BAB II
TINJAUAN PUSTAKA

2.1 Teori Dasar

2.1.1 Rancang Bangun

Rancang bangun yaitu definisi yang digunakan untuk mendesain atau membuat suatu objek dari awal hingga selesai merupakan. Proses rancang bangun meliputi visualisasi, mengatur dan menyusun komponen-komponen yang berbeda untuk menciptakan satu kesatuan yang kohesif dan bermanfaat. Oleh karena itu, rancang bangun dapat diartikan sebagai aktivitas menerjemahkan hasil analisis menjadi *software* dan menciptakan atau memperbaiki sistem (Wulandari, S., et al. 2021).

2.1.2 Aplikasi

Definisi aplikasi ialah sebuah program yang memiliki fungsi memproses perintah untuk menjalankan permintaan *user* dengan suatu tujuan tertentu. Aplikasi dirancang untuk mempermudah pengguna dalam menyelesaikan tugas atau mencapai hasil yang diinginkan. Aplikasi dapat berbentuk berbagai jenis, seperti aplikasi *desktop*, aplikasi web, atau aplikasi *mobile*, masing-masing dengan fungsi dan kegunaan spesifik. Misalnya, aplikasi perkantoran seperti pengolah kata dan *spreadsheet* membantu dalam pembuatan dokumen dan pengelolaan data, sementara aplikasi hiburan seperti pemutar musik dan video memberikan pengalaman multimedia (Malik, D. S., & Zein, A., 2022).

2.1.3 Sistem Informasi

Sistem informasi didefinisikan sebagai sistem yang digunakan oleh suatu organisasi untuk melakukan transaksi sehari-hari dan untuk melaksanakan tugas operasional, manajerial, dan strategis. Laporan yang diperlukan dihasilkan oleh sistem informasi. Sistem informasi terdiri dari sekelompok orang yang berkolaborasi

untuk mencapai tujuan bersama. Ini menyediakan informasi dalam bentuk laporan-laporan kepada pihak-pihak yang memerlukan informasi yang bersifat manajerial (Dewi, R. K., Ardian, Q. J., & Isnaini, F., 2021).

Berikut adalah beberapa contoh sistem informasi yang digunakan dalam berbagai bidang (Sahusilawane, W., dkk., 2023):

1. **Sistem Informasi Manajemen (MIS)**
   Sebuah sistem yang disebut SIM digunakan untuk mengatur dan mengelola data yang dibutuhkan manajemen untuk membuat pilihan taktis, strategis, dan operasional. Contoh: ERP untuk menggabungkan serta mengaitkan fungsi-fungsi bisnis seperti fungsi biaya, fungsi *human resource*, fungsi inventaris, serta fungsi lain-lain ke dalam satu *platform*.
2. **Sistem Informasi Akuntansi (AIS)**
   AIS berfungsi untuk menggabungkan, melakukan penyimpanan, dan mengolah data akuntansi untuk membuat laporan keuangan yang bermanfaat bagi pihak luar dan manajemen. Contoh: QuickBooks atau SAP FinancialAccounting.
3. **Sistem Informasi Sumber Daya Manusia (HRIS)**
   HRIS mengelola data karyawan, penggajian, rekrutmen, pelatihan, dan evaluasi kinerja. Contoh: Workday, SAP SuccessFactors, atau BambooHR.
4. **Sistem Informasi Geografis (GIS)**
   Sistem ini mengumpulkan, melakukan analisis, serta menampilkan data yang memiliki lokasi geografis. Contoh: ArcGIS, Google Earth, atau QGIS.
5. **Sistem Informasi Pemasaran (MKIS)**
   Sistem ini mengelola data pemasaran dan pelanggan untuk mendukung keputusan pemasaran. Contoh: Salesforce, HubSpot, atau Marketo.
6. **Sistem Pendukung Keputusan (DSS)**

Sistem ini membantu manajemen dalam membuat keputusan dengan menyediakan informasi analitis yang relevan. Contoh: IBM Cognos Analytics, Microsoft Power BI, atau Tableau.

7. **Sistem Informasi Eksekutif (EIS)**

   Sistem ini menyediakan informasi penting kepada eksekutif untuk membantu dalam pengambilan keputusan strategis. Contoh: SAP BusinessObjects, Oracle Hyperion, atau IBM Watson Analytics.

8. **Sistem Informasi Medis (MIS)**

   Sistem ini mengelola data medis pasien dan membantu dalam administrasi rumah sakit. Contoh: Electronic Health Records (EHR) seperti Epic, Cerner, atau Allscripts.

9. **Sistem Manajemen Rantai Pasokan (SCM)**

   Sistem ini mengelola aliran barang, informasi, dan uang di seluruh rantai pasokan dari pemasok hingga pelanggan akhir. Contoh: SAP SCM, Oracle SCM, atau JDA Software.

