BAB II
KAJIAN LITERATUR

1.1 Pengembangan Sistem

Berikut ini, akan dilakukan pembahasan mengenai teori-teori yang berkaitan dengan pengembangan sistem antara lain:

2.1.1 Pengertian Pengembangan Sistem

Terdapat beberapa definisi pengembangan dari beberapa sumber-sumber antara lain sebagai berikut:

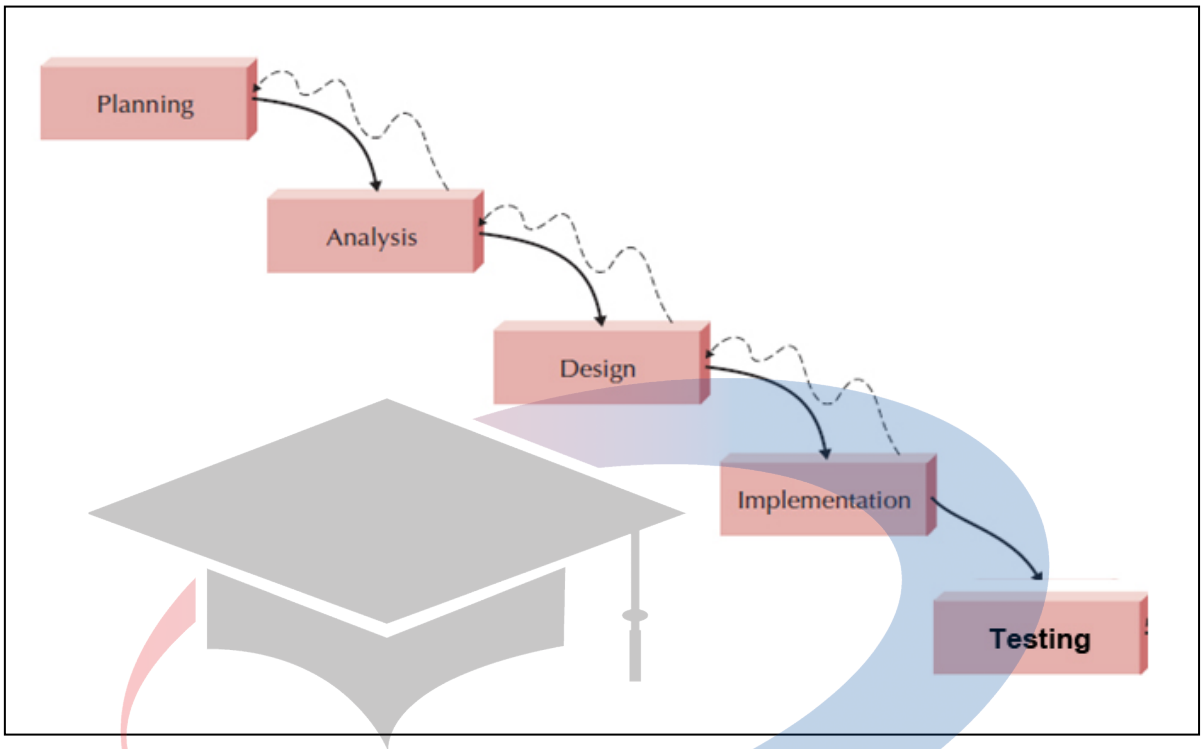
1. Pengembangan sistem adalah proses merancang, membangun, dan memelihara sistem informasi atau perangkat lunak yang efektif dan efisien, sehingga dapat mengoptimalkan kinerja bisnis atau organisasi [12].
2. Pengembangan sistem melibatkan tahap-tahap seperti tahap perencanaan, tahap analisis, tahap desain, tahap implementasi dan tahap penggunaan [13].
3. Pengembangan sistem bertujuan untuk mengembangkan sistem yang sesuai dengan kebutuhan bisnis atau organisasi, dapat mengatasi masalah atau tantangan, dan dapat memberikan manfaat atau keuntungan secara signifikan [14].
4. Pengembangan sistem memerlukan keterampilan dan pengetahuan di berbagai bidang, termasuk teknologi informasi, manajemen proyek, desain antarmuka pengguna, pengujian, dan pemeliharaan sistem [15].

2.1.2 Tahapan Pengembangan Sistem

Tahapan pengembangan sistem adalah serangkaian proses yang dilakukan dalam mengembangkan sistem informasi atau perangkat lunak dari awal hingga selesai. Terdapat beberapa metode pengembangan sistem, namun yang umum digunakan yaitu adalah metode *Waterfall.* Dalam penelitian ini, *website* sistem informasi pencarian dan pemesanan jasa pemandu wisata yang akan dibangun menggunakan metode *Waterfall Model.*

Waterfall Model sering disebut sebagai metode air terjun. Dengan metode berbasis pengembangan air terjun, para analis dan pengguna melanjutkan secara berurutan dari satu fase ke fase berikutnya. Kunci kiriman untuk setiap fase biasanya sangat panjang (seringkali ratusan halaman) dan disajikan kepada sponsor proyek untuk mendapatkan persetujuan saat proyek berpindah dari fase ke fase. Setelah sponsor menyetujui pekerjaan yang dilakukan untuk suatu fase, fase tersebut berakhir dan selanjutnya dimulai. Metode ini disebut

pengembangan air terjun karena bergerak maju dari fase ke fase dengan cara yang sama seperti air terjun [16].



Gambar 2.1 Tahapan Metode Waterfall Model [16]

Berikut ini, akan diuraikan tahapan-tahapan dari metode *Waterfall Model* yang ditunjukkan pada Gambar 2.1 antara lain:

1. *Planning*/Perencanaan

   Di dalam tahapan ini, persyaratan potensial dari aplikasi di analisis ke dalam dokumen spesifik yang berfungsi sebagai dasar untuk semua pengembangan di masa mendatang. Proses ini akan menghasilkan dokumen persyaratan yang menentukan apa yang harus dilakukan aplikasi, bukan bagaimana cara melakukannya.

