

BAB II TINJAUAN PUSTAKA

2.1 Teori Umum

Teori yang mendukung penelitian tentang Pembuatan Dashboard Penjualan Berbasis Mobile Menggunakan Metode Prototyping Pada PT. XYZ dibahas dalam bab ini.

2.1.1 Sistem Informasi Manajemen

Sistem Informasi Manajemen adalah studi tentang bagaimana manusia, teknologi, dan organisasi bekerja sama untuk memaksimalkan investasi dalam personel, peralatan, dan proses bisnis.

Menurut Scott (2019), sistem informasi manajemen terdiri dari sekumpulan subsistem informasi yang sangat terintegrasi dan terorganisir secara logis. Mengubah data menjadi informasi dengan berbagai metode untuk meningkatkan produktivitas, sesuai dengan gaya dan karakteristik manajer serta sesuai dengan standar kualitas yang ditetapkan.

Sistem informasi terdiri dari individu, perangkat lunak, perangkat keras, jaringan komunikasi, sumber daya data, serta aturan dan prosedur. Sistem ini digunakan untuk menyimpan, mengambil, mengubah, dan mendistribusikan informasi dalam sebuah organisasi (O'Brien & Marakas, 2007).

2.1.2 Dashboard

Dashboard penjualan menggambarkan visual data yang digunakan untuk memonitor dan menganalisis data penjualan suatu bisnis dalam periode waktu tertentu. Data yang digambarkan dalam representasi visual ini harus dibuat dengan cermat untuk memastikan aksesibilitas dan pemahaman yang optimal bagi pengguna akhir (Few, 2006).

Dashboard penjualan harus memberikan gambaran yang cepat dan mudah dipahami atas performa penjualan suatu bisnis melalui berbagai metrik, seperti jumlah penjualan, produk terlaris, pelanggan teratas, daerah penjualan, dan sebagainya. *Dashboard* penjualan biasanya berisi grafik, tabel, dan diagram yang interaktif, sehingga pengguna dapat memperoleh informasi secara *real-time* dan melakukan analisis lebih mendalam terhadap data penjualan. Dengan menggunakan *dashboard* penjualan, bisnis dapat mengidentifikasi tren dan pola penjualan, mengambil keputusan yang lebih tepat, serta meningkatkan kinerja penjualan secara keseluruhan.

Menurut Rasmussen et al. (2009), *dashboard* dibagi menjadi tiga kategori utama berdasarkan berbagai pendekatan yang digunakan untuk membedakannya, yaitu:

1. *Dashboard* Strategis (*strategic dashboard*)

Dashboard strategis memiliki tujuan untuk memberikan panduan yang relevan untuk mencapai tujuan strategis, memfasilitasi pengambilan keputusan, mengantisipasi peluang, dan memberikan informasi.

2. *Dashboard* Taktis (*tactical dashboard*)

Fokus pada proses analisis untuk mengidentifikasi dan menentukan faktor-faktor yang menyebabkan suatu situasi atau peristiwa tertentu terjadi.

3. *Dashboard* Operasional (*operational dashboard*)

Sistem *dashboard* yang memungkinkan pemantauan operasi perusahaan, proses bisnis, dan elemen rumit lainnya. *Dashboard* ini memberikan pembaruan *real-time* tentang status proses bisnis organisasi, menawarkan informasi harian, mingguan, atau bergambar.

2.1.3 Visualisasi Data

Visualisasi data merupakan sebagai proses pengubahan data mentah menjadi representasi visual yang membantu dalam memahami informasi secara lebih mudah dan intuitif (Kirk, 2016). Evergreen (2016),

menjelaskan visualisasi data adalah praktik memilih dan menggunakan jenis grafik yang tepat untuk berbagai tipe data agar dapat menyampaikan informasi secara efektif dan mudah dipahami oleh penggunanya.

Mengubah data mentah menjadi representasi visual yang mudah dipahami dan dikomunikasikan disebut visualisasi data (Wilke, 2019). Visualisasi data merupakan seni dan ilmu dalam menampilkan data sehingga dapat dilihat dan dipahami dengan lebih efektif (Telea, 2014). Visualisasi data adalah proses menggambarkan data dalam bentuk grafis yang memungkinkan untuk mengeksplorasi, memahami, dan menyampaikan informasi dengan jelas dan mudah dipahami (Camm et al., 2017).

