BAB II

TINJAUAN PUSTAKA

2.1 Teori Terkait

Sebagai acuan perancangan sistem, kajian terhadap beberapa penelitian terdahulu yang relevan telah dirangkum secara komparatif pada Tabel 2. 1.

Tabel 2. 1 Teori Terkait

No.	Rujukan (Penulis /Tahun)	Data, *Dataset*, Masukan	Model, Algoritma, *Framework*	Capaian / Hasil Evaluasi	*Research Gap* (Celah Penelitian)
1.	Ash-Shafa & Andono (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika (JIPI), 2025) Pendeteksi Visual Makanan dan Jumlah Kalorinya	Menggunakan *dataset AIcrowd* Food *Recognition Challenge* sebanyak 5.545 gambar latih.	Menggunakan algoritma *Mask* R-CNN yang disambungkan dan diimplementasik	Sistem berhasil mengidentifikasi 13 jenis makanan dengan tingkat akurasi	Masih bergantung pada ekosistem platform pihak ketiga (Telegram) sebagai antarmuka dan belum

	Menggunakan Algoritma *Mask* R-CNN Berbasis Bot Telegram[50].		an pada Bot Telegram menggunakan *API*.	secara keseluruhan mencapai 78%.	dikembangkan menjadi aplikasi perangkat bergerak mandiri berbasis bahasa pemrograman Dart.
2.	Supriyadi, dkk. (Jurnal Informatika dan Teknik Elektro Terapan, 2025) Sistem Estimasi Kalori Makanan Berbasis Citra Menggunakan YOLOv8[51].	3.772 citra makanan dalam 9 kelas makanan harian di Indonesia.	Menggunakan deteksi objek YOLOv8 untuk *front-end* dan peladen berbasis *FastAPI* (Python)	Aplikasi seluler mampu mendeteksi makanan dan mengestimasi total kalori dengan	Sistem peladen pada penelitian ini dibangun menggunakan *framework FastAPI* (Python), sedangkan

			.	kecepatan respons 1–2 detik .	penelitian yang diusulkan membangun peladen *REST API* menggunakan *framework Dart Frog* (Dart) agar tercipta ekosistem kode yang selaras dan terintegrasi penuh dengan platform

					klien (*mobile Flutter*).
3.	Nadiyah, dkk. (Jurnal Kecerdasan Buatan, Komputasi dan Teknologi Informasi (COREAI), 2024) Deteksi Objek Untuk Menghitung Perkiraan Kalori Makanan Menggunakan Metode R-CNN *Mask* Berbasis Web[52].	Data citra makanan pemicu obesitas dengan total 220 citra yang dibagi ke dalam 5 kelas makanan.	Menggunakan kecerdasan buatan tipe *Mask* R-CNN yang diimplementasikan menggunakan *framework* Flask berbasis web..	Sistem mampu mendeteksi objek makanan dan memperkirakan jumlah kalori bagi seseorang yang menjalankan program diet.	Antarmuka kecerdasan buatan masih murni berupa *website* yang harus diakses melalui browser, sementara penelitian ini mengemas model cerdas tersebut

					langsung ke dalam aplikasi seluler (*mobile*).
4.	Haryana, dkk. (Amerta *Nutrition*, 2024) Pengembangan Aplikasi *Self-Dietary Assessment "Diary NutriMe"* sebagai Media Pendampingan Gizi bagi Remaja *Overweight* dan Obesitas[22].	Data asupan zat gizi harian dan Indeks Massa Tubuh (IMT) spesifik dari remaja yang mengalami kondisi berat badan lebih dan obesitas.	Dibangun menggunakan metode perancangan perangkat lunak Rapid *Application Development* (RAD) yang dilengkapi fitur	Menghasilkan perangkat lunak pendampingan diet interaktif dengan nilai tingkat kelayakan dari pakar dan ahli materi yang	Masih menganda lkan metode pencatatan gizi harian (*food diary*) secara manual oleh pengguna, dan belum menginteg rasikan fungsi kamera untuk

