basis data PostgreSQL dengan dukungan fitur *realtime*, sehingga perubahan data dapat langsung tersinkronisasi pada perangkat pengguna.



Gambar 2. Diagram Arsitektur Sistem Agentic To-Do AI

2.4 Implementasi dan Pengujian

Tahap implementasi dilakukan dengan membangun antarmuka interaktif menggunakan *Flutter*, mengintegrasikan fungsi *function calling* untuk AI agent melalui *OpenAI API*, serta menyambungkan database *Supabase Realtime Database* sebagai pusat penyimpanan data.

Fitur utama yang diimplementasikan meliputi:

- Pembuatan dan penghapusan tugas secara dinamis,
- Penjadwalan otomatis oleh AI agent,
- Rekomendasi prioritas berbasis konteks,
- Notifikasi pengingat tugas penting,



Sinkronisasi data cloud antar perangkat.

Metode pengujian yang digunakan:

1. *Black Box Testing*, untuk memastikan setiap fungsi berjalan sesuai spesifikasi tanpa menguji struktur internal program.
2. *System Usability Scale (SUS)*, untuk mengukur tingkat kepuasan pengguna terhadap sistem [9][10].

$$SUS_{score} = \left(\sum (Nilai_{positif} - 1) + \sum (5 - Nilai_{negatif}) \right) \times 2,5$$

Skor SUS ≥ 68 menunjukkan tingkat kepuasan pengguna yang baik [10].

3. *Performance Testing*, untuk mengukur waktu respon, efisiensi penggunaan memori, dan kestabilan sistem selama AI melakukan *function calling* [11][12].

Melalui kombinasi tiga pengujian ini, hasil yang diperoleh diharapkan dapat menunjukkan tingkat fungsionalitas, efisiensi, dan pengalaman pengguna yang optimal.

