

**IMPLEMENTASI ALGORITMA A\*(Star) UNTUK MENENTUKAN RUTE TERPENDEK DI KAMPUS UNIVERSITAS MERCUBUANA MERUYA : Backend**

OLEH :

1. Deni Arianto (41513120131)
2. Siti Hajarwati (41513120153)

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS MERCU BUANA**

**JAKARTA**

**2015**



**[JUDUL]**

*Laporan Tugas Akhir*

Diajukan Untuk Melengkapi Persyaratan

Menyelesaikan Gelar Sarjana Komputer

Disusun oleh :

**[Nama]**

**[NIM]**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS MERCU BUANA**

**JAKARTA**

**2015**

# *ABSTRAKSI*

*Template TA ini merupakan panduan dan contoh penyusunan laporan tugas akhir yang digunakan di lingkungan Program Studi Teknik Informatika. Mahasiswa sebaiknya menggunakan template lay out yang sama yaitu menggunakan template lay out ini. Sedangkan substansi (isinya) harus disesuaikan dengan topic dan keperluan laporan penelitian masing – masing. Panduan umum antara lain : Bab 1. Pendahulan, Bab 2. Landasan Teori (atau Studi Terkait). Bab 3. Analisa Sistem, Bab 4. Perancangan, Bab 5. Implementasi dan Pengujian, Bab. 6 Kesimpulan. Bab 1. Secara umum berisi : Latar Belakang, Rumusan Masalah, Batasan Masalah, Tujuan dan Manfaat Penelitian, Ruang Lingkup Penelitian, Metodologi Penelitian, dan Sistematika Penulisan Laporan.* ***Latar Belakang*** *harus menjelaskan gambaran kondisi umum terkait masalah dan alasan mengapa topic penelitian muncul.* ***Rumusan masalah*** *berisi poin-2 permasalahan yang akan diselesaikan,* ***batasan masalah*** *merupakan pembatasan atas permasalahan yang akan diselesaikan. Batasan masalah sebaiknya* ***bukan*** *spesifikasi produk/subyek penelitian yang akan diteliti.* ***Tujuan Penelitian*** *merupakan hasil yang direncanakan untuk dicapai pada akhir penelitian,* ***Manfaat Penelitian*** *adalah benefit apa yang bisa diperoleh setelah tujuan penelitian dicapai. Ruang Lingkup membatasi aktivitas / subyek / hal apa saja yang akan ditangani pada penelitian. Bab 2 dan seterusnya, bisa berubah sesuai dengan keperluan penelitian dan penulisan laporan.*

*Panduan terkait lay out untuk template ini: 1. Menggunakan font Times New Roman ukran 12. Spasi untuk abstraksi spasi tunggal, spasi untuk badan laporan 1.5. Identasi dan penomoran bab, sub bab, sub – sub bab, dst menggunakan lay out heading 1, 2, 3, dst. Judul gambar dan judul table menggunakan* ***style caption.*** *Dengan menggunakan style ini****,*** *maka pembuatan* ***daftar isi, daftar table, daftar gambar,*** *dilakukan secara* ***otomatis.*** *Sitasi dan daftar pustaka, menggunakan mendeley dengan stylenya adalah APA (American Psychologi Association). Dengan bantuan mendeley, maka pengorganisasian (pembuatan, pengacuan, dan pembuatan daftar pustaka dilakukan secara otomatis). Silakan mengacu pada situs mendeley* [*http://mendeley.com*](http://mendeley.com) *untuk membuat akun, menginstall, dan menggunakan mendeley.*

*Untuk menggunakan template ini, cara yang paling efektif adalah : 1. Simpan dengan nama file lain sesuai judul TA anda 2. Tuliskan materi anda langsung di template ini (misalnya di badan latar belakang), hapus tulisan template yang tidak diperlukan 3. Untuk membuat / menambah komponen (sub bab, sub – sub bab, judul gambar, judul table, dll) kopi komponen dari template ini ke tempat yang seharusnya, ganti texnya dengan text yang sesuai. Dengan demikian stylenya tidak berubah. Untuk menyamakan style bagian tertentu dengan style yang diinginkan (misalnya dari body menjadi heading 2), blok heading 2 (sumber salinan style) tersebut, klik ikon* ***Format Painter*** *(gambar kuas kiri atas word), pindahkan mouse ke body yang akan diubah stylnye, sapukan ikon kuas pada body tersebut.*

*Semoga membantu :* [*mujiono.sadikin@mercubuana.ac.id*](mailto:mujiono.sadikin@mercubuana.ac.id)

***Kata Kunci: template TA, lay out, mendeley, …..***

# *ABSTRACT*

*This final exam report template is as a guidance and example as well for informatics students who are preparing their final report documentation. This tamplate is applied in Faculty of Computer Science, Universitas Mercu Buana. All those students are strongly recommended to apply all template style to their report. The substantive content of their final report, of course,…etc. Please type your English abstract here..:)*

