# BAB II

# TINJAUAN PUSTAKA

## 2.1 Teori Terkait

Dalam membangun sistem informasi manajemen pelanggan berbasis *website*, terdapat beberapa teori utama yang mendasari pengembangan sistem ini, salah satunya adalah teori *Customer Relationship Management (CRM)*[4]. *CRM* merupakan pendekatan strategis yang digunakan untuk mengelola interaksi antara perusahaan dan pelanggan guna meningkatkan kepuasan, loyalitas, serta nilai jangka panjang pelanggan terhadap perusahaan. Melalui *CRM*, perusahaan dapat memahami kebutuhan pelanggan secara lebih mendalam dan memberikan layanan yang lebih personal dan responsif.

Selain itu, konsep *Electronic Customer Relationship Management (E-CRM)* menjadi pengembangan dari *CRM* tradisional dengan dukungan teknologi informasi[5]. *E-CRM* menggabungkan perangkat lunak dan sistem berbasis *web* untuk mengotomatisasi proses pemasaran, penjualan, dan pelayanan pelanggan. Sistem ini memungkinkan penyimpanan data pelanggan secara terpusat, pemantauan histori transaksi, serta pengelolaan komunikasi yang lebih efisien melalui saluran digital.

Pengembangan sistem informasi ini juga mengacu pada teori pengembangan perangkat lunak, khususnya metode *Waterfall*, yang memiliki alur kerja sistematis mulai dari tahap analisis kebutuhan, desain sistem,

implementasi, pengujian, hingga pemeliharaan. Metode ini cocok digunakan dalam proyek yang memiliki ruang lingkup dan kebutuhan yang telah didefinisikan dengan jelas sejak awal[6].

Di sisi lain, teori basis data juga menjadi dasar penting dalam perancangan sistem ini, mengingat kebutuhan untuk menyimpan, mengelola, dan mengakses data pelanggan secara cepat dan akurat. Penggunaan sistem manajemen basis data relasional seperti *MySQL* memberikan dukungan dalam pengelolaan struktur data yang kompleks dan terintegrasi. Dengan penggabungan teori-teori ini, sistem informasi manajemen pelanggan yang dibangun diharapkan mampu memberikan nilai tambah bagi perusahaan dalam mengelola hubungan dengan pelanggan secara lebih efektif dan modern[7].

Pemilihan teknologi dan pendekatan yang tepat sangat penting untuk memastikan efisiensi dan efektivitas sistem. *Framework Laravel*, yang berbasis *PHP*, menyediakan struktur *MVC (Model-View-Controller)* yang memudahkan pengembangan aplikasi *web* yang terorganisir dan *skalabel*[8] Penggunaan *Tailwind CSS* memungkinkan pengembangan antarmuka pengguna yang responsif dan konsisten, sementara *Filament* sebagai admin panel *Laravel* mempercepat pembuatan *dashboard* yang intuitif. Integrasi *Midtrans* sebagai *gateway* pembayaran memfasilitasi transaksi *online* yang aman dan efisien. Dalam konteks ini, *MySQL* berperan sebagai sistem manajemen basis data relasional yang handal untuk menyimpan dan mengelola data pelanggan secara terstruktur. Pendekatan ini sejalan dengan

penelitian oleh Amalina , yang mengembangkan sistem informasi E-CRM berbasis *web* pada Dinikoe Keramik, menggunakan metode *Waterfall* dan teknologi *web* untuk meningkatkan hubungan dengan pelanggan. Mereka menekankan pentingnya integrasi antara sistem informasi dan strategi bisnis untuk meningkatkan kepuasan dan loyalitas pelanggan[9].

## 2.2 Landasan Teori

Landasan teori merupakan konsep, prinsip, dan pendapat yang mendukung penelitian terkait perancangan dan pembangunan sistem informasi manajemen pelanggan berbasis *website* pada PT CYB Media. Teori-teori yang disajikan dalam bab ini diperoleh dari berbagai literatur dan penelitian terdahulu yang relevan dengan tema transformasi digital serta penerapan teknologi informasi. Adapun teknologi yang digunakan dalam pengembangan sistem ini meliputi *PHP*, *Laravel*, *Filament*, *Tailwind CSS*, *Midtrans*, *Visual Studio Code*, dan *MySQL*.