10. **Sistem Manajemen Hubungan Pelanggan (CRM)**

    Sistem ini mengelola interaksi dengan pelanggan saat ini dan potensial, dengan tujuan meningkatkan hubungan dan mempertahankan pelanggan. Contoh: Salesforce, Zoho CRM, atau Microsoft Dynamics CRM.

2.1.4 Presensi

Presensi adalah kehadiran seseorang pada lokasi tertentu yang menandakan bahwa kehadiran individu tersebut diantisipasi atau diharapkan oleh pihak-pihak tertentu untuk tujuan tertentu (Prasetyo, E., & Juman, K. K., 2022). Presensi adalah istilah yang merujuk pada kehadiran seseorang pada suatu lokasi atau acara tertentu. Ini menandakan bahwa individu tersebut berada di tempat yang dimaksud, dan kehadirannya diantisipasi atau diharapkan oleh pihak-pihak tertentu untuk tujuan tertentu. Presensi sering kali merupakan bagian penting dari berbagai konteks, termasuk lingkungan kerja, acara sosial, acara formal, dan sebagainya.

2.1.5 QR *Code*

Definisi dari QR *Code* ialah suatu bentuk kode berupa *pixel* yang tersusun dalam gambar serta dapat secara efisien diuraikan menggunakan peralatan pemindai. Kode QR mengandung data secara vertikal dan horizontal, walaupun umumnya hanya satu set informasi yang disimpan secara vertikal. Ini merupakan representasi gambar dalam bentuk *pixel* dua dimensi dan mampu menyimpan berbagai informasi di dalam gambar *pixel* tersebut. QR *Code* merupakan perkembangan dari teknologi identifikasi, yaitu *barcode*. Dikarenakan bersifat dua dimensi, QR Code melakukan penyimpanan informasi secara vertikal dan horizontal dalam bidang datar (Nasution, M. I. P., & Triase, T., 2021).

2.1.6 *Software Development Life Cycle* (SDLC)

Definisi SDLC ialah suatu metode yang diterapkan dengan tujuan pengembangan, pemeliharaan, dan penggunaan sistem informasi. Metode ini melibatkan susunan tahapan dan digunakan dalam proses pengembangan sistem. SDLC dapat dikatakan kerangka kerja yang tertua dalam metode pengembangan untuk rancang bangun sistem informasi. Konsep SDLC ialah untuk rancang bangun atau menciptakan sistem informasi secara terorganisir, dengan memperhatikan tahapan siklus hidup dari konsepsi hingga implementasi (Rizki, W., Rayuwati, R., & Gemasih, H., 2022).

SDLC memiliki tahapan-tahapan atau serangkaian langkah atau fase yang diikuti dalam pengembangan sistem informasi. Meskipun berbagai model SDLC mungkin memiliki varian dalam nama dan jumlah tahapan, umumnya SDLC terdiri dari tahapan-tahapan berikut (Jamal, S., & Kusnadi, K., 2022):

1. **Perencanaan (*Planning*)**

   Tahap di mana rencana proyek disusun. Ini termasuk mengidentifikasi tujuan proyek, menyusun jadwal,

menentukan anggaran, dan menetapkan sumber daya yang dibutuhkan.

2. **Analisis (*Analysis*)**

   Tahap di mana kebutuhan bisnis dan fungsional sistem diidentifikasi dan didokumentasikan secara rinci. Ini melibatkan wawancara pemangku kepentingan, pemetaan proses bisnis, dan menghasilkan dokumen kebutuhan.

3. **Desain (*Design*)**

   Tahap di mana desain teknis sistem direncanakan. Ini mencakup merancang arsitektur sistem, mengembangkan spesifikasi teknis, serta merancang antarmuka pengguna dan basis data.

4. **Implementasi (*Implementation*)**

   Tahap di mana sistem aktual dikembangkan berdasarkan desain yang telah disetujui. Ini melibatkan menulis kode, mengintegrasikan komponen, dan melakukan pengujian unit.

5. **Pengujian (*Testing*)**

   Tahap di mana sistem diuji untuk memastikan bahwa ia memenuhi spesifikasi dan bekerja sesuai yang diharapkan. Ini melibatkan pengujian fungsionalitas, kinerja, keamanan, dan keandalan sistem.

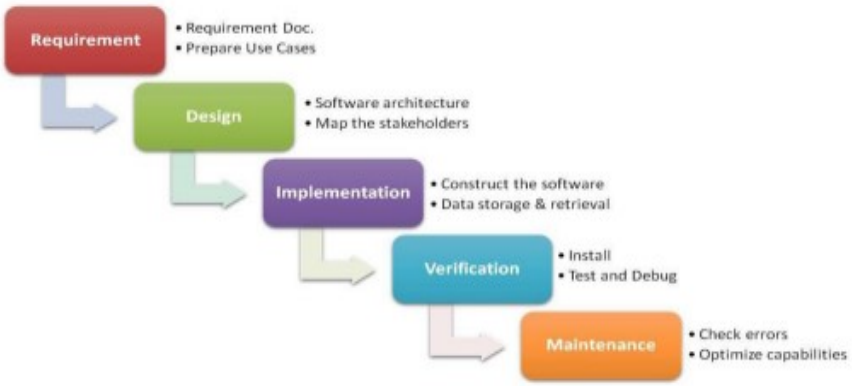
6. **Pemeliharaan (*Maintenance*)**

   Tahap yang terjadi setelah penerapan di mana sistem dipelihara dan diperbaiki sesuai kebutuhan. Ini berisi penyelesaian *error* atau *bug*, perbaikan fungsionalitas, dan peningkatan fleksibilitas dengan perubahan kebutuhan bisnis.

