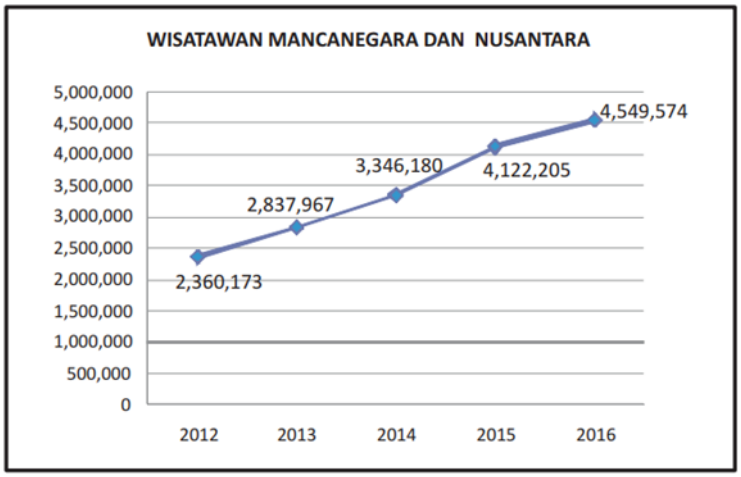
BAB III
METODE PENELITIAN

3.1 Analisis Masalah

Daerah Istimewa Yogyakarta (DIY) dikenal sebagai kota kebudayaan, pusat pendidikan, serta dikenal sebagai kota yang memiliki banyak destinasi wisata. Yogyakarta menjadi daerah tujuan wisata yang cukup terkenal di nusantara dan mancanegara. Kepedulian pemerintah dalam pembangunan pariwisata di DIY juga mempengaruhi perkembangan kepariwisataan maupun jumlah wisatawan di Provinsi DIY.

Menurut data statistik Dinas Pariwisata DIY, jumlah kunjungan wisatawan dari tahun ke tahun selalu mengalami peningkatan yang signifikan, berikut adalah grafik jumlah wisatawan dari tahun 2012-2016 di DIY:



Gambar 3.1 Jumlah wisatawan DIY tahun 2011-2016

Dari data pada Gambar 3.1 tersebut diketahui bahwa jumlah wisatawan dari tahun 2012 ke tahun 2016 terus mengalami peningkatan. Meningkatnya jumlah wisatawan di DIY juga dipengaruhi oleh banyaknya jumlah destinasi wisata yang ditawarkan.

Dengan banyaknya wisatawan dan banyaknya alternatif destinasi wisata yang ada di Daerah Istimewa Yogyakarta tentunya memiliki dampak di berbagai bidang, salah satunya adalah kurangnya alat bantu bagi para wisatawan untuk menentukan urutan kunjungan ke tempat-tempat wisata yang diinginkan. Tidak hanya itu, penentuan *cluster* untuk setiap tempat wisata terhadap jumlah hari berlibur juga menjadi masalah efisiensi perjalanan. Bagi para

travel agent juga pasti mengalami kesulitan untuk menentukan urutan kunjungan wisata serta memberikan paket perjalanan yang menarik dan efisien kepada calon konsumennya.

Permasalahan yang muncul dalam menentukan urutan perjalanan tersebut tentunya membutuhkan solusi demi meningkatkan efisiensi dalam membuat rencana perjalanan.

3.2 Analisis Kebutuhan

Analisis kebutuhan sistem bertujuan untuk menentukan kebutuhan dari aplikasi pembuat *itinerary* wisata di Daerah Istimewa Yogyakarta.

3.2.1 Analisis Kebutuhan Masukan

Berdasarkan hasil analisis kebutuhan masukan, maka diketahui masukan yang dibutuhkan oleh aplikasi adalah sebagai berikut:

a. Pengguna menentukan jumlah hari yang diinginkan untuk berlibur.
b. Pengguna menentukan lokasi awal dimulainya perjalanan.
c. Pengguna menentukan lokasi-lokasi tujuan yang ingin dikunjungi.

3.2.2 Analisis Kebutuhan Proses

Berdasarkan hasil analisis kebutuhan proses, maka diketahui proses yang dibutuhkan oleh aplikasi adalah sebagai berikut:

a. Proses mengakses halaman yang tersedia pada aplikasi.
b. Proses mengakses lokasi awal dimana pengguna berada.
c. Proses mengakses lokasi yang diinginkan pengguna untuk dijadikan lokasi awal rencana perjalanan.
d. Proses menampilkan daftar lokasi tujuan wisata.
e. Proses pembuatan *itinerary*
f. Proses menampilkan hasil *itinerary*

3.2.3 Analisis Kebutuhan Keluaran

Berdasarkan hasil analisis kebutuhan keluaran, maka diketahui hasil keluaran yang diharapkan oleh aplikasi adalah sebagai berikut:

a. Halaman utama dapat menjalankan semua fitur yang tersedia, mulai dari penentuan hari, penentuan lokasi awal, dan penentuan lokasi tujuan.
b. Halaman *Rencana Perjalanan* dapat menampilkan hasil *itinerary* yang paling efisien.

3.2.4 Analisis Kebutuhan Perangkat Keras

Analisis kebutuhan perangkat keras yang digunakan untuk implementasi sistem adalah sebagai berikut :

a. Laptop
    1. Processor intel core i3
    2. RAM 6 Gb
    3. HDD 500 Gb

3.2.5 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak yang digunakan untuk implementasi sistem adalah sebagai berikut :

a. Windows 10
   Sistem operasi berbasis Windows.
b. Sublime Text 3
   Text editor untuk menulis kode.
c. Mozilla Firefox
   Browser untuk menampilkan sistem.
d. Xampp Control Panel v3.2.2
   Web server untuk menampilkan halaman web yang dinamis.

3.3 Perancangan dan Pemodelan Sistem

Setelah melakukan tahap analisis sistem maka tahap selanjutnya adalah perancangan dan pemodelan sistem yang akan dibuat. Dalam pembuatan sistem ini rancangan yang dibuat adalah *framework* sistem, *flowchart* dan perancangan *interface*.