### 2.2.1. *PHP*

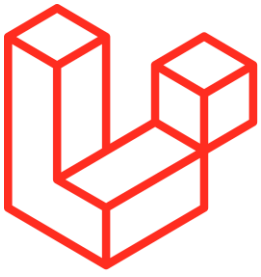
*PHP* adalah bahasa pemrograman *server-side* yang banyak digunakan dalam pengembangan aplikasi *web* dinamis. Keunggulan *PHP* terletak pada kemudahan integrasi dengan berbagai basis data dan dukungan komunitas yang luas[10]. Menurut penelitian oleh Wibowo dan Putra (2021), *PHP* memungkinkan pengembangan aplikasi *web* yang efisien dan fleksibel, serta mendukung berbagai *framework* modern seperti *Laravel*[11].

Gambar 2. 1 Logo Php

Sumber : https://www.php.net/images/logos/new-php-logo.svg

### 2.2.2. *Laravel Framework*

*Laravel* adalah *framework PHP* yang dirancang untuk mempermudah proses pengembangan aplikasi *web* dengan sintaks yang elegan dan ekspresif. *Laravel* menyediakan berbagai fitur seperti *routing*, *middleware*, dan *ORM (Object-Relational Mapping)* yang memudahkan pengelolaan basis data. Versi terbaru, *Laravel* 12, dengan berbagai pembaruan untuk meningkatkan produktivitas pengembang[12].



Gambar 2. 2 Logo *Laravel*

Sumber : https://upload.wikimedia.org/wikipedia/commons/thumb/9/9a/Laravel.svg/1969px-Laravel.svg.png

### 2.2.3. *Filament Admin Panel*

*Filament* adalah paket *Laravel* yang menyediakan antarmuka admin panel yang modern dan responsif. Dengan *Filament*, pengembang dapat dengan cepat membangun panel administrasi tanpa perlu menulis banyak kode dari awal. *Filament* mendukung pembuatan *CRUD (Create, Read, Update, Delete)* dengan cepat dan mudah, serta integrasi dengan *Tailwind CSS* untuk desain antarmuka yang konsisten[13].

### 2.2.4. *Tailwind Css*

*Tailwind CSS* adalah *framework CSS utility-first* yang memungkinkan pengembang membangun desain antarmuka yang responsif dan konsisten dengan menggunakan kelas-kelas *utility* yang telah disediakan. *Tailwind CSS* memungkinkan pembuatan desain yang fleksibel dan efisien tanpa harus menulis *CSS* secara manual[14].


Gambar 2. 3 Logo Tailwind Css

Sumber : https://tailwindcss.com/brand

### 2.2.5. *Midtrans Payment gateway*

*Midtrans* adalah penyedia layanan *payment gateway* di Indonesia yang memungkinkan integrasi berbagai metode pembayaran dalam

aplikasi *web*. Dengan *Midtrans*, perusahaan dapat menerima pembayaran melalui transfer bank, kartu kredit, e-*wallet*, dan metode lainnya. *Midtrans* menyediakan *API* yang mudah digunakan untuk integrasi dengan berbagai platform, termasuk *Laravel*[15].

Gambar 2. 4 Logo Midtrans

Sumber : https://midtrans.com/

### 2.2.6. *Mysql*

*MySQL* adalah sistem manajemen basis data relasional yang banyak digunakan dalam pengembangan aplikasi *web*. *MySQL* memungkinkan penyimpanan dan pengelolaan data secara efisien, serta mendukung integrasi dengan berbagai bahasa pemrograman. *MySQL* menyediakan fitur-fitur seperti transaksi, *indexing*, dan *query optimization* yang mendukung pengembangan sistem informasi yang handal[16].