			chatting.	sangat tinggi (97,3%) .	mendeteksi nutrisi secara otomatis.
5.	Puspitasari & Pangaribuan (Jurnal Comasie, 2025) Deteksi Kalori Pada Citra Makanan Dengan Algoritma *Single Shot Multibox Detector*[53].	Mengumpulkan data citra makanan menggunakan tangkapan kamera dari perangkat ponsel pintar.	Menggunakan pendekatan *Deep Learning* dengan algoritma *Single Shot Multibox Detector* (SSD) *MobileNet.*	Model kecerdasan buatan mampu mendeteksi objek makanan secara *real-time* untuk mengestimasi jumlah kalori secara akurat	Masih difokuskan pada pemantauan asupan kalori mandiri untuk mencegah penyakit obesitas dan diabetes, serta belum diimplementasikan pada Program

				dengan mAP 77,20%.	Makan Bergizi Gratis (MBG).

2.2 Landasan Teori

Landasan teori berperan sebagai dasar ilmiah dalam pengembangan sistem, sehingga penelitian memiliki arah yang jelas dan pijakan yang kuat. Bagian ini memaparkan teori serta konsep yang dijadikan acuan dalam proses pengembangan sistem, termasuk pendekatan dan prinsip ilmiah yang digunakan dalam perancangan serta penerapan aplikasi. Selain itu, dijelaskan pula teknologi dan perangkat yang mendukung proses pengembangan, agar sistem dapat dirancang secara terstruktur dan sesuai dengan tujuan penelitian. Setiap teori yang digunakan dipilih berdasarkan relevansinya terhadap kebutuhan sistem, sehingga dapat menghasilkan aplikasi yang berfungsi optimal, efisien, dan mudah digunakan oleh pengguna. Berikut landasan teori yang digunakan dalam penelitian ini

2.2.1 Aplikasi Perangkat Bergerak

Secara konseptual, aplikasi perangkat bergerak merupakan perangkat lunak yang dirancang khusus untuk digunakan pada perangkat portabel, seperti ponsel pintar dan tablet. Dari segi fungsionalitas, teknologi ini pada dasarnya mengadaptasi kemampuan komputasi layaknya komputer desktop, namun dikemas ke dalam

antarmuka yang jauh lebih ringkas dan praktis guna menunjang aktivitas harian penggunanya[54]. Keunggulan utama aplikasi perangkat bergerak terletak pada tingkat fleksibilitas dan aksesibilitasnya yang tinggi, karena memungkinkan pengguna tetap terhubung dengan jaringan internet, mengakses layanan digital, serta melakukan pertukaran informasi secara *real-time* tanpa terikat oleh batasan ruang dan waktu[55].

2.2.2 Application Programming Interface (API)

Antarmuka pemrograman aplikasi, atau *API*, pada dasarnya merupakan sekumpulan instruksi dan aturan sistematis yang memungkinkan berbagai perangkat lunak untuk saling berkomunikasi, berinteraksi, dan bertukar fungsi secara terprogram[34]. Di dalam arsitektur sistem, antarmuka ini mengambil peran esensial sebagai penghubung utama yang memfasilitasi pertukaran informasi lintas platform sehingga fungsionalitas aplikasi dapat digunakan tanpa perlu merancangnya kembali dari awal[38]. Melalui implementasi layanan web ini, para pengembang dapat mendistribusikan data secara aman dan efisien melintasi jaringan guna mengintegrasikan sistem yang memiliki bahasa pemrograman maupun fondasi kerangka kerja yang sama sekali berbeda[56].