2. *Analysis*/Analisis

   Selama tahap kedua, sistem akan dianalisis untuk menghasilkan model dan logika bisnis yang akan digunakan dalam aplikasi.

3. *Design*/Desain

   Tahap ketiga ini biasanya mencakup kepentingan desain teknis, seperti bahasa pemrograman, lapisan data, layanan, dan sebagainya. Spesifikasi desain biasanya akan dibuat untuk menguraikan bagaimana logika bisnis yang tercakup dalam analisis akan diimplementasikan secara teknis.

4. *Implementation*/Implementasi

   Dari tahap ketiga dan kedua, didapatkan sebuah rancangan sistem secara teknis, kemudian akan dilakukan implementasi dengan membangun aplikasi menggunakan bahasa-bahasa pemrograman yang dipilih.

5. *Testing*/Pengujian

   Tahap terakhir adalah melakukan *testing* terhadap aplikasi yang telah dibangun dan diimplementasikan agar dapat diperbaiki *bug* ataupun *error* yang terdapat pada aplikasi.

2.2 *Website*

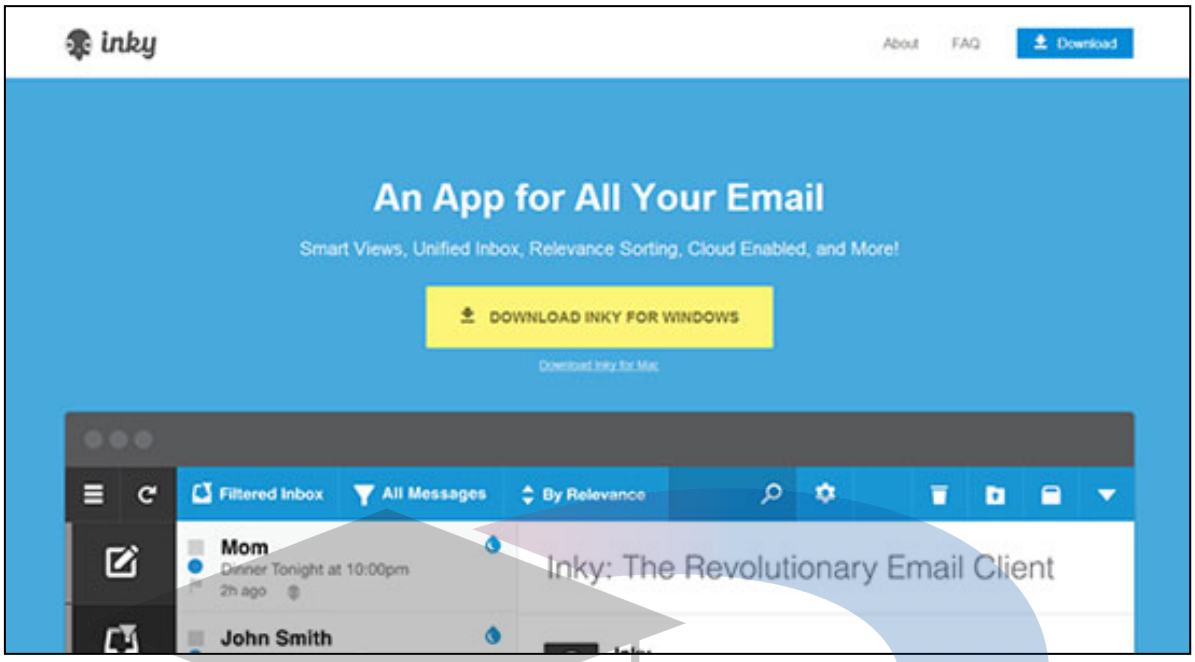
Website merupakan kumpulan halaman-halaman yang dapat menampilkan teks, gambar, animasi, video, suara yang masing-masing dihubungkan dengan jaringan-jaringan halaman. Selain itu, *website* juga sering disebut sebagai suatu sistem yang berkaitan dengan dokumen digunakan sebagai media untuk menampilkan teks, gambar, *multimedia,* dan lainnya pada jaringan internet [17]. Terdapat beberapa ciri-ciri dari sebuah *website* yaitu [17]:

1. Hanya diinstall pada sebuah *web server* saja. Jika ada *update*, maka cukup melakukan *update* pada *server* saja.
2. *Update* yang dilakukan pada *server* akan segera digunakan oleh *user* yang menggunakan aplikasi *web* tersebut.
3. Aplikasi *web* mudah digunakan untuk digunakan oleh *user* yang banyak.
4. Aplikasi *web* tidak bergantung pada sistem operasi, karena hanya memerlukan *web browser*.

Secara umum, berdasarkan jenisnya sebuah *website* terbagi menjadi 2 yaitu sebagai berikut [17]:

1. *Website* statis adalah *website* yang biasanya *user* tidak bisa mengubah *content* dari *web* tersebut secara langsung menggunakan *browser*. Interaksi yang terjadi hanya seputar pemrosesan *link* yang ada.
2. *Website* dinamis merupakan *website* yang biasanya *user* dapat mengubah *content* dari halaman tertentu dengan menggunakan *browser*.

Berikut ini gambar 2.2 menunjukkan contoh tampilan depan dari sebuah *website* [17].