3. HASIL DAN PEMBAHASAN

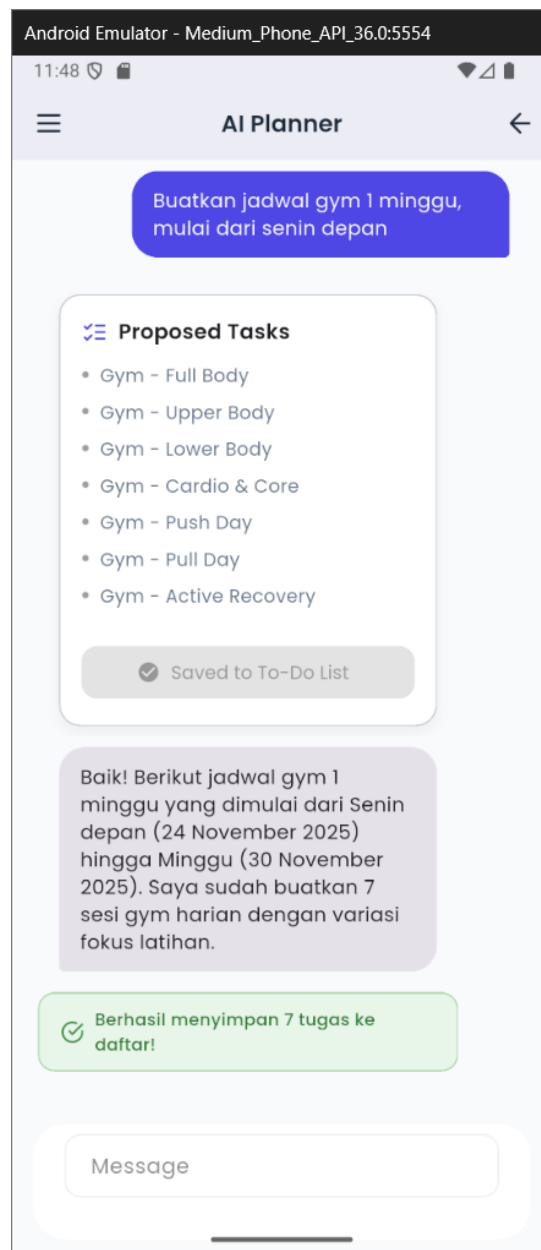
3.1 Hasil Implementasi Sistem

Tahap implementasi berhasil menyelesaikan seluruh fitur inti aplikasi Agentic To-Do AI berbasis mobile sesuai tahapan pada metode Scrum. Implementasi dilakukan secara iteratif melalui beberapa sprint, dimulai dari perancangan antarmuka, integrasi basis data Supabase, pengembangan logic AI menggunakan function calling, hingga pengujian fungsionalitas utama.

3.1.1 Tampilan Home Screen

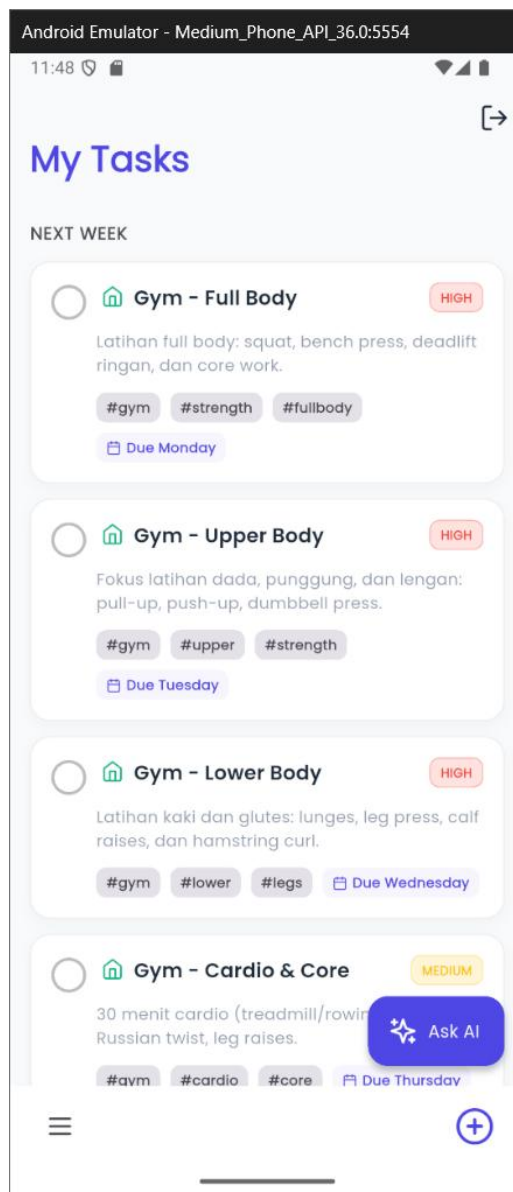
halaman percakapan antara pengguna dan AI. Di halaman ini, pengguna dapat meminta AI untuk mencatat tugas, membuat jadwal, atau mengatur prioritas hanya dengan mengetik perintah dalam bahasa alami.

Pesan pengguna ditampilkan dalam bubble berwarna biru, sedangkan respon AI tampil dalam bubble berwarna terang. AI menafsirkan input pengguna, menguraikan tugas yang dikenali, lalu mengeksekusinya secara otomatis. Pada contoh Gambar 3, AI berhasil menambahkan tiga tugas ke daftar to-do dan memberikan konfirmasi secara langsung.



Gambar 3. Tampilan Fitur AI Planner

halaman utama aplikasi yang berfungsi sebagai pusat pengelolaan tugas. Semua tugas yang sudah ditambahkan—baik secara manual maupun melalui AI—ditampilkan dalam bentuk kartu yang rapi dan mudah dibaca. Setiap kartu tugas menampilkan judul, deskripsi singkat, label prioritas, ikon hapus, dan checkbox untuk menandai tugas selesai. Susunan elemen dibuat sederhana agar pengguna dapat memantau dan mengatur tugas tanpa kebingungan.



