BAB II
TINJAUAN PUSTAKA

2.1. Sistem

Sistem dapat berupa abstrak atau fisik, system yang abstrak adalah susunan yang teratur dari gagasan-gagasan tentang tuhan, manusia dan sebagainya sedangkan System yang bersifat fisik adalah serangkaian unsur yang bekerja sama untuk mencapai tujuan (Tohari, 2017)

Sistem merupakan kumpulan elemen-elemen yang saling terkait dan bekerja sama untuk memproses masukan (*input*) yang ditujukan kepada sistem tersebut dan mengolah masukan tersebut sampai menghasilkan (*output*) yang diinginkan (Tohari, 2017)

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan (Hakim, 2018)

2.2. Metode *Algoritma Best-First Search (BeFS) Algoritma*

Algoritma BeFS menggunakan nilai heuristik pada tiap simpul-simpulnya. Nilai heuristik dalam penelitian ini merupakan nilai estimasi dari jarak antar dua titik. Simpul dengan nilai heuristik terbaik akan dibuka atau dikerjakan lebih dahulu. Nilai heuristik dikatakan terbaik artinya apabila nilai tersebut mendekati nilai sebenarnya. Diasumsikan bahwa dalam permasalahan pencarian rute terpendek, nilai yang dianggap baik apabila nilai tersebut memberikan hasil yang lebih kecil dari nilai yang lainnya karena konteks yang bibahas adalah jarak. Setelah simpul dengan nilai terbaik diperoleh, jika goal state belum ditemukan maka akan dilakukan pemeriksaan pada simpul berikutnya dengan nilai heuristik terbaik pada kedalaman yang sama. Simpul tersebut kemudian dibuka dan diperiksa apakah terdapat goal state pada cabang-cabangnya. Bila goal state belum ditemukan, akan dilakukan proses yang sama pada simpul

berikutnya.Penentuan nilai terbaik dapat dilakukan menggunakan suatu operasi fungsi yang disebut fungsi heuristik yang akan mengestimasi seberapa baik dibangkitkannya setiap node. Fungsi heuristik dikatakan baik atau dapat diterima apabila nilai perkiraan yang dihasilkan tidak melebihi nilai sebenarnya. Semakin mendekati nilai sebenarnya, fungsi heuristik akam semakin optimal. Heuristik yang dipakai dalam penelitian ini, yaitu heuristik jarak garis lurus yang dinotasikan hSLD . Fungsi heuristik yang digunakan dalam algoritma ini hanya memperhitungkan jarak untuk mencapai tujuan dan mengabaikan jarak jalan sebenarnya yang sudah ditempuh untuk sampai ke titik tujuan, sehingga sering disebut sebagai *Greedy Best-First Search*. Secara harfiah, greedy artinya rakus atau tamak (sifat yang berkonotasi negatif).

Sesuai dengan arti tersebut, prinsip greedy adalah mengambil keputusan yang dianggap terbaik untuk saat itu saja yang diharapkan dapat memberikan solusi terbaik secara keseluruhan. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya. *Greedy Best-First Search* akan mencoba untuk memperluas node yang terdekat dengan tujuan, dengan alasan bahwa akan menemukan kemungkinan solusi tercepat. Sehingga fungsi evaluasi node yang digunakan hanya fungsi heuristik, yang dinyatakan dengan (Rusdiyanto, 2017)

$$f(n) = h(n)$$

Keterangan :

$f(n)$= fungsi evaluasi

$h(n)$= fungsi heuristi atau estimasi jarak terpendek dari vertex (node) (node) tujuan (heuristik).

2.3. *Location Based Service* (LBS)

Location Based Service (LBS) mengacu pada sekumpulan aplikasi yang mengeksplorasi pengetahuan / informasi dari lokasi geografis perangkat

mobile untuk mendapatkan layanan berdasarkan informasi tersebut" (*Internet and Mobile Association of India*, 20018).