Gambar 2.2 Contoh Tampilan Depan Sebuah Website [17]

2.3 Pariwisata

Kegiatan wisata merupakan aktivitas wisatawan yang dilakukan dengan dinamis sehingga menumbuhkan berbagai kebutuhan wisata di manapun berada. Pelaku usaha wisata berusaha memenuhi kebutuhan tersebut dan melakukan aktivitasnya pada saat yang bersamaan sehingga merupakan sebuah industri pariwisata. Industri Pariwisata sendiri merupakan kesatuan dari beragam jenis perusahaan yang bekerja sama menyediakan barang-barang dan jasa *(goods and service)* kepada wisatawan selama melakukan lawatan rekreasi dari awal hingga selesai [18].

Pariwisata merupakan aktivitas perjalanan yang dilakukan pada waktu berlibur karena rasa ingin tahu atau ingin menambah pengetahuan. Industri Pariwisata memiliki lingkup sebagai berikut [18]:

1. Daya tarik wisata.

   Klasifikasi daya tarik wisata ada dua yaitu site atraksi (objek wisata) dan event atraksi (atraksi wisata). Site atraksi berbentuk wisata alam dan buatan yang tidak dapat dipindah-pindahkan, sedangkan event atraksi merupakan aktivitas yang bisa berpindah tempat seperti halnya atraksi tarian budaya. Daya Tarik Wisata yang disingkat DTW, merupakan suatu hal yang unik dan bernilai tinggi yaitu aneka ragam sumber daya alam, budaya serta karya kreatif yang diminati wisatawan. Yoeti (1985) mengistilahkan dengan "*tourist attraction*", yaitu semua hal yang menarik dan menimbulkan minat untuk berkunjung ke lokasi tersebut. Sedangkan menurut Pendit (2003) seperti

penuturannya bahwa daya tarik wisata merupakan suatu hal yang memikat hati wisatawan.

2. Fasilitas wisata.

   Wisatawan tetap membutuhkan fasilitas meskipun sedang rekreasi di luar rumah. Fasilitas yang harus tersedia di lokasi rekreasi setidaknya memenuhi kebutuhan makan, minum, tempat istirahat, mandi, beribadah dan sebagainya.

3. Infrastruktur.

   Lingkup infrastruktur merujuk pada kondisi fisik fasilitas umum yang mendukung aktivitas ekonomi dan kehidupan sehari-hari masyarakat setempat, seperti jalan, jembatan, terminal dan sebagainya. Kategori infrastruktur meliputi keras dan keras non fisik seperti aliran air bersih, aliran listrik serta lunak berupa regulasi pemerintah. Kawasan pariwisata merupakan suatu wilayah dengan luasan tertentu yang memiliki fungsi utama sebagai tempat tujuan wisata dan memerlukan infrastruktur yang berhubungan erat dengan pariwisata.

4. Angkutan.

   Angkutan berkaitan erat dengan akses menuju lokasi rekreasi.

5. Keramahan.

   Syarat utama agar wisatawan betah berlama-lama di lokasi wisata adalah diperolehnya pengalaman yang mengesankan dan kenyamanan yang dirasakan merupakan kenangan yang tak terlupakan.

Industri pariwisata memiliki klasifikasi unsur antara lain [18]:

1. Jasa transportasi wisata.

   Layanan yang menyediakan angkutan bukan umum karena hanya dipergunakan untuk aktivitas wisata.

2. Jasa Perjalanan Wisata.

   Layanan yang disediakan selama melakukan lawatan wisata.

3. Jasa Makanan dan Minuman.

   Layanan konsumsi selama dalam lawatan rekreasi.

4. Penyediaan Akomodasi.

   Layanan kebutuhan wisatawan seperti tempat menginap atau beristirahat sejenak, tempat menunaikan ibadah di lokasi rekreasi.

5. Penyelenggaraan Kegiatan Hiburan, Rekreasi.

   Pelayanan kepada wisatawan berupa penyelenggaraan event dan tempat pertunjukkan kesenian sebagai aktivitas pariwisata.

6. Penyelenggaraan Pertemuan, Perjalanan Insentif, Konferensi, Pameran.
   Layanan MICE yaitu *Meeting, Incentive, Convention, Exhibition*.
7. Jasa Informasi.
   Pariwisata Layanan penyediaan berita mengenai pariwisata.
8. Jasa Konsultan.
   Pariwisata Layanan pendampingan atau pemberian masukan terkait kepariwisataan.
9. Jasa Pemandu Wisata.
   Layanan dari pemandu atau guide untuk keperluan pendampingan wisatawan selama perjalanan rekreasi.
10. Wisata Tirta.
    Layanan rekreasi yang berhubungan dengan air seperti olahraga air yang bisa dilakukan di pantai atau danau juga kolam renang.
11. Spa.
    Layanan *body treatment* menggunakan terapi untuk kesehatan dengan mengutamakan kenyamanan wisatawan.

Berikut ini, Gambar 2.3 menunjukkan salah satu destinasi pariwisata Orchid Forest Cikole.

Gambar 2.3 Destinasi Pariwisata Orchid Forest Cikole [18]

Gambar 2.4 Destinasi Pariwisata Pantai Drini [18]

2.4 Pemandu Wisata

Pemandu wisata adalah seseorang yang memberi penjelasan serta petunjuk kepada wisatawan dan *traveller* lainnya tentang segala sesuatu yang hendak dilihat dan disaksikan bilamana mereka berkunjung pada suatu objek, tempat atau daerah wisata tertentu. Pemandu wisata adalah orang-orang yang memang berfokus dalam membangun pariwisata antara lain lebih banyak mengetahui seluk beluk pariwisata dan kebutuhan wisatawan serta memberikan pelayanan pariwisata yang berkualitas kepada wisatawan [18].

Pemandu wisata atau sering dikenal dengan *tour guide* dalam bahasa internasional adalah seseorang yang dibayar untuk menemani wisatawan dalam perjalanan mengunjungi, melihat serta menyaksikan objek dan atraksi wisata sedangkan dari sudut pandang wisatawan pemandu wisata adalah seseorang yang bekerja pada suatu biro perjalanan atau suatu kantor pariwisata *(tourism office)* yang bertugas memberikan informasi, petunjuk secara langsung kepada wisatawan sebelum dan selama perjalanan berlangsung dan orang yang di anggap serba tahu oleh para wisatawan yang dapat menjadi guru sekaligus teman dalam perjalanannya [19].

