JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 14 No. 2, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v14i2.9200

PERANCANGAN APLIKASI MANAJEMEN TUGAS BERBASIS PREDIKSI CUACA DAN INTEGRASI KALENDER UNTUK PENINGKATAN PRODUKTIVITAS

Regani Awalludin [1*], Fathoni Mahardika[2], Dani Indra Junaedi[3]

[1,2,3]Universitas Sebelas April; Jl. Angkrek Situ No. 19, Kota Sumedang; Telp. (0261) 202911

Keywords:
Personal Productivity;
To-do list;
Weather Prediction.

Corespondent Email:
220660121030@student.unsap.ac.id

Abstrak. Aplikasi daftar tugas (to-do list) digital tradisional seringkali kurang memiliki kesadaran kontekstual yang penting untuk optimalisasi pelaksanaan tugas, khususnya terkait faktor lingkungan. Kelalaian utama pada aplikasi yang ada adalah tidak adanya integrasi prediksi cuaca granular yang berdampak signifikan pada kenyamanan aktivitas luar ruangan, serta sinkronisasi kalender yang belum holistik. Penelitian ini menghadirkan Tasktisfying, sebuah aplikasi mobile yang dikembangkan menggunakan framework Flutter. Aplikasi ini dirancang untuk menjembatani kesenjangan tersebut dengan mengintegrasikan prakiraan cuaca 5 hari (interval 3 jam) dari OpenWeather API dan sinkronisasi Google Calendar secara real-time. Metode penelitian menggunakan pendekatan pengembangan sistem waterfall dengan arsitektur client-server berbasis Firebase. Hasil pengujian menunjukkan bahwa integrasi data cuaca prediktif dan jadwal kalender dalam satu antarmuka mampu memberdayakan pengguna untuk membuat keputusan penjadwalan yang lebih cerdas, mengurangi upaya penjadwalan ulang, dan meningkatkan produktivitas personal dengan beradaptasi terhadap kondisi lingkungan.

Abstract. *Traditional digital to-do list applications often lack the contextual awareness crucial for optimizing task execution, particularly regarding environmental factors. A significant omission in existing apps is the integration of granular weather prediction, which profoundly impacts the feasibility of outdoor activities, along with seamless holistic calendar synchronization. This paper presents Tasktisfying, a mobile application developed using the Flutter framework. It is designed to bridge these gaps by incorporating 5-day (3-hour interval) weather forecasts from the OpenWeather API and real-time Google Calendar integration. The research method employs a waterfall system development approach with a Firebase-based client-server architecture. Test results demonstrate that integrating predictive weather data and calendar schedules into a unified interface empowers users to make smarter scheduling decisions, minimize rescheduling efforts, and enhance personal productivity by intelligently adapting to environmental conditions.*

1. PENDAHULUAN

Evolusi teknologi seluler yang pesat dalam dekade terakhir telah menghadirkan berbagai perangkat lunak yang bertujuan meningkatkan produktivitas personal. Di antara berbagai alat tersebut, aplikasi daftar tugas (*to-do list*) digital telah menjadi fondasi utama dalam pengorganisasian dan manajemen tugas harian

[1]. Aplikasi populer yang mendominasi pasar saat ini, seperti Microsoft To Do dan Google Tasks, menyediakan platform yang kuat bagi pengguna untuk mencatat, memprioritaskan, dan melacak tanggung jawab mereka. Namun, meskipun efektif untuk enumerasi tugas, sebagian besar alat yang sudah mapan ini memiliki keterbatasan inheren: kurangnya kesadaran kontekstual dinamis (dynamic contextual awareness), khususnya mengenai faktor lingkungan yang secara signifikan mempengaruhi kelayakan eksekusi tugas [2].

Aplikasi produktivitas saat ini umumnya masih bersifat statis. Pengguna sering kali dipaksa untuk berpindah-pindah antar aplikasi, menggunakan manajer tugas untuk perencanaan dan aplikasi cuaca terpisah untuk memvalidasi kondisi lingkungan. Hal ini menciptakan kesenjangan (gap) fungsional yang kritis. Referensi silang manual ini menimbulkan beban kognitif (cognitive burden) dan diskoneksi antara perencanaan digital dengan kondisi dunia nyata [3], [4]. Sebagai contoh, tugas yang sangat bergantung pada cuaca, seperti olahraga luar ruangan, berkebun, atau perjalanan logistik, sering dijadwalkan tanpa mempertimbangkan kendala lingkungan. Akibatnya, sering terjadi penjadwalan ulang di menit-menit terakhir yang menyebabkan inefisiensi dan frustrasi pengguna. Google Tasks, misalnya, meskipun unggul dalam kesederhanaan, tidak menyediakan peringatan intuitif jika tugas luar ruangan yang direncanakan bertepatan dengan prakiraan hujan lebat .

Selain aspek cuaca, integrasi sinkronisasi yang mulus dengan kalender pribadi untuk manajemen waktu yang holistik juga masih menjadi area yang memerlukan perbaikan. Fragmentasi antara daftar tugas dan acara kalender (events) dapat menyebabkan konflik jadwal. Pengguna membutuhkan pandangan terpadu di mana tugas dan acara kalender dapat dilihat dan dikelola dalam satu antarmuka yang cerdas.

Penelitian ini bertujuan untuk menjembatani kesenjangan ganda tersebut melalui pengembangan Tasktisfying, sebuah aplikasi seluler baru yang dikembangkan menggunakan kerangka kerja Flutter. Kebaruan (novelty) dari penelitian ini terletak pada integrasi proaktif data cuaca granular (prakiraan 5 hari dengan interval 3 jam) dari OpenWeather API dan sinkronisasi dua arah dengan Google Calendar API ke dalam satu ekosistem manajemen tugas [5], [6].