2.2.3 REST API

Layanan *web* yang menerapkan arsitektur *REST* dirancang untuk mendukung pertukaran data antar aplikasi atau sistem dengan memanfaatkan protokol komunikasi *HTTP* [39]. Karakteristik paling mendasar dari pendekatan ini terletak pada sifat operasionalnya yang *stateless*, di mana sistem peladen tidak menahan atau merekam informasi status sesi klien di antara setiap siklus interaksi yang terjadi[57]. Untuk memproses interaksi tersebut, peladen murni mengandalkan metode *HTTP* yang seragam guna mengeksekusi permintaan pengguna, seperti instruksi untuk membaca (GET), mengirim (POST), memperbarui (PUT), dan menghapus (DELETE) sumber daya[40]. Mekanisme arsitektur yang memproses setiap permintaan secara independen ini pada akhirnya menghasilkan sistem yang tangguh, tidak terbebani oleh memori sesi, serta terbukti krusial dalam mendukung *skalabilitas* layanan secara optimal[58].

2.2.4 Desain Antarmuka (UI*)*

Antarmuka pengguna merupakan elemen visual pada perangkat lunak yang membantu pengguna berinteraksi secara langsung dengan sistem[44]. Secara struktural, *UI* bertindak sebagai wujud representasi fisik yang menunjang keindahan antarmuka. Antarmuka pengguna mencakup tata letak, ikon, tombol, warna, tipografi, serta elemen visual lain yang membantu pengguna memahami dan menjalankan fungsi aplikasi. Dalam pengembangan aplikasi perangkat bergerak,

perancangan *UI* yang baik diperlukan agar informasi dapat disajikan secara jelas, fitur utama mudah ditemukan, dan proses penggunaan aplikasi menjadi lebih sederhana[59].

2.2.5 Flutter

Sebagai sebuah inovasi dalam rekayasa perangkat lunak, *Flutter* hadir sebagai kerangka kerja bersumber terbuka besutan Google yang dirancang secara khusus untuk memfasilitasi pengembangan antarmuka visual secara efisien [60]. Nilai tawar utama dari teknologi ini terletak pada kemampuannya mendukung ekosistem lintas platform, sehingga para pengembang dapat membangun aplikasi untuk perangkat Android maupun iOS secara bersamaan hanya dengan berbekal satu basis kode utama[61]. Dari segi arsitektur strukturalnya, sistem ini sepenuhnya digerakkan oleh bahasa pemrograman *Dart* dan beroperasi menggunakan mesin *rendering* mandiri yang mengonstruksi seluruh antarmuka peladennya melalui penggabungan komponen *widget*[62]. Pendekatan desain yang komprehensif tersebut tidak hanya menjanjikan performa aplikasi yang stabil layaknya sistem *native*, tetapi juga menawarkan fitur pemuatan ulang instan yang sangat krusial dalam mempercepat proses evaluasi perubahan kode[35]. Berkat kemampuannya dalam mereduksi kompleksitas teknis sekaligus menekan beban waktu dan biaya produksi ganda, *Flutter* kini menjadi salah satu solusi teknologi

yang paling relevan untuk mengakomodasi tingginya dinamika industri aplikasi seluler modern[63].

2.2.6 Dart

Sebagai fondasi utama dalam membangun perangkat lunak, bahasa pemrograman *Dart* diciptakan secara khusus oleh Google untuk memenuhi berbagai kebutuhan rekayasa teknologi masa kini secara lintas platform[55]. Secara *arsitektural*, sistem instruksi ini dikembangkan dengan mengadopsi paradigma berorientasi objek dan dirancang menggunakan struktur sintaksis yang menyerupai keluarga bahasa C sehingga sangat familier untuk dipelajari oleh para pengembang [62]. Dalam industri aplikasi seluler modern, teknologi tersebut memegang peranan esensial sebagai mesin penggerak sentral pada kerangka kerja *Flutter* guna menyusun antarmuka visual yang konsisten dan interaktif[35]. Keunggulan fundamental dari bahasa ini bermuara pada ketiadaan proses interpretasi perantara saat berkomunikasi dengan sistem, sehingga terbukti mampu menghasilkan eksekusi program yang sangat cepat, stabil, dan terpadu layaknya aplikasi bawaan mesin[64].