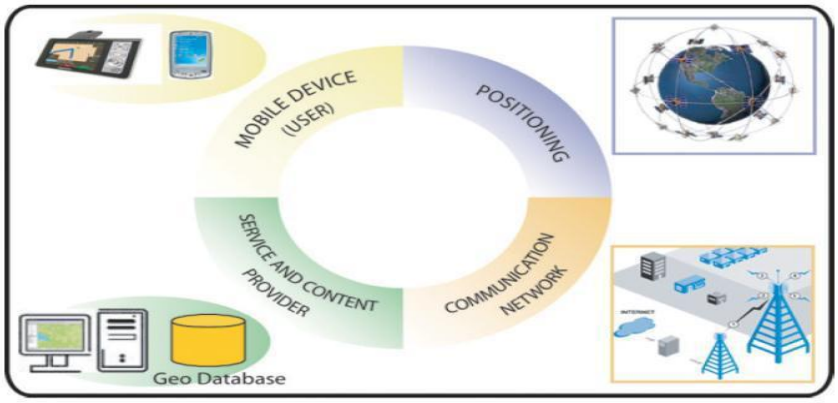
Pemanfaatan LBS memudahkan pengguna perangkat *mobile* mengatur dan memilih layanan sesuai kebutuhan dan dapat dimanfaatkan untuk memberikan berbagai layanan seperti informasi kondisi lingkungan (kemacetan lalu lintas, cuaca, lokasi fasilitas umum terdekat), maupun promosi produk dan jasa.

Pemanfaatan LBS pada sistem operasi perangkat mobile android dimungkinkan dengan adanya dukungan dua unsur utama berikut:

1. *Location Manager* (API *Maps*) Menyediakan *tool* atau *source* untuk LBS, *Application Progrogram Interface* (API). *Maps* menyediakan fasilitas untuk menampilakan, memanipulasi maps atau peta beserta *feature-feature* lainnya seperti tampilan satelit, *street* (jalan), maupun gabungannya. Paket ini berada com.google.android.map.
2. *Location Provider* (API *Location*) Menyediakan teknologi pencarian lokasi yang digunakan oleh *device* atau perangkat. API *Location* berhubungan dengan data GPS (*Global Positioning System*) dan data lokasi *real-time*. API *Location* berada pada paket android yaitu dalam paket *android.location*. *Location Manager* dapat ditentukan lokasi posisi saat ini, *track* gerakan atau perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan.

2.3.1 Komponen Location Based Service (LBS)

Dalam Layanan Berbasis Lokasi terdapat lima komponen penting yaitu Meliputi Gambar 2.1:



Gambar 2. 1 *Komponen LBS*

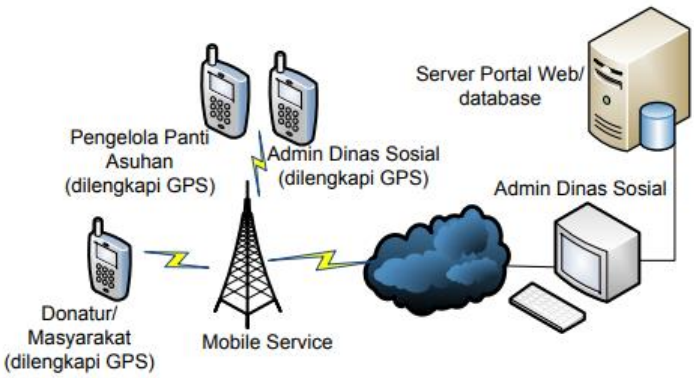
Sumber : *Internet and Mobile Association of India* (2018).

1. *Mobile Devices*: Suatu alat yang digunakan oleh pengguna untuk meminta informasi yang dibutuhkan. Informasi dapat diberikan dalam bentuk suara,gambar, dan teks.
2. *Comunication Network*: Komponen kedua adalah jaringan komunikasi yang mengirim data pengguna dan informasi yang diminta dari *mobile* terminal ke *Service Provider* kemudian mengirimkan kembali informasi yang diminta ke pengguna. *Communication Network* dapat berupa jaringan seluler (*GSM, CDMA), Wireless Local Area Network (WLAN),* atau *Wireless Wide Area Network (WWAN)*
3. *Positioning Component*: Untuk memproses suatu layanan maka posisi pengguna harus diketahui
4. *Service and Aplication Provider*: Penyedia layanan menawarkan berbagaimacam layanan kepada pengguna dan bertanggung jawab untuk memproses informasi yang diminta oleh pengguna.
5. *Data and Content Provider*: Penyedia layanan tidak selalu menyimpan semua data yang dibutuhkan yang bisa diakses oleh pengguna. Untuk itu, data dapat diminta dari data dan *content provider*. (Imaniar, 2011)