Tujuan utama penelitian ini adalah merancang dan mengimplementasikan sistem yang memberdayakan pengguna untuk membuat keputusan penjadwalan yang lebih terinformasi dan adaptif. Dengan menggabungkan data prediktif cuaca dan manajemen kalender yang kuat, aplikasi ini diharapkan dapat meminimalkan gangguan terkait cuaca dan mengoptimalkan produktivitas harian pengguna dengan beradaptasi secara cerdas terhadap kondisi lingkungan dan komitmen waktu yang ada [7]

2. TINJAUAN PUSTAKA

Bagian ini menguraikan landasan teori yang mendukung pengembangan aplikasi Tasktisfying, meliputi konsep sistem sadar konteks, integrasi antarmuka pemrograman aplikasi (API), dan kerangka kerja pengembangan perangkat lunak seluler.

2.1 Sistem Komputasi Sadar Konteks (Context-Aware Computing)

Komputasi sadar konteks adalah paradigma di mana aplikasi memiliki kemampuan untuk menemukan dan bereaksi terhadap perubahan lingkungan pengguna. Dalam manajemen tugas, "konteks" diperluas untuk mencakup faktor eksternal seperti lokasi dan kondisi cuaca.

Penerapan sistem yang mengintegrasikan rekomendasi berbasis konteks (IoT-CARS) telah terbukti meningkatkan efisiensi pengambilan keputusan pengguna [8]. Pada Tasktisfying, konsep ini diwujudkan dengan menjadikan data cuaca prediktif sebagai variabel konteks utama yang memengaruhi visualisasi dan keputusan penjadwalan tugas.

2.2 Integrasi Application Programming Interface (API)

Penggunaan API memungkinkan pertukaran data yang efisien antara aplikasi seluler dan layanan eksternal.

OpenWeather API: Layanan ini menyediakan data meteorologi granular. Studi menunjukkan bahwa integrasi OpenWeather API mampu memberikan prakiraan cuaca yang

interaktif dan akurat untuk perencanaan [9], [10]. Tasktisfying memanfaatkan endpoint prakiraan 5 hari dengan interval 3 jam untuk memberikan tingkat presisi yang diperlukan dalam perencanaan mikro.

Google Calendar API: API ini menggunakan protokol otentikasi OAuth 2.0 untuk membaca dan menulis event ke dalam kalender pengguna [6]. Integrasi ini memungkinkan Tasktisfying mencapai manajemen waktu yang holistik, di mana tugas dan acara kalender disinkronkan secara real-time dalam satu tampilan terpadu.

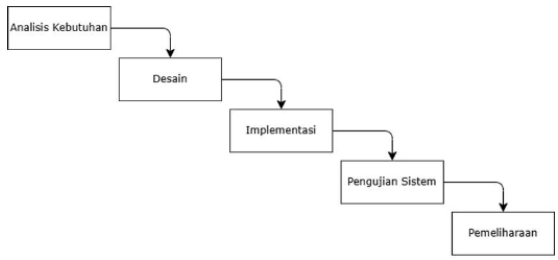
2.3 Pengembangan Aplikasi Lintas Platform (Flutter & Firebase)

Flutter: Merupakan kerangka kerja UI open-source yang memungkinkan pengembangan aplikasi lintas platform (Android dan iOS) dari satu basis kode (Dart) [11] . Flutter dipilih untuk Tasktisfying karena menawarkan efisiensi pengembangan dan kinerja native yang tinggi, penting untuk memastikan kelancaran animasi cuaca dan antarmuka pengguna yang cepat.

Firebase: Digunakan sebagai Backend-as-a-Service (BaaS) untuk menangani autentikasi pengguna dan penyimpanan data real-time. Firebase mendukung skalabilitas aplikasi Tasktisfying dan meminimalkan kebutuhan pengelolaan infrastruktur server fisik.

3. METODE PENELITIAN

Metode penelitian ini berfokus pada pendekatan pengembangan sistem waterfall yang sistematis, dimulai dari analisis kebutuhan, perancangan sistem, implementasi, pengujian, dan pemeliharaan [12]. Pendekatan ini dipilih untuk memastikan integrasi API dan fungsionalitas utama sistem dikembangkan secara terstruktur.



Gambar 1. Diagram Waterfall

Keterangan Gambar 1:

1. Analisis Kebutuhan

Pada tahap ini, dilakukan identifikasi terhadap kebutuhan-kebutuhan utama yang menjadi dasar dalam perancangan dan implementasi sistem secara keseluruhan.

2. Desain

Pada tahapan ini mencakup proses perancangan antarmuka pengguna (user interface) sebagai representasi visual dari sistem. Desain antarmuka dikembangkan berdasarkan hasil analisis kebutuhan, sehingga dapat memberikan gambaran awal mengenai alur interaksi pengguna serta tata letak fitur-fitur utama dalam aplikasi

3. Implementasi

Pada tahap ini dilakukan proses pembangunan aplikasi to-do list dengan fitur prediksi cuaca menggunakan framework Flutter sesuai dengan desain dan spesifikasi teknis yang telah ditetapkan pada tahap perancangan. Implementasi mencakup pengkodean fungsionalitas sistem, integrasi antar komponen, serta konfigurasi awal agar sistem dapat berjalan sesuai harapan.

4. Pengujian Sistem

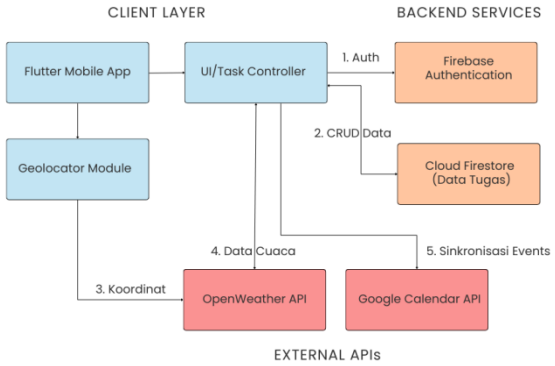
Pada tahap pengujian bertujuan untuk memastikan bahwa sistem berfungsi dengan baik dan sesuai dengan kebutuhan pengguna. Pengujian dilakukan menggunakan metode Blackbox Testing.