2.2.7 Dart Frog

Dalam pengembangan aplikasi modern, *backend* memiliki peran penting dalam mengatur proses pertukaran data dan komunikasi antar sistem. Untuk mendukung kebutuhan tersebut, dibutuhkan *framework backend* yang mampu memberikan kemudahan,

kecepatan, dan efisiensi dalam proses pengembangan aplikasi. Salah satu *framework* yang dapat digunakan adalah *Dart Frog*. *Framework* ini berbasis bahasa pemrograman *Dart* dan digunakan untuk membangun *REST API* serta aplikasi *server-side* dengan cara yang cepat, ringan, dan modern. *Dart Frog* dikembangkan menggunakan *library Shelf* serta menerapkan konsep *file-based routing*, sehingga pengelolaan *endpoint* menjadi lebih mudah dan terstruktur. Selain itu, *Dart Frog* juga menyediakan fitur *middleware* dan *dependency injection* yang membantu pengembang dalam membuat aplikasi yang lebih rapi dan efisien. *Framework* ini dirancang agar dapat terintegrasi dengan ekosistem Flutter, sehingga developer dapat menggunakan satu bahasa pemrograman yang sama, yaitu *Dart*, baik pada sisi *frontend* maupun *backend* aplikasi[65].

2.2.8 PhpMyadmin

Proses pengelolaan basis data sering kali memerlukan alat bantu yang praktis agar pengguna dapat melakukan administrasi data dengan lebih efisien. *PhpMyAdmin* merupakan aplikasi berbasis *web* yang berfungsi untuk mengelola basis data *MySQL* melalui antarmuka grafis yang intuitif, sehingga pengguna dapat melakukan berbagai operasi seperti membuat, mengubah, dan menghapus tabel, mengeksekusi perintah *SQL*, serta mengatur pengguna dan hak akses tanpa harus menulis kode secara manual. Aplikasi ini dikembangkan menggunakan bahasa pemrograman *PHP* dan biasanya diintegrasikan

dengan paket server seperti *XAMPP* atau *LAMP*. Karena kemudahan penggunaan dan kompatibilitas lintas *platform*, *phpMyAdmin* menjadi salah satu alat yang paling populer dalam pengembangan sistem berbasis *web*. Selain itu, aplikasi ini juga meningkatkan efisiensi administrasi data, terutama bagi organisasi dan usaha kecil-menengah, melalui fitur ekspor-impor data yang fleksibel dan tampilan antarmuka yang mudah dipahami bahkan oleh pengguna non-teknis[66].

2.2.9 MySQL

Dalam pengembangan aplikasi, pengelolaan data yang efisien dan terstruktur memerlukan sistem basis data yang andal dan mudah diimplementasikan. *MySQL* adalah sistem manajemen basis data relasional *Relational Database Management System*(RDBMS) bersifat *open-source* yang banyak digunakan di berbagai bidang, seperti *e-commerce*, keuangan, dan kesehatan. Sistem ini memungkinkan pengguna untuk menyimpan, mengelola, serta mengakses data menggunakan *Structured Query Language* (SQL) secara efisien. *MySQL* dikenal memiliki performa tinggi dan mendukung operasi dasar *CRUD* (Create, Read, Update, Delete) yang menjadi fondasi penting dalam pengembangan aplikasi modern. Dengan arsitektur modular, *MySQL* memungkinkan optimasi performa melalui pengaturan penyimpanan fisik dan data *tuning*, sehingga cocok digunakan pada sistem berskala besar maupun

aplikasi berbasis *web*. Keunggulan utama *MySQL* terletak pada kestabilan, serta dukungan komunitas global yang aktif, yang secara berkelanjutan menghadirkan pembaruan fitur modern untuk meningkatkan efisiensi dan keandalan pengelolaan data digital[67].

