

BAB II

TINJAUAN PUSTAKA

2.1. Penelitian Terkait

Penelitian pertama dilakukan oleh Moh. Nizar Arian, Arfan Haqiqi Sulasmoro, dan Irawan Pudja Hardjana (2021) dengan penelitian yang berjudul "Sistem Monitoring Peringatan Banjir Cerdas Berbasis Website". Penelitian ini bertujuan untuk menerapkan sistem monitoring pendeteksi banjir cerdas berbasis *website* guna mempermudah masyarakat mengetahui ketinggian air dan debit air secara *real-time* serta mengirimkan notifikasi peringatan ke WhatsApp. Kelemahan dari sistem ini adalah pemantauan masih terfokus pada penyajian data numerik standar tanpa adanya antarmuka yang secara khusus didesain adaptif terhadap situasi darurat, serta notifikasi yang belum terintegrasi secara otomatis sebagai pelaporan ke *server* instansi resmi kebencanaan [10].

Penelitian selanjutnya dilakukan oleh Sarmada dkk. (2024) dengan judul "Perancangan Antarmuka Sistem Peringatan Dini Banjir Menggunakan Web Thing Speak sebagai Basis Data dan Notifikasi Pesan melalui Aplikasi WhatsApp". Penelitian ini bertujuan untuk merancang antarmuka pengguna (*user interface*) pada sistem peringatan dini banjir yang mengintegrasikan web ThingSpeak untuk menampilkan histori data ketinggian air sungai melalui grafik serta memberikan notifikasi. Namun, kelemahan dari sistem ini terletak pada ketergantungan tampilan visualisasi data yang menggunakan standar grafik bawaan dari platform pihak ketiga (ThingSpeak), sehingga

tampilan antarmuka cenderung kaku dan kurang fleksibel untuk dikustomisasi sesuai kebutuhan visual pengguna awam [16].

Tinjauan berikutnya dilakukan terhadap penelitian Feddriyanto dan Basyir (2020) dengan judul "Aplikasi Pemantau Banjir Berbasis Web (Studi Kasus: Badan Penanggulangan Bencana Daerah DKI Jakarta)". Penelitian ini bertujuan untuk mengembangkan aplikasi pemantau banjir guna memberikan informasi status Siaga dan ketinggian air dalam membantu pihak tertentu dalam penyaluran bantuan. Kelemahan dari pengembangan ini adalah aksesibilitas sistem yang cenderung ditujukan bagi pihak internal atau admin tertentu saja, sehingga kurang optimal dalam memberikan pengalaman pengguna (*user experience*) yang responsif dan *mobile-friendly* bagi warga umum yang membutuhkan informasi cepat melalui ponsel pintar [17].

Dalam aspek sistem notifikasi, penelitian oleh Faturachman dkk. (2023) dengan judul "Design of Flood Early Warning Monitoring System Using a ESP32 Microcontroller Ultrasonic Sensor Based on Website and Telegram Chat Bot" turut ditinjau. Penelitian ini bertujuan untuk merancang sistem monitoring peringatan dini banjir yang memanfaatkan platform *chat bot* untuk penyebaran informasi status siaga, level air, dan tindakan yang harus dilakukan secara *real-time* melalui pesan otomatis. Kritik utama terhadap sistem ini adalah penggunaan media notifikasi yang masih terbatas pada aplikasi Telegram, padahal aplikasi WhatsApp memiliki penetrasi pengguna yang lebih luas di kalangan masyarakat dan aparat pemerintah daerah untuk koordinasi tanggap darurat [18].

Terkait dengan manajemen data visual dan numerik, penelitian dari Arrasyidy, Hallim, dan Rahim (2025) dengan judul "Perancangan Sistem Deteksi Banjir Berbasis IoT Untuk Peringatan Dini dengan Sistem Integrasi Web" juga menjadi referensi. Penelitian ini bertujuan untuk mengembangkan sistem deteksi banjir yang mengintegrasikan data sensor ketinggian air dan dokumentasi kamera ke dalam antarmuka *website* berbasis basis data MySQL untuk menyajikan log data pemantauan dalam format tabel kronologis. Namun, kelemahan mendasar dari sistem ini terletak pada arsitektur pengelolaan datanya yang masih dijalankan menggunakan *server* lokal (XAMPP), sehingga sistem bersifat berdiri sendiri (*standalone*) dan data riwayat kebencanaan tidak dapat diakses atau diintegrasikan secara terpusat (*cloud*) oleh instansi terkait untuk keperluan analisis mitigasi jangka panjang[19].

