BAB II

TINJAUAN PUSTAKA

2.1 Teori Terkait

Penelitian terdahulu yang dilakukan oleh Prastya, E. P., & Misbah (2024). Pengembangan Sistem Presensi Mahasiswa Menggunakan ESP32 Berbasis *Internet of Things* Penelitian ini mengembangkan sistem presensi berbasis fingerprint yang terintegrasi dengan teknologi *Internet of Things* (IoT) dan ESP32. Sistem ini memungkinkan pengelolaan presensi secara *real-time* dengan memanfaatkan sensor *fingerprint*, RTC untuk pengaturan waktu, dan Google *Spreadsheet* untuk penyimpanan data presensi. Penelitian ini relevan dengan pengembangan sistem presensi di SMK Dinamika Kota Tegal karena melibatkan teknologi *fingerprint* dan aplikasi web untuk pengelolaan data kehadiran secara efisien[5].

Penelitian lainya juga dilakukan oleh Nirmala Putri Ismail, Alif Abdul Hakim, Tegar Subagdja, dan Uung Ungkawa (2025) dalam jurnal berjudul Sistem Presensi dan Rekapitulasi di SMP Negeri 1 Batujajar. Penelitian ini mengembangkan sistem presensi digital berbasis teknologi dengan menggunakan framework Laravel dan database terpusat untuk mempermudah pencatatan kehadiran siswa. Sistem ini mengatasi masalah ketidakakuratan dan keterlambatan dalam proses presensi manual, memungkinkan pemantauan kehadiran secara *real-time* dan mempercepat pembuatan laporan presensi Implementasi sistem ini juga membantu meningkatkan efisiensi

pengelolaan presensi dan transparansi data kehadiran di lingkungan pendidikan[6].

Penelitian serupa juga dilakukan oleh oleh Nirsal Nirsal dan St. Aminah (2024) dalam jurnal berjudul Desain Sistem Presensi Fingerprint Berbasis IoT untuk Meningkatkan Efisiensi Pengelolaan Kehadiran di Institusi Pendidikan. Penelitian ini mengembangkan sistem presensi berbasis fingerprint yang terintegrasi dengan *Internet of Things* (IoT), memungkinkan pencatatan kehadiran secara otomatis dan *real-time*. Sistem ini dirancang dengan pendekatan System-Centered Design (SCD) yang lebih berfokus pada efisiensi dan kinerja sistem, memaksimalkan kecepatan, akurasi, dan meminimalkan kesalahan dalam proses presensi[7].

Terdapat juga penelitian yang dilakukan oleh Indra Dharma Wijaya, S.T., M.MT berjudul "Perancangan Sistem Presensi Mahasiswa Berbasis Smart Card dan Fingerprint Menggunakan Framework ITIL." Penelitian ini mengembangkan sistem presensi mahasiswa yang menggunakan smart card dan fingerprint untuk memastikan keakuratan data kehadiran. Dengan mengintegrasikan teknologi fingerprint dan smart card, sistem ini mampu meminimalkan kecurangan seperti menitipkan absen dan memastikan data kehadiran yang lebih akurat dan efisien[8].

2.2 Landasan Teori

Landasan teori merupakan konsep, teori, prinsip, dan pendapat yang mendukung proses perancangan sistem dan sumber daya yang digunakan

dalam perancangan sistem yang dibuat. Berikut dijelaskan teori-teori yang menjadi dasar atau pedoman dalam Rancang Bangun Sistem Presensi Berbasis Fingerprint di SMK Dinamika Tegal untuk Meningkatkan Efektivitas Pengelolaan Kehadiran Website.

2.2.1. *Website*

Website (WEB) atau juga dikenal dengan *Worl Wide Web* atau WWW adalah sebuah salah satu layanan yang didapat oleh pemakai komputer yang terhubung ke internet. *Website* atau situs dapat diartikan sebagai kumpulan halaman-halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara, dan atau gabungan dari semuanya yang saling terkait dimana masing-masing dihubungkan dengan jaringan-jaringan halaman (*hyperlink*)[9].

Gambar 2. 1 Logo *Website*

2.2.2. *Application Programming Interface* (API)

API atau (*Application Programming Interface*) adalah adalah sekumpulan fungsi, *subroutine*, protokol komunikasi, atau kakas/*tools* untuk membuat *software* perangkat lunak[10].

