BAB 2
LANDASAN TEORI

2.1 System Development Life Cycle (SDLC

Software Development Life Cyle (SDLC) adalah sebuah proses yang menggambarkan metode dan strategi seperti bagaimana caranya mengembangkan desain dan memelihara proyek perangkat lunak serta memastikan bahwa semua tujuan, sasaran, fungsional, dan kebutuhan pengguna terpenuhi [7]. Pengertian System Development Life Cycle (SDLC) lainnya adalah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem-sistem dalam rekayasa sistem dan rekayasa perangkat lunak [8]. Hal ini bertujuan untuk memberikan hasil yang sesuai dengan ekspektasi dan keinginan dari pengguna, serta mengurangi pengerjaan yang dilakukan secara berulang-ulang. Langkah penerapan dari metode SDLC adalah sebagai berikut.

1. Identifikasi Masalah

   Pada tahap ini mencari tahu kebutuhan dan masalah untuk menciptakan suatu sistem yang baru maupun memperbaiki sistem yang sudah ada. Dalam melakukan proses ini dibutuhkan beberapa pihak yang terkait seperti *programmer*, *user*, *stakeholder*, dan lain sebagainya. Hal ini bertujuan untuk menerima berbagai masukan dan pandangan terhadap kelebihan dan kekurangan pada sistem yang akan dibuat maupun yang akan diperbaharui dari masing-masing pihak.

2. Perencanaan

   Pada tahap ini akan *programmer* akan merencanakan berbagai persyaratan dalam mengembangkan sistem. Selain itu perlu memikirkan juga resiko yang akan dihadapi selama proses berlangsung termasuk dengan waktu yang digunakan dan menentukan biaya yang perlu dikeluarkan. Dalam menentukan waktu diperlukan evaluasi strategi yang akan digunakan dalam pengembangan sistem.

3. Rancangan

   Pada tahap ini akan dilakukan desain terkait sistem yang akan dibangun berdasarkan kebutuhan klien. Kemudian hasil desain akan diberikan kepada

klien untuk diperiksa ulang dan diberikan umpan balik. Hal ini dilakukan agar klien dapat melihat perkiraan hasil jadi ketika sistem tersebut dibuat.

4. Membangun atau mengembangkan produk

   Pada tahap ini *programmer* akan membuat dan membangun sistem berdasarkan desain yang telah dirancang pada tahap sebelumnya. Sistem yang dibangun menggunakan bahasa pemrograman dalam bentuk modul kecil yang nantinya akan digabungkan pada tahap akhir pembangunan sistem. Selain itu, tahap ini juga membutuhkan waktu yang cukup lama.

5. Pengujian

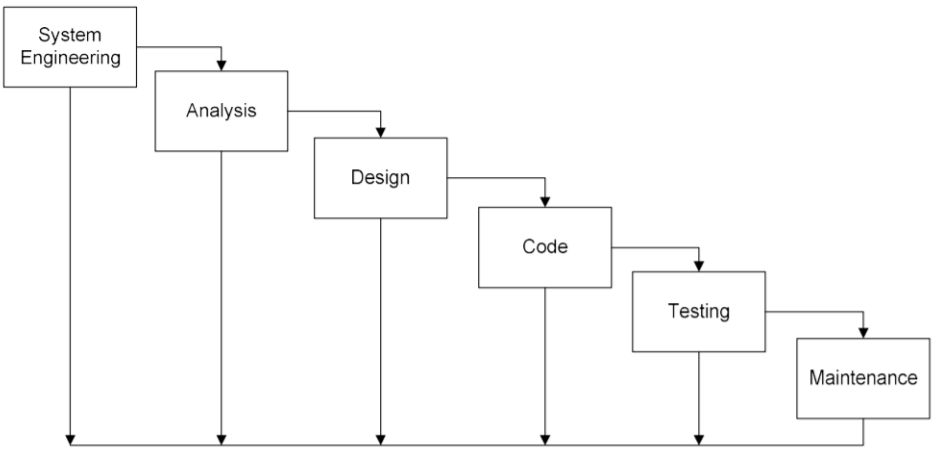
   Pada tahap ini, sistem telah dibangun dan masuk ke dalam tahap menguji sistem tersebut. Hal ini bertujuan agar dapat mengantisipasi adanya kesalahan pada sistem sebelum digunakan langsung oleh masyarakat. Di sisi lain juga menyesuaikan kembali sistem yang dibangun dengan permintaan dari klien.