5. Pemeliharaan

Pemeliharaan mencakup berbagai aktivitas seperti pembaruan sistem, perbaikan bug, serta penyesuaian terhadap perubahan kebutuhan pengguna, guna memastikan aplikasi tetap relevan dan optimal digunakan secara berkelanjutan

3.1 Arsitektur Sistem

Aplikasi Tasktisfying dikembangkan menggunakan model arsitektur client-server terdistribusi, yang memanfaatkan layanan Backend-as-a-Service (BaaS) dan API eksternal.



Gambar 2. Arsitektur Sistem

Mobile Client (Frontend): Dikembangkan menggunakan framework Flutter (bahasa pemrograman Dart) untuk menjamin kompatibilitas lintas platform (Android dan iOS). Bertanggung jawab atas rendering Antarmuka Pengguna (UI), handling input pengguna, logika client-side, dan komunikasi dengan backend serta API eksternal. Flutter juga mendukung pengembangan aplikasi web dan desktop, menjadikannya solusi yang sangat serbaguna untuk berbagai kebutuhan pengembangan aplikasi [13].

Backend Services (Firebase): Digunakan sebagai database real-time untuk menyimpan dan menyinkronkan data tugas (CRUD) antar perangkat pengguna. Menyediakan layanan Firebase Authentication untuk mengelola otentikasi pengguna, khususnya melalui integrasi Google Sign-In [14].

External APIs: OpenWeather API: Digunakan untuk mengambil data prakiraan cuaca saat ini dan 5 hari (interval 3 jam) berdasarkan koordinat geografis pengguna yang diperoleh dari modul Geolocator pada perangkat seluler. Google Calendar API: Menggunakan otorisasi OAuth 2.0 untuk membaca acara kalender yang ada dan membuat acara baru langsung dari aplikasi Tasktisfying, memastikan sinkronisasi jadwal yang terpadu.

3.2 Alur Data dan Integrasi

Integrasi kunci Tasktisfying adalah alur data kontekstual yang memungkinkan perencanaan sadar lingkungan:

1. Pengambilan Lokasi:

Aplikasi menggunakan modul Geolocator untuk mendapatkan koordinat geografis pengguna.

2. Panggilan API Cuaca:

Koordinat lokasi dikirim ke WeatherService untuk mengambil data prakiraan 5 hari/3 jam dari OpenWeather API. Respons JSON dipetakan ke dalam model objek *FiveDayThreeHourForecast.*

3. Sinkronisasi Kalender:

Setiap pembuatan tugas memicu *GoogleAuthClient* untuk mengirimkan permintaan pembuatan event melalui Google Calendar API, memastikan konsistensi jadwal.

4. Visualisasi Terpadu:

Data tugas (Firebase) dan data cuaca (OpenWeather) digabungkan dan divisualisasikan dalam *Unified Schedule View* (Tampilan Jadwal Terpadu).

3.3 Skenario Pengujian Sistem

Pengujian fungsionalitas sistem dilakukan menggunakan metode Black Box Testing untuk memverifikasi kesesuaian output sistem dengan spesifikasi yang telah ditetapkan. Pengujian mencakup:

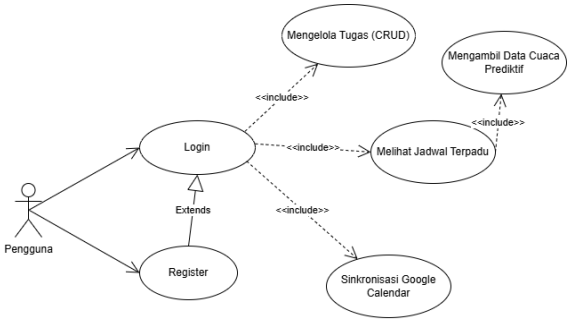
1. Validasi Akurasi Pengambilan Data Cuaca (sesuai lokasi).
2. Verifikasi Fungsionalitas CRUD (Create, Read, Update, Delete) tugas.
3. Keberhasilan dan akurasi sinkronisasi tugas dua arah dengan Google Calendar.

4. HASIL DAN PEMBAHASAN

Bagian ini memaparkan kebutuhan sistem, desain sistem, implementasi antarmuka, hasil pengujian fungsional, dan analisis komparatif dengan aplikasi sejenis.

4.1 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem merupakan tahap awal yang mengidentifikasi serta mengevaluasi permasalahan dan kebutuhan fungsional yang diinginkan. Dalam penelitian ini, digunakan Use Case Diagram untuk menggambarkan kompleksitas fungsionalitas utama yang terintegrasi dalam sistem. Diagram ini berfungsi membantu mengidentifikasi alur sistem sekaligus menjadi model yang menjelaskan bagaimana sistem digunakan oleh aktor sesuai dengan perannya [15].



Gambar 3. Use Case Diagram

Diagram ini mengidentifikasi Pengguna sebagai aktor utama yang berinteraksi dengan sistem Tasktisfying. Kebutuhan fungsional sistem berpusat pada empat fungsionalitas: Otentikasi, Manajemen Tugas, Sinkronisasi Kontekstual, dan Tampilan Terpadu.