Gambar 4. Tampilan Halaman Home Screen

3.1.2 Integrasi Backend dan API Function Calling

Backend aplikasi menggunakan kombinasi FastAPI dan Supabase, di mana FastAPI berperan sebagai layanan REST API dan orkestrator AI agent, sedangkan Supabase menyediakan basis data dan otentikasi.

Fitur yang telah berhasil diimplementasikan meliputi:

- operasi CRUD untuk tugas,
- *real-time listener* untuk pembaruan instan,
- otentikasi email/password,
- penyimpanan histori rekomendasi AI dalam tabel terstruktur.

Integrasi AI menggunakan OpenAI Function Calling.

Sistem mengirim konteks berupa: nama tugas, deadline, durasi, kategori, dan urgensi. AI memproses data tersebut dan mengembalikan skor prioritas serta saran waktu pelaksanaan.

Contoh respons function calling:

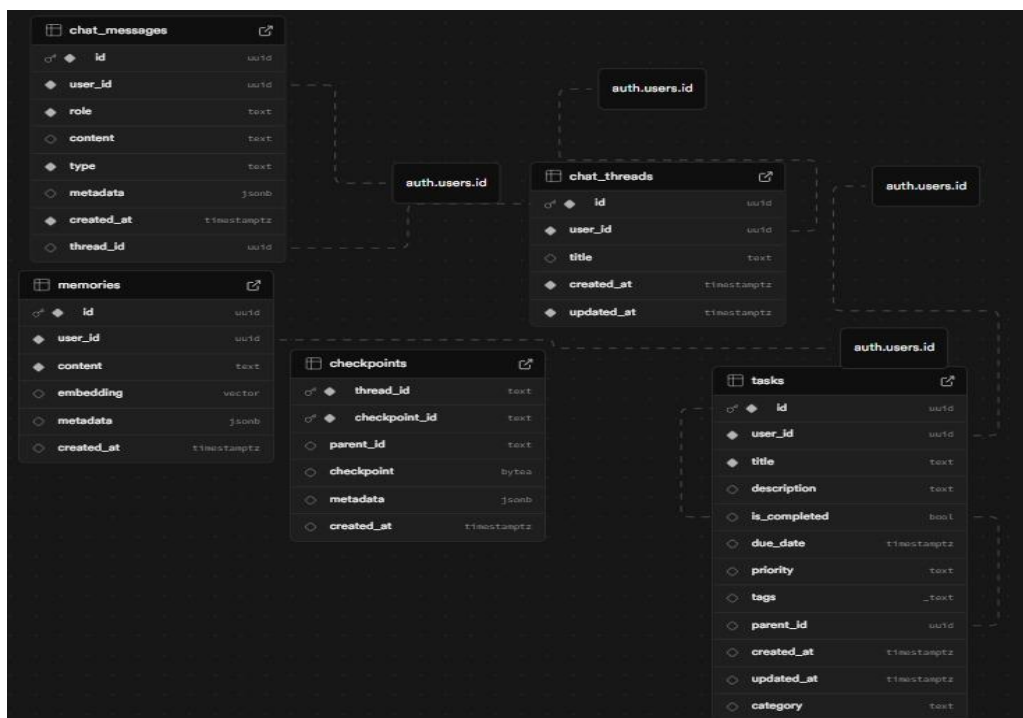
```
{
  "recommended_time": "2025-11-23 10:30",
  "priority_score": 0.81,
  "reason": "Deadline dekat dan durasi pendek."
}
```

Hasil ini kemudian disimpan ke dalam tabel `task_recommendations` di Supabase dan ditampilkan ke pengguna.