6. Melakukan Pemeliharaan (*maintenance*)

   Pada tahap akhir, sistem dapat dioperasikan oleh masyarakat dan dilanjutkan ke proses pemeliharaan. Pemeliharaan dapat berupa perbaikan sistem atas kesalahan yang tidak terdeteksi pada tahap sebelumnya. Contoh dari pemeliharaan sistem meliputi perbaikan kesalahan pemrograman ataupun penyesuaian sistem berdasarkan kebutuhan masyarakat. Selain itu juga meningkatkan mutu dari sistem tersebut.

2.2 Model Pengembangan Waterfall

Model *waterfall* merupakan salah satu model pengembangan perangkat lunak pada model SDLC yang memiliki alur sekuensial linear. Dengan adanya model pengembangan ini, maka dapat membantu manajemen sistem agar mencapai hasil yang jelas, karena setiap langkah-langkah yang terdapat dalam model ini akan dijalankan secara berurutan. Urutan langkah tersebut dimulai dari analisis, desain, implementasi pemrograman, pengujian, dan *maintenance* [9].



Gambar 2.1. Model Pengembangan Waterfall

sumber: [10]

1. *System Engineering*

   System engineering merupakan sebuah proses interdisipliner yang akan memastikan kepuasan pengguna berdasarkan kebutuhan pengguna tersebut dalam keseluruhan siklus pada suatu sistem. Selain itu juga harus meningkatkan probabilitas keberhasilan dari sistem, mengurangi resiko dan mengurangi biaya yang harus dikeluarkan selama membangun sistem tersebut[11].

2. *Analysis* (Analisis kebutuhan)

   Melakukan analisa kebutuhan dari sistem yang nantinya akan dibangun dan dikembangkan. Kebutuhan tersebut dapat berupa kebutuhan perangkat keras dan perangkat lunak yang akan digunakan dalam membangun sebuah sistem, serta kebutuhan fungsional (kebutuhan yang nantinya akan membantu proses sistem tersebut berjalan). Kebutuhan fungsional bisa didapat dengan melakukan proses survei, wawancara, ataupun mengambil data dari pakar [9].

3. *System Design* (Desain sistem)

   Membuat *mockup* atau rancangan dari tampilan dari sistem sebelum sistem tersebut dibangun dan digunakan oleh *client*/pengguna. Dengan adanya tahap ini, membantu *programmer* dalam membangun sistem tersebut. Selain itu tampilan desain ini dibangun untuk memberikan pentunjuk dengan jelas

bagaimana nanti sistem tersebut akan dilihat dan berjalan. Desain dapat dilakukan secara digital maupun secara manual [9].

4. *Coding and Testing* (Implementasi)

   Pada tahap ini *programmer* akan membuat sistem berdasarkan *mockup* yang telah dibuat pada tahap sebelumnya. Implementasi ini dilakukan dengan menggunakan bahasa pemrograman yang telah ditetapkan juga. Setelah sistem telah berhasil dibuat, maka sistem tersebut akan diuji dan diperiksa terlebih dahulu, untuk mengantisipasi adanya kekurangan ataupun kesalahan dalam proses membangun sistem dari sisi kode pemrogramannya [9].

5. *Integration and Testing* (Penerapan / pengujian program)

   Melakukan pengujian terhadap website atau aplikasi yang sudah dibangun agar dapat mengetahui sistem tersebut sudah bisa digunakan oleh masyarakat luas atau terdapat kesalahan setelah dibangun. Hal-hal yang diuji adalah mencoba semua fitur yang tersedia di dalam sistem yang nantinya akan digunakan oleh masyarakat, seperti mengisi formulir, melihat data yang ditampilkan, dan lain-lain. Pengertian lainnya dari integration testing merupakan salah satu metode pengujian software ataupun aplikasi dengan cara membandingkan antara output yang sudah dibangun dengan input yang dihasilkan [9].

6. *Operation and Maintenance* (Pemeliharaan)

   Melakukan pengembangan dan pemeliharaan terhadap sistem yang telah dibangun terutama ketika sistem terdapat sebuah kesalahan. Hal ini dilakukan agar sistem dapat terus digunakan dan diperbaharui sehingga pengguna menjadi aman ketika menggunakannya. Pembaharuan dapat berupa menambahan fitur ataupun memperbaiki *error* yang terjadi selama pengguna menggunakan sistem tersebut [9].