Berdasarkan tinjauan terhadap kelima penelitian di atas, ditemukan *research gap* (celah penelitian) bahwa belum ada sistem yang mengintegrasikan fitur Dashboard Visual Adaptif (yang mampu berubah warna latar atau indikator secara otomatis mengikuti status bahaya agar mudah dipahami orang awam) dengan Integrasi Pelaporan Otomatis ke BPBD melalui platform pesan instan yang paling umum digunakan. Penelitian ini telah dilaksanakan untuk menutup celah tersebut dengan solusi pengembangan *website* yang memiliki antarmuka adaptif serta fitur WhatsApp Gateway yang terhubung langsung untuk mempercepat respons penanggulangan bencana.

Tabel 2. 1 Kajian Pustaka Terkait Topik Penelitian Tugas Akhir

No	Penulis (Tahun)	Topik / Judul	Teknologi / Metode	Media Notifikasi	Keterangan / Kelemahan
1	Moh. Nizar Arian, dkk. (2021)	Sistem Monitoring Peringatan Banjir Cerdas Berbasis Website	ESP8266, Sensor Ultrasonik, Waterflow, Website	WhatsApp	Pemantauan terfokus pada angka numerik standar; belum memiliki antarmuka visual adaptif khusus dan belum terintegrasi ke <i>server</i> instansi resmi (BPBD).
2	Sarmada dkk. (2024)	Perancangan Antarmuka Sistem Peringatan Dini Banjir Menggunakan Web ThingSpeak	Web ThingSpeak (Platform Pihak Ketiga)	WhatsApp	Bergantung pada grafik bawaan ThingSpeak sehingga tampilan antarmuka cenderung kaku dan kurang fleksibel bagi pengguna awam.
3	Feddriyanto & Basyir (2020)	Aplikasi Pemantau Banjir Berbasis Web (Studi Kasus DKI Jakarta)	Aplikasi Berbasis Web	- (Hanya Internal)	Aksesibilitas cenderung ditujukan hanya bagi pihak internal/admin tertentu, kurang responsif untuk masyarakat umum.

No	Penulis (Tahun)	Topik / Judul	Teknologi / Metode	Media Notifikasi	Keterangan / Kelemahan
4	Faturochman dkk. (2023)	<i>Design of Flood Early Warning Monitoring System Using a ESP32...</i>	ESP32, Website	Telegram Bot	Penggunaan media notifikasi masih terbatas pada aplikasi Telegram, belum menggunakan WhatsApp yang penetrasinya lebih luas.
5	Arrasyidy, Hallim, & Rahim (2025)	Perancangan Sistem Deteksi Banjir Berbasis IoT Untuk Peringatan Dini	MySQL, XAMPP (Local Server)	-	Server masih bersifat lokal (<i>standalone</i>) sehingga data riwayat tidak terpusat (<i>cloud</i>) dan tidak bisa diakses jarak jauh oleh instansi.
6	Tugas Akhir (Penelitian Ini, 2026)	Pengembangan Sistem Monitoring Peringatan Dini Banjir Berbasis Website Terintegrasi Dengan BPBD	Framework Laravel, Tailwind CSS, MySQL (Cloud)	WhatsApp Gateway (Fonnte)	Dilengkapi <i>Dashboard Visual Adaptif</i> yang mudah dipahami orang awam dan pelaporan otomatis ganda (pesan siaran ke warga & server BPBD Tegal).

2.2. Landasan Teori

2.2.1 Tinjauan Umum Sistem Monitoring Banjir

Sistem informasi pemantauan dijelaskan sebagai suatu sistem pengawasan yang berlandaskan teknologi informasi yang dirancang untuk mengumpulkan, mengatur, dan menyajikan data secara berkesinambungan untuk mendukung proses pengambilan keputusan[20]. Dalam konteks rekayasa perangkat lunak, sistem *monitoring* tidak lagi hanya dipandang sebagai alat pengamat pasif, melainkan sebagai sebuah arsitektur sistem digital yang mampu mengolah *input* data menjadi visualisasi informasi yang bermakna bagi pengguna. Tujuan utama dari pengembangan sistem informasi *monitoring* berbasis *web* adalah untuk mengatasi latensi informasi yang sering terjadi pada metode pelaporan konvensional. Sistem ini bekerja dengan menerima *input* data digital dari sumber data, memprosesnya melalui logika pemrograman di sisi *server*, dan menampilkannya secara *real-time* pada antarmuka pengguna (*dashboard*). Dengan demikian, pengguna dapat memantau status atau anomali data dari lokasi mana saja tanpa batasan geografis, selama terhubung dengan jaringan internet.

