BAB II

LANDASAN TEORI

2.1 Sistem Informasi

Kata sistem informasi terdiri dari dua kata, yaitu: sistem dan informasi. Menurut Elisabet Yunaeti Anggraeni et. Al (2017) di dalam bukunya yang berjudul "Pengantar Sistem Informasi" dikemukakan bahwa *"Sistem adalah kumpulan orang yang saling bekerja sama dengan ketentuan-ketentuan aturan yang sistematis dan terstruktur untuk membentuk satu kesatuan yang melaksanakan suatu fungsi untuk mencapai tujuan"* (p. 1 – 2) serta *"Informasi adalah data yang diolah menjadi lebih berguna dan berarti bagi penerimanya, serta untuk mengurangi ketidakpastian dalam proses pengambilan keputusan mengenai suatu keadaan"* (p. 1 – 2). Dengan pengertian diatas, dapat kita tarik kesimpulan bahwa sistem informasi adalah sekumpulan orang dan aturan yang terstruktur secara sistematis untuk mengolah informasi.

2.2 Perangkat Lunak

2.2.1 HTML

Hypertext Markup Language (HTML) adalah sebuah bahasa pemrograman terstruktur yang dikembangkan untuk membuat halaman website yang dapat diakses atau ditampilkan menggunakan web browser (Didik Setiawan, 2020). HTML dibuat oleh Tim Berners Lee pada tahun 1989 dan dikembang lanjutkan oleh *World Wide Web Consortium* (W3C).

2.2.2 CSS

Cascading Style Sheet (CSS) merupakan kode pemrograman untuk menghias dan mengatur gaya tampilan/layout halaman web supaya lebih elegan dan menarik (Didik Setiawan, 2020). Dikembangkan pertama kali di SGML pada tahun 1970 dan terus dikembangkan hingga saat ini.

2.2.3 PHP

PHP:Hypertext Preprocessor atau pada awalnya *Personal Home Page* adalah bahasa pemrograman tingkat tinggi yang dipasang pada dokumen HTML yang diciptakan oleh Rasmus Lerdorf (Didik Setiawan, 2020). Sintaks dari bahasa pemrograman PHP mirip sekali dengan C, Java, dan Perl. Tujuan utama dibuatnya PHP adalah untuk merancang website secara dinamis dan bekerja secara otomatis.

2.2.4 Javascript

Pada buku *Eloquent Javascript* karangan Marijn Haverbeke (2018) menceritakan bahwa Javascript pertama kali diperkenalkan pada browser Netscape Navigator pada tahun 1995 sebagai cara untuk menambahkan program pada halaman web. Sejak saat itu bahasa javascript mulai diadopsi oleh web browser lainnya. Berkat javascript, web browser modern dapat berinteraksi kepada usernya tanpa perlu memuat ulang halaman web.

2.2.5 MySQL

MySQL adalah *open-source relational database management system* yang dibuat oleh Michael "Monty" Widenius pada tahun 1995 (Russell J.T. Dyer, 2008). Database MySQL ini menggunakan bahasa *Structured Query Language* (SQL) untuk melakukan manipulasi data yang menggunakan model relasional, dimana setiap data disimpan dalam sebuah tabel yang setiap barisnya memiliki kunci unik untuk mengenali setiap data.

2.2.6 Python

Dalam buku *The Quick Python Book* (Vernon L. Ceder, 2010), Python adalah sebuah bahasa pemrograman yang dikembangkan oleh Guido van Rossum. Kelebihan Python dibandingkan bahasa lain berupa: mudah digunakan, merupakan bahasa yang ekspresif (satu baris kode dapat melakukan banyak hal), mudah dibaca, library bawaan yang lengkap, *cross-platform* dan gratis.

2.3 Pemodelan Diagram

2.3.1 Unified Modeling Language

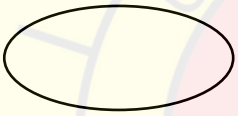
Unified Modeling Language (UML) memberikan kemampuan untuk menggambarkan aspek yang beragam dari sebuah sistem perangkat lunak Pada tahun 1997, UML versi 1.0 diresmikan sebagai jawaban dari seruan *Object Management Group*, sebuah badan standarisasi untuk pemrograman berorientasi objek, tentang spesifikasi pemodelan yang seragam (Martina Seidl et. al., 2014).