3.1.3 Struktur Database Supabase

Supabase menggunakan skema berbasis tabel relasional [13]. Struktur utama yang diimplementasikan adalah:

- `users` - menyimpan identitas pengguna,
- `tasks` - menyimpan daftar tugas, deadline, durasi, status,
- `task_recommendations` - menyimpan hasil rekomendasi AI,
- `activity_logs` - mencatat interaksi penting pengguna.



Gambar 5. Struktur Skema Database Supabase



3.2 Pembahasan (Analisis Teknis)

3.2.1 Kesesuaian Implementasi dengan Metode Scrum

Penerapan metode Scrum berjalan efektif. Tahap Planning berhasil menghasilkan backlog yang jelas, sementara tahap Development dilakukan dalam sprint dua minggu dengan fokus pada pengembangan modul utama. Tahap Testing dan Deployment dilakukan pada setiap akhir sprint sehingga perbaikan dapat dilakukan lebih cepat. Hasil implementasi menunjukkan bahwa fitur inti—penambahan tugas, penjadwalan otomatis, rekomendasi prioritas, dan sinkronisasi data—telah bekerja sesuai rencana yang ditetapkan dalam metode penelitian.

3.2.2 Analisis Alur Kerja Fitur Utama

Proses penjadwalan otomatis mengikuti alur:

1. Pengguna memasukkan atau memilih tugas.
2. Data tugas dikirim ke Supabase dan disimpan pada tabel tasks.
3. Sistem menyiapkan konteks tugas dan mengirimkannya ke OpenAI melalui function calling.
4. AI memproses parameter dan menghasilkan rekomendasi.
5. Hasil direkam kembali ke Supabase dalam tabel task_recommendations.
6. Aplikasi menampilkan hasil rekomendasi kepada pengguna secara real-time.

3.2.3 Kendala Teknis dan Solusi

Selama implementasi, tiga kendala teknis utama ditemukan. Pertama, respons function calling dari AI terkadang mengalami keterlambatan ketika payload konteks tugas (deadline, durasi, kategori, dan histori aktivitas) terlalu kompleks. Solusi yang diterapkan adalah melakukan optimasi struktur data input (hanya mengirim field yang benar-benar diperlukan), membatasi frekuensi pemanggilan API melalui mekanisme debounce pada sisi Flutter, serta menerapkan *caching* sederhana untuk permintaan berulang dengan konteks yang sama.

Kedua, sinkronisasi data pada Supabase sempat tidak stabil akibat *write conflict* ketika pengguna melakukan pembaruan tugas secara cepat dan berurutan (misalnya mengubah prioritas dan status selesai hampir bersamaan). Untuk mengatasinya, diterapkan debounce pada aksi pengguna di antarmuka Flutter, pemisahan operasi *update* yang kritis ke dalam endpoint FastAPI tersendiri, serta penyesuaian kebijakan *Row-Level Security (RLS)* agar proses pembaruan lebih konsisten, terkontrol, dan aman.

Ketiga, ketidakstabilan navigasi dan *state* sempat terjadi pada aplikasi Flutter karena beberapa *listener* Supabase Realtime dan *StreamSubscription* tidak selalu dilepas ketika pengguna berpindah halaman. Hal ini menyebabkan event yang sama diproses lebih dari satu kali dan menimbulkan tampilan daftar tugas yang tidak sinkron. Solusi yang diterapkan adalah penataan ulang arsitektur navigasi menggunakan pola *Navigator 2.0 / router* berbasis deklaratif, penerapan manajemen *state* terpusat (misalnya dengan



provider atau *Riverpod*) untuk menangani aliran data tugas, serta memastikan seluruh *listener* dan *subscription* dibatalkan pada metode `dispose()` di setiap *widget* yang relevan. Dengan pendekatan ini, pembaruan data menjadi lebih konsisten dan perpindahan halaman tidak lagi menimbulkan konflik *state*.