2.1.4 Website

Situs *website* adalah berbagai macam halaman *web* yang terhubung satu sama lain, dapat diakses melalui internet, dan ditampilkan di bawah domain tertentu. Halaman web berisi teks, gambar, animasi, suara atau audio, film atau video, dan interaktivitas seperti formulir dan tombol. *Website* adalah salah satu cara terbaik untuk membangun kehadiran digital untuk bisnis dan organisasi. Seiring dengan meningkatnya penggunaan internet dan perangkat *mobile*, *website* dapat menjadi alat yang sangat penting untuk mencapai audiens secara global dan membangun merek.

2.1.5 Data Warehouse

Sebuah tempat penyimpanan yang mengumpulkan data dari berbagai sumber dan menyediakan data tersebut untuk pengguna akhir dalam format yang mudah dimengerti dan digunakan disebut data *warehouse*. Tujuannya adalah untuk mendukung proses pengambilan keputusan dalam konteks bisnis tertentu.

Data *warehouse* juga berfungsi sebagai sistem pendukung keputusan, dengan menyimpan, mengorganisasi, dan menganalisis data dari berbagai sumber untuk membantu para pengambil keputusan. Namun, data *warehouse* hanya menyajikan informasi untuk memungkinkan pengambilan keputusan yang tepat. Terdapat empat karakteristik utama dari data *warehouse*: berorientasi pada subjek (*subject oriented*), terintegrasi (*integrated*), berjangka waktu (*time variant*), serta data tidak mudah berubah (*non volatile*).

Data *warehouse* merupakan kumpulan data yang komprehensif dan tidak berubah yang terintegrasi, berfokus pada subjek, dan bergantung pada waktu. Ini digunakan oleh manajemen untuk memfasilitasi proses pengambilan keputusan (Inmon, 2005). Menurut Kimball dan Caserta (2004), data *warehouse* adalah sistem yang mengumpulkan dan mengelola beberapa data dari berbagai sumber yang digunakan untuk analisis dan pelaporan bisnis.

Selain *database* relasional, lingkungan data *warehouse* juga mencakup *Online Analytical Process* (OLAP), alat bantu analisis klien, dan program lain yang mengawasi dan menyediakan akses ke data untuk pengguna bisnis.

Extract, Transform, Load (ETL) merupakan fondasi dari sistem data *warehouse*. Desain ETL yang efektif menjamin ekstraksi data dari berbagai sumber yang berbeda sesuai dengan kriteria kualitas dan konsistensi data, sehingga memungkinkan integrasi dan transformasi data yang berbeda ke dalam format yang sesuai untuk ditampilkan.

Proses ETL meliputi ekstraksi data dari sistem sumber, transformasi data agar sesuai dengan standar kualitas dan konsistensi, serta pemuatan data ke dalam format yang siap untuk dianalisis (Kimball & Caserta, 2004).

Sistem ETL adalah aktivitas inti yang tidak terlihat oleh pengguna akhir dari data *warehouse*.

1. *Extract*

Proses ini melibatkan pemilihan dan pengambilan data dari beberapa sumber, serta mengakses data yang telah dipilih.

2. *Transform*

Dalam proses transformasi, data yang telah dipilih akan diubah menjadi format yang sesuai dengan kebutuhan.

3. *Load*

Proses ini bertujuan untuk memasukkan data ke dalam data *warehouse*, yang dilakukan dengan menjalankan skrip SQL.

2.1.6 *Object-Oriented Analysis and Design*

etode sistematis dalam pengembangan perangkat lunak yang memanfaatkan konsep pemrograman berorientasi objek untuk menganalisis, merancang, dan mengimplementasikan sistem disebut *Object-Oriented Analysis and Design* (OOAD). Valacich dan George (2017), mengatakan OOAD merupakan pendekatan teknik dalam menganalisis dan merancang aplikasi, sistem, dan bisnis dengan menggunakan prinsip-prinsip pemrograman yang berorientasi objek.

OOAD menggunakan diagram untuk merepresentasikan struktur dan perilaku sistem secara visual. Diagram ini membantu dalam komunikasi antara anggota tim, serta dalam memahami dan mendokumentasikan sistem.