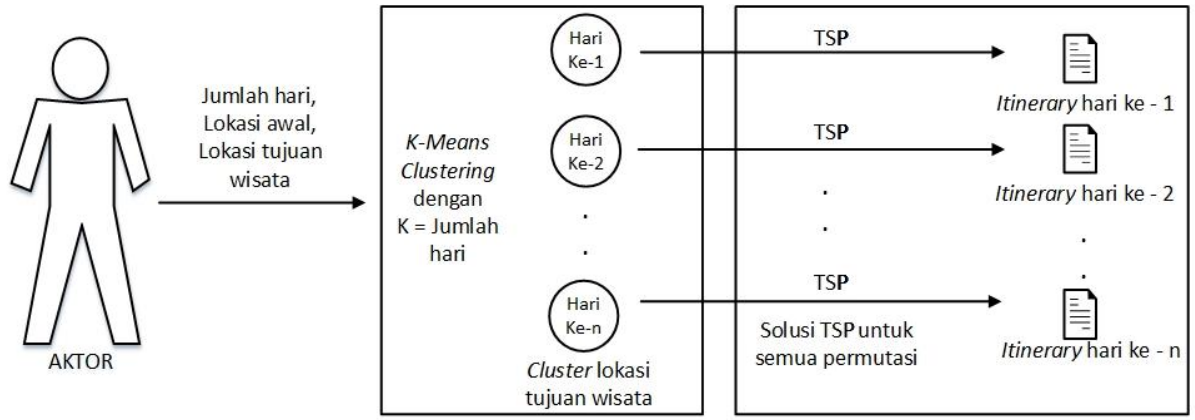
3.3.1 *Framework* Sistem Pembuat *Itinerary* Wisata

Framework sistem merupakan sebuah kerangka kerja untuk memodelkan sistem. Dalam *framework* sistem dilakukan analisis terhadap kebutuhan sistem, dan diidentifikasi beberapa masukan yang dibutuhkan oleh aplikasi. Kebutuhan tersebut berupa jumlah hari yang diinginkan untuk berlibur, lokasi awal dari perjalanan dan juga lokasi tujuan yang ingin

dikunjungi pengguna. Pada aplikasi ini diasumsikan bahwa lokasi awal dan lokasi akhir perjalanan adalah sama, sesuai dengan konsep TSP.

Lokasi awal dapat diisi dengan alamat dimana pengguna akan menginap di kota tujuan wisata. Aplikasi ini juga akan menyediakan daftar lokasi tujuan wisata yang bisa dikunjungi, sehingga pengguna hanya perlu memilih dari daftar yang ada. Dalam pembuatan aplikasi ini, *database* untuk daftar tujuan wisata diambil dari YogYes.com yang merupakan situs panduan wisata Kota Yogyakarta.

Kerangka sistem disajikan pada Gambar 3.2 dimana pendekatan yang dilakukan terdiri dari dua modul utama yaitu modul pengelompokan dengan metode *k-means* yang akan mengelompokkan semua lokasi tujuan yang dipilih oleh pengguna dengan jumlah *K cluster* sama dengan jumlah hari berlibur. Modul kedua ialah untuk menemukan solusi TSP dengan menggunakan semua kemungkinan permutasi.



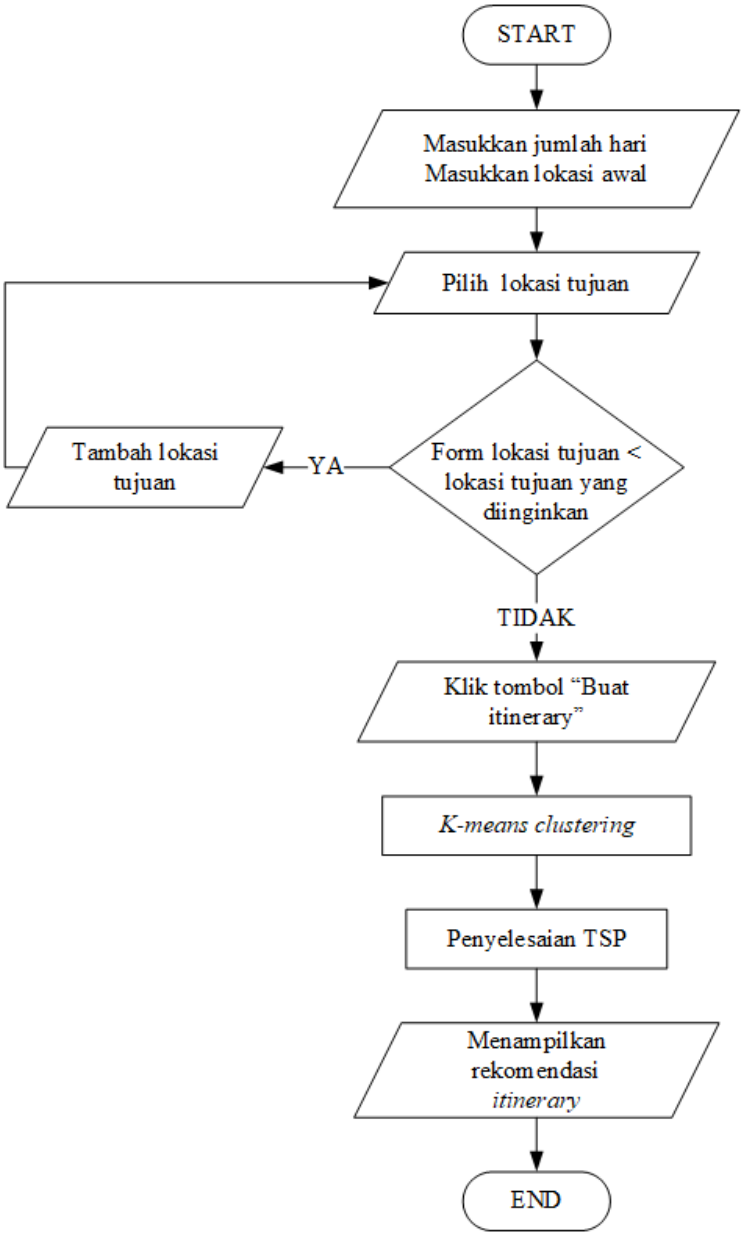
Gambar 3.2 *Framework* sistem

3.3.2 *Flowchart* Sistem Pembuat *Itinerary* Wisata

Flowchart adalah bagan-bagan yang mempunyai arus yang menggambarkan langkah-langkah atau alur penyelesaian suatu masalah serta untuk menggambarkan sebuah algoritma. Ada tiga *flowchart* yang digunakan untuk menggambarkan alur sistem.

