BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Berdasarkan penjelasan yang telah disampaikan secara mendalam, diperlukan juga kajian literatur yang komprehensif untuk mendukung penelitian yang akan dilakukan. Kajian ini bertujuan untuk memberikan dasar teoritis yang kuat, mengidentifikasi kesenjangan dalam penelitian sebelumnya, serta membantu merumuskan hipotesis dan metode penelitian yang relevan. Dengan demikian, tinjauan pustaka yang lengkap dan sistematis akan menjadi landasan penting dalam pengembangan penelitian ini.

Table 2. 1 Daftar *Literatur*

No	Penulis dan Tahun	Judul	Uraian
1	(Kristianto & Supatman, 2024)	Sistem Pakar Diagnosa Kerusakan Kendaraan Bermotor Jenis *Matic* Menggunakan Metode *Naive Bayes*	Masalah utama yang dihadapi dalam mendiagnosis kerusakan pada sepeda motor matic adalah kurangnya pengetahuan teknis dan pengalaman pengguna dalam mengidentifikasi gejala dan penyebab kerusakan yang tepat.
2	(Epriliani et al., 2022)	Implementasi Algoritma *Naive Bayes* Untuk Memprediksi Kerusakan Sepeda Motor Pada Bengkel Citra Djaya Motor	Pengembangan sistem aplikasi prediksi kerusakan sepeda motor dengan menggunakan algoritma *Naive Bayes* telah diusulkan sebagai solusi inovatif. Algoritma *Naive Bayes*, yang dikenal karena kesederhanaannya dan kemampuannya dalam klasifikasi probabilistik.
3	(Nuryahya & Muflihah, 2023)	Sistem Pakar Diagnosa Kerusakan Sepeda	Solusi yang diusulkan adalah pengembangan sistem pakar

		Motor Bebek Karburator Dengan Metode *Certainty Factor* Berbasis *Web Mobile*	menggunakan metode certainty factor untuk mendiagnosa kerusakan sepeda motor bebek karburator. kelemahan Potensial adalah ketergantungan pada data yang dikumpulkan dan aturan yang ditetapkan dalam sistem pakar.
4	(Mulyani & Natsir, 2023)	Sistem Pakar Diagnosis Kerusakan Sepeda Motor Di Bengkel Rahmat Cort Menggunakan Metode *Forward chaining*	Pengembangan sistem pakar berbasis *web* dengan metode *forward chaining* telah diusulkan. Sistem ini dirancang untuk menganalisis gejala kerusakan yang dialami kendaraan dan memberikan solusi yang tepat berdasarkan data yang tersedia.

2.2 Pengertian Bengkel

Bengkel memiliki arti tempat yang digunakan untuk memperbaiki sepeda, mobil dan sebagainya. Dimulai dari transaksi pembelian penjualan spare part, pengolahan data *customer*, pelayanan servis, hingga mendiagnosis kerusakan. Sehingga memperbaiki dan meningkatkan efisiensi kerja (Fauzy et al., 2020).

Kesimpulan dari literatur adalah bahwa bengkel merupakan tempat yang digunakan untuk memperbaiki kendaraan seperti sepeda dan mobil. Aktivitas di bengkel meliputi transaksi pembelian dan penjualan suku cadang, pengolahan data pelanggan, pelayanan servis, serta diagnosis kerusakan. Semua kegiatan ini bertujuan untuk memperbaiki kendaraan dan meningkatkan efisiensi kerja di bengkel.

2.3 Pengertian Sepeda Motor

Sepeda bermotor adalah kendaraan yang di gerakkan oleh peralatan teknik untuk penggeraknya, dan digunakan untuk alat transportasi darat. Umumnya kendaraan bermotor menggunakan mesin pembakaran dalam (perkakas atau alat untuk menggerakan atau membuat sesuatu yang di jalankan dengan roda, di gerakkan oleh tenaga motor penggerak, menggunakan bahan bakar minyak atau tenaga alam). Sepeda motor memiliki roda dua dan berjalan diatas jalanan (Nurhidayati et al., 2024).

Kesimpulan dari literatur adalah bahwa sepeda bermotor merupakan kendaraan darat yang digerakkan oleh peralatan teknik, biasanya menggunakan mesin pembakaran dalam. Sepeda motor menggunakan bahan bakar minyak atau tenaga alam untuk menggerakkan roda-roda. Sepeda motor umumnya memiliki dua roda dan dirancang untuk berjalan di atas jalanan.