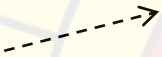
2.3.2 Use Case Diagram

Use case diagram digunakan untuk menggambarkan skenario penggunaan dari sistem yang sedang dikembangkan. *Use case diagram* menjelaskan apa yang harus sistem dapat lakukan tanpa menjelaskan data-data detail, dimana data-data tersebut akan digambarkan pada diagram lain (Martina Seidl et. al., 2014).

Beberapa simbol yang digunakan pada *use case diagram*:

Tabel 2.1 Simbol-simbol pada *use case diagram*

Nama & Simbol	Penjelasan Kegunaan
 Use Case	Untuk menjelaskan fungsionalitas yang diharapkan dalam sistem.
 System	Sebagai batasan kumpulan use case dari satu sistem.
 Actor	Yang akan berinteraksi dengan sistem, dapat berupa manusia (cth. pengguna / end user) ataupun non-manusia (cth. server)

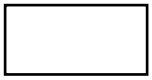
 Association	Menggambarkan adanya interaksi antara actor dan sebuah *use case.*
 Extend & Include	Keduanya digunakan untuk menggambarkan hubungan antar *use case*. Dengan penjelasan secara individu sebagai berikut: **Extend** Menggambarkan dimana *use case* asal (yang menunjuk) dapat melanjutkan/menjalankan *use case* tujuan (yang ditunjuk) namun tidak harus. **Include** Menggambarkan dimana *use case* asal (yang menunjuk) membutuhkan proses *use case* tujuan (yang ditunjuk) terlebih dahulu untuk menjalankan fungsionalitasnya.

2.3.3 Activity Diagram

Activity diagram digunakan untuk menggambarkan alur kerja dari suatu *use case*. Dalam diagram ini, digambarkan secara detil alur yang terjadi dari awal sampai akhir dari sebuah fungsionalitas pada sistem.

Beberapa simbol yang digunakan pada *activity diagram*:

Tabel 2.2 Simbol-simbol pada *activity diagram*

Nama & Simbol	Penjelasan Kegunaan
 Swimlane	Digunakan untuk menjelaskan siapa/apa yang melakukan sebuah aktivitas
 Start Point	Titik awal dari *activity diagram*
 End Point	Titik akhir dari *activity diagram*
	Menggambarkan aktifitas yang dilakukan

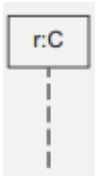
Activity	
 Decision	Percabangan alur aktifitas berdasarkan kondisi yang ditentukan
 Connector	Menggambarkan urutan/alur dari aktifitas
 Join & Fork	Dapat digunakan sebagai join/fork. Menggambarkan pencabangan aktifitas berjalan secara paralel.

2.3.4 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek dalam menjalankan suatu perintah. Diagram ini berfokus dalam urutan interaksi secara kronologis.

Beberapa simbol yang digunakan pada *sequence diagram*:

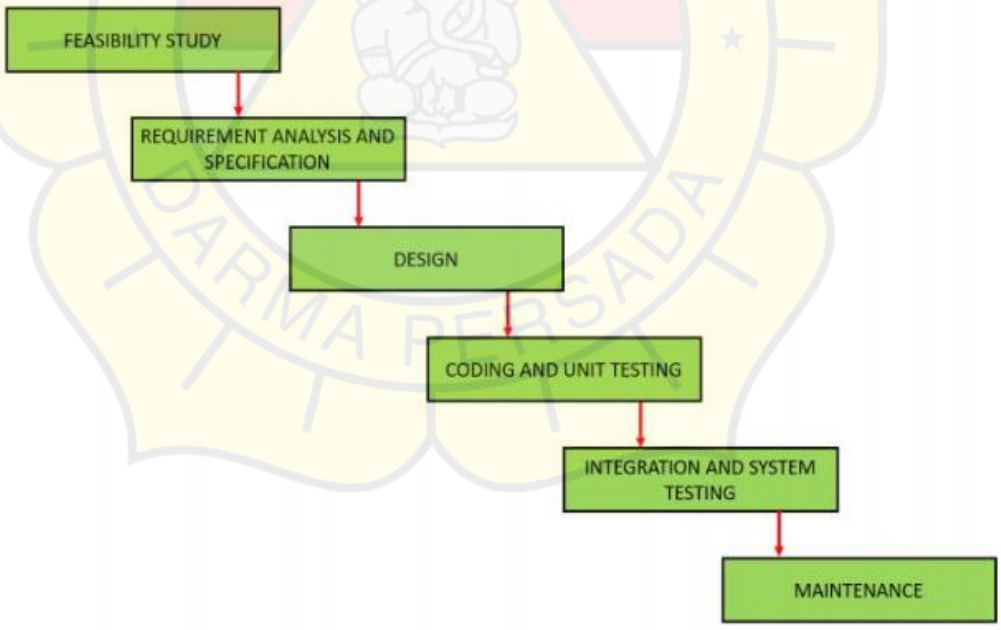
Tabel 2.3 Simbol-simbol pada *sequence diagram*