Gambar 3.3 menjelaskan tentang *flow* (alur) dari aplikasi *itinerary* wisata secara menyeluruh, dimulai dengan memasukkan jumlah hari, lokasi awal dan lokasi tujuan. Langkah selanjutnya adalah dilakukan pengecekan untuk memastikan bahwa jumlah lokasi tujuan sesuai dengan yang diinginkan. Jika *form* lokasi tujuan kurang dari jumlah lokasi tujuan yang diinginkan, maka tambahkan lokasi tujuan kemudian memilih kembali lokasi tujuan. Jika kondisi telah terpenuhi maka klik tombol "Buat *Itinerary*" maka alur selanjutnya adalah melakukan proses pembuatan *itinerary* yang terdiri dari *k-means clustering* dan penyelesaian

TSP. Terakhir adalah menampilkan rekomendasi *itinerary*. *Flowchart* sistem dapat dilihat pada Gambar 3.3.

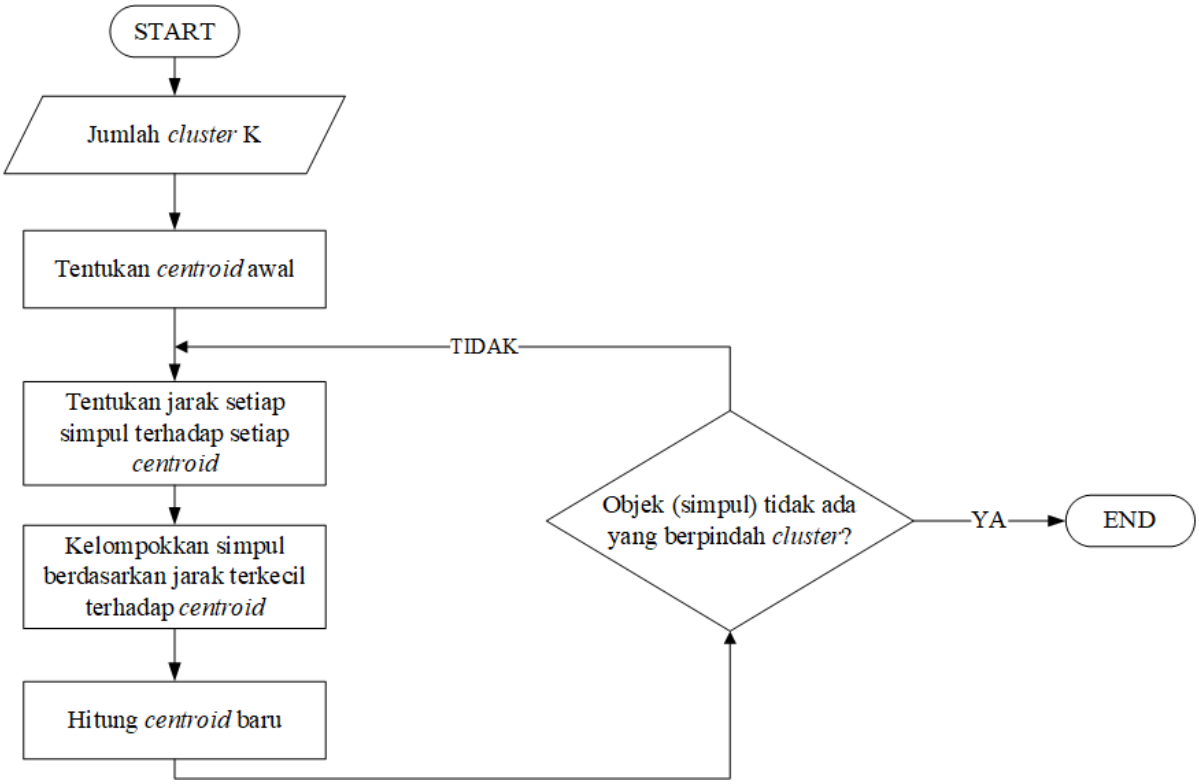


Gambar 3.3 *Flowchart* sistem *itinerary* wisata

Flowchart kedua dan ketiga menjelaskan dengan lebih detail proses yang ada di dalam sistem. *Flowchart* kedua adalah *k-means clustering* yang menjelaskan tentang proses pengelompokkan lokasi tujuan berdasarkan jumlah hari. Proses dimulai dengan menentukan jumlah *cluster* k, dimana k merupakan jumlah hari berlibur. Kemudian setelah didapatkan nilai k, menentukan *centroid* awal atau titik tengah suatu *cluster* dimana *centroid* awal didapatkan

secara acak. Tahap selanjutnya adalah menentukan jarak setiap simpul terhadap *centroid* awal dimana simpul yang dimaksud disini adalah lokasi tujuan wisata yang telah dipilih.

Kemudian mengelompokkan simpul berdasarkan jarak terkecil terhadap *centroid.* Setelah didapatkan kelompok-kelompok sebanyak *cluster* k maka dilakukan pengecekan terhadap masing-masing *cluster.* Apakah setiap simpul sudah tidak lagi mengalami perpindahan *cluster*? Jika "TIDAK" yang berarti simpul masih mengalami perubahan *cluster* maka akan dilakukan pengulangan proses yang dimulai dari menentukan kembali *centoid* awal. Jika "YA" yang berarti simpul sudah tidak lagi mengalami perubahan *cluster* maka proses selesai. Sehingga akan didapatkan *cluster* yang berisi simpul-simpul lokasi tujuan yang memiliki kedekatan jarak. Berikut adalah Gambar 3.4 *flowchart k-means clustering* :