2.2.10 Visual Studio Code

Dalam pengembangan perangkat lunak, diperlukan lingkungan pengembangan yang mendukung efisiensi, fleksibilitas, dan kemudahan dalam penulisan kode. *Visual Studio Code* (VS Code) merupakan penyunting kode gratis dan bersifat sumber terbuka yang dikembangkan oleh *Microsoft*. Perangkat ini dapat digunakan pada berbagai sistem operasi, seperti *Windows, macOS, dan Linux. VS Code* juga mendukung pengembangan lintas platform melalui sistem ekstensi yang fleksibel, sehingga dapat digunakan untuk berbagai bahasa pemrograman, termasuk *JavaScript, Python, Dart*, dan bahasa lainnya. *Edit*or ini menawarkan beragam fitur unggulan seperti *syntax highlighting, debugging, auto-completion (IntelliSense)*, serta integrasi langsung dengan *Git* untuk memudahkan pengelolaan versi kode. Keunggulan utama *VS Code* terletak pada performa yang ringan namun sangat dapat disesuaikan, memungkinkan pengembang menambahkan ekstensi sesuai kebutuhan proyek. Dengan dukungan komunitas global yang aktif dan pembaruan berkelanjutan, *VS Code* telah berkembang menjadi salah satu alat utama dalam pengembangan perangkat lunak modern karena kemampuannya menggabungkan

fleksibilitas, efisiensi, dan kemudahan penggunaan dalam satu platform terpadu[68].

2.2.11 UML (Unifield Modeling Language)

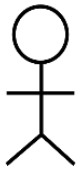
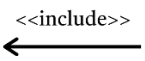
Dalam pengembangan perangkat lunak, diperlukan alat bantu pemodelan yang dapat menggambarkan struktur dan perilaku sistem secara jelas dan terstandar. *Unified Modeling Language* (UML) merupakan bahasa pemodelan standar yang digunakan untuk memvisualisasikan, merancang, dan mendokumentasikan sistem berbasis objek. *UML* menyediakan beberapa jenis diagram, seperti *use case* diagram, *class* diagram, dan *sequence* diagram, yang membantu pengembang memahami kebutuhan, alur kerja, serta struktur sistem. *UML* juga berperan sebagai media komunikasi antara analis, perancang, dan pengembang agar proses pengembangan sistem lebih terarah. Selain itu, *UML* memungkinkan sistem yang kompleks digambarkan dalam bentuk model yang lebih sederhana, sehingga memudahkan proses analisis dan perbaikan sebelum tahap implementasi dilakukan[69].

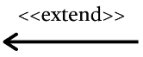
a. Use Case Diagram

Dalam pemodelan sistem, *use case* diagram digunakan untuk menggambarkan hubungan antara aktor dan fungsi-fungsi yang ada di dalam sistem. Diagram ini berperan penting dalam mengorganisir dan memodelkan perilaku sistem sesuai dengan kebutuhan dan harapan pengguna. Melalui diagram ini, hubungan

antara aktor dan fungsi sistem dapat digambarkan secara jelas, sehingga memudahkan analisis kebutuhan sebelum tahap perancangan lebih lanjut[70]. Beberapa tentang *use case* yang mendeskripsikan interaksi tersebut disajikan pada Tabel 2. 2.

Tabel 2. 2 Use Case Diagram

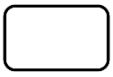
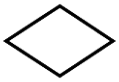
No.	Gambar	Simbol	Keterangan
1.		Aktor	Mewakili peran manusia, sistem lain, atau perangkat saat berinteraksi dengan *use case.*
2.		*Use Case*	Abstraksi dan interaksi antara *system* dan aktor.
3.		*Association*	Abstrak dari penghubung antar aktor dengan *use case*
4.		Generalisasi	Menjelaskan bentuk khusus dari seorang aktor agar dapat terlibat dalam sebuah *use case*.
5.		*Include*	Menjelaskan bahwa sebuah *use case* sepenuhnya merupakan bagian dari fungsionalitas *use case* lain.

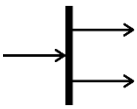
6.		*Extend*	Menggambarkan bahwa sebuah *use case* akan menambahkan fungsi tertentu pada *use case* lain apabila kondisi tertentu terpenuhi.

b. Activity Diagram

Dalam pemodelan sistem, *activity* diagram digunakan untuk menggambarkan alur aktivitas dari satu proses ke proses lainnya di dalam suatu sistem. Diagram ini berperan penting dalam memodelkan fungsi-fungsi sistem serta menyoroti alur kendali antar objek yang terjadi selama proses berlangsung[70]. *Activity* Diagram yang digunakan dalam penelitian ini ditampilkan pada Tabel 2. 3.