***Keywords : template TA, lay out, mendeley, …..***

# Daftar Isi

[*ABSTRAKSI* i](#_Toc431192794)

[*ABSTRACT* ii](#_Toc431192795)

[Daftar Isi iii](#_Toc431192796)

[Daftar Gambar v](#_Toc431192797)

[Daftar Tabel v](#_Toc431192798)

[Definisi vi](#_Toc431192799)

[BAB 1. PENDAHULUAN 1-1](#_Toc431192800)

[1.1. Latar Belakang 1-1](#_Toc431192801)

[**1.1.1** **[Sub sub bab]** 1-1](#_Toc431192802)

[**1.1.2** **[Sub sub bab, italic untuk istilah asing]** 1-1](#_Toc431192803)

[1.2. Rumusan Permasalahan 1-2](#_Toc431192804)

[1.3. Tujuan & Manfaat Penelitian 1-3](#_Toc431192805)

[**1.3.1** **Tujuan Penelitian** 1-3](#_Toc431192806)

[**1.3.2** **Manfaat Penelitian** 1-3](#_Toc431192807)

[1.4. Ruang Lingkup & Batasan Penelitian 1-3](#_Toc431192808)

[1.5. Sistematika Penulisan Laporan 1-4](#_Toc431192809)

[**1.5.1** **Pendahuluan** 1-4](#_Toc431192810)

[**1.5.2** **Landasan Teori** 1-4](#_Toc431192811)

[**1.5.3** **Analisis Sistem** 1-4](#_Toc431192812)

[**1.5.4** **Perancangan Sistem** 1-4](#_Toc431192813)

[**1.5.5** **Implementasi Dan Testing** 1-4](#_Toc431192814)

[**1.5.6** **Penutup** 1-4](#_Toc431192815)

[BAB 2. LANDASAN TEORI 2-1](#_Toc431192816)

[2.1. Interaksi Antar Obyek 2-1](#_Toc431192817)

[2.2. Perolehan Informasi (IR) Berbasis Konteks 2-1](#_Toc431192818)

[BAB 3. ANALISA SISTEM 3-1](#_Toc431192819)

[3.1. […..] 3-1](#_Toc431192820)

[3.2. […..] 3-1](#_Toc431192821)

[3.3. *Pattern Generation* 3-1](#_Toc431192822)

[BAB 4. PERANCANGAN 4-1](#_Toc431192823)

[4.1. Perancangan Algoritma 4-1](#_Toc431192824)

[**4.1.1** **Algoritma A** 4-1](#_Toc431192825)

[**4.1.2** **Algoritma B** 4-1](#_Toc431192826)

[4.2. Perancagan Basis Data 4-1](#_Toc431192827)

[**4.2.1** **E-R-D** 4-1](#_Toc431192828)

[**4.2.2** **Implementasi Fisik Basis Data** 4-1](#_Toc431192829)

[**4.2.3** **………….** 4-1](#_Toc431192830)

[BAB 5. IMPLEMENTASI DAN PENGUJIAN 5-2](#_Toc431192831)

[5.1. Lingkkungan Implementasi 5-2](#_Toc431192832)

[**5.1.1** **Perangkat Keras** 5-2](#_Toc431192833)

[**5.1.2** **Perangkat Lunak Platform** 5-2](#_Toc431192834)

[5.2. Hasil Implementasi 5-2](#_Toc431192835)

[5.3. Hasil Pengujian 5-2](#_Toc431192836)

[**5.3.1** **Skenario Uji Coba** 5-2](#_Toc431192837)

[**5.3.2** **Hasil Uji Coba** 5-2](#_Toc431192838)

[BAB 6. PENUTUP 6-1](#_Toc431192839)

[6.1. Kesimpulan 6-1](#_Toc431192840)

[6.2. Saran 6-1](#_Toc431192841)

[**Daftar Pustaka** A](#_Toc431192842)

# Daftar Gambar

[*Gambar 1‑1**Ilustrasi Permasalahan dan Solusi 1-2*](#_Toc431041434)

[*Gambar 3‑1 Kerangka Global Penelitian 3-1*](#_Toc431041435)

[*Gambar 4‑1 Representasi Relasi Antar Obyek 4-1*](#_Toc431041436)

# Daftar Tabel

[*Tabel 3‑1 Contoh Tabel 3-1*](#_Toc431041430)