Tahapan-tahapan ini sering kali diikuti secara berurutan, tetapi dalam praktiknya, beberapa tahapan mungkin tumpang tindih atau berulang tergantung pada metode pengembangan yang digunakan dan kompleksitas proyek.

2.1.7 *Waterfall*

Definisi *Waterfall* ialah metode dalam pengembangan aplikasi dengan mengikuti urutan langkah-langkah yang berurutan dan sekuensial. Pada metode ini, setiap tahap, seperti menalaah kebutuhan, arsitektur aplikasi, implementasi *code*, *testing*, dan pemeliharaan, dilaksanakan dengan berturut-turut. Artinya, tahapan berikutnya dimulai setelah tahapan sebelumnya selesai sepenuhnya. Analoginya seperti air terjun, di mana air mengalir ke bawah secara berurutan dan tidak ada mundur ke tahapan sebelumnya. Metode *Waterfall* menekankan perencanaan yang cermat di awal proyek dan setiap tahap memiliki *deliverables* (hasil kerja) yang jelas sebelum memulai tahap berikutnya (Badrul, M., 2021).



Gambar 2. 1 Metode *Waterfall*

(Sumber: Heriyanti, F., & Ishak, A., 2020)

Metode *Waterfall* cocok untuk proyek-proyek yang memiliki persyaratan yang stabil dan terdefinisi dengan jelas di awal proyek, serta memiliki sedikit perubahan yang diharapkan selama siklus pengembangan. Berikut adalah situasi-situasi di mana Metode *Waterfall* cocok digunakan (Pricillia, T., 2021):

1. Proyek dengan persyaratan stabil dan jelas. Ketika persyaratan proyek telah ditetapkan dengan jelas dan tidak banyak berubah selama siklus pengembangan, Metode *Waterfall* dapat menjadi

pilihan yang baik karena fokus pada perencanaan awal dan eksekusi bertahap.

2. Proyek dengan batas waktu yang tidak singkat. Metode *Waterfall* memungkinkan perencanaan yang cermat dari awal, yang dapat membantu dalam menetapkan jadwal yang ketat dan memastikan pengiriman tepat waktu.
3. Proyek dengan kebutuhan risiko yang rendah. Metode *Waterfall* cocok untuk proyek-proyek di mana risiko perubahan atau kegagalan sistem rendah, karena mengharuskan semua persyaratan dan desain ditetapkan di awal.

Meskipun cocok untuk situasi-situasi tersebut, penting untuk diingat bahwa setiap pendekatan pengembangan memiliki kelebihan dan kelemahan tersendiri. Sebab itu, perlu diteliti lebih dalam faktor-faktor pengembangan dan memilih metodologi yang paling sesuai dengan karakteristik dan kebutuhan proyek tertentu.

2.1.8 Analisis dan Perancangan

Istilah ini mengacu pada tahap awal pembangunan sistem secara keseluruhan. Untuk memahami sistem yang akan dibangun dan dirancang, proses analisis dan perancangan sangat penting. Proses ini bertujuan untuk menemukan dan membuat solusi sistem baru yang berguna untuk menangani masalah yang dihadapi perusahaan (Indyah Hartami, 2020).

Salah satu pendekatan dalam analisis dan perancangan sistem adalah OOAD (analisis dan desain dengan focus *object*), yang bersifat kontemporer dalam analisis dan desain aplikasi dengan berfokus pada penggunaan objek dan kelas dalam sistem. Metode ini memungkinkan identifikasi objek sistem serta hubungan antara objek dan perilakunya dengan implementasi diagram UML. OOAD berorientasi pada objek, menggunakan konsep seperti pewarisan, enkapsulasi, dan polimorfisme,

menghasilkan desain yang modular dan fleksibel (Apandi, A., 2023).

2.1.9 *Unified Modelling Language* (UML)

UML berdefinisi bahasa pemodelan dan dikembangkan oleh Grup Manajemen Objek (OMG). Struktur kelas, diagram use case, dan diagram aktivitas adalah beberapa aspek sistem yang dapat digambarkan dan diwakili dalam UML. Pengembang perangkat lunak menggunakan bahasa pemodelan visual ini untuk memberikan dukungan dalam proses desain, dokumentasi, dan analisis aplikasi. UML adalah bahasa pemodelan visual yang digunakan untuk menunjukkan konsep, struktur, dan perilaku sistem yang kompleks. UML digunakan sebagai standar industri untuk menggambarkan jenis-jenis sistem, termasuk perangkat lunak serta aplikasi alur bisnis. Dengan fleksibilitas dan kesederhanaan UML, ia dapat disesuaikan dengan kebutuhan pengguna dan digunakan dalam berbagai konteks (Fitrani, L. D., & Puspitaningrum, A. C., 2023).