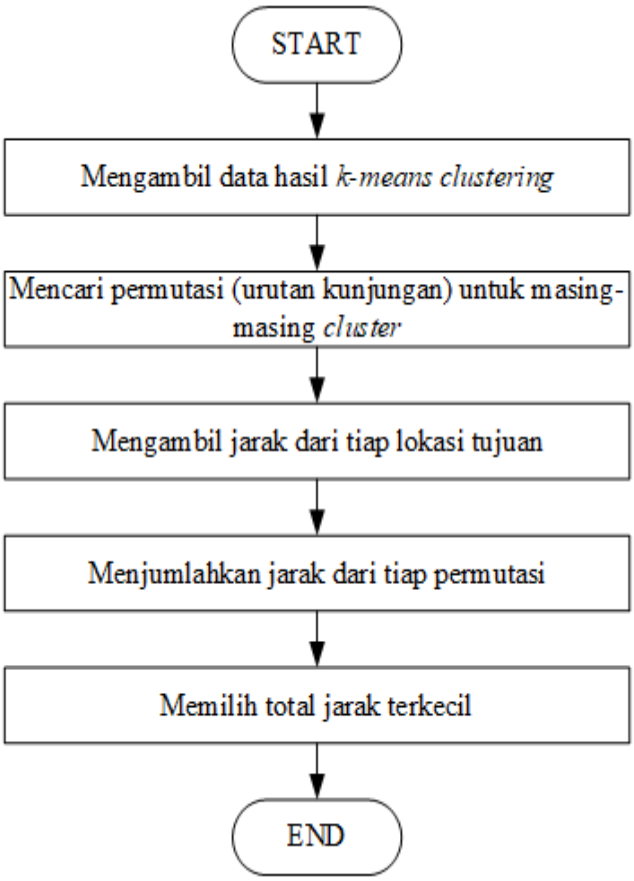


Gambar 3.4 *Flowchart k-means clustering*

Flowchart ketiga merupakan *flowchart* yang menggambarkan proses penyelesaian TSP untuk mendapatkan rute perjalanan terpendek. Dimulai dengan mengambil data hasil *k-means clustering*, tahap selanjutnya adalah mencari permutasi untuk masing-masing kelompok atau *cluster* untuk mencari kemungkinan-kemungkinan dari rute yang ada.

Setelah didapatkan hasil permutasi dari rute perjalanan, maka proses selanjutnya adalah mengambil jarak dari tiap lokasi tujuan yang ada di *database.* Kemudian menjumlahkan jarak

dari tiap permutasi, setelah jumlah dari masing-masing permutasi didapatkan, maka masuk ke proses terakhir, yaitu memilih total jarak terkecil. Kemudian alur berakhir dengan didapatkannya hasil terkecil dari total jarak permutasi.



Gambar 3.5 *Flowchart* proses penyelesaian TSP

3.4 Contoh Kasus TSP

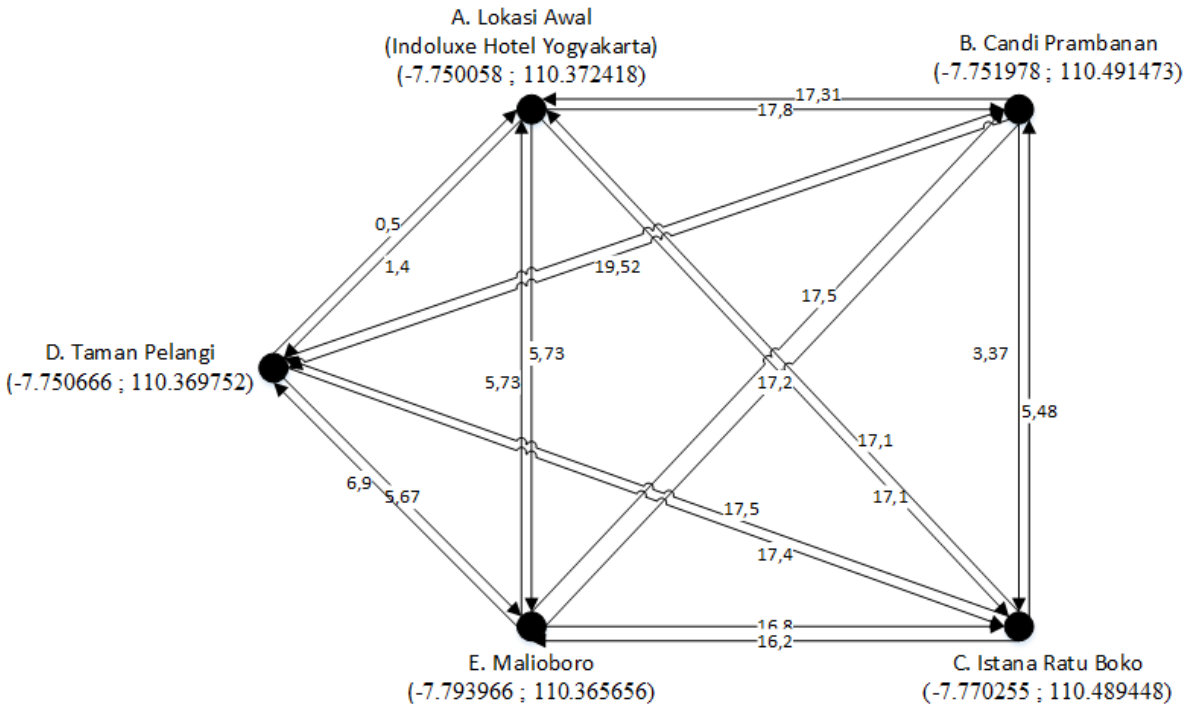
Untuk memberikan gambaran mengenai cara kerja sistem itinerary wisata dengan *k-means clustering* dan *Traveling Saleman Problem* berikut adalah contoh kasusnya :

Jumlah hari : 2 hari

Lokasi awal : Indoluxe Hotel Yogyakarta

Tujuan wisata : Candi Prambanan, Istana Ratu Boko, Taman Pelangi, Malioboro.

Langkah awal penyelesaiannya adalah dengan mengelompokkan tempat tujuan wisata dengan metode *k-means clustering* dimana k = jumlah hari. Kemudian setelah didapatkan masing masing *cluster* maka langkah selanjutnya adalah mencari permutasi untuk setiap *cluster* yang telah didapatkan. Karena permutasi yang digunakan merupakan hasil dari simpul berarah dan berbobot, maka abc ≠ cba. Visualisasi contoh kasus TSP dapat dilihat pada Gambar 3.6.