# Definisi

|  |  |
| --- | --- |
| **Istilah** | **Pengertian** |
| Dokumen | Pengertian dokumen dalam penelitian ini adalah file yang berisi terminologi – terminologi yang merupakan represenatasi Bahasa Alami. Format file representasi dokumen ini: \*.txt atau \*.pdf |
| Dokumen Terstruktur | Adalah dokumen yang terminologi di dalamnya mengikuti pola tertentu, bisa dipahami secara alami oleh manusia atau mengikuti pola tata bahasa alami, dalam hal ini adalah Bahasa Indonesia |
| Obyek | Pengertian obyek pada penelitian ini kata benda atau yang dibendakan dalam dokumen / corpus yang menjadi topik pembahasan dokumen tersebut. Contoh obyek adalah: nama obat dalam dokumen label obat; judul penelitian atau metoda penelitian atau hasil penelitian dalam dokumen makalah; atau nama barang, komponen barang, atau produsen barang pada brosur produk |
| Obyek Utama | Adalah obyek yang menjadi pembahasan utama pada dokumen. Contoh obyek utama dalam hal ini misalnya nama obat pada label obat, nama produk pada brosur promosi barang/jasa, atau pembayar pajak pada dokumen pajak. Jika obyek utama ditiadakan dari satu dokumen, maka dokumen tersebut menjadi tidak bermakna. |
| Obyek Pendukung | Adalah obyek yang keberadaanya terkait langsung atau mendukung obyek utama. Keberadaan obyek pendukung tergantung pada obyek utama dan peniadaan obyek pendukung dari obyek utama tidak mempengaruhi makna obyek utama. Contoh obyek pendukung dalam hal ini adalah kandungan kimawi obat untuk obat, obyek pajak untuk pembayar pajak pada dokumen pajak, spesikasi produk / jasa pada dokumen brosur promosi penjualan produk |
| Relasi | Mengacu pada jenis hubungan antara satu obyek utama dengan obyek pendukung. Contoh relasi dalam hal ini adalah: relasi penjual dengan barang barang yang dijual, relasi pembeli dengan barang - barang yang dibeli, relasi kandungan kimiawi dengan obat, relasi obyek pajak dengan pembayar pajak |
| Interaksi | Yang dimaksud interaksi pada penelitian ini adalah istilah yang lebih umum yang digunakan untuk mengacu keterkaitan antara obyek utama dengan obyek utama yang lain. Contoh interaksi dalam hal ini adalah interaksi penjual dengan pembeli, interaksi pembayar pajak dengan pemungut pajak, atau interaksi antar obat |
| bi-set | Adalah kesatuan representasi interaksi antar obyek, satu bi-set terdiri dari dua himpunan <MO,SO>. MO = {MO1, MO2, …MOp} dan SO={SO1, SO2,…SOq}. Satu Bi-Set <MO, SO> artinya sekumpulan obyek utama MO1…MOq berinteraksi satu dengan lainnya berdasarkan SO1, SO2,…SOq dengan p > 1, dan q > 0. Untuk menyederhanakan, pada laporan ini selanjutnya : bi-set ditulis *b, MO* sebagai *S,* dan *SO* sebagai *G* |

# PENDAHULUAN

## Latar Belakang

Pembuatan Web Sistem Informasi Geografis (SIG) salah satunya didorong karena penggunaan internet yang sangat luas dimasyarakat dan pemerintah, karena internet maka peta sekarang bisa diakses oleh semua pihak. Oleh karena itu dibuatlah sebuah web SIG di Universitas Mercubuana. Karena terdapat kurang lebih 18 gedung dan ruangan yang ada serta jalur yang harus ditempuh untuk menuju lokasi tempat wisata yang dimaksud.

Pencarian jalur terpendek merupakan suatu permasalahan untuk menemukan sebuah jalur antara dua node. Pada kasus pencarian jalur tercepat antara dua lokasi yang berbeda dalam sebuah peta, node akan merepresentasikan lokasi pada peta untuk melakukan perjalanan antara dua lokasi tersebut. Salah satu metode yang dapat digunakan untuk menyelesaikan permasalahan pencarian jalur terpendek dengan menggunakan algoritma A\* (Star). Algoritma A\*(Star) akan mencari jalur terpendek mulai dari node awal sampai node tujuan. Algroitma ini akan membandingkan bobot terkecil dari node awal sampai ke node tujuan untuk menemukan jalur paling efisien.

Universitas Mercubuana saat ini memiliki 46 tempat yang bisa didatangi, namun saat ini belum ada aplikasi yang dapat memberikan rute yang dapat dilalui para mahasiswa khususnya mahasiswa baru untuk mendapatkan informasi menuju ruangan atau gedung yang akan dituju. Oleh karena itu dibutuhkan suatu aplikasi agar mahasiswa dapat mendapatkan informasi tempat dan jalur terpendek menuju ke tempat tersebut.

Algoritma A\* (Shortest Path Algorithm) adalah algoritma untuk menemukan jarak terpendek dari suatu node ke node yang lainnya pada suatu graph yang berbobot, dimana jarak antar node adalah bobot dari tiap edge pada graph tersebut. Algoritma A\* mencari jarak terpendek untuk tiap node dari suatu graph yang berbobot. Algoritma A\* mencari jarak terpendek dari node asal ke node terdekatnya, kemudian ke node kedua, dan seterusnya. Node ini yang merupakan kandidat dari algoritma A\* untuk memilih node berikutnya dari node asal. Algoritma A\* merupakan salah satu varian bentuk algoritma popular dalam pemecahan persoalan yang terkait dengan masalah optimasi. Sifatnya sederhana dan lempang (straight-forward). Sesuai dengan arti greedy yang secara harafiah berarti tamak atau rakus, namun tidak dalam konteks negative. Algoritma ini hanya memikirkan solusi terbaik yang akan diambil pada setiap langkah tanpa memikirkan konsekuensi ke depan.