Selain itu, UML dapat diterapkan oleh banyak pihak dengan keterlibatan pada perancangan sistem, contohnya analis, manajer proyek, *programmer*, dan *user*. Sebagai contoh, UML menyediakan berbagai jenis diagram:

1. ***Use Case Diagram***

   Definisinya ialah diagram dalam UML dengan fungsi memodelkan atau visualisasi interaksi sistem-sistem dan aktor-aktor di luar sistem tetapi terlibat (Hartiwi, Y., & Nurhayati, N., 2022). Diagram ini berupa pemodelan atau visualisasi dari cara sistem bekerja dari sudut pandang pengguna atau aktor. Ini juga membantu memahami bagaimana sistem berinteraksi dengan lingkungan eksternal dan kebutuhan pengguna. **Tabel 2.1** menunjukkan elemen atau ikonnya.

Tabel 2. 1 Elemen *Use Case Diagram*

Simbol	Nama	Keterangan
	*Actor*	Menentukan kumpulan *role* yang *user* gunakan saat beraktivitas di dalam *use case*.
	*Dependency*	Relasi yang jika ada perubahan pada aktivitas atau entitas independen akan memberikan pengaruh kepada entitas yang terikat padanya, yang bersifat tidak independen.
	*Generalization*	Relasi turunan (*descendent*) mewarisi sifat dan bentuk dari atasannya (*ancestor*).
	*Include*	Menspesifikasikan bahwa *use case* berasal dari entitas yang bersifat eksplisit.
	*Extend*	Menentukan bahwa target *use case* mengembangkan sifat dari asalnya.
	Association	Yang menghubungkan satu entitas dengan entitas lainnya.
	*System*	Menggambarkan entitas yang memperlihatkan sistem.
	*Use Case*	Penjelasan dari aktivitas-aktivitas yang diperlihatkan sistem dan dapat dilihat oleh *actor*.
	*Collaboration*	Korelasi peraturan dan entitas-entitas terhubung untuk aktivitas yang lebih luas dari berapa banyak entitas dan sinerginya.
	*Notes*	Berupa dokumen fisik nyata di dalam sistem.

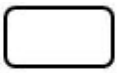
(Sumber: Jacobson, L., & Booch, J. R. G., 2021)

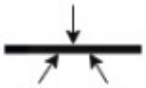
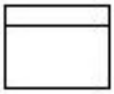
Melalui *Use Case Diagram*, pihak-pihak terkait dapat memahami secara visual interaksi antara pengguna dan sistem, mengidentifikasi fungsionalitas utama yang harus disediakan oleh sistem, serta memperjelas dan memvalidasi kebutuhan pengguna. *Use Case Diagram* juga berfungsi menjadi media percakapan yang efektif antara *programmer*, *analyst*, dan pihak bersangkutan yang lain dalam proyek rancang bangun aplikasi atau *software* .

2. ***Activity Diagram***

Salah satu diagram UML dengan fungsi memodelkan aliran kerja dan kegiatan yang berada pada suatu proses adalah aktivitas diagram. Diagram ini membantu dalam memodelkan aktivitas, keputusan, dan aliran kontrol antara aktivitas tersebut (Hartiwi, Y., & Nurhayati, N., 2022). **Tabel 2.2** menunjukkan elemen dan ikonnya.

Tabel 2. 2 Elemen *Activity Diagram*

Simbol	Nama	Keterangan
	*Start*	Kondisi permulaan dari sebuah kegiatan di dalam sistem yang digambarkan dengan *activity diagram*.
	*Activity*	Kegiatan yang terjadi di dalam sistem dan dimulai menggunakan kata kerja.
	*Decision*	Situasi di mana terdapat dua atau lebih aktivitas.
	Join	Kondisi saat dua atau lebih aktivitas dipersatukan menjadi satu.

	*End*	Kondisi akhir sistem, dan setiap sistem diwajibkan mempunyai satu *end*.
	*Fork*	Menunjukkan kegiatan yang dilakukan secara paralel atau bersamaan.
	*Swimlane*	Membagi aktor-aktor dengan kepentingan terkait aktivitas-aktivitas di dalam sistem.

(Sumber: Jacobson, L., & Booch, J. R. G., 2021)

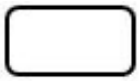
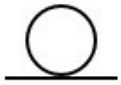
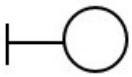
Activity Diagram membantu dalam memodelkan aliran kerja secara visual dan memahami bagaimana aktivitas-aktivitas terkait dan memiliki interaksi satu dengan lainnya dalam suatu proses. Diagram ini juga dapat digunakan untuk mengidentifikasi masalah dalam aliran kerja, mengoptimalkan proses, dan memvalidasi desain sistem sebelum implementasi.