2.3.2. Cara Kerja *Location Based Service* (LBS)

Cara kerja LBS pada aplikasi pencarian objek terdekat berdasarkan posisi *user.*

a. *Smartphone* akan membuka aplikasi yang me manfaatkan layanan LBS yang sudah ter-*install*.
b. Aplikasi akan melakukan sambungan dengan jaringan *provider* yang dipakai oleh *user*.
c. Selanjutnya informasi posisi *user* pada perangkat *mobile* yang diperoleh dari *Location Sensor*. Hal ini dapat dilakukan oleh perangkat dengan menggunakan GPS sendiri atau layanan posisi jaringan yang berasal dari *provider*.
d. Perangkat *mobile* pengguna akan mengirimkan permintaan informasi ke satelit untuk menentukan *longitude* dan *latitude* dari pengguna aplikasi tersebut.
e. *Provider* menghubungkan aplikasi pada *smartphone* dengan *server* LBS dan meminta data yang diinginkan *user* beserta informasi tentang jarak dan cara yang diperlukan dalam menjangkau lokasi tujuan.
f. Terakhir, *user* mendapatkan data dan ditampilkan di *smartphone* melalui aplikasi. Dapat dilihat pada Gambar 2.2



Gambar 2. 2 Cara Kerja Location Based Service.

Sumber : Justino , Syahputri, & Nurfiana (2020).

2.3.3 Google Maps

Google Maps merupakan layanan dari *google* yang mempermudah penggunanya untuk melakukan kamampuan pemetaan untuk aplikasi

yang dibuat. Sedangkan *Google Maps API* memungkinkan pengembangan untuk mengintegrasikan *Google Maps* ke dalam situs web. Dengan menggunakan *Google Maps API* memungkinkan untuk menanamkan situs *Google Maps* ke dalam situs eksternal, di mana situs data tertentu dapat dilakukan *overlay*. Meskipun pada awalnya hanya *JavaScript API, API Maps* sejak diperluas untuk menyertakan sebuah *API* untuk *Adobe Flash* aplikasi, layanan untuk mengambil gambar peta statis, dan layanan web untuk melakukan *geocoding*, menghasilkan petunjuk arah mengemudi, dan mendapatkan profil elevasi. Kelas kunci dalam perpustakaan *Maps* adalah *MapView*, sebuah *subclass* dari *ViewGroup* dalam standar perpustakaan Android. Sebuah *MapView* menampilkan peta dengan data yang diperoleh dari layanan *Google Map* (Firly, 2018).

2.3.4 MySQL

MySQL merupakan *software* yang tergolong sebagai *DBMS* (*Database Management System*) yang bersifat *open source*. *Open source* menyatakan bahwa *software* ini dilengkapi dengan *source code* (kode yang dipakai untuk membuat *MySQL*), selain itu tentu saja bentuk *executable*- nya atau kode yang dapat dijalankan secara langsung dalam sistem operasi, dan bisa diperoleh dengan men-*download* (mengunduh) di internet secara gratis (A.S Rosa & Shalahuddin, 2018).

2.3.5 XAMPP

XAMPP adalah aplikasi yang berfungsi sebagai *server* yang berdiri sendiri (*localhost*), yang terdiri beberapa program antara lain: Apache HTTP Server, MySQL database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP sendiri merupakan singkatan dari X empat sistem operasi, yang meliputi Apache, MySQL, PHP dan Perl. Program ini tersedia dalam GNU, merupakan web server yang mudah untuk digunakan yang dapat menampilkan halaman web yang dinamis (Sadeli, 2017).