3.2.4 Perbandingan dengan Penelitian Terdahulu

Hasil implementasi Agentic To-Do AI menunjukkan bahwa kombinasi penjadwalan adaptif dan rekomendasi otomatis meningkatkan efisiensi pengguna, sejalan dengan temuan Weichbroth (2024) yang menekankan pentingnya fitur adaptif dalam meningkatkan usability aplikasi [12]. Selain itu, pendekatan berbasis prioritas dan context-aware scheduling mendukung hasil penelitian Giagkos dkk. (2024) yang menunjukkan bahwa sistem berbasis analisis QoS dapat menghasilkan rekomendasi yang lebih akurat dan relevan [14]. Integrasi AI agent pada aplikasi mobile dengan Supabase juga relevan dengan tren agentic AI yang dibahas oleh Bandi dkk. (2025) [1].

3.2.5 Analisis Matematis Logika Prioritas

Algoritma prioritas menggunakan persamaan berikut:

$$Skor_{prioritas} = w_1 \left(\frac{1}{deadline} \right) + w_2 \left(\frac{1}{durasi} \right) + w_3 (urgensi)$$

Bobot w_1, w_2, w_3 bersifat dinamis dan dapat disesuaikan berdasarkan perilaku pengguna. Semakin kecil nilai *deadline* (sisa waktu) dan *durasi* (perkiraan lama pengerjaan), serta semakin tinggi nilai *urgensi*, maka skor prioritas tugas akan semakin besar, sehingga tugas tersebut direkomendasikan untuk dikerjakan lebih dahulu.

KESIMPULAN

Penelitian ini berhasil mengembangkan aplikasi Agentic To-Do AI berbasis **Flutter** yang mampu melakukan penjadwalan otomatis dan penentuan prioritas tugas melalui penerapan mekanisme *function/tool calling* pada model kecerdasan buatan. Sistem yang dirancang menawarkan pendekatan baru dalam manajemen tugas mobile, yaitu mengubah aplikasi to-do list yang sebelumnya bersifat pasif menjadi sistem yang proaktif, adaptif, dan otonom.

Melalui metode pengembangan Scrum, seluruh fitur inti—mulai dari perancangan antarmuka pengguna berbasis Flutter, integrasi basis data **Supabase**, hingga pengolahan konteks dan rekomendasi AI—dapat diselesaikan melalui iterasi bertahap yang terukur. Pengujian menggunakan Black Box Testing, System Usability Scale (SUS), dan performance testing



menunjukkan bahwa aplikasi dapat berfungsi dengan baik, stabil, serta mudah digunakan oleh pengguna.

Integrasi function calling memungkinkan AI untuk memproses konteks tugas (deadline, durasi, kategori, dan urgensi), kemudian menghasilkan rekomendasi prioritas dan waktu pelaksanaan yang lebih optimal dibandingkan pengaturan manual. Hasil implementasi juga menunjukkan bahwa sistem mampu menampilkan rekomendasi secara real-time melalui arsitektur tiga lapisan (Frontend-Middleware-Backend) dengan dukungan **Supabase** sebagai basis data yang menyediakan sinkronisasi cepat dan fitur realtime.

Dari hasil pengujian awal, aplikasi terbukti memberikan peningkatan efisiensi manajemen tugas, terlihat dari waktu penyelesaian yang lebih cepat, prioritas yang lebih akurat, serta skor kepuasan pengguna yang lebih tinggi dibandingkan aplikasi to-do list konvensional. Temuan ini selaras dengan penelitian terdahulu tentang agentic AI dan sistem rekomendasi adaptif, serta memperkuat potensi penerapan Agentic AI di dalam aplikasi mobile modern berbasis Flutter.