2.3 Sistem Pendukung Keputusan

Sistem pendukung keputusan diciptakan pertama kali oleh Michael S. Scoott pada tahun 1970-an dengan istilah *Management Decision System*. Sistem ini dibuat untuk mendukung seluruh tahap pengambilan keputusan nilai dari mencari masalah, memilih data yang sesuai, dan menemukan pendekatan yang dipakai dalam proses pengambilan keputusan, sampai mengevaluasi pemilihan alternatif (produk)

[12]. Pemahaman lainnya tentang sistem pendukung keputusan adalah sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data. Sistem ini digunakan untuk membantu pengambilan keputusan dalam sitasi yang semistruktur dan situasi yang tidak terstruktur, dimana tidak seorang pun mengetahui secara pasti bagaimana keputusan seharusnya dibuat [13].

2.4 Algoritma Visekriterijumsko Kompromisno Rangiranje (VIKOR)

VIKOR (Vise Kriterijumska Optimizajica I Kompromisno Resenje) berarti *multi-criteria optimization and compromise solution* (optimasi multi kriteria dan solusi kompromis), merupakan salah satu dari sekian banyak teknik MCDM. VIKOR diperkenalkan pertama kali oleh Serafim Opricovic pada tahun 1998. Metode VIKOR adalah metode perankingan dengan menggunakan indeks peringkat multikriteria berdasarkan ukuran tertentu dari kedekatan dengan solusi yang ideal. Konsep dasar VIKOR adalah menentukan ranking dari sampel yang ada dengan melihat hasil dari nilai-nilai utilitas dengan regrets dari setiap sampel. Langkah-langkah dari metode VIKOR ini sebagai berikut [6][14].

1. Rumus normalisasi

   Jika tipe kriteria adalah *benefit*:

$$R_{ij} = \left(\frac{X_j^+ - X_{ij}}{X_j^+ - X_j^-} \right) \tag{2.1}$$

   Jika tipe kriteria adalah *cost*:

$$R_{ij} = \left(\frac{X_{ij} - X_j^-}{X_j^+ - X_j^-} \right) \tag{2.2}$$

   Keterangan:

   - R_{ij} dan X_{ij} adalah elemen dari matriks pengambilan keputusan
   - X_{ij}^+ adalah elemen terbaik dari kriteria j
   - X_{ij}^- adalah elemen terburuk dari kriteria j
   - i adalah alternatif

- j adalah kriteria

2. Rumus menghitung nilai S dan R

$$S_i = \sum_{j=1}^{n} W_j \left(\frac{X_j^+ - X_{ij}}{X_j^+ - X_j^-} \right) \tag{2.3}$$

dan

$$R_i = Max j \left[W_j \left(\frac{X_j^+ - X_{ij}}{X_j^+ - X_j^-} \right) \right] \tag{2.4}$$

Keterangan:

- S_i atau R_i preferensi alternatif yang dianalogikan sebagai vektor V
- X adalah nilai dari kriteria
- W adalah bobot dari kriteria
- i adalah alternatif
- j adalah kriteria
- n adalah banyaknya kriteria yang disungguhkan

3. Rumus untuk menentukan nilai indeks

$$Q_i = \left(\frac{S_i - S^-}{S^+ - S^-} \right) (v) + \left(\frac{R_i - R^-}{R^+ - R^-} \right) (1 - v) \tag{2.5}$$

Keterangan:

- S^- merupakan min S_i
- S^+ merupakan max S_i
- R^- merupakan min R_i
- R^- merupakan max R_i

- v merupakan 0.5

Hasil ranking adalah urutan nilai dari S, R, dan Q
Solusi alternatif peringkat terbaik berdasarkan nilai Q minimum menjadi peringkat terbaik dengan syarat:

$$Q(A_2) - Q(A_1) \geq \left(\frac{1}{m-1}\right) \tag{2.6}$$

Keterangan:

- A_2 adalah alternatif dengan urutan kedua pada perangkingan Q
- A_1 adalah alternatif dengan urutan terbaik pada perangkingan Q
- m adalah jumlah alternatif

Alternatif $A_{(1)}$ harus berada pada ranking terbaik pada S dan/atau R

2.5 End User Computing Satisfaction (EUCS)

End User Computing Statisfaction adalah penilaian atas semua sistem informasi atau aplikasi yang dijalankan oleh pelanggan sebuah sistem atau aplikasi yang terkait dengan kemahiran penggunaan aplikasi tersebut. Kemahiran pemanfaatan aplikasi tersebut dihitung untuk mendapatkan informasi apakah aplikasi tersebut berdaya guna dan cocok seperti yang diharapkan[15].