Menurut Kementrian Pendidikan Dan Kebudayaan Pusat Pengembangan Pendidikan Anak di Usia Dini Dan Pendidikan Masyarakat Jawa Barat tahun 2019 yang berjudul Kursus dan Pelatihan Bidang Kepemanduan Wisata Melalui Pemagangan bahwa pelayanan seorang pemandu wisata terhadap tamu dapat saja dimulai dari penjemputan di bandara, atau di

stasiun kereta api, atau di terminal bus atau dititik penjemputan yang telah disepakati bersama, kemudian diantara ke hotel atau penginapan lainnya.

Pemandu wisata memiliki peranan yang sangat penting karena selama dalam masa liburan wisatawan lebih banyak bersinggungan atau beradaptasi dengan pemandu wisata. Baik buruknya kesan yang diterima wisatawan banyak ditentukan oleh peran seorang pemandu wisata dalam mempromosikan produk wisata dan mendampingi wisatawan saat berkunjung ke suatu objek wisata yang diinginkan.

2.5 Metode Penghitungan Jarak

2.5.1 Metode *Euclidean Distance*

Euclidean Distance merupakan salah satu metode perhitungan jarak yang digunakan untuk mengukur jarak dari 2 buah titik dalam *euclidean space* (meliputi bidang *Euclidean* dua dimensi, tiga dimensi atau bahkan lebih) [20]. Dalam matematika, jarak *Euclidean* untuk mengukur dua titik dalam satu dimensi, menghasilkan hasil yang mirip dengan perhitungan Pythagoras [8]. Rumus *Euclidean Distance* untuk mengukur tingkat kemiripan data adalah sebagai berikut [20]:

$$d(x,y) = |x - y| = \sqrt{\sum_{i=1}^{N}(xi - yi)^2} \quad (1)$$

Keterangan:

d = jarak antara x dan y

x = data pusat klaster

y = data pada atribut

i = setiap data

n = jumlah data

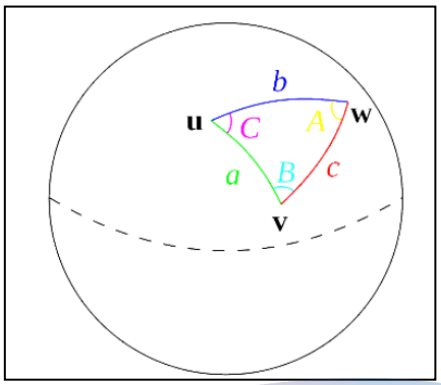
xi = data pada pusat klaster ke i

yi = data pada setiap data ke i

2.5.2 Metode *Haversine*

Metode *Haversine* adalah rumusan yang diterapkan dalam navigasi dengan meggunkana nilai jarak lingkaran besar antara dua titik pada permukaan bola bumi berdasarkan bujur dan lintang. Metode *Haversine* digunakan untuk mengukur jarak antar dua titik dengan menganalogikan bahwa bumi bukanlah sebuah bidang datar namun adalah sebuah bidang yang memiliki nilai derajat kelengkungan. Penggunaan rumus ini, cukup akurat untuk sebagian besar perhitungan dengan pengabaian efek ellipsoidal, ketinggian

bukit, dan kedalaman lembah di permukaan bumi [21]. Berikut ini, Gambar 2.5 menunjukkan gambaran implementasi dari metode *Haversine* [22].



Gambar 2.5 Metode *Haversine*

Untuk mencari jarak antara lokasi pengguna dan tujuan lokasi, perhitungan ini dipengaruhi oleh a derajat kelengkungan tertentu. dengan melakukan perhitungan sebagai berikut [22]:

$$x = (lon2 - lon1)\cos(\frac{lat1-lat2}{2}) \quad (2)$$

$$y = lat2 - lat1 \quad (3)$$

$$d = \sqrt{(x * x) + (y * y)R} \quad (4)$$

Keterangan:

R = jari-jari bumi adalah 6,371 (km)

1 *degree* = (0,0174532925 Radians)

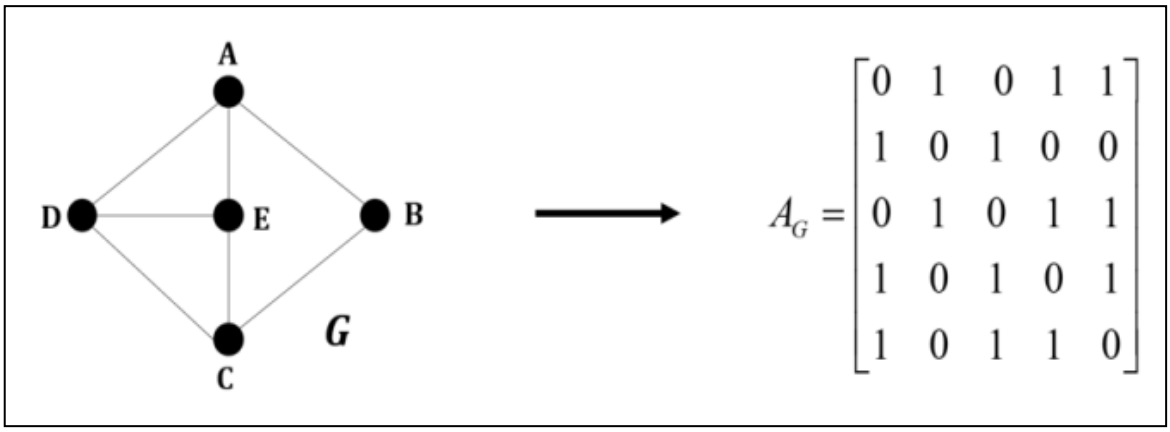
x = *latitude*

y = *longtitude*

d = *distance* (km)