Tabel 2. 3 Activty Diagram

No.	**Gambar**	**Simbol**	**Keterangan**
1.		*Initial node*	Menandai titik dimulainya suatu aktivitas yang akan dilaksanakan.
2.		*Activity*	Menunjukkan cara setiap kelas antarmuka saling berinteraksi satu sama lain.
3.		*Decision*	Menjelaskan pilihan atau langkah yang perlu ditentukan ketika berada pada kondisi tertentu.

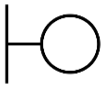
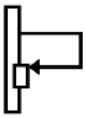
4.		*Fork node*	Satu alur proses pada suatu tahap yang kemudian bercabang menjadi beberapa alur.
5.		*Fork Join Node*	Berfungsi untuk menunjukkan aktivitas yang berlangsung secara paralel atau untuk menggabungkan dua aktivitas paralel menjadi satu alur.
6.		*Activity* Final *Node*	Menandai titik berakhirnya aktivitas yang telah dilaksanakan.
4.		*Fork node*	Satu alur proses pada suatu tahap yang kemudian bercabang menjadi beberapa alur.

c. Sequence Diagram

Dalam pemodelan sistem, *Sequence* diagram adalah diagram interaksi yang digunakan untuk menggambarkan urutan atau alur pesan antar objek dalam suatu sistem. Diagram ini menunjukkan bagaimana dan kapan objek saling berkomunikasi untuk menjalankan sebuah proses atau fungsi tertentu. Dengan demikian, *Sequence* Diagram membantu memahami alur logika dan hubungan dinamis antar komponen sistem[70]. *Sequence*

Diagram yang digunakan dalam penelitian ini di tampilkan pada Tabel 2. 4.

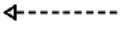
Tabel 2. 4 Sequence Diagram

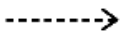
No.	Gambar	Simbol	Keterangan
1.		*Enity Class*	Menunjukkan titik awal sistem yang menjadi dasar dalam penyusunan basis data.
2.		*Boundary Class*	Menunjukkan interaksi atau pertukaran informasi antara sistem, seperti melalui *form*.
3.		*Control Class*	Berfungsi sebagai penghubung antara *boundary* dan tabel.
4.		*Recursive*	Mengirimkan pesan yang ditujukan kembali ke dirinya sendiri.
5.		*Activation*	Menggambarkan lamanya waktu suatu proses dijalankan.
6.		*Life Line*	Mengaitkan setiap objek yang memiliki aktivitas

d. Class Diagram

Salah satu diagram dalam pemodelan sistem berbasis objek adalah *class* diagram, yang berfungsi untuk menggambarkan keterkaitan antar kelas di dalam sistem. Diagram ini menampilkan atribut, metode, serta relasi antar kelas, seperti asosiasi, pewarisan, atau dependensi, sehingga membantu memahami struktur dan rancangan logika sistem secara keseluruhan[70]. yang digunakan dalam penelitian ini di tampilkan pada Tabel 2. 5.

Tabel 2. 5 Class Diagram

No.	Gambar	Simbol	Keterangan
1.		*Enity Class*	Menunjukkan titik awal sistem yang menjadi dasar pembuatan basis data.
2.		*Boundary Class*	Menunjukkan interaksi antara sistem, misalnya melalui *form*.
3.		*Control Class*	Berperan sebagai penghubung antara *boundary* dan tabel.
4.		*Recursive*	Mengirim pesan yang ditujukan kembali kepada dirinya sendiri.
5.		*Activation*	Mewakili proses durasi mengaktifkan sebuah proses

6.		*Life Line*	Menghubungkan setiap objek yang memiliki *aktivasi*.