Gambar 3.6 Visualisasi contoh kasus TSP

Jarak untuk masing-masing lokasi dapat dilihat pada Tabel 3.1 berikut :

Tabel 3.1 Jarak antar lokasi contoh kasus TSP

	A(km)	**B(km)**	**C(km)**	**D(km)**	**E(km)**
A	0	17,8	17,1	1,4	5,73
B	17,31	0	3,37	19,52	17,2
C	17,1	5,48	0	17,5	16,2
D	0,5	18,0	17,4	0	5,67
E	5,73	17,5	16,8	6,9	0

Keterangan :

A : Lokasi awal (Indoluxe Hotel)

B : Candi Prambanan

C : Istana Ratu Boko

D : Taman Pelangi

E : Malioboro

Koordinat untuk masing-masing lokasi adalah sebagai berikut :

Tabel 3.2 Koordinat lokasi

Nama Lokasi	Latitude	Longitude
A	-7.750058	110.372418
B	-7.751978	110.491473
C	-7.770255	110.489448
D	-7.750666	110.369752
E	-7.793966	110.365656

Proses *clustering*

Berikut adalah tahapan dalam *k-means clustering* untuk menyelesaikan contoh kasus TSP di atas :

a. Menentukan banyaknya *cluster* adalah dua (k= 2).

b. Menentukan *centroid* setiap *cluster*.

Untuk menentukan *centroid* awal untuk setiap *cluster* adalah dengan mengambil data dari koordinat lokasi pada Tabel 3.2 secara acak, namun untuk *clustering* ini **Indoluxe Hotel tidak disertakan karena merupakan lokasi awal**. Dipilih tujuan B dan C sebagai *centroid* awal (sel yang berwarna kuning dan biru di Tabel 3.2). Berikut adalah *centroid* pada perulangan ke-0 :

Tabel 3.3 *Centroid* pada perulangan ke-0

Centroid	**Latitude**	**Longitude**
C1	-7.751978	110.491473
C2	-7.770255	110.489448

Keterangan :

C1 : *Centroid* awal untuk *cluster* 1 dipilih dari lokasi B

C2 : *Centroid* awal untuk *cluster* 2 dipilih dari lokasi C

Penentuan *centroid* baru untuk perulangan berikutnya (perulangan ke-1 sampai selesai) dipilih dengan **menghitung rata-rata data pada setiap *cluster***. Jika *centroid* baru berbeda dengan *centroid* sebelumnya, maka proses dilanjutkan kelangkah berikutnya. Namun jika *centroid* baru sama dengan *centroid* sebelumnya, maka proses *clustering* selesai.

c. Menghitung jarak data dengan *centroid*

Untuk menghitung jarak data dengan *centroid* digunakan rumus *Euclidean Distance* sebagai berikut :

(3.1)

$$d(x_j, c_k) = \sqrt{\sum_{i=1}^{n}(x_{ji} - c_{ki})^2}$$

Keterangan :

d = jarak	x = data
n = jumlah data	i = parameter
j = banyaknya data	k = *cluster*
c = *centroid*	

Jarak data dengan *cluster* 1 adalah :

$$\mathrm{d}(\mathrm{x_{B,}c_1}) = \sqrt{(\mathrm{X_{Ba}} - \mathrm{C_{1a}})^2 + (\mathrm{X_{Bo}} - \mathrm{C_{1o}})^2}$$
$$= \sqrt{(-7.751978 - (-7.751978))^2 + (110.491473 - 110.491473)^2}$$
$$= 0$$

$$\mathrm{d}(\mathrm{x_{C,}c_1}) = \sqrt{(\mathrm{X_{Ca}} - \mathrm{C_{1a}})^2 + (\mathrm{X_{Co}} - \mathrm{C_{1o}})^2}$$
$$= \sqrt{(-7.770255 - (-7.751978))^2 + (110{,}489448 - 110.491473)^2}$$
$$= \sqrt{0.000334 + 0.000004}$$
$$= \sqrt{0.000338}$$
$$= 0.018388$$

$$d(x_{\mathrm{D}}, c_1) = \sqrt{(X_{Da} - C_{1a})^2 + (X_{Do} - C_{1o})^2}$$
$$= \sqrt{(-7.750666 - (-7.751978))^2 + (110.369752 - 110.491473)^2}$$
$$= \sqrt{0.000001 + 0.014816}$$
$$= \sqrt{0.014817}$$
$$= 0.121728$$

$$\mathrm{d}(\mathrm{x_E}, \mathrm{c_1}) = \sqrt{(\mathrm{X_{Ea}} - \mathrm{C_{1a}})^2 + (\mathrm{X_{Eo}} - \mathrm{C_{1o}})^2}$$
$$= \sqrt{(-7.793966 - (-7.751978))^2 + (110.365656 - 110.491473)^2}$$
$$= \sqrt{0.001762 + 0.015829}$$
$$= \sqrt{0.017592}$$
$$= 0.132638$$

Jarak dengan *cluster* 2 adalah :

$$d(x_{B,}c_2) = \sqrt{(X_{Ba} - C_{2a})^2 + (X_{Bo} - C_{2o})^2}$$
$$= \sqrt{(-7.751978 - (-7.770255))^2 + (110.491473 - 110.489448)^2}$$
$$= \sqrt{0.000334 + 0.000004}$$
$$= \sqrt{0.000338}$$
$$= 0.018388$$

$$d(x_{C,}c_2) = \sqrt{(X_{Ca} - C_{2a})^2 + (X_{Co} - C_{2o})^2}$$
$$= \sqrt{(-7.770255 - (-7.770255))^2 + (110{,}489448 - 110{,}489448)^2}$$
$$= 0$$

$$d(x_{D,}c_2) = \sqrt{(X_{Da} - C_{2a})^2 + (X_{Do} - C_{2o})^2}$$
$$= \sqrt{(-7.750666 - (-7.770255))^2 + (110.369752 - 110{,}489448)^2}$$
$$= \sqrt{0.000383 + 0.014327}$$
$$= \sqrt{0.014710}$$
$$= 0.121288$$

$$d(x_{E,}c_2) = \sqrt{(X_{Ea} - C_{2a})^2 + (X_{Eo} - C_{2o})^2}$$
$$= \sqrt{(-7.793966 - (-7.770255))^2 + (110.365656 - 110.489448)^2}$$
$$= \sqrt{0.000562 + 0.000562}$$
$$= \sqrt{0.015886}$$
$$= 0.126042$$

Hasil jarak pada setiap baris data dapat dilihat pada Tabel 3.4.