2.6 Graf

Teori Graf merupakan bagian dari ilmu matematika dan komputer mengenai suatu graf struktur matematika, yang dinyatakan dalam G dimana G terdiri atas himpunan V yang berisikan titik (*vertex* atau *node*) [23]. Graf merupakan himpunan objek-objek yang berhingga dan tak kosong, disebut dengan simpul dan himpunan pasangan tak berurut (yang mungkin kosong) dari simpul-simpul tersebut, yang disebut dengan sisi. Gambar 2.6 merupakan contoh dari graf G dengan 5 simpul dan 7 sisi dengan himpunan simpulnya {A, B, C, D, E} dan himpunan sisi {(A,B), (A,D), (A,E), (B,C), (C,D), (C,E), (C,D)}. Simpul pada suatu graf memiliki derajat simpul, yaitu banyak sisi yang melekat pada simpul tersebut. Pada graf G, derajat simpul A, B, C, D, dan E berturut-turut adalah 3, 2, 3, 3, 3 [24].



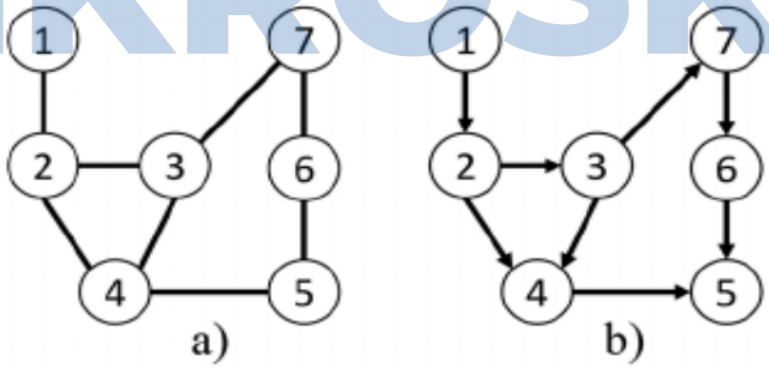
Gambar 2.6 Gambaran Mengenai Graf [24]

Graf juga dapat direpresentasikan dalam bentuk matriks. Pada Gambar 2.6, matriks AG merupakan matriks ketetanggan dari graf G. Matriks ketetanggan adalah matriks yang baris dan kolomnya disusun berdasarkan urutan simpulnya dengan pemberian 1 jika dua simpul terhubung oleh sisi, dan 0 jika sebaliknya [24].

Berdasarkan orientasi arahnya, sebuah graf dapat dikelompokkan menjadi beberapa kategori yaitu [25] [26]:

1. Graf berarah adalah graf yang sisinya mempunyai orientasi arah yang jelas. Pada graf berarah ini dinyatakan 2 arah yang berbeda (vj,vk) tidak sama dengan (vk,vj).
2. Graf tidak berarah adalah graf yang sisinya tidak mengandung arah. Urutan pasangan simpul yang dihubungkan oleh sisi diabaikan maka (vj,vk) sama dengan (vk,vj).

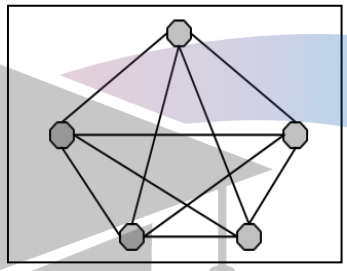
Berikut ini akan ditunjukkan contoh Gambar 2.7 menunjukkan graf berarah dan graf tidak berarah [27].



Gambar 2.7 a) Graf Tidak Berarah dan b) Graf Berarah [27]

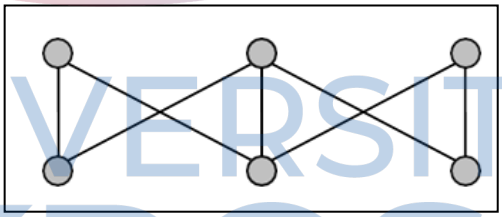
Sebuah struktur graf bisa dikembangkan dengan memberi bobot atau nilai pada tiap *edge* di mana merupakan suatu nilai yang dapat berupa biaya atau jarak, graf semacam ini disebut graf berbobot (*weighted graph)* [28]. Dalam pengajaran teori graf. terdapat graf khusus beberapa diantaranya adalah sebagai berikut :

1. *Complete Graph* adalah graf di mana setiap verteks berhubungan dengan semua verteks yang lain (semua verteks saling berhubungan). Biasanya direpresentasikan dengan simbol K_n, dimana K adalah *complete graph* dan *n* jumlah verteks. Sebuah *complete graph* dengan *n* verteks akan mempunyai rusuk sebanyak $n(n-1)/2$.



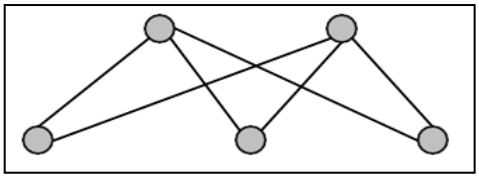
Gambar 2.8 Contoh Model *Complete Graph* [29]

2. *Bipartite Graph* adalah graf dimana satu verteksnya dibagi kedalam dua subset verteks *m* dan *n*, sedemikian sehingga tidak ada rusuk yang menyebabkan verteks-verteks dalam *subset* yang sama. Biasanya direpresentasikan dengansimbol $K_{m,n}$, di mana K adalah *bipartite graph*, dan *m* adalah jumlah sunset verteks *m*, dan *n* adalah jumlah *subset n*.