2.2.3. Laravel

Laravel adalah framework PHP yang dirancang untuk membangun aplikasi web secara efisien dan terstruktur. Dalam konteks sistem presensi berbasis fingerprint di SMK Dinamika Kota Tegal, Laravel menyediakan berbagai fitur seperti routing autentikasi, dan integrasi database yang memungkinkan pengelolaan data kehadiran secara *real-time*. Dengan menggunakan Laravel, pengembangan aplikasi presensi dapat dilakukan dengan cepat, memudahkan pembuatan sistem yang kuat dan mudah dikelola[11].

Gambar 2. 2 Logo Laravel

2.2.4. MySQL

MySQL adalah aplikasi sistem manajemen basis data yang digunakan untuk membuat dan mengelola basis data dengan menggunakan perintah SQL, *Structured Query Languange* (SQL) adalah bahasa database komputer yang dirancang untuk mengelola data dalam sistem manajemen bisnis rasional[12].

Gambar 2. 3 Logo MySQL

2.2.5. Database

Database adalah sebuah sistem yang dibuat untuk mengorganisasi, menyimpan, dan menarik data dengan mudah. Fungsi dari database antara lain meminimalisir suatu data ganda, menjadi alternatif lain terkait masalah penyimpanan ruang dalam suatu aplikasi, serta mempermudah pengguna user dalam berbagai hal misalnya pada saat penginputan data baru[13].

2.2.6. *PhpMyAdmin*

PhpMyAdmin adalah sebuah aplikasi atau perangkat berbasis *opensouce* yang bisa digunakan secara gratis untuk melakukan pemrograman ataupun administrasi pada database MySQL. PhpMyAdmin sendiri menggunakan bahasa PHP untuk pemrogramannya, selain itu PhpMyAdmin mendukung berbagai operasi MySQL, diantaranya (mengelola basis data, tabel-tabel, bidang (*fields*). Relasi (*relations*). Indeks, pengguna (*users*), perijinan (*permissions*), dan lain-lain[14].

Gambar 2. 4 Logo *PhpMyAdmin*

2.2.7. *Visual Studio Code* (VS Code)

Visual *Studio Code* (VS Code) adalah teks editor yang diciptakan Microsoft sebagai sistem operasi multiplatform, bahasa

pemrograman JavaScript, TypeScript, Node.js, dan bahasa pemrograman lain menggunakan pertolongan plugin pada via marketplace Visual Studio Code (seperti C++, C#, Pyhton, Go, Java, dst)[15].

Gambar 2. 5 Logo *Visual Studio Code*

2.2.8. *Unifed Modeling Languange* (UML)

Unified Modeling Language (UML) adalah sebuah model visual atau sebuah standar penulisan yang dapat disebut sebagai blueprint yang mengandung bisnis proses, serta penulisan kelas dituangkan ke dalam bentuk bahasa yang spesifik. UML juga biasa dipakai dalam proses perancangan terhadap sistem atau software yang akan dibangun yang berorientasikan terhadap objek[10].

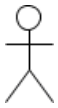
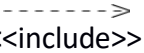
Terdapat beberapa diagram UML yang sering digunakan dalam pengembangan sebuah sistem yaitu:

1. *Use Case* Diagram

   Use Case Diagram bertujuan untuk mengilustrasikan bagaimana individu akan menggunakan dan memanfaatkan sistem, sementara aktor merupakan entitas yang berinteraksi dengan sistem. Diagram *use case* biasa digunakan untuk

mengilustrasikan aktivitas seperti apa yang seharusnya dapat dilakukan pengguna terhadap sistem yang dibuat. Simbol-simbol yang digunakan dalam *use case* diagram yaitu:

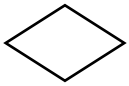
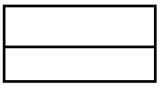
Tabel 2. 1 *Use Case* Diagram

No	Simbol	Keterangan
1.		*Use Case*, abstraksi dan interaksi antara sistem dan aktor.
2.		*Aktor*, menggambarkan manusia atau suatu hal yang berinteraksi dengan sistem.
3.		*Association*, Penghubung antara aktor dengan *usecase.*
4.		*Generalisasi*, Menunjukan spesialisasi aktor untuk dapat berpartisipasi dengan *usecase.*
5.	 <<include>>	*Include*, Menunjukan bahwa suatu *usecase* seluruhnya merupakan fungsionalitas dari *usecase* lainnya.
6.	 <<extend>>	*Extend*, Menunjukkan bahwa suatu *usecase* merupakan tambahan fungsional dari *usecase* lainnya jika suatu kondisi terpenuhi.