2.4. UML (*Unified Modelling Language*)

UML merupakan sistem arsitektur yang bekerja dalam OOAD (*Object-Oriented Analysis/Design*) dengan satu bahasa yang konsisten untuk menentukan, visualisasi, mengkontruksi, dan mendokumentasikan artifact (sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa software, dapat berupa model, deskripsi, atau software) yang terdapat dalam sistem software. UML merupakan bahasa pemodelan yang paling sukses dari tiga metode OO yang telah ada sebelumnya, yaitu Booch, OMT (*Object Modeling Technique*), dan OOSE (*Object-Oriented Software Engineering*). UML merupakan kesatuan dari dari ketiga pemodelan tersebut dan ditambah kemampuan lebih karena mengandung metode tambahan untuk mengatasi masalah pemodelan yang tidak dapat ditangani ketiga metode tersebut. UML dikeluarkan oleh OMG (*Object Management Group, Inc*) yaitu organisasi internasional yang dibentuk pada 1989, terdiri dari perusahaan sistem informasi, *software, developer,* dan para user sistem komputer.

Dengan adanya UML, diharapkan dapat mengurangi kekacauan dalam bahasa pemodelan yang selama ini terjadi dalam lingkungan industri. UML diharapkan juga dapat menjawab masalah penotasian dan mekanisme tukar menukar model yang terjadi selama ini.

Tujuan UML diantaranya adalah :

a) Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.

b) Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.

c) Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.

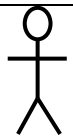
Bahasa Pemodelan Pengembangan Sistem (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri

untuk mendefinisikan *requirement*, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (Rosa and Shalahudin, 2018). Beberapa jenis diagram *UML* antara lain sebagai berikut:

2.4.1. *Use Case Diagram*

Use case diagram atau diagram *use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat (Rosa and Shalahudin, 2018), simbol-simbol yang ada pada diagram *use case* dapat dilihat pada gambar 2.1 di bawah ini:

Tabel 2. 1 Simbol Diagram Use Case
Sumber : (Rosa and Shalahudin, 2018)

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*
Aktor/*actor* 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor
Ekstensi/*extend* *<<extend>>* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek biasanya *use case* tambahan memiliki nama depan.
Asosiasi/*association*	Komunikasi antara aktor dan *use case* yang berpatisipasi pada *use case* atau *use case* memiliki interaksi dengan *actor*
Generalisasi/***generalization*** 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
Menggunakan/*Include/uses*	Relasi *use case* tambahan ke sebuah *use*

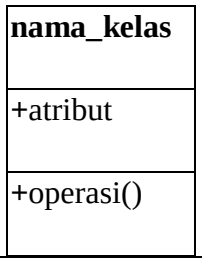
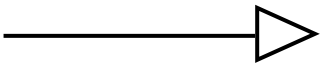
Tabel 2.1 Simbol Diagram *Use Case* (Lanjutan)
Sumber : (Rosa and Shalahudin, 2018)

Simbol	Deskripsi
<<*include*>> 	*case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya

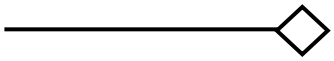
2.4.2. *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi (Rosa and Shalahudin, 2018), simbol-simbol yang ada pada diagram kelas pada tabel *class diagram* 2.2 di bawah ini:

Tabel 2. 2 *Simbol Class Diagram*
Sumber : (Rosa and Shalahudin, 2018)

Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem
Antarmuka/*Interface*  **nama_*interface***	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi/*asociation*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah/*directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi(umum khusus)

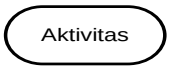
Tabel 2.2 Simbol *Class Diagram* (Lanjutan)
Sumber : (Rosa and Shalahudin, 2018)

Simbol	Deskripsi
Kebergantungan/*dependecy* 	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi/*agregation* 	Relasi antar kelas dengan makna semua bagian (*whole-part*)