2.4 Pengertian Diagnosa

Diagnosa adalah Proses menemukan kelemahan atau penyakit apa yang dialami seseorang seseorang dengan melalui pengujian dan studi yang seksama mengenai gejala-gejalanya. Diagnosa memiliki proses tidak langsung ketika mengidentifikasi jenis penyakit dengan cara mengetahui jenisnya. Sehingga kita dapat dikatakan bahwa itu penyakit yang sedang di alami (Nurhidayati et al., 2024).

Kesimpulan dari literatur adalah bahwa diagnosa merupakan proses menemukan kelemahan atau penyakit yang dialami seseorang melalui pengujian dan studi yang seksama terhadap gejala-gejalanya. Diagnosa sering kali melibatkan proses tidak langsung untuk mengidentifikasi jenis penyakit dengan memahami jenis gejalanya. Dengan demikian, diagnosa memungkinkan kita untuk menentukan penyakit yang sedang dialami seseorang.

2.5 Sistem Pakar

Sistem pakar adalah memindahkan kepakaran seseorang kedalam sebuah sistem atau aplikasi tertentu dengan harapan akan menghasilkan sebuah informasi sebagaimana yang diinginkan (Wiyandra et al., 2021). Sistem pakar pada umumnya mempunyai tiga elemen, yaitu basis pengetahuan (*Knowledge Base*), Mesin

Inferensi (*Inference Engine*), dan Antar muka Pemakai (*User Interface*) (Sihombing & Sitio, 2021).

Sistem pakar merupakan suatu sistem yang dibangun untuk memindahkan kemampuan dari seorang atau beberapa orang pakar dalam sistem yang digunakan untuk memecahkan masalah yang dihadapi oleh pemakai dalam bidang tertentu (Simamora, 2021)

Kesimpulan dari literatur yang diberikan adalah bahwa sistem pakar merupakan teknologi yang mentransfer keahlian manusia ke dalam sistem komputer. Tujuannya adalah untuk menghasilkan informasi yang dibutuhkan oleh pengguna dengan cara yang serupa dengan pakar manusia. Sistem ini terdiri dari tiga elemen utama: basis pengetahuan, mesin inferensi, dan antarmuka pemakai. Sistem pakar dirancang untuk mengatasi masalah tertentu dengan menggunakan kemampuan yang biasanya dimiliki oleh seorang pakar dalam bidang tersebut.

2.6 Alat Pendukung

Alat pendukung pengembangan sistem web meliputi text editor pengembangan seperti Sublime Text, framework untuk membangun aplikasi, serta alat pengujian dan pengiriman aplikasi. Alat-alat ini memfasilitasi pengembangan sistem web dengan efisiensi dan kualitas yang tinggi.

2.6.1 *Database Management Sistem* (DBMS)

Database Management Sistem (DBMS) merupakan paket program (Software) yang dibuat agar memudahkan dan mengefisienkan pemasukan, pengeditan, penghapusan dan pengabilan informasi terhadap database. Software yang tergolong kedalam DBMS anatar lain, Microsoft SQL, MySQL, Oracle, MS.Access dan lain-lain. Database Management System (DBMS) adalah sekumpulan program yang digunakan untuk mendefinisikan, mengatur administrasi, dan memproses basis data dan aplikasi-aplikasi yang terkait dengan basis data (Ardhani, 2020).

2.6.2 *Xampp*

XAMPP adalah sebuah *software web server apache* yang didalamnya sudah tersedia *database server MySQL* dan dapat mendukung pemrograman *PHP* (Tan & Hajjah, 2020).

2.6.3 PHP (*Hypertext Preprocessor*)

PHP (*Hypertext Preprocessor*) merupakan salah satu bahasa pemograman yang berjalan dalam sebuah *web server* dan berfungsi sebagai pengolah data pada sebuah *server*. Data yang dikirim oleh *user client* akan diolah dan disimpan pada database *web server* dan dapat ditampilkan kembali apabila diakses. Untuk menjalankan kode-kode program *PHP*, file harus di upload kedalam *server* (Rahman et al., 2024).