## Rumusan Permasalahan

Adapun rumusan masalah yang akan diselesaikan pada penelitian tugas akhir ini adalah :

1. Bagaimana dosen, mahasiswa khususnya mahasiswa baru dapat menemukan suatu tempat di Universitas Mercubuana Meruya ?

## Tujuan & Manfaat Penelitian

### **Tujuan Penelitian**

Tujuan yang ingin dicapai pada penelitian ini adalah:

1. Menyediakan aplikasi yang dapat digunakan oleh dosen dan mahasiswa agar dapat menemukan rute terpendek yang dapat di tempuh dari suatu tempat ke tempat lainnya yang dapat di unduh di Play Store ataupun diakses melalui website.

### **Manfaat Penelitian**

Manfaat yang ingin dicapai pada penelitian ini adalah:

1. Mahasiswa dapat menggunakan sebuah aplikasi yang bisa menunjukan rute terpendek untuk menuju ke sebuah gedung / ruangan yang ada di Universitas Mercu Buana.
2. Mengetahui bagaimana proses algoritma Dijkstra bekerja.

## Ruang Lingkup & Batasan Penelitian

Untuk menghindari meluasnya pembahasan, maka permasalahan dibatasi sebagai berikut:

1. Pembuatan aplikasi ini dibuat dengan menggunakan algoritma A\*(Start).
2. Penulis tidak melakukan perbandingan dengan algoritma pencarian rute terpendek lainnya seperti A star, Bellman-Ford, Floyd-Warshall, dll.

## Sistematika Penulisan Laporan

Laporan hasil penelitian ini disusun menurut sistematika sebagai berikut:

### **Pendahuluan**

Pada bab ini akan diberikan gambaran umum mengenai alasan mengapa penelitian ini dilakukan. Dijelaskan uraian dari permasalahan yang ada sehingga menjadi topik tugas akhir lalu disusun berdasarkan rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, metodologi penelitian, dan sistematika penulisan.

### **Landasan Teori**

Bab ini memberikan teori-teori yang mendukung dalam pelaksanaan penelitian dan penyusunan tugas akhir mengenai algoritma A\*(Star).

### **Analisis Sistem**

Menjelaskan tentang gambaran sistem serta deskripsi dari hasil analisis sistem yang akan dijadikan sebagai petunjuk untuk perancangan pada tahapan berikutnya.

### **Perancangan Sistem**

Berisi tentang PerancanganSistemdenganUML, Perancangan Data,Perancangan Arsitektural, Perancangan Prosedural dan Perancangan Antarmuka.

### **Implementasi Dan Testing**

Menjelaskan mengenai implementasi dari perancangan system yang telah dilakukan, pembahasan mengenai cara bekerjanya Algoritma Dijkstra.

### **Penutup**

Mengemukakan kesimpulan yang diambil dari hasil penelitian dan penulisan Skripsi ini, serta saran-saran untuk pengembangan selanjutnya, agar dapat dilakukan perbaikan-perbaikan di masa yang akan datang.

# LANDASAN TEORI

## Penelitian Sebelumnya

Rujukan penelitian pertama yaitu jurnal ilmiah milik Nor Amalina Mohd Sabri, Abd Samad Hasan Basari, Burairah Husin dan Khyrina Airin Fariza Abu Samah mahasiswa Universitas Teknikal Malaysia Melaka, Malaysia dengan judul The Utilisation of Dijkstra’s Algorithm to Assist Evacuation Route in Higher and Close Building. Dalam penelitiannya peneliti menggunakan Algoritma Dijkstra untuk (Amalina, Sabri, Samad, Basari, & Husin, 2015) Finding a shortest path in a high rise building during critical incident or evacuation is facing two main issues which are evacuees find difficulties to find the best routes and their behavior makes the process more difficult. These problems are important since it is related to the human’s life. By providing the shortest path and control the evacuee behavior, these can lead to successfulness of evacuation. To overcome these issues, two main objectives have been carried on which is initiated by identifying the shortest path algorithm for evacuation. Then follows by design and develop an evacuation preparedness model via shortest path algorithm to choose a suitable exit route to evacuate. Three steps are involved to achieve the objectives. The first step is Building Layout Plan, followed by creating the Visibility Graph and finally implements Dijkstra Algorithm to find the shortest path. Based on the experimental study, the result shows that Dijkstra Algorithm has produced a significant route to exit the building safely. Even though there are other factors need to be considered, this preliminary result has shown a promising outcome which can be extended to improve the robustness of the algorithm. In conclusion, the obtained shortest path is believed can assist evacuee to choose a suitable exit route to evacuate safely.