DAFTAR PUSTAKA

- [1] A. Bandi, B. Kongari, R. Naguru, S. Pasnoor, dan S. Vilipala,
“The Rise of Agentic AI: A Review of Definitions, Frameworks, Architectures, Applications, Evaluation Metrics, and Challenges,”
Future Internet, vol. 17, no. 9, art. 404, 2025.
DOI: <https://doi.org/10.3390/fi17090404>
- [2] B. Ranganathan,
“Function Calling and Agents in Agentic AI: How They Work Together,”
DZone, 2024.
Tersedia: <https://dzone.com/articles/function-calling-and-agents-in-the-agentic-ai>
- [3] Y. Paddalwar,
“Agents and Tool Calling in Agentic Frameworks - The Ultimate Guide,”
Medium, 2024.
Tersedia: <https://medium.com/@yashpaddalwar/agents-and-tool-calling-in-agentic-frameworks-the-ultimate-guide-0ec446e89b55>
- [4] S. Ahn,
“Everything about AI Function Calling and MCP, the Keyword for Agentic AI,”
DEV Community, 2024.
Tersedia: <https://dev.to/snowflake1078/everything-about-ai-function-calling-mcp-the->



[keyword-for-agentic-ai-2id7](#)

[5] Jetlink AI Lab,

“Comprehensive Guide to Tool Calling for Developers: Unlocking Agentic AI,”

Jetlink Blog, 2024.

Tersedia: <https://jetlink.io/blog/comprehensive-guide-to-tool-calling-for-developers-unlocking-agentic-ai>

[6] S. Liu, W. Wu, X. Xu, T. Li, B. Pang, B. Guo, dan Z. Yu,

“Adaptive and Resource-efficient Agentic AI Systems for Mobile and Embedded Devices: A Survey,”

arXiv preprint, arXiv:2510.00078, Sept. 2025.

Tersedia: <https://arxiv.org/abs/2510.00078>

[7] C. G. Endacott, S. Aarseth, H. E. Eskedal, dan M. A. T. Fiske,

“Enacting Machine Agency When AI Makes One’s Day,”

Journal of Computer-Mediated Communication, vol. 29, no. 4, 2024.

DOI: <https://doi.org/10.1093/jcmc/zmae011>

[8] S. Pal, N. Z. Jhanjhi, A. S. Abdulbaqi, D. Akila, F. S. Alsubaei, dan A. A. Almazroi,

“An Intelligent Task Management Framework for AI-Enabled Applications,”

Sustainability, vol. 15, no. 6, art. 5104, 2023.

DOI: <https://doi.org/10.3390/su15065104>

[9] P. Weichbroth,

“Usability of Mobile Applications: A Systematic Literature Study,”

IEEE Access, vol. 8, pp. 55 563-55 577, 2020.

DOI: <https://doi.org/10.1109/ACCESS.2020.2981892>

[10] A. Bangor, P. T. Kortum, dan J. T. Miller,

“Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale,”

Journal of Usability Studies, vol. 4, no. 3, pp. 114-123, 2009.

Tersedia: <https://uxpajournal.org/determining-what-individual-sus-scores-mean-adding-an-adjective-rating-scale/>

[11] M. Yu, R. Zhou, Z. Cai, C.-W. Tan, dan H. Wang,

“Unravelling the Relationship between Response Time and User Experience in Mobile Applications,”

Internet Research, vol. 30, no. 5, pp. 1353-1382, 2020.

DOI: <https://doi.org/10.1108/INTR-05-2019-0223>

[12] P. Weichbroth,



“Usability Testing of Mobile Applications: A Methodological Framework,”

Applied Sciences, vol. 14, no. 5, art. 1792, 2024.

DOI: <https://doi.org/10.3390/app14051792>

[13] Supabase Team,

“Supabase Documentation - Database, Auth, Realtime, and Storage,”

Supabase Docs, 2024.

Tersedia: <https://supabase.com/docs>

[14] D. Giagkos, A. Tzenetopoulos, D. Masouros, S. Xydis, F. Catthoor, dan N. S. Voros,

“AI-Driven QoS-Aware Scheduling for Serverless Video Analytics at the Edge,”

Information, vol. 15, no. 8, art. 480, 2024.

DOI: <https://doi.org/10.3390/info15080480>