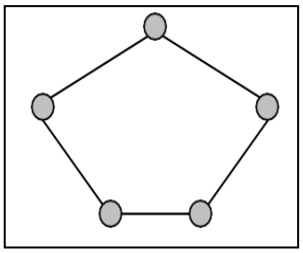
Gambar 2.9 Contoh Model *Bipartite Graph* [30]

3. *Complete Bipartite Graph* adalah *Bipartite Graph* di mana setiap verteks dari *m* harus memiliki rusuk yang berhubungan ke semua verteks dari *n*. Biasanya direpresentasikan dengan simbol $K_{m,n}$, sama seperti *Bipartite Braph*.



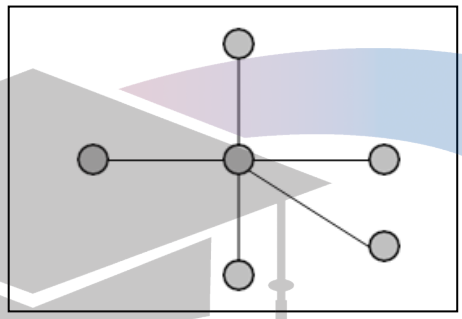
Gambar 2.10 Contoh Model *Complete Bipartite Graph* [30]

4. *Reguler Graph* adalah graf dimana setiap verteksnya memiliki derajat yang sama.



Gambar 2.11 Contoh Model *Reguler Graph* [31]

5. *Tree* adalah graf yang tidak memiliki *cycle*. Jika jumlah verteks pada *tree* adalah n, maka jumlah rusuk pada *tree* adalah n-1.



Gambar 2.12 Contoh Model *Tree Graph* [32]

Berikut ini proses sebuah graf dibentuk untuk menggambarkan lokasi-lokasi dalam menyelesaikan persoalan rute terpendek [33]:

1. Gabungkan tiap titik dari setiap rute menjadi sebuah *connected graph* (graf terhubung).
2. Berikan arah lintasan pada rute sebagai aliran *(flow)* sehingga terbentuk suatu *directed graph* dari *connected graph* yang ada.
3. Informasi dari jarak tempuh yang didapatkan diubah menjadi bobot jarak. Implementasikan bobot jarak yang ada menjadi sebuah aliran beban *directed graph* sehingga menjadi sebuah *weighted graph* (graf berbobot).

Berikut ini representasi dari graf antara lain [33]:

1. *Adjacency Matrix*

   Sebuah matriks digunakan untuk menunjukkan *adjacency set* dari setiap verteks dalam baris dan kolomnya. Dimana baris pada matriks menunjukkan nomor verteks adjacency berasal sedangkan kolom pada matriks menunjukkan nomor verteks kemana arah *adjacency*. Elemen matriks [x,y] bernilai 1 apabila terdapat sisi dari x ke y, namun bernilai 0 apabila terdapat lintasan yang berawal dan berakhir pada simpul yang sama.

2. *Adjacency List*

   Pada *adjacency list* representasi matriksnya berupa matriks sparse, dimana sebagian besarnya berisikan bilangan nol. Untuk efisiensi ruang, tiap baris pada matriks

digantikan list yang hanya terdapat verteks-verteks dalam *adjacency set* Vx dari setiap verteks x.

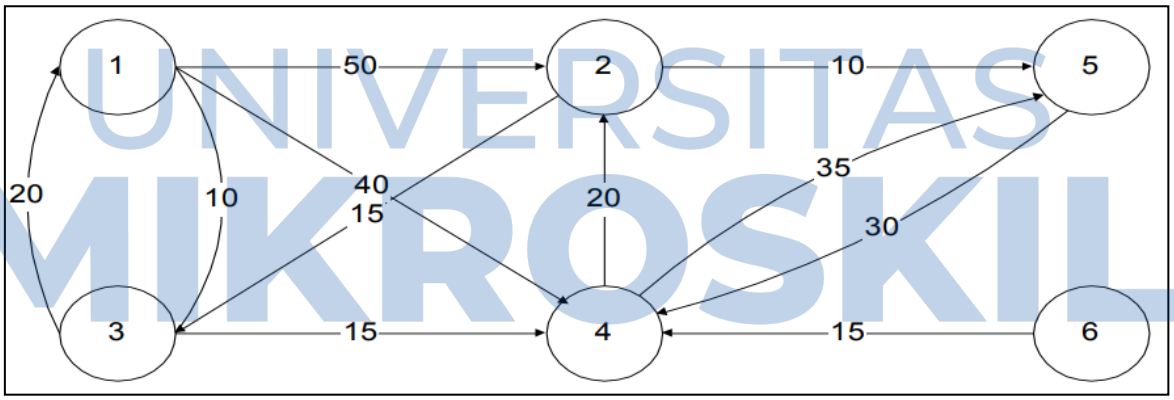
2.7 *Shortest Path*

Shortest Path Problem adalah sebuah permasalahan untuk menemukan sebuah jalan antara 2 *vertex*, *dimana* weight dari unsur pokok diminimalisasi. Salah satu contoh dari *Shortest Path Problem* yang paling sering dibahas adalah *Travelling Salesman Problem* dimana permasalahan tersebut adalah permasalahan untuk menemukan jarak terpendek untuk memutari semua *vertex* yang ada dan kemudian kembali pada titik awal pencarian rute [33].

Terdapat beberapa jenis persoalan lintasan terpendek, antara lain [34]:

1. Lintasan terpendek antara dua buah simpul tertentu.
2. Lintasan terpendek antara semua pasangan simpul.
3. Lintasan terpendek dari simpul tertentu ke semua simpul yang lain.
4. Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu.