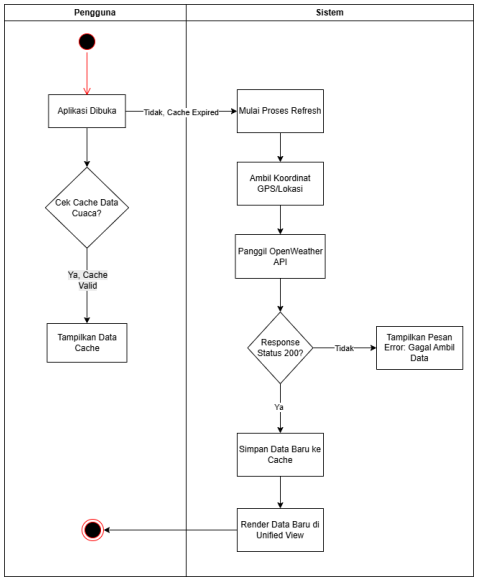
Kebutuhan paling krusial yang dipecahkan oleh penelitian ini diwakili oleh use case Melihat Jadwal Terpadu. Use case ini memiliki relasi <<*include*>> terhadap Mengelola Tugas (CRUD) dan Mengakses Data Cuaca Prediktif. Relasi inklusif ini secara eksplisit memodelkan desain sistem di mana output utama (jadwal) wajib menyajikan informasi yang telah diperkaya oleh konteks cuaca dinamis. Hal ini menjamin bahwa sistem memenuhi kebutuhan akan manajemen tugas yang sadar konteks (context-aware). Selain itu, sistem mendukung Sinkronisasi Google Calendar dan Otentikasi Pengguna, memastikan interoperabilitas dan keamanan data pengguna.

4.2 Desain Sistem

Pada tahap ini, model UML digunakan untuk memvisualisasikan proses yang terjadi dalam sistem, termasuk kondisi percabangan dan urutan pesan antar objek.

4.2.1 Activity Diagram

Activity Diagram merupakan jenis diagram Unified Modeling Language (UML) yang digunakan untuk menggambarkan dan mendeskripsikan alur sistem atau urutan aktivitas aktor maupun sistem [16]. Diagram ini memvisualisasikan Alur Sinkronisasi dan *Caching* Data Cuaca, yang merupakan proses background penting untuk menjaga efisiensi Tasktisfying.

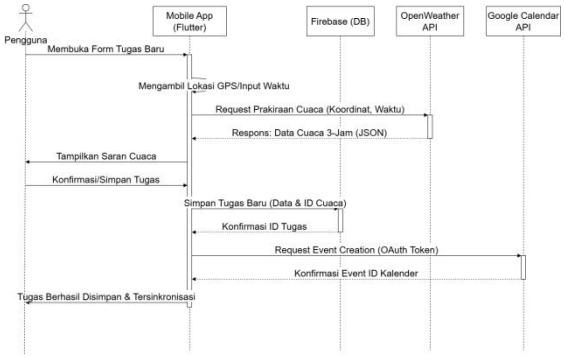


Gambar 4. Activity Diagram

Alur diawali ketika aplikasi Dibuka atau terjadi Perubahan Lokasi Terdeteksi. Proses ini segera menuju decision node *Cek Cache Data Cuaca*. Strategi *caching* ini diterapkan untuk menghindari panggilan berulang ke OpenWeather API, yang penting untuk optimasi performa dan kepatuhan rate limit API. Jika cache dinyatakan invalid atau expired, alur mengarahkan sistem untuk Panggil OpenWeather API. Keberhasilan pengambilan data dikonfirmasi melalui decision node *Response Status 200?*. Jika panggilan sukses, data prakiraan akan Simpan Data Baru ke Cache sebelum di-render di Unified View. Alur ini menunjukkan desain sistem yang efisien dan andal dalam mengelola konteks lingkungan eksternal.

4.2.2 Sequence Diagram

Diagram ini berfokus pada urutan pesan yang dikirim antar objek dalam sebuah skenario tertentu. Diagram digunakan untuk menggambarkan interaksi antara class atau objek class dan juga bisa digunakan dalam aktifitas pengujian sistem [17]. Diagram ini memvalidasi proses unik dan inovatif Tasktisfying dalam skenario Alur Penjadwalan Tugas Baru dengan Kesadaran Cuaca.



Gambar 5. Sequence Diagram

Interaksi dimulai oleh Pengguna (P) yang mengakses formulir. Mobile App (A) kemudian melakukan aksi internal untuk Mengambil Lokasi GPS/Input Waktu. Inti dari fitur context-aware terbukti pada urutan berikutnya, di mana Mobile App secara proaktif memanggil OpenWeather API (O) untuk Request Prakiraan Cuaca dan menampilkan data cuaca tersebut kembali kepada Pengguna (P) sebagai Saran Cuaca. Tindakan ini terjadi sebelum tugas disimpan, memberikan kesempatan kepada Pengguna untuk menyesuaikan jadwal secara cerdas (misalnya memindahkan jadwal lari pagi jika diprediksi hujan). Setelah Konfirmasi/Simpan Tugas oleh Pengguna, Mobile App melanjutkan dengan dua proses activation yang terpisah: penyimpanan data tugas ke Firebase (F) dan sinkronisasi event ke Google Calendar API (G). Urutan pesan ini secara teknis menunjukkan bagaimana sistem mengintegrasikan data prediktif eksternal sebagai bagian integral dari proses input data.