2.2.2 Konsep Integrasi Sistem dan API

Dalam pengembangan perangkat lunak modern yang kompleks, integrasi antar-sistem menjadi elemen krusial. Salah satu metode utama untuk menghubungkan berbagai komponen perangkat lunak

adalah melalui *Application Programming Interface (API)*. *API* merupakan sekumpulan protokol dan definisi yang memungkinkan satu aplikasi perangkat lunak berkomunikasi dengan aplikasi lainnya. Dalam arsitektur sistem ini, *API* berfungsi sebagai jembatan yang membawa permintaan (*request*) dari klien ke *server* dan mengembalikan respons (*response*) data yang sesuai, seringkali menggunakan format standar seperti JSON.

Salah satu implementasi *API* yang telah digunakan dalam sistem ini adalah *WhatsApp Gateway*. *WhatsApp Gateway* merupakan sistem perangkat lunak yang menjembatani komunikasi antara sistem basis data aplikasi dengan *platform* pesan instan *WhatsApp* melalui protokol HTTP [21]. Secara teoritis, *gateway* ini bekerja dengan menerima pemicu (*trigger*) dari logika program utama misalnya ketika data *input* digital mencapai ambang batas tertentu dan kemudian mengeksekusi skrip untuk mengirimkan pesan notifikasi secara otomatis ke nomor pengguna yang terdaftar. Penggunaan *API* ini memungkinkan sistem *monitoring* untuk memberikan peringatan dini (*early warning*) secara proaktif langsung ke perangkat seluler pengguna tanpa memerlukan intervensi manual.

2.2.3 Framework Laravel

Laravel adalah kerangka kerja (*framework*) berbasis bahasa pemrograman PHP yang digunakan untuk membangun aplikasi *web* yang kuat dan skalabel [22]. *Laravel* menggunakan arsitektur *Model-*

View-Controller (MVC) yang memisahkan logika aplikasi menjadi tiga komponen utama:

- 1 *Model*: Bertanggung jawab untuk pengelolaan data dan interaksi dengan basis data. *Model* merepresentasikan struktur data logis dari aplikasi.
- 2 *View*: Menangani presentasi data atau antarmuka pengguna. Bagian ini yang dilihat dan diinteraksikan langsung oleh pengguna pada layar peramban.
- 3 *Controller*: Berfungsi sebagai pengatur alur kerja yang menerima *input* dari pengguna, memprosesnya menggunakan logika bisnis, dan menentukan respons apa yang akan dikirimkan kembali melalui *View*.

Penggunaan Laravel dalam sistem ini bertujuan untuk meningkatkan keamanan, mempercepat proses *coding* melalui fitur bawaan seperti *routing* dan otentikasi, serta mempermudah pemeliharaan kode program (*maintainability*).

2.2.4 Website dan Internet

Internet (*Interconnected Network*) adalah jaringan global yang menghubungkan miliaran komputer dan perangkat digital di seluruh dunia menggunakan standar protokol TCP/IP untuk pertukaran data [23]. Dalam ekosistem ini, *website* berfungsi sebagai antarmuka utama (*user interface*) yang memungkinkan pengguna berinteraksi dengan sistem informasi. *Website* adalah kumpulan halaman digital

yang saling terkait yang mengandung teks, grafik, dan data dinamis yang diakses melalui aplikasi peramban (*browser*). Arsitektur sistem ini menggunakan model *Client-Server* dengan alur kerja yang meliputi *input* data digital, pemrosesan *server* divalidasi dan disimpan ke dalam basis data, serta visualisasi klien dalam bentuk grafik visual atau tabel informasi.