3. ***Sequence Diagram***

Definisi *Sequence Diagram* ialah diagram dalam *Unified Modeling Language* (UML) dengan fungsi memodelkan keterkaitan antara objek yang berada di dalam sistem dalam tahapan waktu tertentu (Syarif, M., & Nugraha, W., 2020). *Sequence Diagram* berisi pesan yang diantarkan setiap objek satu sama lain di dalam urutan waktu tertentu. **Tabel 2.3** menunjukkan elemen dan ikonnya.

Tabel 2. 3 Elemen *Sequence Diagram*

Simbol	Nama	Keterangan
	*Actor*	Menentukan *user* yang sedang melakukan kegiatan di dalam sistem.

	*Object*	Entitas yang saling memberikan interaksi dan menyampaikan pesan atau kelas yang berisi nama objek dan dimulai dengan titik dua.
	*Entitiy Class*	Menunjukkan relasi yang dilakukan oleh entitas.
	*Boundary Class*	Menunjukkan visualisasi dari *form.*
	*Control Class*	Menghubungkan *boundary* satu sama lain dan *model*.
	*Focus of Control & Life Line*	Menunjukkan lokasi *start* serta *end* suatu objek yang berinteraksi dengan pesan.
	Message	Menggambarkan komunikasi atau pengiriman pesan antar objek.

(Sumber: Jacobson, L., & Booch, J. R. G., 2021)

Sequence Diagram membantu dalam memodelkan keterkaitan objek-objek satu sama lain yang berada di sistem berdasarkan suatu urutan waktu. Diagram ini memungkinkan pengembang untuk memahami aliran logika dalam sistem, mengidentifikasi proses asinkron atau sinkron, dan memvalidasi desain sistem sebelum implementasi. *Sequence Diagram* juga mendukung dan memberikan bantuan dalam percakapan antara *programmer*, analis, dan pihak bersangkutan yang lain dalam proyek rancang bangun aplikasi atau *software*.

4. ***Class Diagram***

Definisinya ialah diagram UML dengan fungsi memodelkan struktur statis dari sistem atau perangkat lunak yang saat ini dibangun (Hartiwi, Y., & Nurhayati, N., 2022). Diagram ini

memodelkan *class*, *attribute*, metode, serta relasi setiap kelas tersebut. **Tabel 2.4** menunjukkan elemen atau ikonnya.

Tabel 2. 4 Simbol *Class Diagram*

Simbol	Nama	Keterangan
	Generalization	Relasi turunan (*descendent*) mewarisi sifat dan bentuk dari atasannya (*ancestor*).
	*n-ary Association*	Simbol yang menggambarkan gabungan dari 3 objek atau lebih.
	*Class*	Kumpulan beberapa objek dengan *attribute* dan *operation* yang serupa.
	*Collaboration*	Keterkaitan setiap peraturan serta elemen-elemen untuk aktivitas lebih luas dari banyaknya elemen dan sinerginya.
	*Realization*	Aktivitas secara nyata dan dijalankan oleh objek.
	*Dependency*	Relasi yang jika ada perubahan pada aktivitas atau entitas independen akan memberikan pengaruh kepada entitas yang terikat padanya, yang bersifat tidak independen.
	Association	Yang menghubungkan satu entitas dengan entitas lainnya.

(Sumber: Jacobson, L., & Booch, J. R. G., 2021)

Class Diagram membantu dalam memodelkan struktur statis dari sistem, termasuk kelas-kelas yang ada, *attribute* dan

metode dari kelas-kelas tersebut, dan juga relasi dari setiap kelas. *Class Diagram* memungkinkan pengembang memahami rancangan logika dari aplikasi atau sistem perangkat lunak, mengidentifikasi relasi setiap *class*, dan memvalidasi desain sistem sebelum implementasi. *Class Diagram* memberikan bantuan dalam percakapan antara *programmer*, analis, dan pihak bersangkutan yang lain dalam proyek rancang bangun aplikasi atau *software*.

2.1.10 *Website*

Definisi *website* atau situs web ialah halaman-halaman dengan isi terdiri dari informasi yang memiliki berbagai macam tipe, seperti *text*, *image*, *video*, *audio*, atau *animation*, dan diakses dengan menggunakan koneksi atau jaringan internet. Dengan kata lain, situs web ialah kumpulan halaman dengan informasi di dalamnya dan diakses menggunakan *browser* untuk menyampaikan manfaat bagi yang mengaksesnya. Berdasarkan sifatnya, terdapat dua macam situs web (Muhammad Ibnu Sa`ad, 2020):

1. **Web Statis**

   Definisinya ialah jenis situs dengan isi di dalamnya yang tetap dan tidak berubah-ubah. Artinya, konten di dalam halaman-halaman web tidak dapat dimodifikasi secara langsung dan sulit untuk dimodifikasi. Hal tersebut dikarenakan penggunaan teknologi untuk membangun web tersebut tidak memiliki fleksibilitas sehingga sulit untuk mengubah isi dan data di dalamnya. Teknologi untuk membangun web statis biasanya HTML dan *Cascading Style Sheet* (CSS) yang merupakan *client-side scripting*. Untuk mengubah isi atau data pada halaman web statis, caranya hanya dapat dilakukan dengan melakukan perubahan secara langsung pada isinya di dalam *file* mentah atau melakukan perubahan pada *script*.