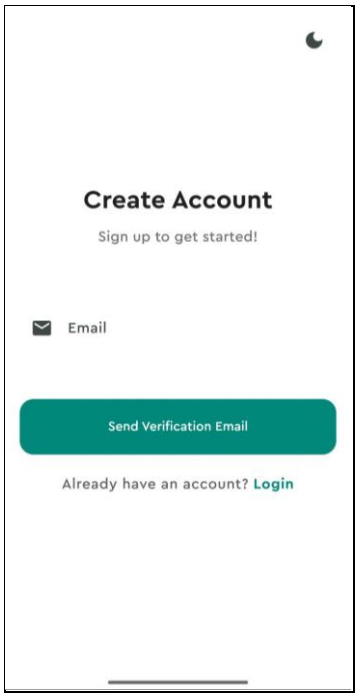
4.3 Implementasi

Implementasi desain sistem yang telah dirancang, yang dikembangkan menggunakan framework Flutter dan diintegrasikan dengan layanan backend Firebase serta API eksternal OpenWeather dan Google Calendar. Fokus implementasi adalah menciptakan pengalaman pengguna yang mulus dalam manajemen tugas yang sadar konteks.

4.3.1 Tampilan Registrasi

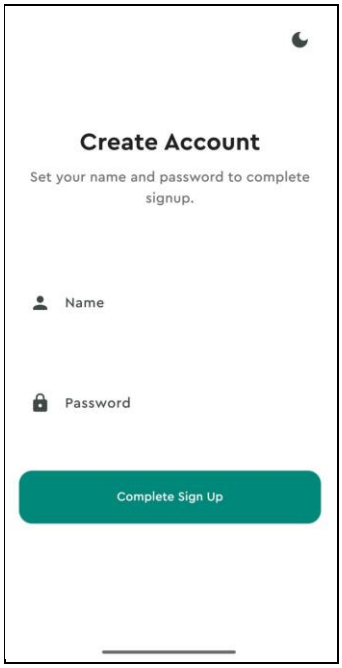
Tampilan registrasi akun merupakan langkah esensial bagi pengguna baru untuk mengakses layanan aplikasi. Proses ini dirancang dalam dua tahapan utama yang berurutan untuk memastikan verifikasi identitas pengguna dan pengaturan kredensial yang aman. Setiap tampilan didesain minimalis dengan latar belakang putih dan fokus utama diletakkan pada formulir masukan.

Tahap 1: Verifikasi Email, tampilan ini secara eksklusif berfokus pada verifikasi alamat email pengguna. Pengguna diharuskan mengisi kolom masukan berlabel "Email" yang dilengkapi dengan ikon amplop sebagai penanda visual. Setelah alamat email dimasukkan, pengguna melanjutkan proses dengan menekan tombol berlabel "Send Verification Email". Tampilan ini juga menyediakan jalur pintas bagi pengguna yang sudah memiliki akun melalui tautan "Login" di bagian bawah halaman.



Gambar 6. Tampilan Registrasi (Tahap 1)

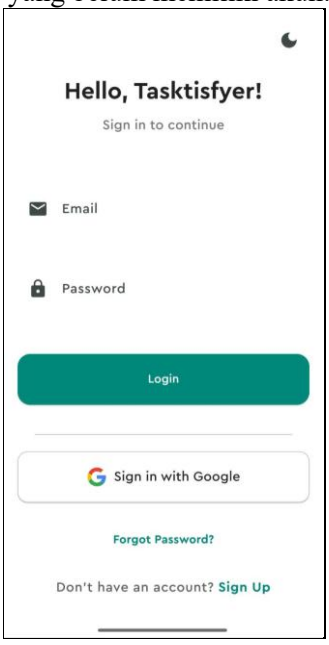
Tahap 2: Pengaturan Nama dan Kata Sandi, setelah proses verifikasi email berhasil diselesaikan tahap kedua muncul setelah verifikasi berhasil, bertujuan untuk menyelesaikan pendaftara. Pengguna mengisi kolom untuk Name dan Password, lalu mengakhiri pembuatan akun dengan menekan tombol aksi "Complete Sign Up". Kedua tampilan mempertahankan desain yang bersih dan minimalis.



Gambar 7. Tampilan Registrasi (Tahap 2)

4.3.2 Tampilan Login

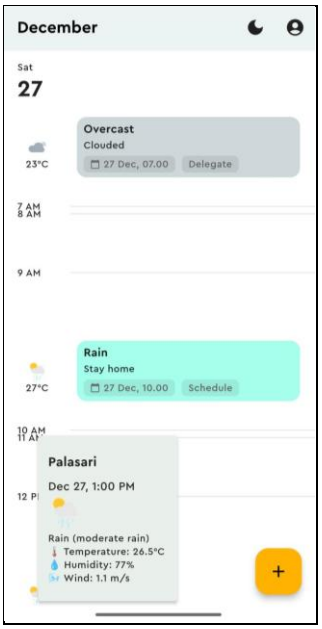
Tampilan Login menyambut pengguna dengan sapaan "Hello, Tasktisfyer!". Tampilan ini memungkinkan pengguna untuk masuk menggunakan dua cara: pertama, melalui formulir tradisional yang meminta Email dan Password dan tombol "Login"; dan kedua, melalui opsi alternatif "Sign in with Google". Sebagai bantuan tambahan, disediakan tautan "Forgot Password?" untuk pemulihan akun dan tautan "Sign Up" di bagian bawah bagi pengguna yang belum memiliki akun.



Gambar 8. Tampilan Login

4.3.3 Antarmuka Pengguna

Tasktisfying diimplementasikan dengan antarmuka yang mengedepankan clean design dan penyampaian informasi kontekstual yang proaktif. Tampilan Jadwal Terpadu (Unified Schedule View): Seperti ditunjukkan pada Gambar 9, tugas disajikan dalam urutan kronologis. Setiap blok waktu 3 jam menampilkan ikon cuaca animasi (Lottie) dan suhu prediksi.



