BAB II
TINJAUAN PUSTAKA

2.1. Teori Terkait

Sistem *monitoring* keamanan brankas pernah dibuat dan dikembangkan dengan beragam fungsinya, penggunaannya dapat disesuaikan kebutuhan penelitian. Tujuan adanya rancang bangun alat ini sebagai pembantu menjaga sebuah arsip penting di ke pemerintahan desa.

Penelitian sebelumnya telah melakukan penelitian mengenai *monitoring* sistem keamanan brankas, diantaranya penelitian yang dilakukan oleh Muhammad Ichsan Nudin dan Tatang Wirawan Wisjhnuadji (2022) pada jurnal berjudul "PENERAPAN SISTEM MONITORING DAN KONTROLING PADA KEAMANAN BRANKAS BERBASIS IOT" merancang sistem keamanan brankas menggunakan mikrokontroler *NodeMCU* dengan metode autentikasi menggunakan kartu *RFID* serta *keypad* sebagai pengaman tambahan. Sistem ini terhubung melalui aplikasi sehingga dapat memberikan notifikasi kepada pemilik apabila terjadi akses masuk pada brankas [6]. Sistem ini telah berhasil melakukan kontrol dan *monitoring* pada brankas, namun belum memiliki integrasi manajemen *monitoring* berbasis *website* serta belum dilengkapi dengan dokumentasi aktivitas yang dapat dilihat pada periode tertentu sebagai bentuk manajemen keamanan.

Penelitian lainnya juga di kemukakan oleh Desnanjaya (2022) pada jurnal berjudul **"**SISTEM BRANKAS BERBASIS INTERNET OF

THINGS MENGGUNAKAN ARDUINO MEGA 2560" berkaitan dengan prototipe brankas IoT berbasis *web* telah mengintegrasikan sistem *database* untuk pencatatan aktivitas, namun pencatatan masih bersifat pasif tanpa mekanisme verifikasi pengguna berlapis. Sistem pencatatannya tidak memberikan fitur filtrasi riwayat akses berdasarkan identitas pengguna, perangkat, *timestamp*, atau lokasi aktivitas, sehingga pendekatan *monitoring* lebih menekankan pada notifikasi insiden, bukan analisis keamanan berbasis data *(data-driven)*. Dengan demikian sistem belum dapat mendukung fungsi *security investigation*, misalnya melihat pola percobaan akses ilegal, jam kritis terjadinya percobaan pembobolan, atau korelasi waktu notifikasi terhadap respons pengguna [7].

Sementara itu Ramadhan (2024) pada jurnal berjudul "PROTOTYPE SISTEM KEAMANAN BRANKAS BERBASIS SENSOR SIDIK JARI" dengan rancangannya menggunakan *Firebase* telah memaksimalkan *push notification real-time*, memberikan benefit dalam *incident response time*, namun penelitian tersebut belum mengintegrasikan dengan *website* sebagai pusat kendali dan dokumentasi. Data sensor dikirim secara *real-time* tetapi tidak disediakan modul visual analisis data seperti grafik percobaan akses atau ringkasan aktivitas harian yang memungkinkan pemilik menganalisis tren percobaan keamanan dalam jangka panjang. Sistem masih bersifat *linear* (mengirim, menerima, selesai) tanpa manajemen akses atau *historical intelligence* [8].

Penelitian selanjutnya dilakukan oleh Mahendra (2025) pada jurnal berjudul **"**PERANCANGAN SISTEM KEAMANAN BRANKAS MENGGUNAKAN FINGERPRINT DAN GPS TRACKING BERBASIS INTERNET OF THINGS**"** yang merancang sistem keamanan brankas dengan mengimplementasikan *GPS tracking* sebagai salah satu fitur *monitoring* lokasi secara *real-time* serta mengirimkan notifikasi melalui Telegram [9]. Penggunaan *GPS* memberikan fitur keamanan tambahan terhadap pemindahan fisik brankas, namun sistem ini masih terbatas dalam hal dokumentasi riwayat akses pengguna dan belum menggunakan *website* sebagai pusat pemantauan dan pengelolaan akses secara *online*.