Tabel 3.4 Hasil perhitungan jarak pada perulangan ke-0

Nama Lokasi	Latitude	Longitude	dC1	dC2
B	-7.751978	110.491473	0	0.018388
C	-7.770255	110,489448	0.018388	0
D	-7.750666	110.369752	0.121728	0.121288
E	-7.793966	110.365656	0.132638	0.126042

d. Kelompokkan data sesuai dengan *cluster*nya, yaitu data yang memiliki jarak terpendek. Contoh : karena $d(x_{B,}c_1) < d(x_{B,}c_2)$ maka x_B masuk ke dalam *cluster* 1. Pada Tabel 3.7 data untuk lokasi B masuk ke dalam *cluster* 1 karena dc1 < dc2, sedangkan data C,D,E masuk ke dalam *cluster* 2 karena dc2 < dc1. Berikut adalah Tabel 3.5 dimana data telah dikelompokkan :

Tabel 3.5 Hasil perhitungan jarak dan pengelompokan data pada perulangan ke-0

Nama Lokasi	Latitude	Longitude	dC1	dC2	C1	C2
B	-7.751978	110.491473	0	0.01838	Ok	
C	-7.770255	110,489448	0.01838	0		Ok
D	-7.750666	110.369752	0.12172	0.12128		Ok
E	-7.793966	110.365656	0.13263	0.12604		Ok

e. Proses kembali ke langkah b yaitu menghitung *centroid* baru.

Nilai *centroid* baru didapatkan dari rata-rata data dalam satu *cluster* yang sama pada Tabel 3.5. Kemudian dilakukan lagi perhitungan seperti pada perulangan sebelumnya. Hasil perhitungan pada perulangan ke-1 dapat dilihat pada Tabel 3.7.

Perhitungan *centroid* baru :

C1 : Latitude = latitude B

= -7.751978

Longitude = longitude B

= 110.491473

C2 : Latitude $= \frac{\text{latitude C} + \text{latitude D} + \text{latitude E}}{3}$

$= \frac{-7.770255+(-7.750666)+(-7.793966)}{3}$

= -7,771629

Longitude $= \frac{\text{longitude C} + \text{longitude D} + \text{longitude E}}{3}$

$$= \frac{110{,}489448 + 110.369752 + 110.365656}{3}$$

= 110,408285

Perulangan ke-1

Tabel 3.6 *Centroid* pada perulangan ke-1

Cluster	**Latitude**	**Longitude**
C1	-7.751978	110.491473
C2	-7,771629	110,408285

Tabel 3.7 Hasil perhitungan jarak dan pengelompokan data pada perulangan ke-1

Nama Lokasi	**Latitude**	**Longitude**	**dc1**	**dc2**	**C1**	**C2**
B	-7.751978	110.491473	0	0,08547	Ok	
C	-7.770255	110,489448	0,01838	0,08117	Ok	
D	-7.750666	110.369752	0,12172	0,04386		Ok
E	-7.793966	110.365656	0,13263	0,04812		Ok

Perulangan ke-2

Tabel 3.8 *Centroid* pada perulangan ke-2

Cluster	**Latitude**	**Longitude**
C1	-7.751978	110.491473
C2	-7,771629	110,408285

Tabel 3.9 Hasil perhitungan jarak dan pengelompokan data pada perulangan ke-2

Nama Lokasi	**Latitude**	**Longitude**	**dc1**	**dc2**	**C1**	**C2**
B	-7.751978	110.491473	0,00919	0,020363	Ok	
C	-7.770255	110,489448	0,00919	0,12176	Ok	
D	-7.750666	110.369752	0,12115	0,02174		Ok
E	-7.793966	110.365656	0,12905	0,02174		Ok

Perulangan ke-3

Tabel 3.10 *Centroid* pada perulangan ke-3

Cluster	**Latitude**	**Longitude**
C1	-7.751978	110.491473
C2	-7,771629	110,408285

Karena *centroid* baru sama dengan *centroid* sebelumnya maka proses *clustering* selesai. Didapatkan hasil *cluster* berupa :

C1 = B (Candi Prambanan) dan C (Istana Ratu Boko)

C2 = D (Taman Pelangi) dan E (Malioboro)

Cluster 1 dikunjung pada hari ke-1 kemudian *cluster* 2 dijunjungi pada hari ke-2, ataupun sebaliknya.

Proses TSP

Setelah proses *clustering* telah selesai maka tahapan selanjutnya adalah mencari solusi *traveling salesman problem*. Dari *cluster* yang didapatkan sebelumnya maka akan dilakukan proses TSP sebagai berikut :

a. Mencari permutasi setiap *cluster*

Rumus permutasi yang digunakan :

$$P = n! \tag{3.2}$$

Keterangan :

P = permutasi

n = banyaknya data

Cluster yang didapatkan sebelumnya :

C1 = Candi Prambanan dan Istana Ratu Boko (n=2)

C2 = Taman Pelangi dan Malioboro (n=2)

Permutasi untuk C1 :

$P = n!$

$= 2!$

$= 2 \times 1$

$= 2$

Dari perhitungan di atas didapatkan bahwa banyaknya kemungkinan urutan pada C1 adalah 2. Karena jumlah n untuk C2 sama dengan C1 maka hasil kemungkingkan permutasinya

pun sama, yaitu 2. Karena setiap *cluster* diawali dan diakhiri oleh lokasi awal maka urutannya adalah sebagai berikut :

C1 :

1. Indoluxe Hotel – Candi Prambanan – Istana Ratu Boko – Indoluxe Hotel
2. Indoluxe Hotel – Istana Ratu Boko – Candi Prambanan – Indoluxe Hotel

C2 :

1. Indoluxe Hotel – Taman Pelangi – Malioboro – Indoluxe Hotel
2. Indoluxe Hotel – Malioboro – Taman Pelangi – Indoluxe Hotel

b. Mengambil jarak dari setiap lokasi tujuan

C1 :

1. Indoluxe Hotel – Candi Prambanan – Istana Ratu Boko – Indoluxe Hotel
   = 17,8 + 3,37 + 17,1
   = 38,27 km
2. Indoluxe Hotel – Istana Ratu Boko – Candi Prambanan – Indoluxe Hotel
   = 17,1 + 5,48 + 17,31
   = 39,89 km

C2 :