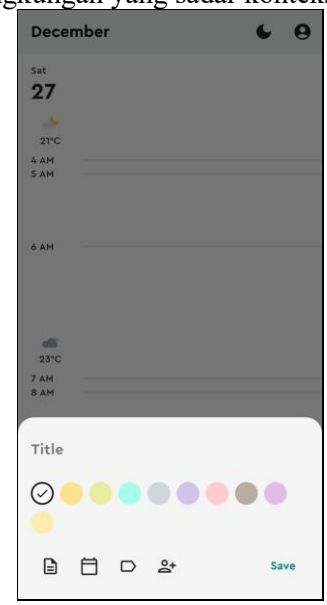
Gambar 9. Tampilan Antarmuka

Detail Cuaca Granular: Pengguna dapat mengetuk entri tertentu untuk melihat hamparan detail cuaca, termasuk kelembapan, kecepatan angin, dan probabilitas presipitasi. Misalnya, data menunjukkan pada 27 Desember pukul 1 siang di "Palasari", kondisi hujan ringan dengan suhu 26.5°C. Tingkat detail ini memungkinkan pengguna memutuskan apakah perlu membawa payung atau menunda aktivitas.

4.3.4 Tampilan Input Task

Antarmuka pembuatan tugas memungkinkan pengguna memasukkan judul, catatan, warna, tanggal jatuh tempo, label dan kolaborator, yang kemudian disinkronkan sebagai event di Google Calendar. Fungsionalitas ini merupakan bagian integral dari manajemen tugas yang komprehensif dan memperkuat kemampuan pengguna untuk

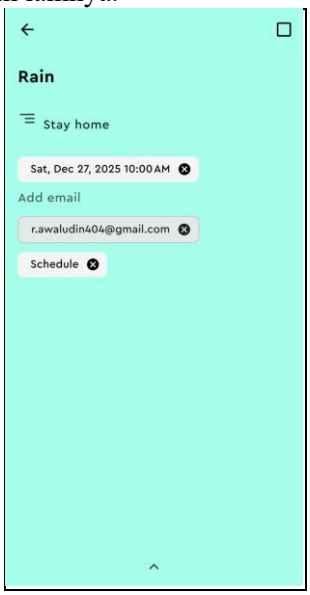
mengatur komitmen mereka secara efektif dalam lingkungan yang sadar konteks [3].



Gambar 10. Tampilan Input Task

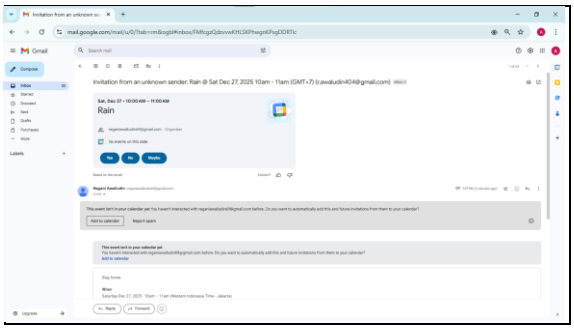
4.3.5 Tampilan Detail Tugas

Detail Tugas memberikan pandangan pada detail dan pengeditan tugas. Tugas berjudul "Rain" ini memiliki deskripsi singkat "Stay home". Detail waktu acara ditunjukkan sebagai "Sat, Dec 27, 2025 10:00 AM" dan mencantumkan email *r.awalludin404@gmail.com* di bagian "Add email". Formulir ini memungkinkan pengguna untuk melihat semua atribut tugas dalam satu tampilan dan dilengkapi dengan tombol aksi di bagian bawah seperti "Done", "Collaborators", "Label" dan lainnya.



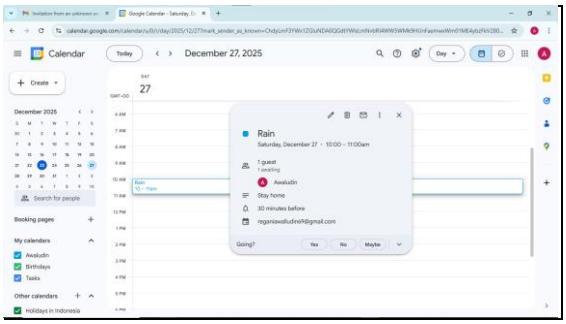
Gambar 11. Tampilan Detail Tugas (1)

Detail Tugas disajikan melalui notifikasi Gmail,yang berfungsi sebagai sarana untuk menampilkan informasi lengkap mengenai suatu tugas atau acara yang dibagikan oleh pengguna lain. Email mengidentifikasi tugas yang dijadwalkan pada hari Sabtu, 27 Desember.



Gambar 12. Tampilan Detail Tugas (2)

Ketika pengguna klik “Add to calendar” pada Google Calendar akan muncul task. Task ini juga bersifat interaktif, memberikan opsi kepada pengguna untuk langsung merespons undangan tersebut. Opsi respons yang tersedia adalah tombol "Yes" (Ya), "Maybe" (Mungkin), dan "No" (Tidak), yang memungkinkan pengguna untuk mengkonfirmasi kehadiran mereka.

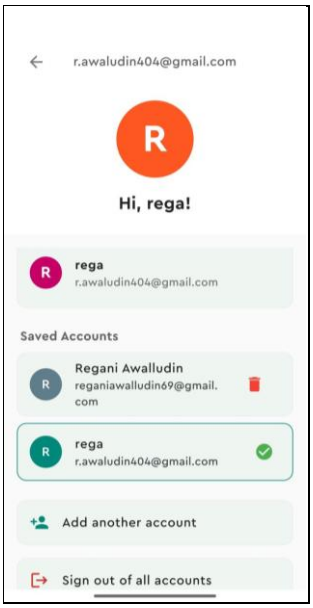


Gambar 13. Tampilan Detail Tugas (3)

4.3.6 Tampilan Logout

Tampilan Logout berfungsi sebagai pusat manajemen sesi dan akun, seperti yang ditunjukkan pada gambar pop-up akun yang menyambut pengguna "Hi, rega!". Tampilan ini memperlihatkan daftar akun yang tersimpan, termasuk akun yang sedang aktif (rega) dan akun yang dapat dihapus dari perangkat (Regani). Fungsi utamanya adalah menyediakan dua opsi aksi krusial: tombol "Add another account" untuk menambahkan sesi baru, dan yang paling penting, tombol

"Sign out of all accounts". Tombol ini memungkinkan pengguna untuk mengakhiri semua sesi aktif secara aman, yang merupakan mekanisme standar untuk menyelesaikan penggunaan sistem.