Rujukan penelitian yang kedua yaitu jurnal ilmiah milik Tjan Marco Orlando mahasiswa Institute Teknologi Bandung, Bandung dengan judul Penerapan Algoritma Dijkstra Untuk Menentukan Rute Wisata Minimum Kota Semarang. Menurut (Orlando & Teknik, 2015) Efisiensi adalah faktor yang sangat penting dalam proses penyelesaian suatu masalah. Penyelesaian masalah dengan cara yang efisien akan meminimalkan sumber daya yang diperlukan. Dalam perancangan panduan rute minimum bagi wisatawan Kota Semarang, efisiensi dalam hal jarak rute juga menjadi hal yang penting. Dengan rute minimum, wisatawan dapat menuju objek wisata yang diinginkannya dengan cost (jarak) minimum. Oleh karena itu, diperlukan algoritma yang tepat untuk menghasilkan panduan rute terpendek ini. Makalah ini membahas penerapan Algoritma Dijkstra untuk menghasilkan panduan rute terpendek bagi wisatawan dengan merepresentasikan objek-objek wisata sebagai verteks dalam graf. Pada akhirnya, dapat ditentukan rute minimum bagi wisatawan untuk menuju ke objek wisata yang diingingkannya.

## Rekayasa Perangkat Lunak

Rekayasa perangkat lunak atau biasa dikenal dengan *Software* merupakan kombinasi antara program, basis data dan dokumentasi di dalamnya, yang dibuat dengan tujuan untuk memecahkan masalah yang sudah di tentukan *requirement-*nya atau kebutuhan. *Software* menambah nilai komponen dari perangkat keras dari sistem komputer. Tanpa *software* , komputer hanyalah sebuah kotak elektronik yang tidak digunakan khusus untuk sebagian umat manusia. [Elvis C. Foster, 2014 : 83]

*Software* terdiri dari data yang diprogram dan disimpan secara digital yang berisi instruksi-instruksi yang akan dibaca dan ditulis oleh komputer. *Software* menyediakan fungsi dan kinerja yang diinginkan, struktur data yang memungkinkan program memanipulasi informasi yang cukup untuk dimengerti, serta dokumen yang menggambarkan operasi dan penggunaan dari program. [Roger S. Pressman, 2015 : 4]

Tujuan dari rekayasa perangkat lunak adalah :

1. Memperoleh biaya produksi perangkat lunak yang rendah.
2. Menghasilkan perangkat lunak yang kinerjanya tinggi, andal dan tepat waktu.
3. Menghasilkan perangkat lunak yang dapat bekerja pada berbagai jenis *platform*.
4. Menghasilkan perangkat lunak yang biaya perawatannya rendah.

Rekayasa perangkat lunak adalah teknologi berlapis. Hal itu dapat ditunjukan dengan gambar sebagai berikut :

Gambar 1.1 Lapisan Teknologi Rekayasa Perangkat Lunak.[Roger S. Pressman, 2015 : 16]

Dari gambar 2.1 dapat diartikan bahwa  rekayasa perangkat lunak ditujukan untuk peningkatan kualitas produk, fokus pada kualitas.

Dasar untuk rekayasa perangkat lunak adalah lapisan proses. Proses rekayasa perangkat lunak adalah proses yang terus berulang, karena karakteristik perangkat lunak yang membutuhkan pemeliharaan dan pengembangan berkelanjutan agar perangkat lunak tidak kadarluasa. Dalam proses pemeliharaan dilakukan koreksi kesalahan, adaptasi kebutuhan, peningkatan kemampuan atau fungsi dan bentuk pencegahan lainnya agar perangkat lunak tersebut tidak kadarluasa. [Roger S. Pressman, 2015 : 15]

Alat rekayasa perangkat lunak  merupakan unsur yang mendukung proses dan metode. Ketika alat-alat yang terhubung satu sama lain dan memberi  informasi, serta  informasi yang dibuat oleh salah satu alat dapat digunakan oleh yang lain, sistem untuk mendukung pengembangan perangkat lunak dapat dibangun dengan menggunakan bantuan komputer. [Roger S. Pressman, 2015 : 16]

Berikut fase generik yang berlaku untuk sebagian besar proyek perangkat lunak : [Roger S. Pressman, 2015 : 17]

1. **Komunikasi**

Kegiatan kerangka ini melibatkan komunikasi berat dan bekerja sama dengan pelanggan (dan *stakeholder* lainnya) dan mencakup pengumpulan persyaratan dan kegiatan terkait lainnya. Pada fase ini mengidentifikasi informasi apa yang akan diproses, apa fungsi dan kinerja yang diinginkan, perilaku sistem apa yang dapat diharapkan, apa antarmuka yang akan didirikan, apa desain kendala yang ada, dan apa kriteria validasi yang diperlukan untuk menentukan sistem yang sukses. Meskipun metode yang diterapkan selama fase definisi akan bervariasi tergantung pada paradigma rekayasa perangkat lunak (atau kombinasi paradigma) yang diterapkan, tiga tugas utama akan terjadi dalam beberapa bentuk: sistem atau teknik informasi, perencanaan proyek perangkat lunak, dan analisis kebutuhan.