Suatu uraian pemecahan persoalan tentang graf berbobot G = (V,E) dan sebuah simpul a. Penentuan lintasan terpendek dari a ke setiap simpul lainnya di G. Asumsi yang kita buat adalah bahwa semua sisi berbobot positif. Perhatikan Gambar 2.13 dibawah ini [35].



Gambar 2.13 Graf Contoh Untuk Persoalan Lintasan Terpendek [35]

Lintasan terpendek dari simpul 1 ke semua simpul lain diberikan pada Tabel 2.1 dibawah ini (diurut dari lintasan terpendek pertama, kedua ketiga dan seterusnya) [35].

Tabel 2.1 Lintasan Terpendek Dari Simpul 1 ke Semua Simpul [35].

Simpul Asal	Simpul Tujuan	Lintasan Terpendek	Jarak
1	3	1→3	10
1	4	1→3→4	25

1	2	1→3→4→2	45
1	5	1→5	45
1	6	Tidak Ada	-

Dari Tabel 2.1 di atas, terlihat bahwa lintasan terpendek dari 1 ke 2 berarti juga melalui lintasan terpendek dari 1 ke 3 dan dari 1 ke 4.

2.8 Algoritma Pencarian Rute Terpendek

2.8.1 Algoritma A*

Algoritma A* adalah algoritma pencarian rute terpendek yang merupakan algoritma yang dituntun oleh fungsi heuristiknya, yang menentukan urutan simpul mana yang akan dikunjungi terlebih dahulu. Heuristik merupakan penilai yang memberi harga pada tiap simpul yang memandu A Star mendapatkan solusi yang diinginkan. Untuk mencari jarak terdekat pada sebuah peta, peta harus direpresentasikan dengan sebuah graf. Simpul-simpul di graf tersebut merupakan representasi persimpangan-persimpangan di wilayah peta tersebut dan sisinya merupakan representasi jalan yang dapat dilalui. Sisi-sisi graf ini harus merupakan sisi berbobot yang nilainya mempresentasikan panjang lintasan. A* mengevaluasi node dengan menggabungkan g(n), yaitu *cost* untuk mencapai *node*, dan h(n), yaitu *cost* yang diperlukan *dari* node untuk mencapai tujuan, dalam notasi matematika dituliskan sebagai [10]:

$$f(n) = g(n) + h(n) \quad (5)$$

Keterangan:

f(n) adalah jumlah dari g(n) dan h(n) adalah perkiraan jalur terpendek sementara. maka f(n) adalah jalur terpendek yang sebenarnya yang tidak ditelusuri sampai Algoritma A-Star (A*) diselesaikan.

g(n)/*Geographical Cost* adalah total jarak yang didapat dari verteks awal ke verteks sekarang (halangan).

h(n)/Heuristic Cost adalah perkiraan jarak dari vertex sekarang (yang sedang dikunjungi) ke vertek tujuan. sebuah fungsi heuristic digunakan untuk membuat perkiraan seberapa jauh lintasan yang akan diambil ke vertex tujuan.

2.8.2 Algoritma *Floyd-Warshall*

Algoritma *Floyd-Warshall* adalah salah satu algoritma untuk mencari jalur terpendek *(shortest path)* antara semua pasangan simpul dalam sebuah graf berbobot, baik itu dengan

bobot positif atau negatif. Algoritma *Floyd-Warshall* merupakan salah satu jenis dari pemrograman dinamis. Pemrograman dinamis merupakan suatu cara memecahkan suatu masalah dengan cara memandang solusi yang akan didapatkan sebagai keputusan yang saling terkait. Solusi yang dibentuk ini didapatkan dari solusi-solusi yang berasal dari tahap sebelumnya dan memiliki kemungkinan bahwa solusi yang didapatkan lebih dari satu. Yang membedakan dengan algoritma *Dijkstra (greedy)* adalah jika pada algoritma *greedy* pada setiap langkah yang dilakukannya hanya berdasarkan nilai optimum yang didapatkan tanpa memikirkan konsekuensi kedepannya jika memilih suatu langkah pada suatu tahap. Karena hal tersebutlah algoritma *greedy* gagal memberikan solusi yang terbaik. Berbeda dengan pemrograman dinamis yang mencoba memberikan solusi dengan memikirkan konsekuensi yang mungkin akan terjadi kedepannya jika mengambil suatu langkah [36].

Prinsip optimalitas merupakan prinsip yang dipakai oleh pemrograman dinamis, jika solusi total optimal, maka bagian dari solusi sampai pada suatu tahap (misalnya tahap ke-i) juga optimal. Beberapa karakteristik yang dimiliki oleh algoritma *Floyd-Warshall* antara lain [36]:

1. Persoalan dibagi atas beberapa tahap, yang setiap tahapnya hanya akan diambil satu keputusan.
2. Masing-masing tahap terdiri atas sejumlah status yang saling berhubungan dengan status tersebut. Status yang dimaksud di sini adalah berbagai kemungkinan masukan yang ada pada tahap tersebut.
3. Ketika masuk ke suatu tahap, hasil keputusan akan transformasi.
4. Bobot pada suatu tahap akan meningkat secara teratur seiring bertambahnya jumlah tahapan.
5. Bobot yang ada pada suatu tahap tergantung dari bobot tahapan yang telah berjalan dan bobot pada tahap itu sendiri.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan pada tahap sebelumnya.
7. Terdapat hubungan rekursif yang menyatakan bahwa keputusan terbaik dalam setiap status pada tahap k akan memberikan keputusan terbaik untuk setiap status pada tahap k + 1.
8. Prinsip optimalitas berlaku pada persoalan yang dimaksud.