Gambar 14. Tampilan Logout

4.4 Pengujian Sistem

4.4.1 Analisis Fungsionalitas dan Performa Sistem

Pengujian fungsionalitas menggunakan metode Black Box Testing memverifikasi kesesuaian output fitur utama sistem.

Tabel 1. Pengujian Fungsionalitas

Skenario Pengujian	Hasil Pengujian	Status
Login via Google	Berhasil masuk & token valid	Valid
Ambil Data Cuaca	Ikon & suhu sesuai lokasi	Valid
Tambah Tugas + Sync	Tugas tersimpan & muncul di Calendar	Valid
Ubah Lokasi	Data ter-refresh otomatis	Valid

Pengujian kinerja pengambilan data (latency) menunjukkan bahwa waktu respons rata-rata untuk pengambilan data prakiraan 5 hari dari OpenWeather API adalah 1,25 detik, sementara waktu muat rata-rata aplikasi Tasktisfying (setelah otentikasi) adalah 2,12 detik. Data ini menunjukkan bahwa mekanisme caching dan implementasi asinkron Flutter telah berhasil menjaga performa aplikasi tetap optimal, meminimalkan lag yang mungkin timbul dari panggilan API eksternal.

4.4.2 Pembahasan Kontribusi dan Analisis Komparatif

Tasktisfying memberikan kontribusi signifikan dalam domain produktivitas pribadi dengan mengatasi masalah context switching dan kurangnya kesadaran lingkungan.

4.4.2.1 Peningkatan Efisiensi Perencanaan (Context-Awareness)

Pendekatan ini secara langsung mengurangi beban kognitif pengguna yang sebelumnya harus membandingkan rencana tugas dengan data cuaca secara manual [3], [4]. Kemampuan sistem untuk secara proaktif memberikan indikasi cuaca pada linimasa tugas menunjukkan pergeseran dari manajer tugas pasif menjadi asisten perencanaan cerdas.

Tasktisfying bertransisi dari sekadar manajemen tugas sederhana menjadi pengalaman perencanaan yang cerdas. Umpan balik awal menunjukkan bahwa visualisasi data cuaca secara langsung mengurangi beban kognitif pengguna. Pengguna tidak perlu lagi melakukan context switching (berpindah aplikasi) untuk memeriksa apakah cuaca mendukung untuk lari pagi atau rapat di luar kantor.

Granularitas data 3 jam terbukti sangat berguna untuk perencanaan mikro (micro-planning) dalam satu hari. Sebagai contoh, pengguna dapat mengidentifikasi jendela waktu cerah di antara prediksi hujan untuk melakukan pengiriman barang atau olahraga. Integrasi ini menjawab kelemahan yang disebutkan mengenai kurangnya peringatan intuitif pada Google Tasks terkait kondisi eksternal.

4.4.2.2 Analisis Komparatif Fitur dan Novelty

Tabel 2 memposisikan Tasktisfying relatif terhadap pesaing utama. Hasil pada Tabel 2 secara eksplisit menunjukkan bahwa Tasktisfying mengisi kekosongan fungsional dengan memasukkan variabel lingkungan sebagai komponen perencanaan utama.

Tabel 2. Pengujian Fungsionalitas

Fitur	Tasktisfying	Google Tasks	Microsoft To Do
Integrasi Cuaca Prediktif	Ya (3-jam interval)	Tidak	Tidak
Sinkronisasi Kalender	Ya (Tugas dan Event Terpadu)	Ya (Hanya Event)	Ya (Hanya Outlook)
Peringatan Berbasis Lokasi	Ya	Tidak	Terbatas (via OS)

Berdasarkan temuan ini, Tasktisfying berhasil mengimplementasikan sistem yang sadar konteks, memenuhi tuntutan yang dikemukakan dalam literatur tentang perlunya sistem yang dapat bereaksi terhadap lingkungan pengguna.

4.4.2.3 Optimasi Teknis

Penggunaan Flutter terbukti efisien untuk deployment lintas platform. Tantangan utama yang ditemukan adalah batas laju (rate limits) API OpenWeather. Solusi yang diterapkan adalah mekanisme caching lokal untuk meminimalkan permintaan berulang yang tidak perlu, sehingga menghemat kuota API dan mempercepat waktu muat aplikasi bagi pengguna [9].

4.5 Pemeliharaan

Tahap pemeliharaan pada aplikasi dilakukan untuk memastikan sistem tetap berjalan optimal setelah diimplementasikan. Pemeliharaan meliputi perbaikan bug yang muncul, peningkatan keamanan dengan penerapan patch terbaru, serta penyesuaian aplikasi terhadap kebutuhan baru pengguna [17].