2. *Class* Diagram

Class Diagram menggambarkan dengan jelas struktur serta deskripsi *class,* atribut, metode, dan hubungan dari setiap objek. *Class* diagram bersifat statis yang mana hanya menjelaskan hubungan apa yang terjadi. Simbol-simbol yang digunakan dalam c*lass* diagram yaitu:

Tabel 2. 2 *Class* Diagram

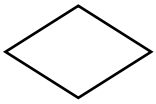
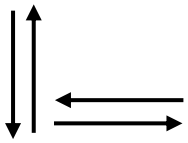
No	Simbol	Keterangan
1.		*Generalization*, Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor).*
2.		*Nary Association*, Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3.		*Realization*, Operasi yang benar-benar dilakukan oleh suatu objek.
4.		*Dependency*, Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
5.		*Association*, Apa yang menghubungkan antara objek satu dengan objek lainnya.
6.		*Class*, Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.

3. *Activity* Diagram

Penggambaran berbagai alur aktivitas data yang sedang dirancang dilakukan di *activity* diagram, yang akan menggambarkan proses berjalan dan memahami proses sistem secara menyeluruh. Simbol-simbol yang digunakan dalam *activity* diagram yaitu:

Tabel 2. 3 *Activity* Diagram

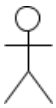
No	Simbol	Keterangan
1.		*Activity*, Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.

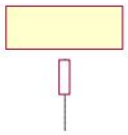
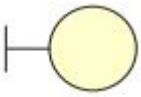
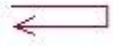
No	Simbol	Keterangan
2.		*Action*, State dari sistem yang mencerminkan eksekusi dari suatu aksi.
3.		*Start Point*, Diletakkan pada pojok kiri atas dan merupakan awal aktivitas.
4.		*End Point*, Akhir aktivitas.
5.		*Decision*, Digunakan untuk menggambarkan suatu keputusan / tindakan yang harus diambil pada kondisi tertentu.
6.		*Line Connector*, Digunakan untuk menghubungkan satu simbol dengan simbol lainnya.

4. *Sequence* Diagram

Sequence Diagram menjelaskan dan memodelkan *use case*, berfungsi memodelkan sebuah logika dari sebuah metode operasi, fungsi, atau prosedur. Simbol-simbol yang digunakan dalam *Sequence* Diagram yaitu:

Tabel 2. 4 *Sequence* Diagram

No	Simbol	Keterangan
1.		*Actor*, Mereprentasikan entitas yang berada diluar sistem dan berinteraksi dengan sistem.
2.		*Lifeline*, Menghubungkan objek selama *sequence* (*message* dikirim atau diterima dan aktifitasinya).

No	Simbol	Keterangan
3.		*General*, Merepsentasikan entitas tunggal dalam *sequence* diagram.
4.		*Boundary*, Berupa tepi dari sistem, seperti *user interface* atau suatu alat yang berinteraksi dengan sistem yang lain.
5.		*Control*, Elemen mengatur aliran dari informasi untuk sebuah scenario. Objek ini umunya mengatur perilaku dan perilaku bisnis.
6.		*Entitas*, Elemen yang bertanggung jawab menyimpan data atau informasi. Berupa beans atau model objek.
7.		*Activation*, Suatu titik dimana objek mulai berpartisipasi dalam sebuah *sequence* yang menunjukkan kapan sebuah objek mengirim atau menerima objek.
8.		*Message Entry*, Berfungsi untuk menggambarkan pesan/hubungan antar objek yang menunjukkan urutan kejadian yang terjadi.
9.		*Message to Self*, Simbol ini menggambarkan pesan/hubungan objek itu sendiri, yang menunjukkan urutan kejadian yang terjadi.
10.		*Message Return*, Menggambarkan hasil dari pengiriman *message* dan digambarkan dengan arah dari kanan kekiri.