End User Computing Satisfaction merupakan sebuah sistem informasi adalah yang akan mengevaluasi secara keseluruhan dari pengguna sistem informasi yang berdasarkan pengalaman mereka dalam menggunakan sistem tersebut. Evaluasi dengan menggunakan model ini lebih menekankan kepuasan (*satisfaction*) pengguna akhir terhadap aspek teknologi, dengan isi, keakuratan, format, waktu dan kemudahan penggunaan dari sistem. Berikut ini adalah kategori yang diukur dengan metode *End User Computing Satisfaction*, yaitu[16].

1. *Content*

   Kategori ini mengukur kepuasan pengguna ditinjau dari sisi isi dari suatu sistem. Isi dari sistem berupa fungsi dan modul yang dapat digunakan oleh

pengguna dan juga informasi yang dihasilkan oleh sistem. Salah satu hal yang diukur adalah apakah sistem tersebut menghasilkan informasi yang sesuai dengan kebutuhan pengguna. Semakin lengkap modul dan informasi sistem tersebut, maka akan semakin meningkat juga kepuasan dari pengguna[16].

2. *Accuracy*

   Kategori ini mengukur kepuasan pengguna dari sisi keakuratan data ketika sistem menerima input kemudian mengolah menjadi informasi. Keakuratan sistem diukur dengan melihat seberapa sering sistem menghasilkan *output* yang salah ketika mengolah *input* dari pengguna. Selain itu dapat dilihat pula seberapa sering terjadi *error* atau kesalahan dalam proses pengolahan data[16].

3. Format

   Kategori ini mengukur kepuasan pengguna dari sisi tampilan dan estetika dari antarmuka sistem. Format dari laporan atau informasi yang dihasilkan oleh sistem apakah anatarmuka dari sistem tersebut menarik dan apakah sistem memudahkan pengguna ketika menggunakan sistem. Sehingga secara tidak langsung dapat berpengaruh terhadap tingkat efektifitas dari pengguna[16].

4. *Ease of Use*

   Kategori ini mengukur kepuasan pengguna dari sisi kemudahan pengguna (*user friendly*) dalam menggunakan sistem. Contoh dari kategori ini adalah proses memasukan data, mengelolah data, dan mencari informasi yang dibutuhkan[16].

5. *Timeliness*

   Kategori ini mengukur kepuasan pengguna dari sisi ketepatan waktu sistem dalam menyediakan data atau informasi yang dibutuhkan oleh pengguna. Sistem yang tepat waktu dapat dikategorikan sebagai sistem *real-time*. Contoh dari kategori *timeliness* adalah setiap permintaan atau input yang dilakukan oleh pengguna akan langsung diproses dan *output* akan ditampilkan secara cepat tanpa harus menunggu lama. Kategori ini juga berdampak pada penerapan aplikasi[16].

2.6 Skala Likert

Skala likert merupakan salah satu metode yang dapat mengukur sikap, pendapat, dan persepsi seseorang atau kelompok terkait fenomena sosial. Pengukuran tersebut diawali dengan meminta pengguna menjawab beberapa pertanyaan dengan pilihan skala dari yang bernilai positif sampai dengan negatif. Semakin positif jawaban tersebut, akan semakin besar juga nilai / skor yang diberikan[17]. Nilai tersebut dapat dilihat pada Tabel 2.1.

Tabel 2.1. Tabel Nilai Likert

Skala	Nilai Likert
Sangat Tidak Setuju (STS)	5
Tidak Setuju (TS)	4
Netral (N)	3
Setuju (S)	2
Sangat Setuju (SS)	1

Cara untuk mengukur hasil survei dari responden, dapat menggunakan Rumus 2.7 yang berguna untuk menghitung total skor dari jawaban tersebut, serta Rumus 2.8 yang bertujuan untuk menghitung nilai persentasenya. Apabila telah mendapatkan nilai persentasenya, maka akan memperoleh keterangan dari nilai tersebut berdasarkan tabel interval dari skala likert itu sendiri. Tabel interval tersebut dapat dilihat pada Tabel 2.2.

$$Total\ Skor = T\ \times\ P_n \tag{2.7}$$

$$\text{Index}\% = \frac{Total\ Skor}{Y}\ x\ 100\% \tag{2.8}$$

Berikut ini adalah keterangan dari Rumus 2.7 dan Rumus 2.8.