Gambar 2. 5 Logo Mysql

Sumber : https://www.mysql.com/downloads/

### 2.2.7. *Visual Studio Code*

*Visual Studio Code (VS Code)* adalah editor kode sumber yang ringan namun kaya fitur, dikembangkan oleh *Microsoft*. *VS Code* mendukung berbagai bahasa pemrograman dan dilengkapi dengan fitur seperti *IntelliSense*, *debugging*, dan integrasi dengan *Git*. Editor ini sangat cocok untuk pengembangan aplikasi *web* dengan teknologi seperti *PHP* dan *Laravel*[17].

Gambar 2. 6 Logo *Visual Studio Code*

Sumber : https://code.visualstudio.com/brand

### 2.2.8. *Unifed Modeling Languange* (*UML*)

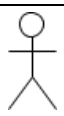
*Unified Modeling Language* (*UML*) merupakan bahasa pemodelan standar yang digunakan secara luas dalam rekayasa perangkat lunak untuk memvisualisasikan, merancang, dan mendokumentasikan sistem perangkat lunak[18]. *UML* dikembangkan oleh *Object Management Group* (OMG) sebagai alat bantu komunikasi antara pengembang dan pemangku kepentingan dalam menggambarkan struktur maupun perilaku sistem secara grafis dan terstruktur[19].

Dalam *UML* terdapat beberapa jenis diagram yang umum digunakan, antara lain: *use case* diagram untuk menunjukkan interaksi antara pengguna (aktor) dan sistem, *activity* diagram untuk menggambarkan alur proses atau aktivitas dalam sistem, *sequence* diagram untuk memperlihatkan urutan interaksi antar objek dalam sistem secara kronologis, dan *class* diagram yang merepresentasikan struktur kelas dan relasi antar kelas. Masing-masing diagram ini memberikan sudut pandang yang berbeda dan saling melengkapi dalam mendeskripsikan sistem yang akan dikembangkan.

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

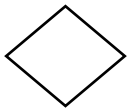
| No | Simbol | Keterangan |
|---|---|---|
| 1. |  | *Use Case*, abstraksi dan interaksi antara sistem dan aktor. |
| 2. |  | *Aktor*, menggambarkan manusia atau suatu hal yang berinteraksi dengan sistem. |
| 3. |  | *Association*, Penghubung antara aktor dengan *usecase.* |

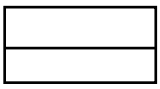
| | | |
|---|---|---|
| 4. |  | *Generalisasi*, Menunjukan spesialisasi aktor untuk dapat berpartisipasi dengan *usecase.* |
| 5. |  <<include>> | *Include*, Menunjukan bahwa suatu *usecase* seluruhnya merupakan fungsionalitas dari *usecase* lainnya. |
| 6. |  <<extend>> | *Extend*, Menunjukkan bahwa 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. |

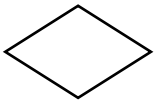
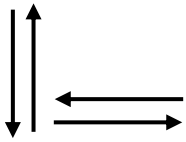
| No | Simbol | Keterangan |
|---|---|---|
| 4. |  | *Dependency*, Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri. |
| 5. |  | *Association*, Apa yang menghubungkan antara objek satu dengan objek 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

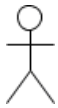
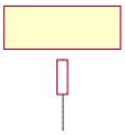
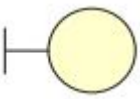
| No | Simbol | Keterangan |
|---|---|---|
| 1. |  | *Activity*, Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain. |
| 2. |  | *Action*, State dari sistem yang mencerminkan eksekusi dari suatu aksi. |
| 3. |  | *Start Point*, Diletakkan pada pojok kiri atas dan merupakan awal aktivitas. |
| 4. |  | *End Point*, Akhir aktivitas. |

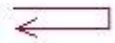
| No | Simbol | Keterangan |
|---|---|---|
| 5. |  | *Decision*, Digunakan untuk menggambarkan suatu keputusan / tindakan yang harus diambil pada kondisi tertentu. |
| 6. |  | *Line Connector*, Digunakan untuk menghubungkan satu simbol dengan simbol lainnya. |

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). |
| 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. |

| No | Simbol | Keterangan |
|---|---|---|
| 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. |