Penelitian selanjutnya dilakukan oleh Ahmad Nadiyan Ishom Abu Sahal dkk. (2024) pada jurnal berjudul "PROTOTYPE ALAT PENGAMAN LEMARI PENYIMPANAN BARANG MENGGUNAKAN ESP3D CAM BERBASIS IOT DI YAYASAN PANTI ASUHAN AR ROHMAH" mengembangkan sistem keamanan lemari penyimpanan dengan memanfaatkan modul *ESP32-CAM*, sensor getar *SW-420*, dan *solenoid door lock* sebagai pengunci elektronik. Sistem ini dihubungkan ke internet sehingga mampu menjalankan fungsi *monitoring* jarak jauh melalui Bot Telegram [10]. Pengguna dapat membuka dan mengunci lemari dari aplikasi Telegram serta menerima notifikasi *real-time* ketika terdeteksi getaran atau percobaan akses ilegal, lengkap dengan pengiriman foto kondisi terbaru dari kamera sebagai bukti visual.

Walaupun telah mendukung notifikasi cepat dan pemantauan langsung, sistem ini masih memiliki kekurangan, yaitu belum tersedia fitur dokumentasi riwayat akses secara historis dan belum menggunakan *website* sebagai pusat kendali maupun manajemen aktivitas. *Monitoring* hanya berfokus pada peringatan saat kejadian (*event-based*) tanpa penyimpanan data jangka panjang, sehingga tidak dapat digunakan untuk analisis pola percobaan pembobolan atau evaluasi keamanan secara menyeluruh.

2.2. Landasan Teori

Adapun landasan teori pada "Sistem *monitoring* dan manajemen keamanan brankas berlapis ganda berbasis *website*" berpacuan dengan berbagai penelitian terdahulu di antaranya:

2.2.1. Sistem Monitoring

Monitoring adalah proses pengumpulan dan analisis informasi berdasarkan indikator yang ditetapkan secara sistematis dan kontinu tentang suatu kegiatan atau program sehingga mampu dilaksanakan tindakan koreksi untuk penyempurnaan kegiatan itu selanjutnya. *Monitoring* akan memberikan informasi tentang status dan kecenderungan bahwa pengukuran dan evaluasi yang diselesaikan berulang dari waktu ke waktu [11]. Pemantauan umumnya dilakukan untuk tujuan tertentu, untuk memeriksa terhadap proses berikut objek atau untuk mengevaluasi kondisi maupun kemajuan menuju tujuan hasil manajemen atas efek tindakan dari beberapa jenis antara lain tindakan untuk mempertahankan manajemen yang sedang

berjalan. *Output monitoring* berguna pada perbaikan mekanisme proses kegiatan dimana *monitoring* dilakukan.

2.2.2. Website

Website adalah sekumpulan halaman digital yang saling terhubung dan disimpan pada *server* sehingga dapat diakses melalui jaringan internet menggunakan *web browser*. *Website* berfungsi sebagai media penyedia informasi, komunikasi, yang memungkinkan pengguna berinteraksi secara langsung ataupun tidak langsung. Melalui protokol *HTTP* atau *HTTPS* [12].

2.2.3. Laragon

Laragon menyediakan fitur seperti *auto*-virtual *host* yang memungkinkan setiap proyek otomatis mendapatkan *domain* lokal sendiri, *hashing password*, dan validasi input. Selain itu, terdapat *Artisan CLI*, yaitu alat baris perintah untuk mempercepat pembuatan file penting dan Laragon adalah sebuah *local development environment* yang dirancang untuk memudahkan developer dalam membuat dan menjalankan aplikasi web secara lokal tanpa perlu melakukan konfigurasi rumit. Sebagai *software* yang ringan dan portabel, Laragon dapat dijalankan dengan cepat dan tidak membebani sistem, bahkan bisa dipindahkan ke perangkat lain hanya dengan menyalin folder instalasinya. Selain itu, Laragon mendukung penggunaan berbagai versi PHP yang dapat diganti dengan mudah, sehingga sangat membantu ketika mengerjakan proyek yang

membutuhkan versi berbeda. Dengan integrasi komponen seperti *Apache/Nginx*, *MySQL/MariaDB*, *PHP*, dan *Node.js* dalam satu paket, Laragon menawarkan pengalaman pengembangan yang efisien, terorganisir, dan nyaman, menjadikannya pilihan ideal untuk pemula maupun *developer* berpengalaman yang ingin bekerja lebih cepat tanpa konfigurasi yang kompleks [13].