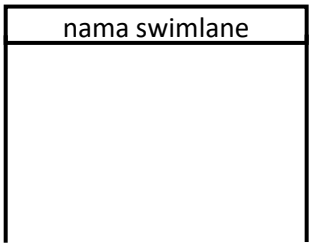
2.4.3. *Activity Diagram*

Activity diagram atau Diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem (Rosa and Shalahudin, 2018). simbol-simbol yang ada pada *activity diagram*dapat dilihat pada tabel 2.3 di bawah ini :

Tabel 2. 3 Simbol Activity Diagram
Sumber : (Rosa and Shalahudin, 2018)

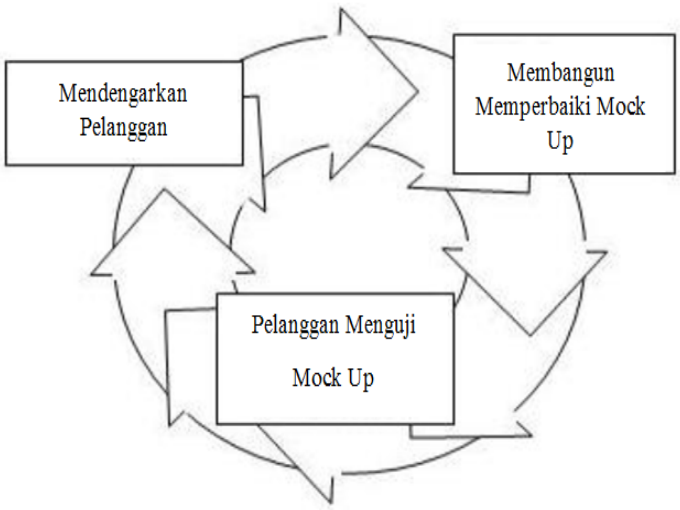
Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem,sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
Percabangan/*decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan/*join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Dokumen 	Menunjukan dokumen sumber atau laporan
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

Tabel 2.3 Simbol *Activity Diagram* (Lanjutan)
Sumber : (Rosa and Shalahudin, 2018)

Simbol	Deskripsi
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

2.5. Metode Pengembangan Perangkat Lunak

Model prototipe dapat digunakan untuk menyambung ketidakpahaman pelanggan mengenai hal teknis dan memperjelas spesifikasi kebutuhan yang diinginkan pelanggan kepada pengembang perangkat lunak (A.S Rosa, & Shalahuddin, M., 2018) dapat dilihat pada Gambar 2.3



Gambar 2. 3 Ilustrasi model prototipe.
Sumber : (Rosa and Shalahudin, 2018)

Menurut Rosa and Shalahudin (2018) pada Gambr 2.3 ilustrasi model *prototype* terdapat beberapa tahapan dalam proses *prototipe* yaitu:

1. Mendengarkan Pelanggan

   Pada tahap ini, dilakukan pengumpulan kebutuhan dari sistem dengan cara mendengar kebutuhan pelanggan sebagai pengguna sistem perangkat lunak untuk menganalisis serta mengembangkan kebutuhan pengguna.

2. Merancang dan Membuat *Prototipe*

   Pada tahap ini, dilakukan perancangan dan pembuatan prototipe sistem yang disesuai dengan kebutuhan pengguna.

3. Uji Coba

   Pada tahap ini, dilakukan pengujian *prototipe* sistem oleh pengguna kemudian dilakukan evaluasi sesuai dengan kekurangan-kekurangan dari kebutuhan pelanggan. Jika sistem sudah sesuai dengan prototipe, maka sistem akan diselesaikan sepenuhnya. Namun, jika masih belum sesuai kembali ke tahap pertama.