OOAD dalam pengembangan perangkat lunak mempunyai tujuan, yaitu :

1. Membantu tim pengembangan untuk memahami persyaratan dan desain sistem secara mendalam melalui representasi visual.
2. Menyediakan bahasa umum yang dapat digunakan oleh pengembang, analis, dan pemangku kepentingan lainnya untuk berkomunikasi secara efektif.
3. Membantu dalam merencanakan dan melaksanakan pengembangan perangkat lunak secara iteratif dan inkremental.

Beberapa manfaat menggunakan OOAD dalam pengembangan perangkat lunak, sebagai berikut :

1. *Reusability*

Elemen-elemen yang dibuat secara efisien dapat digunakan lagi dalam upaya pengembangan perangkat lunak alternatif, sehingga menghasilkan konservasi waktu dan energi.

2. *Maintainability*

Desain yang modular dan jelas memudahkan pemeliharaan dan pengembangan lebih lanjut. Pengembangan tidak hanya pada perangkat lunak tersebut tetapi integrasi ke aplikasi lain.

3. *Scalability*

Sistem yang dirancang dengan baik dapat dengan mudah diskalakan untuk memenuhi kebutuhan yang lebih besar.

4. *Flexibility*

Desain berorientasi objek memungkinkan perubahan dan penyesuaian dilakukan dengan dampak minimal pada sistem keseluruhan. Pengembangan satu perangkat lunak yang sudah berjalan tidak mengganggu operasional perusahaan.

2.1.7 *Unified Modeling Language*

Unified Modeling Language (UML) menyediakan metode visual untuk memahami dan mencatat berbagai aspek sistem perangkat lunak, seperti struktur dan perilakunya.

UML mencakup berbagai diagram yang dapat diklasifikasikan ke dalam dua kategori utama: diagram struktural dan diagram perilaku. Diagram struktural digunakan untuk menunjukkan aspek statis dari sistem perangkat lunak. Diagram struktural fokus pada bagaimana elemen-elemen sistem diatur dan bagaimana mereka saling terkait. Diagram perilaku digunakan untuk menunjukkan aspek dinamis dari sistem perangkat lunak.

Diagram perilaku fokus pada bagaimana elemen-elemen sistem berinteraksi satu sama lain dan bagaimana sistem berperilaku sepanjang waktu.

Beberapa manfaat penggunaan UML dalam pengembangan perangkat lunak sebagai berikut :





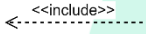
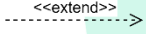

1. UML menawarkan bahasa standar yang dimengerti oleh berbagai pemangku kepentingan, seperti pengembang, analis, dan pengguna bisnis.
2. Diagram UML membantu dalam visualisasi struktur dan perilaku sistem, sehingga meningkatkan pemahaman dan memfasilitasi komunikasi.
3. UML membantu dalam mendokumentasikan berbagai aspek sistem secara rinci, yang penting untuk pemeliharaan dan pengembangan lebih lanjut.
4. UML membantu dalam analisis kebutuhan dan desain sistem, memungkinkan identifikasi potensi masalah dan solusi sebelum implementasi.

Berikut beberapa definisi diagram UML yang perlu diketahui:

1. *Use Case Diagram*

Salah satu jenis diagram dalam UML yang digunakan untuk memperlihatkan interaksi antara pengguna (*actor*) dan sistem perangkat lunak (*system*) disebut *use case diagram*. Diagram ini digunakan untuk mengidentifikasi dan menggambarkan berbagai *use case* atau skenario yang mungkin terjadi dalam sistem perangkat lunak. *Use case* tersebut menggambarkan *actor* (pengguna), *use case* (kasus penggunaan), termasuk interaksi antara *actor* dengan *use case*. *Actor* dalam *use case* diagram mewakili pengguna atau sistem eksternal yang berinteraksi dengan sistem perangkat lunak. *Use case* mewakili aksi atau fungsi yang dilakukan pengguna atau sistem eksternal dalam sistem perangkat lunak.