1. Indoluxe Hotel – Taman Pelangi – Malioboro – Indoluxe Hotel
   = 1,4 + 5,56 + 5,73
   = 12,69 km
2. Indoluxe Hotel – Malioboro – Taman Pelangi – Indoluxe Hotel
   = 5,73 + 6,9 + 0,5
   = 13,13 km

c. Memilih jarak terpendek untuk setiap *cluster*

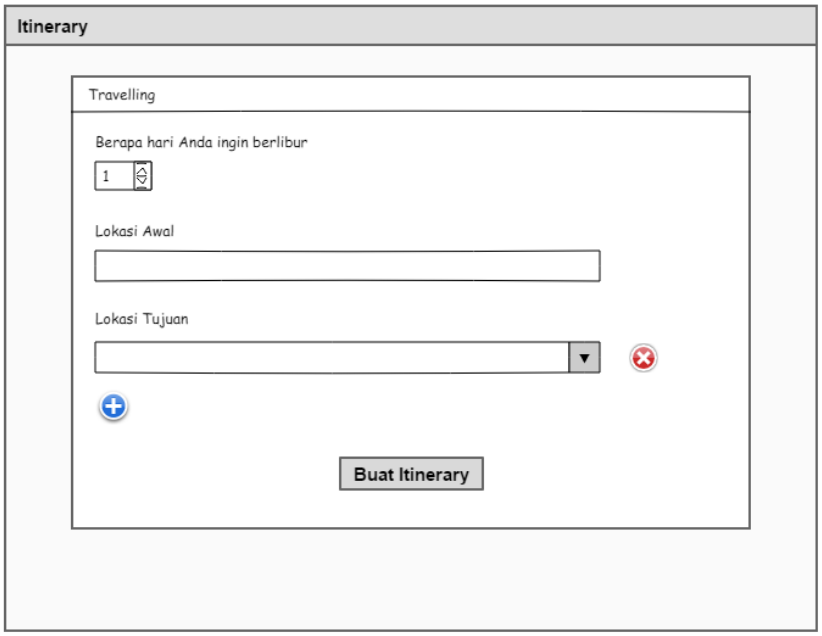
Jarak terpendek untuk C1 adalah permutasi pertama dengan total jarak 38,27 km. Sedangkan jarak terpendek untuk C2 adalah permutasi pertama dengan total jarak 12,69 km. Proses selesai untuk TSP.

Sehingga hasil dari *k-means clustering* dan penyelesaian TSP dalam contoh kasus di atas adalah C1 dengan rekomendasi urutan kunjungan mulai dari Indoluxe Hotel, Candi Prambanan,

Istana Ratu Boko, Indoluxe Hotel dengan jarak tempuh sebesar 38,27 km. Kemudian untuk C2 dengan rekomendasi urutan kunjungan mulai dari Indoluxe Hotel, Taman Pelangi, Malioboro, Indoluxe Hotel dengan jarak tempuh sebesar 12,69 km. Kedua rekomendasi tersebuat merupakan rekomendasi perjalanan dengan jarak tempuh terpendek.

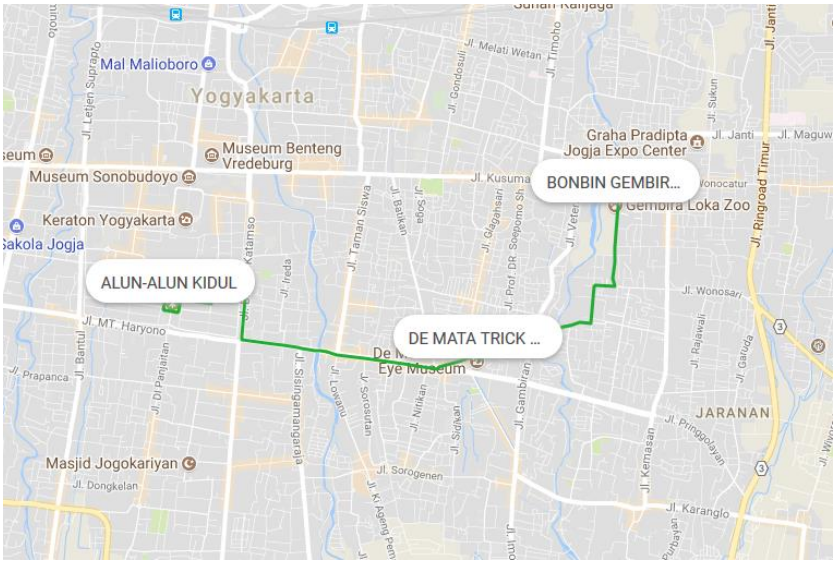
3.4.1 Perancangan Antarmuka

Perancangan antaramuka untuk aplikasi *itinerary* wisata ini terdiri dari 2 rancangan, yaitu halaman utama dan halaman rencana perjalanan. Berikut adalah Gambar 3.7 yaitu rancangan antarmuka halaman utama:



Gambar 3.7 Rancangan antarmuka halaman utama

Pada halaman utama ini pengguna dapat menentukan jumlah hari berlibur, kemudian menentukan lokasi awal perjalanan serta menentukan lokasi tujuan wisata yang diinginkan. Perancangan antarmuka untuk halaman rencana perjalanan terdapat pada Gambar 3.8.



Gambar 3.8 Rancangan antarmuka rencana perjalanan

Gambar 3.8 di atas merupakan rancangan antarmuka rencana perjalanan dari hasil proses pembuatan *itinerary*, pada halaman ini pengguna dapat melihat informasi rute perjalanan wisata yang direkomendasikan oleh sistem.

3.5 Skenario Pengujian

Skenario pengujian yang akan dilakukan pada sistem ini terdiri dari *black box testing*, *user acceptance test* (UAT) serta adanya analisis lama waktu pemrosesan.

3.5.1 *Black Box Testing*

Black box testing merupakan sebuah pengujian yang hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari sistem (Irwan, 2017). Skenario pengujian sistem dengan uji coba *black box* untuk aplikasi pembuat *itinerary* wisata dapat dilihat pada Tabel 3.11 berikut:

Tabel 3.11 Skenario *black box testing*

Kelas Uji	Butir Uji	Jenis Pengujian
Jumlah Hari	Menentukan jumlah hari	*Black Box*
Lokasi Awal	Deteksi otomatis lokasi awal	*Black Box*
	Mengubah lokasi awal	*Black Box*
Lokasi Tujuan	Memilih lokasi tujuan	*Black Box*
	Menambahkan lokasi tujuan	*Black Box*
	Menghapus lokasi tujuan	*Black Box*

Proses *Itinerary*	Membuat rute (urutan) rencara perjalanan	*Black Box*

3.5.2 Rancangan *User Acceptance Test* (UAT)

User acceptance test atau uji penerimaan pengguna adalah suatu proses pengujian oleh pengguna yang dimaksudkan untuk menghasilkan dokumen yang dijadikan bukti bahwa *software* yang telah dikembangkan telah dapat diterima oleh pengguna, apabila hasil pengujian (*testing*) sudah bisa dianggap memenuhi kebutuhan dari pengguna (Sidik, 2006). Perancangan UAT untuk aplikasi *itinerary* wisata di DIY dibagi menjadi 2, yaitu untuk calon pengguna (wisatawan) serta untuk *travel agent* atau penyedia layanan wisata.