- T adalah jumlah responden
- P_n = Nilai Likert
- Y = Skor Likert Tertinggi x Jumlah Responden x Jumlah Pertanyaan

Tabel 2.2. Tabel Interval Persentase Skala Likert

No	Tingkatan	Persentase
1	Sangat Tidak Setuju (STS)	0% s/d 19.99%
2	Tidak Setuju (TS)	20% s/d 39.99%
3	Netral (N)	40% s/d 59.99%
4	Setuju (S)	60% s/d 79.99%
5	Sangat Setuju (SS)	80% s/d 100%

2.7 Smartwatch

Smartwatch adalah perangkat pintar yang memiliki kemampuan lebih daripada jam tangan biasa. Smartwatch memiliki fitur *bluetooth* agar jam tangan dapat terintegrasi dengan *smartphone*. Cara pemakaian smartwatch adalah menggunakan antarmuka arloji untuk memulai dan menjawab panggilan telepon dari ponsel, membaca pesan, mendapatkan laporan cuaca, mendengarkan musik. Smartwatch juga memiliki tujuan tertentu seperti mengumpulkan data tentang kesehatan pengguna, memantau detak jantung, dan menyediakan fitur GPS [18].

2.8 Blackbox Testing

Blackbox testing merupakan metode yang digunakan dalam menguji sebuah sistem / program, dimana proses melakukan pengujian tersebut tidak memperhatikan detail kode pemrograman dari sistem tersebut. Pengujian sistem diperlukan agar sistem yang telah dibangun sama dengan kebutuhan / tujuan dari penelitian ataupun perusahaan. Apabila tidak sama dengan kebutuhan diawal, maka perlu dilakukan evaluasi agar nantinya dilakukan perbaikan pada sistem tersebut. Tahap yang dilakukan dari metode ini adalah mencoba sistem yang telah dibangun dengan memasukan data disetiap formulir yang tersedia [19]. Terdapat beberapa pendekatan dalam penguji sistem dari metode ini adalah sebagai berikut.

1. Decision Table Technique

   Pendekatan ini akan membuat berbagai kemungkinan dalam bentuk tabel keputusan. Setiap kombinasi input ataupun beberapa kemungkinan dari sistem akan diperiksa. Hal ini sesuai untuk fungsi yang memiliki hubungan logis antara dua dan lebih dari dua input [20].

2. Boundary Value Technique

   Boundary Value Technique sering digunakan oleh peneliti, karena menjadi dasar dalam pengujian kesetaraan. Pendekatan ini digunakan untuk mengindentifikasi kekurangan atau kesalahan yang timbul akibat keterbatasan input data. Keterbatasan yang dimaksud merupakan nilai yang berisi batasan atas maupun batasan bawah suatu variabel. Pada saat memasukan nilai batasan, maka pendekatan iniakan memeriksa apakah sistem menghasilkan hasil yang benar atau tidak [20].

3. State Transition Technique

   State Transition Technique merupakan pendekatan yang menggunakan *input, output*, dan keadaan sistem selama fase pengujian. Dimana pendekatan ini akan memeriksa sistem terhadap urutan transisi ataupun peristiwa pada data uji. Perubahan pada sistem akan diperiksa dalam keadaan lain sembari mempertahankan input yang sama [20].

4. Equivalence Partitioning Technique

   Equivalence Partitioning Technique merupakan salah satu pendekatan pengujian yang akan memeriksa *input* dan *output* dengan membagi *input* ke dalam kelas yang setara. Data tersebut akan diuji sebanyak satu kali atau lebih untuk memastikan cakupan pengujian data yang maksimal. Pengujian dengan pendekatan ini sangat lengkap dan dapat mengantisipasi adanya redundansi input [20].

5. Error Guessing Technique

   Error Guessing Technique merupakan pengujian yang menebak hasil yang dikeluarkan dan yang dimasukan untuk memperbaiki kesalahan yang kemungkinan berasal dari dalam sistem. Hal ini di dasari oleh pengalaman analis uji dari penguji untuk menebak area yang bermasalah dari sistem tersebut [20].