5. KESIMPULAN

Penelitian ini menyimpulkan bahwa Tasktisfying berhasil mencapai tujuan utama dalam meningkatkan produktivitas personal melalui manajemen tugas yang sadar konteks.

a. Integrasi Proaktif Cuaca dan Kalender: Tasktisfying berhasil mengintegrasikan prediksi cuaca granular (5 hari, interval 3 jam) dan sinkronisasi Google Calendar ke dalam satu aplikasi manajemen tugas.
b. Peningkatan Efisiensi Perencanaan: Implementasi ini secara efektif memberikan proactive information delivery, mengurangi beban kognitif pengguna, dan memungkinkan pengguna untuk membuat keputusan penjadwalan yang lebih cerdas dan adaptif terhadap kondisi lingkungan.
c. Inovasi Fungsional: Tasktisfying memberikan nilai tambah yang signifikan dibandingkan aplikasi to-do list konvensional dengan menyediakan fitur kesadaran lokasi dan kondisi lingkungan, memposisikannya sebagai tool perencanaan cerdas.
d. Untuk penelitian selanjutnya dan pengembangan aplikasi lebih lanjut, disarankan untuk: 1) Menerapkan algoritma Machine Learning untuk menganalisis kebiasaan pengguna, sehingga dapat memberikan saran penjadwalan otomatis yang dioptimalkan. 2) Menambahkan notifikasi push berbasis perubahan cuaca mendadak (misalnya: "Hujan diprediksi turun 30 menit lagi,"). 3) Melakukan uji usabilitas berskala besar (misalnya menggunakan System Usability Scale / SUS) untuk mengukur tingkat kepuasan pengguna secara kuantitatif.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Fakultas Teknologi Informasi, Universitas Sebelas April Sumedang yang telah memfasilitasi penelitian ini, serta seluruh pihak yang telah berkontribusi dalam perancangan dan implementasi aplikasi Tasktisfying.

DAFTAR PUSTAKA

[1] C. G. Thomas and A. Jayanthila Devi, "A Study and Overview of the Mobile App Development Industry," *Int. J. Appl. Eng. Manag. Lett.*, vol. 5, no. 1, pp. 115–130, 2021, doi: 10.5281/zenodo.4966320.

[2] S. G. Gollagi, M. M. Math, and A. A. Daptardar, "A survey on pervasive computing over context-aware system," *SN Comput. Sci*., vol. 1, no. 2, 2020, Art. no. 110, doi: 10.1007/s42486-020-00030-6

[3] S. Inshi, R. Chowdhury, H. Ould-Slimane, and C. Talhi, "Secure adaptive context-aware ABE for smart environments," *Internet of Things*, vol. 4, no. 2, pp. 112–130, 2023, doi: 10.3390/iot4020007.

[4] M. Czerwinski, G. Jones, and K. Shahi, "The Impact of Proactive Intelligent Assistants on Human Attention and Task Interruption in Knowledge Work," *in Proc. IEEE Int. Conf. Human-Centric Comput. (HCC)*, Boston, MA, USA, 2021, pp. 123–130, doi: 10.1109/HCC.2021.00024.

[5] OpenWeatherMap, "Open Weather API Documentation." Accessed: Oct. 21, 2025. [Online]. Available: https://openweathermap.org/api.

[6] Google Cloud, "Google Calendar API overview." Accessed: Oct. 21, 2025. [Online]. Available: https://developers.google.com/workspace/calendar/api/guides/overview

[7] A. Hussain, A. Aslam, S. Tripura, V. Dhanawat, and V. Shinde, "Weather forecasting using machine learning techniques: Rainfall and temperature analysis," *J. Adv. Inf. Techno*l., vol. 15, no. 12, pp. 1329–1338, 2024, doi: 10.12720/jait.15.12.1329-1338.

[8] D. Nawara and R. Kashef, "Context-Aware Recommendation Systems in the IoT Environment (IoT-CARS)—A Comprehensive Overview," *IEEE Access*, vol. 8, pp. 103214–103249, 2020, doi: 10.1109/ACCESS.2020.2999330.

[9] U. Sharma, E. Das, Diksha, and P. Yadav, "Interactive weather forecasting system using OpenWeather API and web technologies," *Int. J. Res. -GRANTHAALAYAH*, vol. 13, no. 1, pp. 130–140, 2024, doi: 10.29121/granthaalayah.v13.i1.2025.6119.

[10] R. Pushpakumar et al., "Human-Computer Interaction: Enhancing User Experience in Interactive Systems," *in E3S Web Conf.*, vol. 399, 2023, Art. no. 04037, doi: 10.1051/e3sconf/202339904037.

[11] M. Hu and D. You, "A Comparative Study of Cross-platform Mobile Application Development," *in Proc. 12th Annu. Conf. Comput. Inf. Technol. Res. Educ. New Zealand (CITRENZ)*, 2021, pp. 66–71, doi: 10.33965/ciawi2021_202102L015.

[12] T. Thesing, C. Feldmann, and M. Burchardt, "Agile versus waterfall project management: Decision model for selecting the appropriate approach to a project," Procedia Comput. Sci., vol. 181, pp. 746–756, 2021, doi: 10.1016/j.procs.2021.01.227.

[13] M. A. S. Ramadhinata, A. Agussalim, and N. C. Wibowo, "Rancang Bangun E-Marketplace UMKM Pastry & Bakery (Bakehouse) Berbasis Mobile Menggunakan Framework Flutter," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 13, no. 1, 2025, doi: 10.23960/jitet.v13i1.5601.

[14] F. H. Anmi *et al.*, "Implementation of Flutter and Firebase technologies in modern live streaming application development," *JOISIE (J. Inf. Syst. Inform. Eng.)*, vol. 9, no. 1, pp. 246–259, 2025, doi: 10.35145/joisie.v9i1.4808.

[15] S. Narulita, A. Nugroho, and M. Z. Abdillah, "Diagram Unified Modelling Language (UML) untuk perancangan sistem informasi manajemen penelitian dan pengabdian masyarakat (SIMLITABMAS)," *Bridge: J. Publ. Sist. Inf. Telekomun.*, vol. 2, no. 3, pp. 244–256, 2024, doi: 10.62951/bridge.v2i3.174.

[16] N. Perera, D. I. De Silva, C. Serasinghe, et al., "Examining the role of software maintenance in ensuring software quality," Authorea, 2023, doi: 10.22541/au.168296460.09065818/v1.