Pengujian ini akan dilakukan dengan memberikan kuisioner kepada calon pengguna dan *traver agent.* Kuisioner berisi pertanyaan-pertanyaan yang berhubungan dengan aplikasi dan diisi dengan 5 (lima) macam skala penilaian meliputi Sangat Tidak Setuju (STS), Tidak Setuju (TS), Kurang Setuju (KS), Setuju (S), dan Sangat Setuju (SS).

Rancangan UAT Untuk Calon Pengguna Dari Sisi Wisatawan

Rancangan UAT untuk calon pengguna dari sisi wisatawan dapat dilihat pada Tabel 3.12 untuk mengetahui apakah aplikasi *itinerary* wisata ini sudah memenuhi kebutuhan wisatawan sebagai calon pengguna.

Tabel 3.12 Rancangan kuisioner aplikasi *itinerary* wisata untuk wisatawan

No	**Pernyataan**	**Penilaian**				
		STS	**TS**	**KS**	**S**	**SS**
1.	Sistem sudah dapat digunakan untuk membuat *itinerary* wisata					
2.	Sistem dapat digunakan untuk mencari lokasi awal sesuai yang Anda inginkan					
3.	Sistem dapat digunakan untuk mencari lokasi tujuan sesuai yang Anda inginkan					
4.	Sistem mempunyai antarmuka yang menarik					
5.	Sistem memberikan informasi yang jelas					

	mengenai *itinerary* wisata yang Anda buat					
6.	Sistem dapat menampilkan jarak atar lokasi pada *itinerary* wisata Anda dengan akurat					
7.	Sistem dapat memberikan rute perjalanan yang paling akurat (pendek dan cepat) untuk dilalui					
8.	Sistem dapat memberikan pesan kesalahan jika jumlah hari lebih dari jumlah lokasi tujuan					
9.	Sistem membantu wisatawan dalam menyusun rencana perjalanan					

Rancangan UAT Untuk Calon Pengguna Dari Sisi *Travel Agent*

Rancangan UAT untuk *travel agent* dapat dilihat pada Tabel 3.13 untuk mengetahui apakah aplikasi *itinerary* wisata ini sudah memenuhi kebutuhan *travel agent* sebagai calon pengguna.

Tabel 3.13 Rancangan kuisioner aplikasi *itinerary* wisata untuk *travel agent*

No	**Pernyataan**	**Penilaian**				
		STS	**TS**	**KS**	**S**	**SS**
1.	Sistem dapat memberikan rute perjalanan yang paling akurat (pendek dan cepat) untuk dilalui					
2.	Dengan menggunakan sistem ini, maka biaya perjalanan dapat ditekan					
3.	Dengan menggunakan sistem ini, maka Anda dapat memberikan rekomendasi rencana perjalanan yang tepat kepada pelanggan					
4.	Dengan sistem ini, Anda dapat membantu pelanggan Anda untuk menentukan rencana perjalanan yang sesuai					

5.	Dengan sistem ini, Anda dapat dengan mudah memberikan rencana perjalanan baru jika pelanggan merubah lokasi tujuan yang diinginkan sewaktu-waktu					
6.	Sistem mempunyai antarmuka yang menarik					
7.	Sistem dapat digunakan dengan mudah, tidak membingungkan					

Bobot nilai yang digunakan di dalam kuisoner terdiri dari 5 (lima) macam skala penilaian yaitu:

Nilai 1 = Sangat Tidak Setuju (STS)

Nilai 2 = Tidak Setuju (TS)

Nilai 3 = Kurang Setuju (KS)

Nilai 4 = Setuju (S)

Nilai 5 = Sangat Setuju (SS)

Pengambilan kesimpulan dari hasil kuisioner memiliki standar penilaian untuk menentukan apakah sistem yang dibuat sudah dapat digunakan dengan baik oleh pengguna ataukah belum. Berikut adalah standar penilaian yang digunakan:

0 – 19.99 % = Tidak Baik

20 – 39.99 % = Kurang Baik

40 – 59.99 % = Cukup Baik

60 – 79.99 % = Baik

80 – 100 % = Sangat Baik

Pengambilan kesimpulan didapatkan dengan melakukan perhitungan bobot nilai yang didapatkan dari hasil kuisioner. Presentase untuk pengambilan keputusan menggunakan rumus sebagai berikut:

$$Hasil\ Pengujian = \frac{Nilai\ Total\ Kuisioner}{Nilai\ Maksimum\ Kuisioner}\ x\ 100\ \% \qquad (3.3)$$

3.5.3 Rancangan Analisis Waktu Pemrosesan

Lamanya waktu yang diperlukan untuk menjalanankan sistem tergantung dari banyaknya destinasi wisata yang dikunjungi dan jumlah hari berlibur yang diinginkan. Faktor lain yang

dapat mempegaruhi waktu pemrosesan adalah kecepatan koneksi internet, kecepatan prosesor, serta kapasitas RAM yang digunakan. Rancangan analisis waktu pemrosesan sistem dapat dilihat pada Tabel 3.14 beikut:

Tabel 3.14 Rancangan analisis waktu pemrosesan

No	Jumlah Hari	Jumlah Destinasi	Waktu Pemrosesan (detik)
1.	1	1	
2.	1	2	
3.	1	3	
4.	1	4	
5.	1	5	
6.	1	6	
7.	1	7	
8.	1	8	
9.	1	9	
10.	1	10	
11.	2	2	
12.	2	3	
13.	2	4	
14.	2	5	
15.	2	6	
16.	2	7	
17.	2	8	
18	2	9	
19.	2	10	