2.6.4 *HTML* (*HyperText Markup Language*)

HTML (*HyperText Markup Language*) merupakan sebuah bahasa markup (tanda) yang digunakan untuk membuat sebuah "halaman web" dan menampilkan berbagai informasi didalam sebuah *browser internet*. *HTML* berupa kode-kode tag yang mengintruksikan *browser* untuk menghasilkan tampilan sesuai dengan yang diinginkan. Sebuah *file* yang merupakan *file HTML* dapat dibuka dengan menggunakan *browser web* seperti *Mozilla Firefox* atau *Microsoft Internet Explorer*. *HTML* juga dapat dikenali oleh aplikasi pembuka *email* ataupun dari PDA dan program lain yang memiliki kemampuan *browser* (Ardhani, 2020).

2.6.5 MySQL

MySQL merupakan *database server* yang mampu bekerja pada sistem operasi *windows*, *Linux* dan *Unix* dengan linsensi *freeware* sehingga *user* tidak melanggar hak cipta ketika menggunakannya (Tan & Hajjah, 2020).

2.6.6 *Web Browser*

web browser digunakan untuk menampilkan hasil *website* yang telah dibuat. *Web browser* yang paling sering digunakan, di antaranya *Mozilla Firefox*, *Google Chrome*, *Internet Explorer*, *Oper*,, dan *Safari* (Rahman et al., 2024).

2.7 *Website*

Website atau disingkat *web* dapat diartikan sekumpulan halaman yang terdiri atas beberapa laman yang berisi informasi dalam bentuk data digital, baik berupa teks, gamber, video, audio dan animasi lainnya yang disediakan melalui jalur koneksi *internet*. *Website* adalah sekumpulan halaman yang menampilkan konten atau sesuatu yang bisa diakses atau dibuka apabila kita mengakses *internet*.

2.8 Metode *K-Nearest Neighbors* (KNN)

K-Nearest Neighbor (KNN) adalah salah satu metode *machine learning* yang masuk dalam kategori *supervised learning* untuk melakukan klasifikasi teks dan data. Metode *K-Nearest Neighbor* melakukan klasifikasi terhadap objek berdasarkan data yang jaraknya paling dekat dengan objek tersebut atau biasa dapat disebut neighborhood (Riyadi et al., 2021).

Metode *K-Nearest Neighbor* (KNN) merupakan metode *machine learning* yang tergolong dalam *supervised learning*, dan koordinat data akan diklasifikasikan berdasarkan koordinat data pembelajaran yang paling berdekatan (Priyandoko, 2022).

Metode *K-Nearest Neighbor* (KNN) merupakan salah satu algoritma pembelajaran mesin yang bersifat *nonparametrik*. Model *nonparametrik* adalah model yang tidak mengasumsikan distribusi di dalam data. Kelebihannya adalah garis keputusan kelas yang dihasilkan model tersebut bisa jadi sangat fleksibel dan sangat nonlinier. Dalam menentukan k optimum, seringkali mengalami kesulitan. Walaupun sebenarnya algoritma KNN adalah algoritma sederhana yang mudah dilakukan, tidak sedikit peneliti yang meragukan nilai dari k yang dihasilkan. Nilai k yang tinggi akan mengurangi efek noise, tetapi akan membuat hasil prediksi semakin kabur, sedangkan jika nilai k terlalu kecil atau 1, akan mengakibatkan hasil prediksi terasa kaku (Yulianto & Darwis, n.d.). Algoritma KNN (*K-Nearest Neighbors*) sangat baik digunakan sebagai *machine learning*, karena dapat di kombinasikan dengan metode yang ada (Maulana et al., 2021).

Kesimpulan dari literatur tersebut adalah bahwa *K-Nearest Neighbor* (KNN) adalah metode *machine learning* yang digunakan untuk klasifikasi teks dan

data dalam *supervised learning*. KNN melakukan klasifikasi berdasarkan jarak terdekat antara objek dengan data-data lainnya yang dikenal sebagai *neighborhood*. Metode ini bersifat *nonparametrik*, artinya tidak membuat asumsi tentang distribusi data, dan mampu menghasilkan garis keputusan yang fleksibel dan *nonlinier*. Berikut adalah rumus dari metode *K-Nearest Neighbor* (KNN).

$$d(A,B) = \sum_{i=1}^{t} \sqrt{(A_1 - B_1)^2 + \cdots + (A_i - B_i)^2}$$

Source : (Riyadi et al., 2021)

Keterangan:

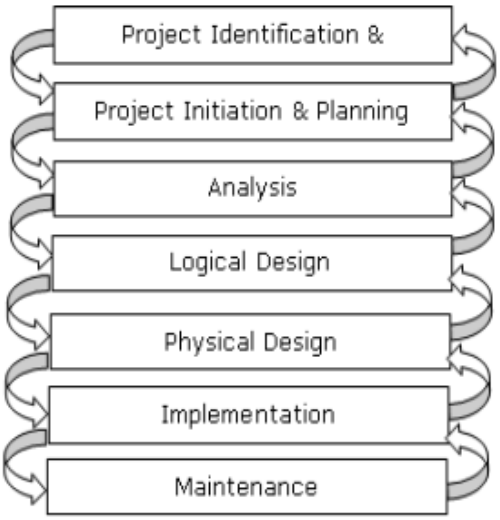
Ai = kata ke-I dokumen *Testing* atau Uji

Bi = kata ke-I dokumen *Training* atau Latih

d (A, B) = Jarak dokumen A ke dokumen B

2.9 SDLC (*Sofware Development Life Cycle*)

Sistem Penelitian ini menggunakan model SDLC (*Sofware Development Life Cycle*) adalah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sebuah sistem. SDLC juga merupakan pola yang digunakan untuk mengembangkan sebuah sistem perangkat lunak (Tan & Hajjah, 2020). Berikut adalah tahapan SDLC (*Sofware Development Life Cycle*) dibawah ini.

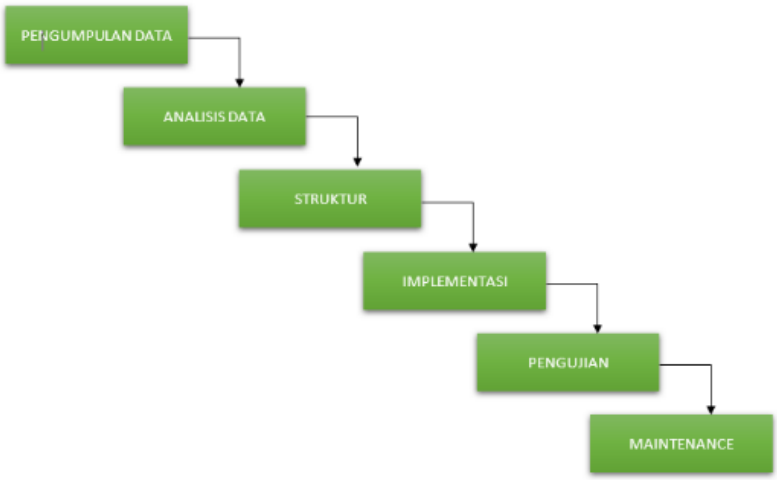


Gambar 2. 1 Tahapan SDLC (*Sofware Development Life Cycle*)

Source : (Tan & Hajjah, 2020)

2.10 Metode *Waterfall*

Metode *waterfall* yang merupakan sebuah metode yang disusun secara berurutan pada suatu langkah atau proses pembuatan sebuah penelitian. Berikut gambaran dari metode *waterfall* dibawah ini.



Gambar 2. 2 Metode *Waterfall*

Source : (Maulana et al., 2021)

1. **Pengumpulan Data (*Requirements Gathering*)**

   Tahap ini melibatkan pengumpulan dan spesifikasi kebutuhan dari klien atau pengguna akhir. Ini mencakup pemahaman tentang apa yang diharapkan dari sistem yang akan dikembangkan, termasuk fitur, fungsi, dan batasan. Hasil dari tahap ini adalah dokumen yang mencakup semua kebutuhan sistem yang jelas dan terdefinisi.

2. **Analisis Data (*System Analysis*)**

   Setelah kebutuhan dikumpulkan, tahap selanjutnya adalah menganalisis data yang dikumpulkan untuk memahami kebutuhan secara rinci. Analis sistem akan memecahkan kebutuhan menjadi komponen yang lebih kecil dan menganalisis bagaimana setiap komponen akan berinteraksi satu sama lain. Tahap ini juga melibatkan pemilihan teknologi dan pendekatan terbaik untuk memenuhi kebutuhan yang telah ditentukan.

3. **Struktur (*System Design*)**

   Tahap desain melibatkan merencanakan arsitektur dan struktur sistem. Ini termasuk merancang antarmuka pengguna, basis data, dan komponen

perangkat lunak lainnya. Desain harus mencakup detail tentang bagaimana sistem akan dibangun, termasuk pemilihan bahasa pemrograman, kerangka kerja, dan alat-alat yang akan digunakan.