Gambar 2.1 Logo *Laragon*

2.2.4. MySql

MySQL adalah implementasi *open source* yang dikembangkan menggunakan *Structured Query Language (SQL)* berfungsi sebagai sistem manajemen basis data. Saat ini, *MySQL* menjadi salah satu pilihan *database* paling populer karena kemudahan penggunaan, berbagai keperluan seperti membuat dan mengelola *database*, menyimpan serta mengolah data, *MySQL* sangat populer sebagai *database* utama [14].

Gambar 2.2 Logo *MySql*

2.2.5. Laravel

Laravel adalah sebuah *framework PHP* modern yang digunakan untuk membangun aplikasi *web* dengan struktur yang rapi dan aman. *Laravel* memakai arsitektur *MVC*, sehingga logika program, tampilan, dan data dipisahkan dengan jelas. *Framework* ini dilengkapi *Eloquent ORM*, yang memudahkan interaksi dengan *database* tanpa harus menulis *query SQL* secara manual, serta *Migration* yang mengatur perubahan struktur database melalui kode sehingga lebih terkontrol.

Untuk bagian tampilan, *Laravel* menggunakan *Blade Template Engine,* yang memungkinkan pengembang membuat *layout HTML* secara dinamis dan terstruktur. Sistem *routing Laravel* sangat fleksibel sehingga mudah mengatur alur *URL* dan dukungan *middleware* memungkinkan pengecekan keamanan atau kondisi-kondisi tertentu.

Dalam hal keamanan berkat fitur otomatis seperti *CSRF protection, hashing password*, dan validasi input. Selain itu, terdapat

Artisan CLI, yaitu alat baris perintah untuk mempercepat pembuatan file penting dan berbagai proses pengembangan. Dengan ekosistem yang luas seperti *Laravel Breeze*, *Jetstream*, *Sanctum*, dan *Livewire*, *Laravel* menjadi *framework* yang lengkap untuk membangun aplikasi modern mulai dari *website* sederhana hingga aplikasi besar dan kompleks [15].

Gambar 2.3 Logo *Laravel*

2.2.6. Boostrap

Bootstrap merupakan *framework front-end* berbasis *HTML*, *CSS*, dan *JavaScript* yang dirancang untuk mempermudah proses pengembangan antarmuka pengguna pada aplikasi berbasis *web*. *Bootstrap* menyediakan *grid system* yang fleksibel serta berbagai komponen antarmuka siap pakai seperti navigasi, tombol, formulir, tabel dan *alert* yang dapat digunakan kembali. *Framework* ini menerapkan konsep desain responsif sehingga tampilan *website* dapat menyesuaikan secara otomatis dengan berbagai ukuran layar perangkat, baik pada komputer maupun perangkat bergerak. Selain

itu, *Bootstrap* mendukung konsistensi desain antarmuka serta mempercepat proses pengembangan karena struktur dan komponen yang telah terstandarisasi. Dengan demikian, penggunaan *Bootstrap* dapat meningkatkan efisiensi serta memberikan pengalaman pengguna yang lebih baik melalui antarmuka yang rapi, responsif, mudah dioperasikan, dan mendukung kenyamanan pengguna [16].