2.6. Pengujian Sistem *Black – Box*

Pengujian *black-box* berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian, pengujian black-box memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program (Dodenti, 2013). Pengujian black-box berusaha menemukan kesalahan dalam kategori sebagai berikut :

a. Fungsi – fungsi yang tidak benar atau hilang,
b. Kesalahan interface
c. Kesalahan dalam struktur data atau akses eksternal
d. Kesalahan kinerja

2.7. Penelitian Terkait

Berikuti ini adalah penelitian yang terkait dengan peneliti, yaitu dapat dilihat pada Tabel 2.4

Tabel 2. 4 Penelitian Terkait

No	Judul	Penulis (Tahun)	Masalah	Metode	Hasil
1	Sistem Informasi Geografis dengan Google Map Untuk Pencarian Rumah Kost	Sundari, & Arumaryawan (2018)	Permasalahan yang terjadi yaitu mencari tempat kos yang sesuai dengan kebutuhan tidaklah mudah, masalah yang dihadapi terkadang susahnya mencari lokasi yang dekat dengan tempat kerja ataupun tempat kuliah dan harga yang terjangkau	Metode research dan development	Hasil penelitian adalah aplikasi baru dengan pemanfaatan teknologi melalui Sistem Informasi Geografis (SIG).
2	Implementasi Kos Seeker Di Wilayah Pekanbaru Dengan Metode *Content Based Recommender System* Berbasis Web	Djusar, Fajrizal , Rhoma, & Rofiantoro (2017)	Membuang waktu dalam pencarian lokasi	Metode *Content Based Recommender System*	Hasil penelitian adalah aplikasi Kos Seeker pencari kos lebih mudah mencari informasi kosan dan memilih kosan sesuai kriteria seperti fasilitas kamar, fasilitas umum, harga dan lokasi kampus yang direkomendasikan dari Aplikasi Kos Seeker.
3	Aplikasi Pencarian Lokasi Kos Di Kota Tembilahan Berbasis Web	Usman, & Masdi (2018)	Sulitnya mendapatkan informasi mengenai lokasi kos-kosan.	Metode SDLC	Aplikasi ini akan memanajemen data-data kos dan memberikan informasi kos secara detail dan

	Mobile		Mereka terkendala dengan terbatasnya pengetahuan tentang seluk beluk kota dan nama-nama jalan.		lokasi yang akurat dengan memanfaatkan peta Google
4	Web Sistem Informasi Pencarian Info Kostan Menggunakan Google Maps API 3	Sagita & Simpony (2018)	Lama dalam pencarian lokasi	Google Maps API 3	Hasil dari pembuatan website ini adalah untuk memberikan kemudahan terhadap konsumen dan pemilik kost dalam setiap aktivitas.
5	Sistem Informasi Pemesanan Salon *Online* Berbasis *Location Based Service.*	Setianni, & Syahputri (2019)	Lamanya waktu yang dibutuhkan untuk perawatan dan biaya jasayang akan ditagihkan	*Location Based Service.*	Aplikasi ini dapat menghitung waktu perawatan dan jasa dengan cepat. Sedangkan penentuan lokasi yang menggunakan *Location Based Service* telah menampilkan hasil yang akurat

Berdasarkan Tabel 2.4 dari beberapa penelitian yang telah dilakukan mengenai pencarian lokasi terdapat masalah yang sama yaitu lamanya waktu yang dibutuhkan untuk pencarian lokasi dengan menerapkan metode yang berbeda seperti metode *Content Based Recommender System*, SDLC, Google Maps API 3 dan *Location Based Service.* sedangkan peneliti menggunakan metode *prototype.* Pada sistem yang dibangun menghasilkan aplikasi pencarian lokasi secara cepat tanpa harus membuang waktu untuk mencari lokasi yang dituju. Berdasarkan penelitian yang diteliti maka tertarik membangun sistem sistem yang dibangun dapat membantu pengguna untuk mencari lokasi, menghemat waktu dan melihat

informasi lengkap mengenai lokasi objek tujuan dengan menggunakan metode *Greedy Best First Search*. Algoritma *Best-First Search* bekerja menggunakan fungsi perkiraan *heuristic* yaitu dengan memprioritaskan pemeriksaan *node-node* yang berurut dan berada pada arah yang benar, karena hanya menggunakan fungsi heuristic tanpa memperhitungkan biaya untuk menuju suatu node, sehingga jalur yang ditemukan dengan algoritma ini kemungkinan adalah jalur terpendek, tetapi belum tentu jalur tersebut memiliki biaya terkecil.