Tabel 2.1 Notasi dan Simbol Use Case Diagram

Simbol	Nama	Keterangan
	<i>Actor</i>	Individu atau sistem yang memperoleh manfaat dari dan berada diluar subjek.
	<i>Use Case</i>	Merupakan bagian utama dari fungsionalitas sebuah sistem yang dapat memperluas dan menggabungkan dengan <i>use case</i> lain. <i>Use case</i> menggunakan kata kerja dan berada di dalam subjek <i>boundary</i> .
	<i>Subject</i>	Batasan, atau batas, yang mewakili ruang lingkup sebuah subjek, seperti sistem atau prosedur bisnis tertentu.
	<i>Association</i>	Menunjukkan hubungan interaksi <i>actor</i> dengan <i>use case</i> .
	<i>Include</i>	Menunjukkan hubungan atau integrasi fungsionalitas dari <i>use case</i> penggunaan ke <i>use case</i> lainnya.
	<i>Extend</i>	Menunjukkan hubungan perluasan <i>use case</i> untuk mencakup perilaku opsional, serta menghubungkan <i>use case</i> ekstensi ke <i>use case</i> utama.
	<i>Generalization</i>	Menunjukkan <i>use case</i> khusus ke <i>use case</i> yang lebih general atau umum.




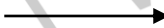
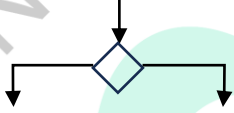

Sumber : (Dennis, Wixom, dan Tegarden, 2015)

2. Activity Diagram

Diagram UML yang dirancang khusus untuk merepresentasikan aktivitas atau alur kerja dalam sistem perangkat lunak secara visual. Diagram ini menggambarkan perkembangan berurutan dari aktivitas dalam alur kerja atau proses bisnis sistem perangkat lunak.

Komponen *Activity diagram* termasuk aktivitas, keadaan, keputusan, dan *fork/join*. Keadaan adalah kondisi di mana aktivitas dapat dilakukan atau tidak dapat dilakukan. Keputusan digunakan untuk membuat keputusan dalam aliran kerja, seperti bercabang ke aktivitas yang berbeda tergantung pada kondisi yang diberikan. *Fork/join* digunakan untuk menggambarkan pemrosesan paralel dari aktivitas atau tugas.

Tabel 2.2 Notasi dan Simbol Activity Diagram


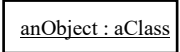
Simbol	Nama	Keterangan
	<i>Initial</i>	Menggambarkan permulaan dari serangkaian aktivitas atau tindakan.
	<i>Final-activity</i>	Menggambarkan bahwa semua aliran kontrol dan aliran objek dalam sebuah aktivitas (atau tindakan) sudah dihentikan.
	<i>Activity</i>	Menggambarkan serangkain tindakan dan diberi nama.
	<i>Control Flow</i>	Menunjukkan urutan eksekusi mulai dari <i>Initial</i> hingga aktivitas berakhir (<i>Final-activity</i>).
	<i>Decision</i>	Menunjukkan kondisi pengujian yang menjamin aliran objek atau kontrol hanya melalui satu jalur.
	<i>Swimlane</i>	Memecah diagram aktivitas menjadi baris dan kolom, kemudian menetapkan aktivitas individu (atau tindakan) ke orang atau objek yang bertanggung jawab untuk melakukannya.



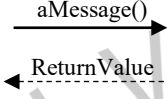
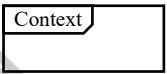
Sumber : (Dennis, Wixom, dan Tegarden, 2015)

3. Sequence Diagram

Sequence diagram merupakan representasi grafis yang menggambarkan urutan dan aliran interaksi antara elemen-elemen dalam sistem perangkat lunak disebut. *Sequence diagram* menampilkan urutan panggilan atau pesan antara objek, yang mencakup nilai dan parameter yang dikirimkan serta tanggapan yang diterima.

Tabel 2.3 Notasi dan Simbol Sequence Diagram

Simbol	Nama	Keterangan
	<i>Actor</i>	Individu atau sistem yang berada di luar sistem dan memperoleh manfaat darinya. Mengirim dan/atau menerima pesan dalam urutan.
	<i>Object</i>	Menggambarkan partisipasi dalam urutan dengan mengirim dan/atau menerima pesan.