1. **Perencanaan**

Kegiatan ini menetapkan rencana untuk pekerjaan rekayasa perangkat lunak yang mengikuti. Ini menggambarkan tugas-tugas teknis yang akan dilakukan, risiko yang mungkin, sumber daya yang akan diperlukan, produk pekerjaan yang harus diproduksi, dan jadwal kerja.

1. **Modelling**

Kegiatan ini meliputi pembuatan model pengembang dan pelanggan untuk lebih memahami kebutuhan perangkat lunak dan desain yang akan mencapai kebutuhan tersebut. Kegiatan ini akan semakin mendetailkan permasalahan dan bagaimana cara menyelesaikan masalah tersebut.

1. **Konstruksi**

Apa yang didesain sebelumnya akan mulai dibuat pada fase ini. Kegiatan ini menggabungkan kode generasi (baik manual atau otomatis) dan pengujian yang diperlukan untuk mengungkap kesalahan dalam kode.

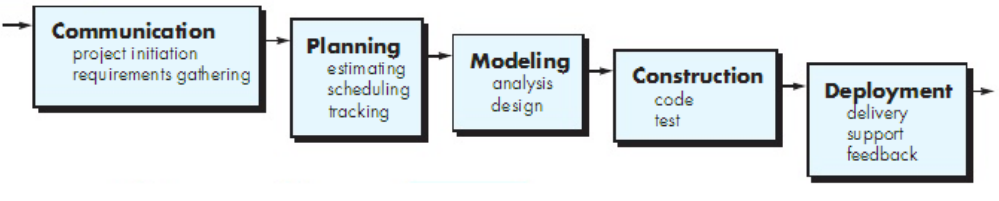
1. ***Deployment***

Perangkat lunak (sebagai entitas lengkap atau entitas yang sebagian sudah selesai) dikirimkan ke pelanggan agar pelanggan mengevaluasi perangkat lunak yang sudah dibuat dan memberikan umpan balik berdasarkan evaluasi tersebut agar bisa diperbaiki atau dikembangkan.

### **Pengembangan System Informasi**

Menurut Pressman model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Berikut ini ada dua gambaran dari *waterfall* model. [Roger S. Pressman, 2015]

Fase-fase dalam model *waterfall* menurut referensi Pressman:



* + 1. *Communication*

Langkah ini merupakan analisis terhadap kebutuhan *software*, dan tahap untuk mengadakan pengumpulan data dengan melakukan pertemuan dengan *customer,* maupun mengumpulkan data-data tambahan baik yang ada di jurnal, artikel, maupun dari internet.

* + 1. *Planning*

Proses *planning* merupakan lanjutan dari proses *communication (analysis requirement).* Tahapan ini akan menghasilkan dokumen *user requirement* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan *software*, termasuk rencana yang akan dilakukan.

* + 1. *Modeling*

Proses *modeling* ini akan menerjemahkan syarat kebutuhan ke sebuah perancangan *software* yang dapat diperkirakan sebelum dibuat *coding*. Proses ini berfokus pada rancangan struktur data, arsitektur *software*, representasi *interface*, dan detail (algoritma) prosedural. Tahapan ini akan menghasilkan dokumen yang disebut *software requirement*.

* + 1. *Construction*

*Construction* merupakan proses membuat kode. *Coding* atau pengkodean merupakan penerjemahan desain dalam bahasa yang bisa dikenali oleh komputer. *Programmer* akan menerjemahkan transaksi yang diminta oleh *user*. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu *software*, artinya penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap sistem yang telah dibuat tadi. Tujuan *testing* adalah menemukan kesalahan-kesalahan terhadap sistem tersebut untuk kemudian bisa diperbaiki.

* + 1. *Deployment*

Tahapan ini bisa dikatakan final dalam pembuatan sebuah *software* atau sistem. Setelah melakukan analisis, desain dan pengkodean maka sistem yang sudah jadi akan digunakan oleh *user*. Kemudian *software* yang telah dibuat harus dilakukan pemeliharaan secara berkala.