Algoritma *Floyd-Warshall* mempunyai jalur yang berbobot dan keluaran dari algoritma ini adalah dengan menghitung bobot minimum pada jalur yang saling berkaitan pada pasangan *node*, dan mengeksekusi pada semua pasangan *node* [33]. Algoritma *Floyd-*

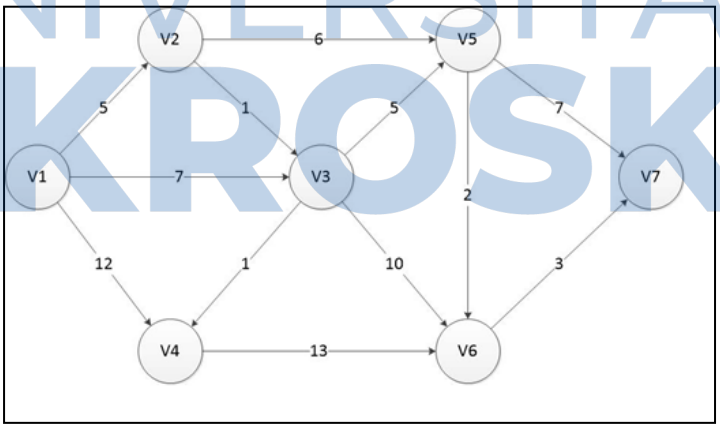
Warshall adalah teknik yang mengambil keuntungan dari subproblem yang tumpang tindih, substruktur optimal, dan memperdagangkan ruang untuk waktu guna meningkatkan kompleksitas runtime algoritma. Eksekusi algoritma tunggal akan menemukan jalur terpendek antara semua pasangan simpul [37].

Berikut ini digunakan Formulasi Rekursif untuk algoritma Floyd Warshall antara lain [33]:

1. *Vertex-vertex* antara dalam *short path*.
2. Jika V = {1,2,3,..,N), untuk k=0,....,n. Maka $d_{ij}^{(k)}$ =
   a. w_{ij} jika k = 0 (6)
   b. $\min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{ij}^{(k-1)}$, untuk k > 0 (7)
3. Solusi dari $d_{ij}(n)$ merupakan matriks *shortest path* dari *vertex* i ke *vertex* j.

2.8.3 Algoritma *Dijkstra*

Algoritma yang paling banyak digunakan untuk menentukan masalah pemilihan *rute*, baik penentuan *rute* terpendek maupun *rute* yang paling optimal adalah metode *Dijkstra*. Metode Dijkstra menawarkan *node-node* yang dilalui untuk mencari rute terpendek dari *node* awal hingga *node* tujuan. Berdasarkan nilai minimum bobot bagian yang diberikan pada serangkaian tahapan-tahapan solusi. Setiap graf terdapat sumber node untuk kemudian secara strategi algoritma *greedy* dilakukan pencarian minimal bobot terkecil sehingga didapati nilai terpendek yang merupakan rute atau jarak terpendek [38]. Contoh graf yang telah siap untuk memberikan jalur rute terpendek dapat dilihat pada Gambar 2.14 [39].



Gambar 2.14 Ilustrasi Contoh Graf [39]

Berdasarkan Gambar 2.14 dapat dirincikan hasil langkah-langkahnya berdasarkan implementasi algoritma *Dijkstra* pada Tabel 2.2 berikut .

Tabel 2.2 Hasil Pencarian Jalur Terpendek Algoritma *Dijkstra* Untuk Graf [39]

Iteration	Unvisited (Q)	Visited(S)	Current	Node: Min = (dist[node], prev[node]) iteration						
	Intialization (V1,V2,V3,V4,V5,V6,V7)	(-)		V1	V2	V3	V4	V5	V6	V7
				(0,-)0	(∞,-)0	(∞,-)0	(∞,-)0	(∞,-)0	(∞,-)0	(∞,-)0
1	{V2,V3,V4,V5,V6,V7}	{V1}	V1		(5,V1)1	(7,V1)1	(12,V1)1	(12,V1)1	(∞,V1)1	(∞,V1)1
2	{V3,V4,V5,V6,V7}	{V1,V2}	V2			(6,V2)2	(12,V1)1	(11,V2)2	(∞,V2)2	(∞,V2)2
3	{V4,V5,V6,V7}	{V1,V2,V3}	V3				(7,V3)3	(11,V3)3	(16,V3)3	(∞,V3)3
4	{V5,V6,V7}	{V1,V2,V3,V4}	V4					(11,V3)3	(16,V3)3	(∞,V3)3
5	{V6,V7}	{V1,V2,V3,V4,V5}	V5						(13,V5)5	(18,V5)5
6	{V7}	{V1,V2,V3,V4,V5,V6}	V6							(16,V6)6

Dengan demikian jarak terpendek dari V1 ke V7 adalah 16 dengan jalur V1->V2->V3->V5->V6->V7.

Pada penelitian ini, algoritma rute terpendek yang akan diimplementasikan adalah algoritma *Dijkstra*.

2.9 *Black Box Testing*

Metode *Black Box Testing* merupakan salah satu metode yang mudah digunakan karena hanya memerlukan batas bawah dan batas atas dari data yang di harapkan. Estimasi banyaknya data uji dapat dihitung melalui banyaknya *field* data entri yang akan diuji, aturan entri yang harus dipenuhi serta kasus batas atas dan batas bawah yang memenuhi. Melalui metode ini dapat diketahui jika fungsionalitas masih dapat menerima masukan data yang tidak diharapkan maka menyebabkan data yang disimpan kurang valid [40].

Pengujian adalah satu set aktifitas yang direncanakan dan sistematis untuk menguji atau mengevaluasi kebenaran yang diinginkan. Pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program untuk mengetahui apakah fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian pada sistem menggunakan metode *Black Box Testing*, tujuannya mengetahui kelemahan dari sistem agar data yang dihasilkan sesuai dengan data yang dimasukkan setelah data dieksekusi dan menghindari kekurangan dan kesalahan pada aplikasi sebelum digunakan oleh *user* [40].