2. Web Dinamis

Definisi dari web dinamis ialah jenis situs web dengan konten yang dapat diubah setiap waktu dan harus diubah dalam file atau dengan istilah bongkar *script* pada halaman admin. Sebuah website yang menampilkan tampilan dinamis melalui halaman admin, tetapi tidak menggunakan *flash,* belum pasti termasuk web dinamis. Biasanya, web dinamis dibangun menggunakan *database* untuk menyimpan data.

Aplikasi web merupakan aplikasi yang arsitekturnya memiliki bentuk *client-server.* Dalam bentuk ini, program *client* terkoneksi dengan *server* untuk menerima informasi yang diperlukan untuk menyelesaikan tugas yang ditetapkan oleh *user*. *Sripting Server-side* dan *Scripting Client-side* sering digunakan pada rancang bangun aplikasi web. *Sripting Server-side* berdefinisi bahasa program web yang diolahkan komputer *server* atau penyedia. Pada saat pengguna mengakses web, data yang disimpan di dalam *database* akan dipindahkan oleh *server* dan tampil di web. Kode pada *server-side* tidak dapat diakses atau terlihat oleh pengguna karena kode tersebut dikunci. Contoh *script server-side* adalah PHP, ASP, dan Java. *Script client-side* yaitu bahasa pemrograman web dengan diolah oleh klien atau pengguna. Komputer pengguna melakukan *download* terhadap *script client-side* setiap kali mereka mengunjungi web. HTML, CSS, JavaScript, dan XML merupakan contoh *script client-side* (Geovanny, A., 2022).

2.1.11 *Database*

Dikenal basis data, berdefinisi data atau informasi tersimpan secara sistematis dan teratur dalam sistem komputer. Penggunaan *database* bertujuan untuk mengarsip, memproses, dan memanggil data dengan cepat dan efisien. *Database* berisi jenis-jenis data terkait kegiatan atau aktivitas bisnis di dalam suatu organisasi. *Database* berisi data yang memiliki kolom dan *attribute*

untuk menunjukkan karakteristik dari data yang disimpan (Elmasri, R. & Navathe, S. B., 2020).

Database terdiri dari data-data yang memiliki relasi satu sama lain dan terorganisir dengan baik. Data-data tersebut terdapat pada isi *database* berbentuk tabel, arsip, atau *file* saling berhubungan dan tersimpan secara elektronis pada suatu media penyimpanan. Seringnya, aplikasi yang berkaitan dengan kelola data pasti menggunakan *database* untuk media penyimpanan datanya. Ini adalah komponen penting untuk mendukung program yang terkomputerisasi (Kadarsih, 2022).

Perangkat lunak *Database Management System* (DBMS) diperlukan untuk melakukan *import* atau *export* data dari *database*. DBMS mengandung manfaat sebagai media penghubung pengguna dan *database*. Karenanya, diperlukan instruksi serta sintaks untuk interaksi pengguna dan database (Tri Rachmadi, 2020). *Data Base Management System* (DBMS) memiliki dua jenis perintah DDL dan DML (Setyawati, E., Wijoyo, H., & Soeharmoko, N., 2020).

Basis data *relational* mengarsip data berbentuk tabel dan saling terkoneksi satu sama lain menggunakan *key* yang tedapat pada kolom di setiap tabel. MySQL, Oracle, dan SQL Server merupakan beberapa contoh *database* jenis ini. Dalam pembuatan *relational database*, normalisasi memiliki fungsi menghilangkan kemungkinan anomali yang terjadi saat mengelola data pada *database* (Setyawati, E., Wijoyo, H., & Soeharmo, M.).

2.1.12 *Black Box Testing*

Definisinya ialah metode untuk menguji aplikasi yang berfokus pada fungsi aplikasi dan tidak memperhatikan kinerja atau struktur internalnya. Teknik ini dapat diterapkan di berbagai tingkatan pengujian, seperti penerimaan, sistem, unit, dan integrasi (Syaiful, R. R. & Saifudin, A., 2022). Untuk melakukan pengujian *black box*, *input* tertentu diberikan ke sistem atau aplikasi, dan

kemudian dilakukan analisis pada *output* yang dihasilkan. *Black box testing* memiliki fungsi memastikan bahwa aplikasi berfungsi dengan tepat seperti yang diharapkan, tetapi tidak memperhatikan bagaimana aplikasi dibangun.