Nama & Simbol	Penjelasan Kegunaan
 Lifeline	Menggambarkan objek yang berinteraksi
 Event	Menggambarkan satu proses interaksi yang terjadi
 *Synchronous Message*	Menggambarkan pesan yang diberikan secara *synchronous*
 *Asynchronous Message*	Menggambarkan pesan yang diberikan secara *asynchronous*

Response Message	Menggambarkan pesan yang diterima yang merupakan hasil langsung saat melakukan interaksi

2.4 Metode Pengembangan Waterfall

Metode *classic waterfall* adalah sebuah metode pengembangan sistem yang bersifat berurutan. Nama asli dari metode ini adalah *linear sequential model* yang dapat disebut juga *classic life cycle* (Mei Prabowo, 2020). Metode pengembangan *classic waterfall* hanya berlangsung satu arah tanpa kembali ke proses sebelumnya. Berikut penggambaran dari fase-fase metode classic waterfall:



Gambar 2.1 Alur Metode Classic Waterfall

a. *Feasibility Study*

Pada tahap ini adalah tahap untuk menentukan kelayakan finansial dan teknis dalam pengembangan perangkat lunak.

b. *Requirement Analysis and Specification*

Tahap ini bertujuan untuk memahami kebutuhan pada aplikasi dan pendokumentasian. Biasanya pada tahap ini dilakukan pengumpulan data terhadap calon pengguna perangkat lunak yang dikembangkan.

c. *Design*

Pada tahap ini sudah mulai untuk menggambarkan garis besar dari aplikasi yang akan dikembangkan. Mulai dari penggambaran sketsa, alur algoritma dan diagram-diagram dari perangkat lunak yang akan dikembangkan.

d. *Coding and Unit Testing*

Pada tahap ini sudah mulai mengembangkan perangkat lunak dan mengimplementasikan alur pada diagram yang telah didesain. Setelah melakukan pengembangan, alur dan modul yang telah dikembangkan diuji untuk melihat apakah ada kesalahan yang terjadi.

e. *Integration and System Testing*

Pada tahap ini perangkat lunak sudah mulai diaplikasikan dan digunakan yang dimulai dari tahap pengujian alpha, pengujian beta dan pengujian pengguna.

Setelah testing selesai, aplikasi akan mulai diimplementasikan dalam kegiatan sehari-hari.

f. *Maintenance*

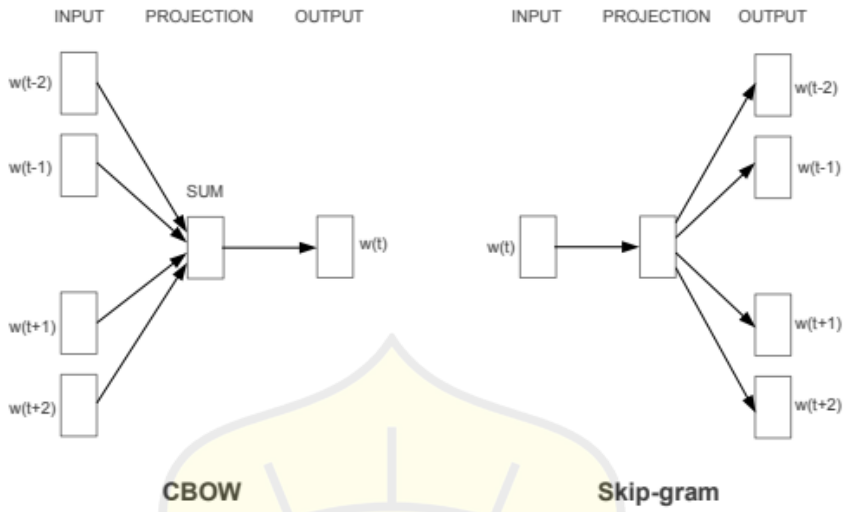
Tahap terakhir merupakan perawatan perangkat lunak apabila terjadi sebuah kesalahan saat penggunaan ataupun pengembangan dan penyesuaian perangkat lunak terhadap perkembangan perangkat keras dan gawai lainnya.