Gambar 2.4 Logo *Boostrap*

2.2.7. Visual Studio Code

Visual Studio Code adalah editor kode sumber *open-source* yang dikembangkan oleh Microsoft dan tersedia untuk Windows, macOS, serta Linux. Meskipun tergolong ringan, VS Code memiliki fitur yang sangat lengkap untuk kebutuhan pengembangan perangkat lunak. Secara bawaan, editor ini mendukung *JavaScript*, *TypeScript*, dan *Node.js*, sementara bahasa pemrograman lain seperti *Python*, *C++*, *C#*, *PHP*, *Java*, dan ditambahkan dengan mudah melalui ekstensi [17].

Gambar 2.5 Logo *Visual Studio Code*

2.2.8. Figma

Figma adalah aplikasi desain grafis berbasis web yang dirancang khusus untuk pembuatan antarmuka pengguna *user interface* (UI) dan pengalaman pengguna *user experience* (UX). Secara fungsional, Figma sangat berfokus pada perancangan grafis vektor dan pembuatan *prototipe* aplikasi, sehingga relatif mudah digunakan. Figma bekerja secara kolaboratif menggunakan teknologi *cloud*, memungkinkan pengembang untuk merancang desain [21].

Gambar 2.6 Logo *Figma*

2.2.9. PHP

PHP singkatan dari *Hypertext Preprocessor*, adalah bahasa pemrograman bersifat *open source* yang dirancang khusus untuk pengembangan aplikasi web dan dapat disisipkan langsung ke dalam kode *HTML*. Secara *sintaks*, *PHP* memiliki kemiripan dengan beberapa bahasa pemrograman seperti *C*, *Java*, sehingga relatif mudah dipahami. *PHP* bekerja sebagai bahasa *scripting* di sisi *server*, artinya proses eksekusi kode dilakukan oleh *server* sebelum hasilnya dikirimkan kepada *client* Dengan kata lain, *server* akan mengolah skrip *PHP* terlebih dahulu, kemudian mengirimkan *output* berupa halaman yang sudah diproses kepada pengguna [18].

Gambar 2.7 Logo *PHP*

2.2.10. UML (*Unifield Modeling Language*)

Unified Modeling Language (UML) adalah bahasa standar yang digunakan untuk membuat diagram atau model perangkat lunak. UML memungkinkan pengembang atau *developer* memvisualisasikan, mendefinisikan, mendokumentasikan sistem perangkat lunak secara jelas dan terstruktur. Analogi yang sering

digunakan adalah seperti seorang arsitek bangunan yang membuat denah sebagai acuan bagi perusahaan konstruksi; demikian pula seorang *arsitek* perangkat lunak membuat diagram UML agar pengembang dapat membangun sistem sesuai dengan rancangan yang telah ditetapkan. Dengan memahami notasi dan kosakata UML, seseorang dapat lebih mudah memahami alur dan struktur sistem, serta menjelaskan desain perangkat lunak kepada pihak lain yang terlibat dalam pengembangan [19].

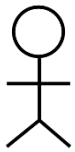
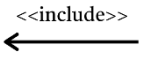
Unified Modeling Language sendiri termasuk salah satu metode pemodelan visual yang banyak digunakan dalam perancangan perangkat lunak berbasis objek. berfungsi sebagai standar penulisan atau semacam *blueprint*, yang di dalamnya mencakup bisnis proses, penulisan kelas, dan berbagai elemen sistem lainnya dalam bahasa yang spesifik. Terdapat beragam diagram yang sering digunakan dalam pengembangan sistem untuk menggambarkan struktur, perilaku, dan interaksi antar elemen sistem.

1. Use Case Diagram

Menggambarkan fungsionalitas yang diharapkan dari sebuah sistem dan merepresentasikan sebuah interaksi antara aktor dan sistem.

Didalam *Use Case* Diagram terdapat (6 enam aktor) yang merupakan sebuah gambaran entitas dari manusia atau sebuah sistem yang melakukan pekerjaan di sistem.