Metode *black box testing* dapat mencakup berbagai teknik pengujian, seperti menguji fungsi aplikasi, non-fungsional aplikasi, uji kasus, keamanan, dan lain-lain. Keuntungan dari *black box testing* yaitu bahwa *testing* dapat dilakukan dengan tidak memerlukan pemahaman detail mengenai kode program atau struktur internal aplikasi sehingga orang yang tidak memiliki keterlibatan dalam rancang bangun aplikasi dapat melakukan *testing*. Namun, kelemahan dari *black box testing* adalah bahwa pengujian mungkin tidak dapat mengungkapkan *bug* atau masalah yang terkait dengan struktur internal aplikasi. Oleh karena itu, *black box testing* sering dikombinasikan dengan *white box testing*, di mana pengujian dilakukan dengan memperhatikan struktur internal aplikasi (Giansyah, Q. A., & Hamzah, M., 2023).

2.2 Tinjauan Studi

Penulis melakukan tinjauan studi dengan tujuan untuk mendukung penyusunan penelitian ini berdasarkan penelitian-penelitian terdahulu sebagai referensi. Tinjauan studi dilakukan dengan mengidentifikasi lima penelitian sebagai referensi utama yang dipilih karena merupakan penelitian terkini dan relevan terhadap rumusan masalah yang diajukan. Berikut adalah pembahasan terkait lima penelitian tersebut:

1. Penelitian dengan judul "**Sistem Informasi *Payroll* Pegawai dengan Absensi QR *Code***" yang ditulis oleh Maharani, R. B. N., Nasution, M. I. P., & Triase, T pada tahun 2021 dalam *Jurnal Informatika Dan Teknologi Pendidikan*. Fokus penelitian menciptakan aplikasi penggajian dengan implementasi kode QR guna mengidentifikasi absensi pegawai. PT. Anugrah Azzahra Utama sering mengalami kesalahan dalam memvalidasi data upah pegawai. Akibatnya, perlu dilakukan penghitungan ulang gaji untuk memastikan bahwa

perusahaan benar-benar membayar karyawannya. Sistem informasi *payroll* karyawan dapat mendukung perusahaan menghitung gaji. Analisis kebutuhan, perancangan sistem, pengkodean, pengujian, dan pemeliharaan adalah langkah-langkah dalam proses desain sistem *waterfall*. Dengan kamera eksternal sebagai pemindai dan terhubung ke *database*, sistem ini menggunakan kode QR untuk memastikan absensi karyawan. Ini memberi kemudahan bagi pengguna serta dapat meningkatkan efisiensi dan efektivitas. Aplikasi dikembangkan dengan penggunaan PHP dan basis data. Metode untuk menguji aplikasi menggunakan *black-box*. Aplikasi ini memberikan kemudahan dalam absensi karyawan menggunakan kode QR menggunakan kamera eksternal sebagai pemindai sehingga hasilnya adalah upah pegawai yang bejumlah sesuai dan valid. Sehingga, tidak perlu menghitung ulang gaji, serta data tersimpan dengan baik untuk admin, karyawan, dan direktur.

2. Penelitian dengan judul "**APLIKASI ABSENSI GURU PADA SEKOLAH BERBASIS ANDROID DENGAN KEAMANAN QR CODE (STUDI KASUS : SMP NEGERI 4 BATANG GANSAL)**" yang ditulis oleh Rahmalisa, U., Irawan, Y., & Wahyuni, R pada tahun 2020 dalam *Riau Journal of Computer Science*. Aplikasi absens guru dengan implementasi dan penggunaan QR basis Android adalah subjek penelitian ini. Kehadiran adalah elemen krusial di setiap lembaga pendidikan. Namun, absensi di SMP Negeri 4 Batang Gansal saat ini masih menggunakan tanda tangan pada kertas dan mencatat waktu masuk pada jam mengajar, yang dapat menimbulkan kecurangan karena tingkat disiplin yang sulit dikendalikan. Karena pencatatan waktu kehadiran dan waktu kerja masih dilakukan dengan Microsoft Excel, administrasi juga menghadapi kesulitan dalam menulis laporan serta memberikan informasi kehadiran kepada guru-guru. Ini karena Microsoft Excel rentan terhadap kesalahan pengetikan dan penghapusan data. Tujuan aplikasi untuk mengambil peran utama memudahkan proses kehadiran guru di sekolah.