2.2.5 Tailwind CSS

Tailwind CSS adalah kerangka kerja CSS (*Cascading Style Sheets*) yang bersifat *utility-first* [24]. Berbeda dengan kerangka kerja tradisional, Tailwind menyediakan kelas-kelas utilitas tingkat rendah yang memungkinkan pengembang untuk membangun desain antarmuka yang sepenuhnya kustom tanpa harus meninggalkan kode HTML. Penerapan *Tailwind CSS* dalam penelitian ini difokuskan pada penciptaan antarmuka pengguna (*User Interface*) yang responsif, sehingga tampilan sistem *monitoring* akan secara otomatis menyesuaikan tata letak berdasarkan ukuran layar perangkat pengguna.

2.2.6 Basis Data MySQL

MySQL merupakan sebuah sistem yang mengelola basis data *relasional* (atau dikenal sebagai *RDBMS*) yang bersifat terbuka dan memanfaatkan bahasa *Structured Query Language (SQL)* untuk pengaturan data. Dalam sistem ini, MySQL berfungsi sebagai wadah penyimpanan data terpusat yang mencatat seluruh riwayat *aktivitas*

sistem, meliputi data konfigurasi akun pengguna dan *log* data riwayat *input* digital. Struktur tabel dirancang agar memiliki relasi yang efisien, memastikan integritas data, dan kecepatan akses saat sistem melakukan *query* untuk menampilkan laporan pada *dashboard* [25].

2.2.7 Teori Metode Waterfall

Metode Waterfall merupakan salah satu jenis model pengembangan perangkat lunak yang termasuk ke dalam siklus hidup klasik (*classic life cycle*) atau model sekuensial linier (*linear sequential model*) [26]. Pendekatan *System Development Life Cycle* (SDLC) ini menuntut tahapan kerja yang sangat terstruktur dan sistematis dari atas ke bawah, yang meliputi tahapan perencanaan (*planning*), analisis kebutuhan (*requirement analysis*), desain sistem (*system design*), implementasi (*implementation/coding*), pengujian (*integration & testing*), dan pemeliharaan (*operation and maintenance*).

2.2.8 Unified Modeling Language (UML)

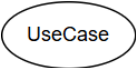
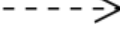
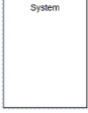
Unified Modeling Language (UML) merupakan bahasa yang diakui secara standar dan telah diterapkan dalam studi ini untuk menggambarkan, merancang, serta mengarsipkan struktur dari sistem perangkat lunak yang berbasis objek[27]. Diagram-diagram UML menawarkan berbagai alat pemodelan yang berfungsi untuk menampilkan interaksi serta perilaku dinamis yang terdapat dalam

sistem. Adapun diagram yang telah diterapkan dalam pengembangan sistem ini meliputi:

1. Use Case Diagram

Use Case Diagram Merupakan alatvisual yang menggambarkan interaksi antara pengguna (*actor*) dengan sistem. Diagram ini berfungsi untuk menyediakan deskripsi fungsionalitas sistem dan menguraikan proses-proses utama dari perspektif pengguna. Simbol-simbol yang digunakan dalam pembuatan *Use Case Diagram* beserta perannya dapat dilihat pada Tabel 2.2

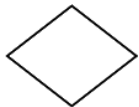
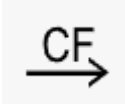
Tabel 2. 2 Use Case Diagram

No.	Simbol	Nama	Keterangan
1		<i>Actor</i>	Mewakili peran orang, sistem lain, atau alat yang berinteraksi dengan sistem.
2		<i>Use Case</i>	Fungsionalitas atau layanan yang disediakan oleh sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.
3		<i>Association</i>	Menghubungkan link antar elemen, biasanya antara aktor dan use case.
4		<i>Include</i>	Menunjukkan bahwa suatu use case seluruhnya merupakan fungsionalitas dari use case lainnya.
5		<i>System</i>	Mendefinisikan paket untuk menggambarkan sistem secara terbatas.