2.5 Algoritma yang Dipakai

2.5.1 *Word2Vec*

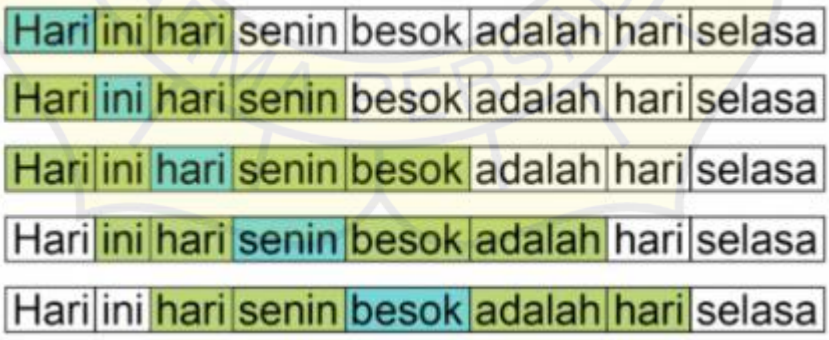
Word2vec atau *Word-to-Vector* adalah sebuah algoritma embedding yang dikembangkan oleh Thomas Mikolov dan rekan sejawatnya pada tahun 2013. *Word2vec* adalah sebuah model *shallow neural network* yang mengubah kata-kata menjadi sebuah vector. Dimana vector tersebut memiliki *"relationship"* terhadap satu sama lain.

Arsitektur *neural network* yang digunakan oleh *Word2vec* adalah *Continous Bag-of-Word* (CBOW) dan *Skip-gram*.



Gambar 2.2 Gambaran CBOW dan Skip-gram

Model *Continous Bag-of-Word*, bekerja dengan cara menggunakan beberapa kata sebelum dan sesudah dari sebuah kata yang ingin diprediksi dalam jarak yang telah ditentukan. Model *Skip-gram* bekerja sebaliknya, model ini bekerja dengan cara memprediksi kata-kata sekitar dengan jarak yang ditentukan menggunakan kata yang ditetapkan.



Gambaran 2.3 Ilustrasi Proses CBOW dan Skip-gram

2.5.2 *Smooth Inverse Frequency*

Smooth Inverse Frequency (SIF) adalah sebuah skema pembobotan untuk meningkatkan performa dalam proses *sentence embedding*. *Smooth Inverse Frequency* berguna untuk mengurangi "kepentingan" sebuah kata dalam kalimat.

Smooth Inverse Frequency bekerja dengan asumsi bahwa kata yang memberikan makna penting pada suatu kalimat cenderung untuk lebih sedikit keberadaannya dalam suatu kalimat. Rumus pembobotan kata pada *Smooth Inverse Frequency* adalah :

$$\mathbf{a / (a + p(w))}$$

Keterangan:

a : parameter

p(w) : frekuensi kata dalam model

Setelah dilakukan pembobotan, akan dilakukan *sentence embedding* yaitu setiap kata dalam kalimat akan dihitung menjadi sebuah vektor yang menangkap makna dari kalimat. Algoritma yang digunakan adalah sebagai berikut:

Algorithm 1 Sentence Embedding

Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$, parameter a and estimated probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.
Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$
1: **for all** sentence s in $\mathcal{S}$ **do**
2: $v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$
3: **end for**
4: Form a matrix X whose columns are $\{v_s : s \in \mathcal{S}\}$, and let u be its first singular vector
5: **for all** sentence s in $\mathcal{S}$ **do**
6: $v_s \leftarrow v_s - uu^\top v_s$
7: **end for**

Gambar 2.4 Algoritma untuk Sentence Embedding

2.5.3 Cosine Similarity

Dewa Ayu et. al. (2016) dalam jurnalnya "*Cosine Similarity* adalah ukuran kesamaan antara dua buath vektor dalm sebuah ruang dimensi yang didapat dari nilai cosinus sudut dari perkalian dua buah vektor yang dibandingkan karena cosinus dari 0 adalah 1 dan kurang dari 1 untuk nilai sudut yang lain, maka nilai *similarity* dari dua buah vektor dikatakan mirip ketika nilai dari *cosine similarity* adalah 1".

Rumus dari cosine similarity adalah:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Keterangan:

A : vektor 1

B : vektor 2

A_i : bobot *term i* dalam blok A_i

B_i : bobot *term i* dalam blok B_i

i : jumlah term dalam kalimat

n : jumlah vektor