Simbol	Nama	Keterangan
	<i>Lifeline</i>	Menampilkan kehidupan objek secara berurutan.
	<i>Execution occurrence</i>	Menyatakan kapan objek mengirim atau menerima pesan.
	<i>Message</i>	Mengirimkan data dari objek ke objek lainnya, dan pengembalian dilabeli dengan nilai yang dikembalikan.
	<i>Frame</i>	Konteks diagram urutan ditampilkan dalam bingkai.

Sumber : (Dennis, Wixom, dan Tegarden, 2015)

4. Class Diagram

Class diagram menggambarkan struktur sistem atau aplikasi yang sedang dibangun dengan memodelkan *class*, relasi antar *class*, *attribute*, serta *method*. *Class diagram* digunakan untuk menggambarkan arsitektur sistem secara visual dan dapat membantu dalam perancangan sistem yang lebih baik dan efisien.

2.1.8 Black Box Testing

Metode pengujian perangkat lunak yang dikenal sebagai "*Black Box Testing*" memungkinkan penguji untuk menilai fungsionalitas aplikasi tanpa memeriksa struktur internal atau kode sumbernya. pengujian ini adalah untuk memverifikasi bahwa perangkat lunak beroperasi sesuai dengan standar dan persyaratan yang ditetapkan. Metode ini juga dikenal sebagai pengujian berbasis spesifikasi atau pengujian fungsional.

Pengujian berbasis spesifikasi dilakukan berdasarkan persyaratan fungsional perangkat lunak yang ditentukan dalam dokumen spesifikasi. Penguji tidak memerlukan pengetahuan tentang desain atau implementasi internal.

Pengujian berbasis fungsional berpusat pada output yang dihasilkan oleh *input* yang berbeda, dan juga memeriksa bagaimana sistem menangani input ini untuk mencapai *output* yang diinginkan.

2.2 Tinjauan Studi

Terdapat beberapa penelitian sebelumnya yang memiliki korelasi dengan penelitian ini, seperti:

1. Sulistiawati dan Heni Sulistiani (2018). “*Perancangan Dashboard Interaktif Penjualan (Studi Kasus : PT Jaya Bakery)*” (jurnal). Sistem yang dirancang memiliki kemampuan untuk menampilkan data penjualan dalam bentuk grafik dan mengeluarkan nota transaksi penjualan sebagai hasilnya. Penjualan roti Jaya Bakery dapat dipantau dengan lebih mudah dengan tampilan ini.

2. Susan Dian Purnamasari dan Alek Wijaya (2017). “*Dashboard Sistem Informasi Eksekutif Penjualan*” (jurnal). Temuan studi dari sistem ini meliputi *dashboard* kinerja distributor, data penjualan semen berdasarkan waktu, penjualan semen yang dikategorikan berdasarkan jenis semen dan area distribusi, analisis kinerja ekspediter, dan *dashboard* yang menampilkan penjualan semen berdasarkan jenis pembayaran.

3. Dede Renza Apriliandi dan Fatoni (2022). “*Dashboard Sistem Informasi Penjualan Obat (Studi Kasus Apotek RSUD Prabumulih)*” (jurnal). Penelitian ini menggunakan *Business Intelligence Roadmap*.

Hasil penelitian berhasil menyajikan informasi secara cepat dan mudah dipahami untuk bidang pelaporan serta dapat membantu dalam analisis masalah, pengambilan keputusan, dan peningkatan layanan karena disajikan dalam bentuk visualisasi grafik.

4. Herly Nurrahmi dan Andri Susanto (2018). “*Perancangan Sistem Informasi Dashboard Penjualan dan Sales Report*” (jurnal). Sistem dirancang menggunakan OOAD dan UML.

Sistem informasi dashboard ini berfungsi sebagai alat bantu khusus bagi manajer untuk mencatat transaksi dan memantau laporan penjualan.

5. Muhtad Fadly, Dina Ros Muryana, dan Adhie Thyo Priandika (2020). “*Sistem Monitoring Penjualan Bahan Bangunan Menggunakan Pendekatan Key Perfomance Indicator*”. Sistem dirancang menggunakan OOAD.

Penelitian ini menyimpulkan bahwa pengembangan aplikasi monitoring penjualan akan memudahkan general manager dalam memantau penjualan setiap sales melalui tampilan dashboard.