2. Activity Diagram

Untuk menggambarkan urutan proses dan aliran kerja (*workflow*) secara dinamis di dalam sistem, tahap pengembangan ini memanfaatkan *Activity Diagram*. Pemodelan tersebut sangat krusial dalam membantu pengembang memahami logika eksekusi program dari awal hingga akhir secara detail. Deskripsi mengenai setiap simbol yang diterapkan pada tahap perencanaan *Activity Diagram* dapat dilihat lebih lanjut pada Tabel 2.3

Tabel 2. 3 Activity Diagram

No	Simbol	Nama	Keterangan
1		<i>Start Point</i>	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2		<i>End Point</i>	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
3		<i>Activity</i>	Aktivitas yang dilakukan sistem, biasanya diawali dengan kata kerja.
4		<i>Decision</i>	Asosiasi percabangan di mana jika ada pilihan aktivitas lebih dari satu.
5		<i>Control Flow</i>	Arah aliran aktivitas dari satu status ke status lainnya.

3. Sequence Diagram

Diagram yang menunjukkan bagaimana objek-objek dalam sistem berinteraksi dan menggambarkan komunikasi antara

objek-objek tersebut menurut urutan waktu pelaksanaannya. Diagram ini digunakan untuk menjelaskan detail pesan (*messages*) yang dikirimkan antar entitas, misalnya bagaimana data sensor dikirim dari mikrokontroler ke database dan ditampilkan ke dashboard pengguna. Adapun simbol-simbol notasi yang digunakan dalam perancangan *Sequence Diagram* beserta fungsinya dapat dilihat pada Tabel 2.4

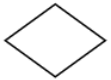
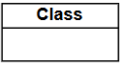
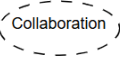
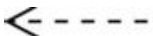
Tabel 2. 4 Sequence Diagram

No	Simbol	Nama	Keterangan
1.		<i>Lifeline</i>	Objek entity, antara muka yang saling berinteraksi.
2.		<i>Message</i>	Spesifikasi dari komunikasi antar objek yang memuat informasi tentang aktivitas yang terjadi.
3.		<i>Actor</i>	Menggambarkan orang yang sedang berinteraksi dengan system.
4.		<i>Boundary Class</i>	Menggambarkan penggambaran dari form.
5.		<i>Entity Class</i>	Menggambarkan hubungan kegiatan yang akan dilakukan.
6.		<i>Control Class</i>	Menggambarkan penghubung antara Boundary dengan label.
7.		<i>Activation</i>	Sebagai sebuah objek yang akan melakukan sebuah aksi.
8.		<i>Message</i>	Mengindikasikan komunikasi antar objek dengan objek.
9.		<i>Self Message</i>	Mengindikasikan komunikasi kembali kedalam sebuah objek itu sendiri.

4. Class Diagram

Class Diagram adalah tipe diagram yang menggambarkan struktur sistem, menjelaskan kelas, paket, serta hubungan antar objek seperti pewarisan, asosiasi, dan lain sebagainya. Dalam pengembangan aplikasi berbasis web, *Class Diagram* sering digunakan sebagai representasi logis dari struktur basis data dan relasi antar tabel. Simbol-simbol yang terdapat pada *Class Diagram* dapat dilihat dalam Tabel 2.5

Tabel 2. 5 Simbol Class Diagram

No	Simbol	Nama	Keterangan
1.		<i>Generalization</i>	Hubungan di mana objek anak (<i>descendant</i>) mewarisi perilaku dan struktur dari objek induk (<i>ancestor</i>).
2.		<i>Nary Association</i>	Upaya ntuk menghindari asosiasi dengan lebih dari 2 objek.
3.		<i>Class</i>	Himpunan dari objek-objek yang berbagai atribut serta operasi yang sama.
4.		<i>Collaboration</i>	Deskripsi dari urutan aksi yang ditampilkan dalam sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5.		<i>Dependency</i>	Operasi yang benar-benar dilakukan oleh suatu objek.
6.		<i>Dependency</i>	Hubungan Dimana perubahan yang terjadi pada suatu elemen (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya

No	Simbol	Nama	Keterangan
			elemen yang tidak mandiri.
7.	—————	<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.

2.2.9 Pengujian Sistem (Black Box Testing)

Black Box Testing adalah metode pengujian perangkat lunak yang dilakukan untuk memverifikasi apakah aplikasi telah memenuhi persyaratan fungsional tanpa perlu mengetahui struktur internal kode program (*source code*). Metode ini berfokus pada aspek eksternal aplikasi dengan mengamati hasil keluaran (*output*) berdasarkan masukan (*input*) yang diberikan. Pengujian ini krusial untuk memastikan seluruh fitur *website* dapat berjalan dengan benar sebelum diimplementasikan secara luas [28].