Tabel 2.1 *Use Case* Diagram

No	Gambar	Simbol	Keterangan
1.		Aktor	Mewakili peran seorang *system* yang lain, atau alat ketika berkomunikasi dengan *Use Case.*
2.		Use Case	Mewakiili Abstraksi dan interaksi antara sebuah *System* dan *Actor.*
3.		*Association*	Abstraksi penghubung antar *Actor* dengan *Use Case.*
4.		*Generalisasi*	Menunjukkan spesialisasi Actor yang terlibat *Use Case*
5.		*Include*	Menunjukan bahwa suatu *Use Case* seluruhnya merupakan fungsionalitas dari *Use Case* lainnya.

6.	<<extend>> ←	*Extend*	Menunjukan bahwa suatu *Use Case* merupakan tambahan fungsional dari *Use Case* lainnya jika suatu kondisi terpenuhi.

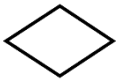
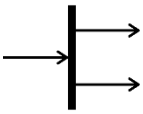
2. Activity Diagram

Menggambarkan alur kerja dari suatu proses dan urutan aktivitas, yang dimodelkan sebagai alur kerja dari satu aktivitas ke aktivitas berikutnya atau dari satu aktivitas ke aktivitas lainnya

Di dalam *Activity* Diagram terdapat beberapa simbol yang digunakan untuk menggambarkan alur aktivitas dan proses kerja dalam sistem.

Tabel 2.2 *Activity* Diagram

No	Gambar	Simbol	Keterangan
1.	●	*Initial node*	Merupakan awal aktivitas yang akan dibuat.
2.	▭	*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi.

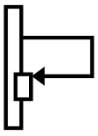
3.		*Decicion*	Menggambarkan suatu keputusan atau tindakan yang harus diambil pada kondisi tertentu.
4.		*Fork Join node*	Digunakan untuk menunjukan kegiatan yang dilakukan secara pararel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
5.		*Fork node*	Satu aliran pada tahap yang berubah menjadi beberapa aliran.
6.		*Activity Final node*	Merupakan akhir aktivitas yang telah dibuat.

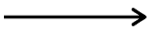
3. Sequence Diagram

Menggambarkan interaksi antar objek didalam dan disekitar sistem yang berupa *message* yang digambarkan terhadap waktu.

Di dalam *Sequence* Diagram terdapat beberapa simbol yang digunakan untuk alur komunikasi pesan antar objek dalam sistem.

Tabel 2.3 *Sequence* Diagram

No	Gambar	Simbol	Keterangan
1.		*Entity Class*	Menggambarkan awal dari sebuah sistem, menjadi dasar penyusunan basisdata.
2.		*Boundary Class*	Menggambarkan komunikasi antara sistem seperti *Form.*
3.		*Control Class*	Penghubung antara *boundary* dengan *table.*
4.		*Rescursive*	Mengirimkan pesan yang akan dikirim ke dirinya sendiri.
5.		*Activation*	Menghubungkan proses durasi aktivasi sebuah proses.
6.		*Life Line*	Menghubungkan tiap-tiap objek yang mana terdapat *activation.*

7.		*Message*	Sebagai pengirim pesan ke tiap-tiap *Class.*

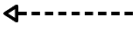
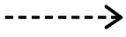
4. Class Diagram

Mengambaran struktur dan deskripsi dari *class, package* dan objek yang saling berhubungan seperti diantaranya pewarisan, asosiasi dan lainnya

Di dalam *Class* Diagram terdapat beberapa simbol yang digunakan untuk hubungan dan struktur data dalam sistem.

Tabel 2.4 *Class* Diagram

No	Gambar	Simbol	Keterangan
1.		*Class*	Menggambarkan kumpulan objek-objek dengan atribut dan operasi yang sama.
2.		*Nary Association*	Sebagai upaya menghindari asosiasi dengan lebih dari 2 objek.

3.		*Collaboration*	Urutan aksi-aksi yang ditampilkan oleh sistem yang menghasilkan suatu hasil yang tertukar bagi suatu *actor.*
4.		*Generalization*	Menggambarkan dimana hubungan objek *descendent* berperilaku dan struktur data objek yang di atasnya objek induk *ancestor.*
5.		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6.		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri akan mempengaruhi elemen yang tidak mandiri.