3. Penelitian dengan judul "**Aplikasi Absensi Pegawai pada Dinas Komunikasi dan Informatika Kabupaten Deli Serdang dengan QR *Code* Menggunakan Algoritma Bcrypt**" yang ditulis oleh Akbar, M. D., & Antoni, A pada tahun 2022 dalam *sudo Jurnal Teknik Informatika*. Penelitian tersebut berisi tentang perancangan aplikasi absensi pegawai dengan QR *Code* menggunakan algoritma Bcrypt. Di kantor Dinas Kominfo Kabupaten Deli Serdang, pengaturan absensi dilakukan menggunakan cara konvensional dengan menandatangani buku untuk absen. Selain itu, rekapitulasi data kehadiran dilakukan juga menggunakan cara konvensional dengan menuliskan dan memperhatikan jumlah pegawai yang hadir, sakit, atau tidak memiliki keterangan. Untuk meminimalisir kesalahan dalam penghitungan jumlah kehadiran, dibutuhkan ketelitian tinggi. Selain itu, kecurangan seperti penggunaan *link*, nomor *handphone*, SMS, *VCard*, atau lainnya dapat terjadi pada sistem absensi konvensional ini. Teknologi QR *Code* berbasis web digunakan dalam penelitian untuk menyelesaikan masalah tersebut. Kode QR memungkinkan penyimpanan dan penyebaran data dengan cepat. Metode Bcrypt digunakan untuk enkripsi *password*, yang dapat meningkatkan keamanan karena orang lain sulit membacanya. Aplikasi ini ialah aplikasi dengan basis web dinamis dan data disimpan menggunakan *database*.

4. Penelitian dengan judul "**Perancangan Sistem Informasi Absensi Siswa Menggunakan QR *Code* Berbasis Web**" yang ditulis oleh Ali, G., Rohman, W. N., & Novalia, M pada tahun 2023 dalam *KLIK: Kajian Ilmiah Informatika dan Komputer*. Studi ini membahas pengembangan sistem absensi dengan mengimplementasikan teknologi QR dengan basis web. Dalam pendidikan, termasuk di sekolah, sistem kehadiran biasanya dilakukan dengan cara konvensional, banyak sekali kesalahan saat merekapitulasi data kehadiran. Tujuannya untuk menciptakan sistem kehadiran siswa dengan mengimplementasikan teknologi QR dengan basis web untuk memudahkan pengelolaan data kehadiran. Penelitian berjenis R&D dengan menggunakan *waterfall*

sebagai metodenya, serta mencakup berbagai langkah seperti tahap analisis, tahap perancangan, pengimplementasian, *testing*, dan *maintenance. Black box testing* digunakan untuk menguji sistem, lalu ahli sistem melakukan validasi. Hasil validasi dinilai sebagai "sangat baik" melalui analisis aspek kegunaan melalui kuesioner, dan hasil nilai konversi juga dinilai sebagai "sangat baik". Hal tersebut menunjukkan bahwa sistem kehadiran yang mengimplementasikan kode QR dengan basis web yang dibangun dapat mempermudah manajemen data dan menyederhanakan proses absensi.

5. Penelitian dengan judul "**PERANCANGAN SISTEM INFORMASI ABSENSI KARYAWAN DENGAN QR *CODE* BERBASIS WEB PADA PT. SALIM IVOMAS PRATAMA Tbk.**" yang ditulis oleh Girsang N. D. pada tahun 2023 dalam *Circle Archive*. Penelitian ini membahas pengembangan sistem absensi karyawan yang mengimplementasikan kode QR dengan basis web. PT Salim Ivomas Pratama Tbk berfokus di bidang agribisnis, terutama kelapa sawit, dengan operasional yang sudah terpadu. Meskipun sistem *finger scan* saat ini digunakan untuk mengidentifikasi absensi karyawan, kesalahan identifikasi masih sering terjadi. Selain itu, staf masih menggunakan metode konvensional saat mereka ingin mengambil laporan absensi, karena *file* yang mereka peroleh berupa *spreadsheet*, mereka harus mengolah data lagi. Oleh karena itu, sistem absensi karyawan yang dapat diakses melalui web dengan implementasi kode QR dikembangkan. Setiap kartu identitas karyawan dilengkapi dengan kode QR untuk mempermudah proses absensi, dan *database* digunakan untuk menyimpan data pegawai. Lalu, laporan absensi dapat diakses oleh staf. Sistem ini juga mencakup laporan kehadiran untuk setiap hari, bulan, dan tahun.

Berdasarkan tinjauan-tinjauan studi tersebut, dapat disimpulkan beberapa perbedaan dari aplikasi presensi kelas mahasiswa yang dibangun dengan aplikasi di dalam tinjauan-tinjauan tersebut:

- Pada aplikasi yang dibangun, terdapat fitur Lihat Mahasiswa oleh mahasiswa dan dosen untuk melihat list mahasiswa di dalam kelas tertentu.
- Pada aplikasi yang dibangun, dosen dapat membuka dan menutup presensi kelas tertentu sehingga QR *Code* tidak dapat dipindai oleh mahasiswa apabila dosen belum membuka presensi kelas.
- Pada aplikasi yang dibangun, terdapat fitur Kelola Mahasiswa oleh dosen sehingga dosen tetap bisa melakukan presensi secara manual apabila terjadi kendala dalam proses presensi menggunakan QR *Code* oleh mahasiswa.
- Pada aplikasi yang dibangun, mahasiswa dapat melihat jadwal kelas hari ini secara langsung di *Home* sehingga mahasiswa tidak perlu memilih terlebih dahulu kelas yang ingin dilakukan presensi.
- Pada aplikasi yang dibangun, dosen dapat memantau langsung secara *real-time* presensi mahasiswa.