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

   Implementasi adalah tahap di mana kode benar-benar ditulis berdasarkan desain yang telah dibuat. Pengembang perangkat lunak akan memulai pembuatan sistem berdasarkan spesifikasi yang telah ditetapkan. Tahap ini juga dikenal sebagai tahap pembangunan atau koding.

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

   Setelah implementasi selesai, sistem harus diuji untuk menemukan dan memperbaiki kesalahan atau bug. Pengujian melibatkan serangkaian tes yang dirancang untuk memastikan bahwa sistem bekerja sesuai dengan yang diharapkan dan memenuhi semua kebutuhan yang telah ditetapkan. Jenis tes yang dilakukan bisa berupa unit testing, integration testing, system testing, dan *user acceptance testing*.

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

   Setelah sistem diuji dan dianggap siap untuk digunakan, sistem dipasarkan atau diberikan kepada pengguna akhir. Namun, proses pengembangan belum selesai karena sistem akan memerlukan pemeliharaan untuk memperbaiki *bug* yang muncul, menambahkan fitur baru, atau meningkatkan kinerja. *Maintenance* adalah tahap terakhir dalam metode *Waterfall*, dan biasanya berlangsung selama siklus hidup sistem.

2.11 Perancangan Sistem *Unified Modeling Language* (*UML)*

Menurut (Rosa & Shalahuddin, 2020) UML merupakan bahasa visual untuk permodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. *UML* terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori (Rosa & Shalahuddin, 2020) yaitu:

a. *Stucture Diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.

b. *Behavior Diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
c. *Interactions Diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

Dari 13 model diagram tersebut, penelitian ini hanya mengambil 2 model diagram yaitu *Usecase Diagram* dan *Activity Diagram* (Rosa & Shalahuddin, 2020).

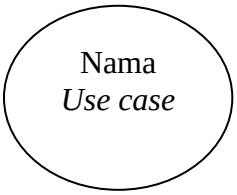
2.1.1 *Usecase Diagram*

Menurut (Rosa & Shalahuddin, 2020). "*Use case diagram* atau diagram *use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat oleh pengembang sistem sebelum melakukan perancangan antarmuka pada suatu perangkat lunak". *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. simbol-simbol yang ada pada diagram *use case* dapat dilihat pada gambar 2.3 di bawah ini.

Table 2. 2 *Usecase Diagram*

No.	Simbol	Keterangan
1.	Aktor/*Actor*  Nama Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem Informasi yang akan dibuat itu sendiri.

Table 2. 3 *Usecase Diagram*

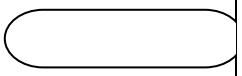
No.	Simbol	Keterangan
2.	*Use case* 	Fungsionalitas yang disediakan sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal *frase* nama *use case*.
3.	*Assocation* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
4.	*Extend/Ekstensi* <<*extend*>> 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
5.	*Generalization/ Generalisasi* 	Hubungan generalisasi dan spesialisasi (umum khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih dari lainnya.
6.	*Menggunakan/ Include/Uses* <<*include*>> «uses» 	Relasi *use case* tambahan ke sebuah *use case* di mana *use case* yang ditambahkan memerlukan *use case* ini ntuk menjalankan fungsinya atau sebagai syarat.

Sumber: (Rosa & Shalahuddin, 2020)

2.1.2 *Activity Diagram*

Diagram aktifitas atau *activity Diagram* menggambarkan workflow (aliran kerja) atau aktivitas sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan adalah bahwa diagram aktivitas menggambarkan aktifitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem (Rosa & Shalahuddin, 2020).

Table 2. 4 *Activity Diagram*

No	Simbol	Keterangan
1	Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3	Percabangan 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4	penggabungan/ *join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5	Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
6	*Swimlane* Nama *Swimlane*	*Swimlane* memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Sumber :(Rosa & Shalahuddin, 2020)

2.12 Pengujian *Black Box*

Salah satu metode pengujian yang berfokus pada spesifikasi fungsionalitas dari perangkat lunak disebut *Black Box Testing* Tujuan dari pengujian *Black Box Testing* adalah untuk menemukan fungsi yang tidak sesuai, kesalahan *interface*, kesalahan struktur data, performansi, inisialisasi dan terminasi (Mulki et al., 2023). Pengujian *black box* menggunakan tipe pengujian *functional testing*. Pengujian ini dilakukan dengan menguji fungsi *input - output* sebuah program apakah keluaran program sesuai dengan apa yang diinputkan (Selan et al., 2023).