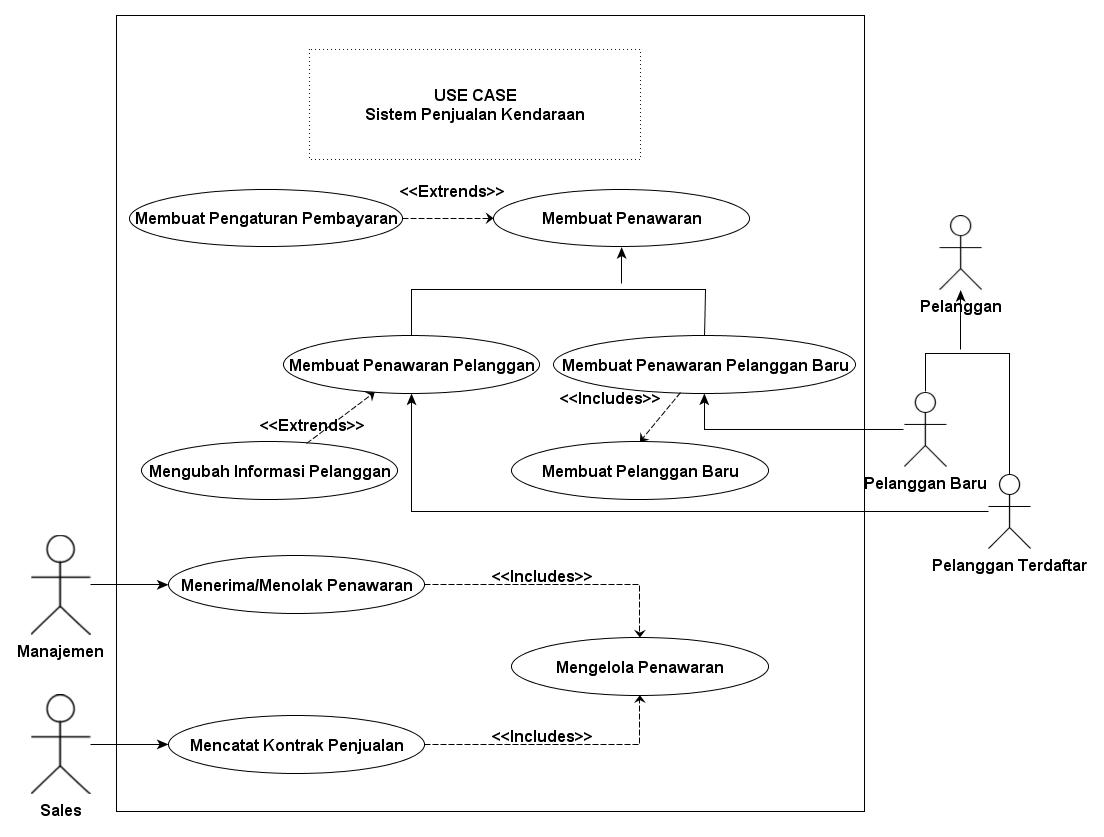
## Unified Model Language (UML)

Pemodelan rancangan program menggunakan UML *(Unified Modeling Language)*. Menurut Bernd Bruegge & Allen H. Dutoit, 2010 *Unified Modeling Language* (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat tool untuk mendukung pengembangan sistem tersebut. Tujuan dari *Unified Modeling Language* adalah untuk menyediakan istilah dalam hal teknik berbasis objek dan teknik menskemakan diagram yang cukup banyak untuk menggambarkan bermacam proyek pengembangan sistem mulai dari analisa sampai desain. (Dennis, Alan. 2012) Terdapat 4 bagian dalam UML, yaitu:

### **Use Case**

*Use case diagram* adalah gambar dari beberapa atau seluruh aktor dan *use case* dengan tujuan mengenali interaksi mereka dalam suatu sistem. *Use case diagram* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem, yang ditekankan adalah “apa” yang diperbuat sistem, buka “bagaimana”. Sebuah *use case* mempresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* menggambarkan kata kerja seperti login ke sistem, *maintenance user* dan sebagainya. Oleh karena itu, *use case diagaram* dapat membantu menganalisa kebutuhan suatu sistem. (Verdi Yasin, 2012:269)

*Use case diagram* digunakan oleh para analis untuk mengetahui lebih baik lagi mengenai fungsionalitas sistem dari tingkatan yang sangat tinggi. (Dennis, Alan. 2012).



Gambar 2.3 Contoh Use Case Diagram (Dennis, Allan. 2012)

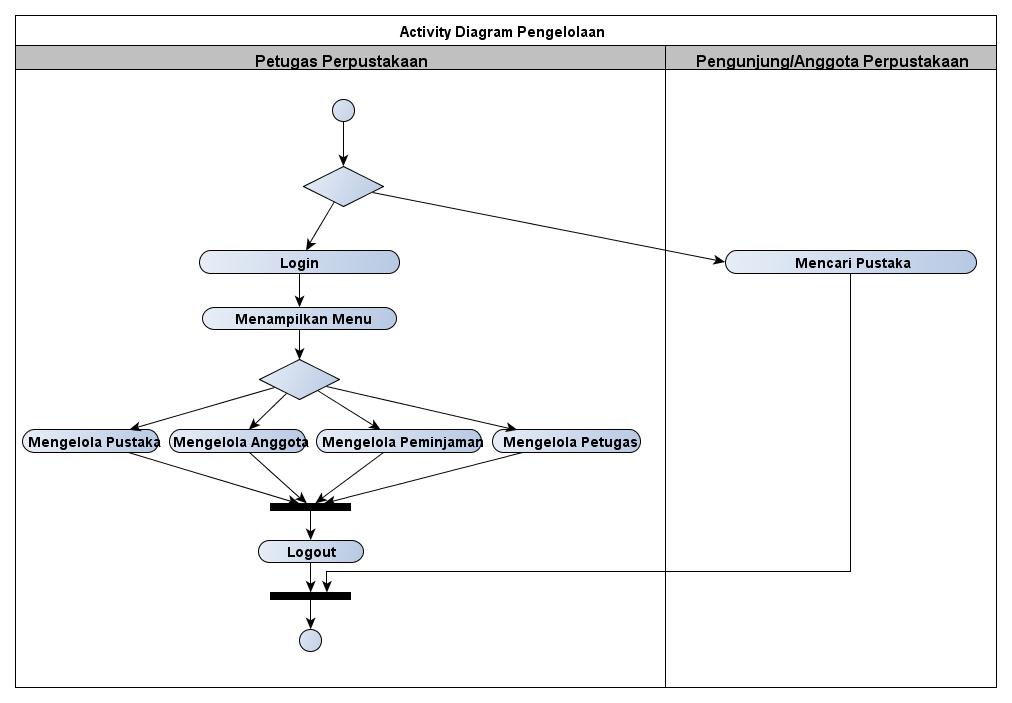
Di bawah ini merupakan elemen-elemen dari *use case diagram*:

Tabel 2.1 Elemen-Elemen Use Case Diagram (Dennis, Alan. 2012)

| **No.** | **Nama Elemen** | **Simbol** | **Definisi** |
| --- | --- | --- | --- |
| 1 | Batasan Sistem |  | * Batasan sistem terdiri dari nama sistem yang terdapat di dalam atau di atas simbol * Batasan sistem merupakan ruang lingkup system |
| 2 | Relasi Asosiasi |  | * Relasi asosiasi menghubungkan aktor dengan use case yang berinteraksi |
| 3 | Aktor |  | * Aktor adalah orang atau sistem yang memperoleh manfaat dari dan /atau di luar sistem. * Aktor harus diberikan label sesuai dengan peranannya * Aktor dapat dikaitkan dengan aktor-aktor lain oleh spesialisasi atau asosiasi *superclass* yang dilambangkan dengan anak panah dengan panah berongga * Aktor ditempatkan di luar batas sistem |
| 4 | Use Case |  | * Use case merupakan bagian utama dari fungsionalitas sistem * Use case dapat memperpanjang use case lain (extend) * Use case dapat menggunakan use case lain   Use case ditempatkan di dalam batas sistem   * Use case diberi label dengan penjelasan kata kerja atau kata benda |

### **Activity**

Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistem dari pada bagaimana sistem itu dirakit. Diagram aktivitas menunjukkan aktivitas sistem dalam bentuk kumpulan aksi-aksi. (Prabowo Pudjo Widodo Herlawati, 2011 : 143). Diagram aktivitas atau activity diagram menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. (Sommervile, Ian 2011).



Gambar 2.4 Contoh Activity Diagram (Sommervile, Ian. 2011)

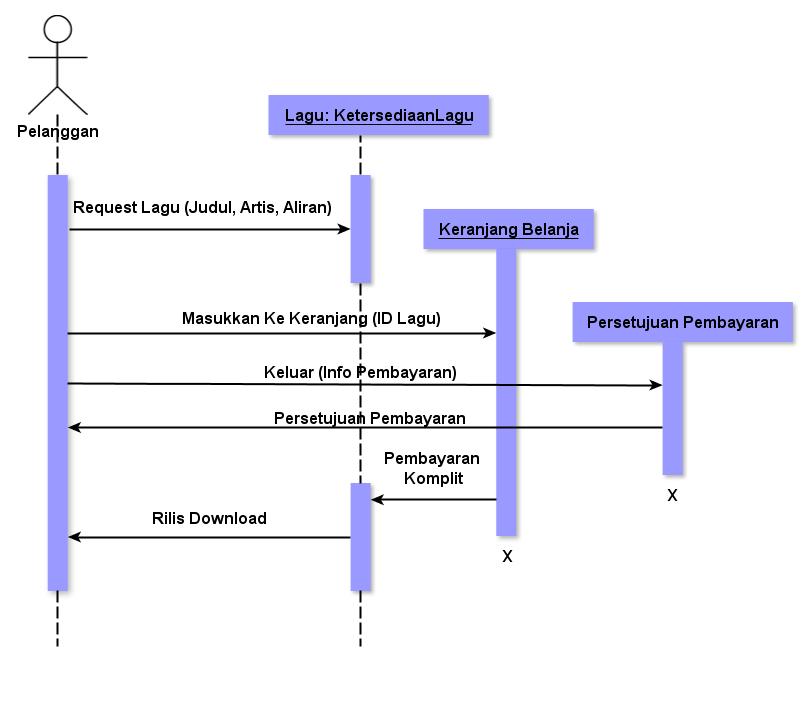
Berikut elemen-elemen dari *activity diagram:*

Tabel 2.2 Elemen-Elemen Activity Diagram (Sommervile, Ian. 2011)

| **No.** | **Nama Elemen** | **Simbol** | **Definisi** |
| --- | --- | --- | --- |
| 1 | Percabangan atau *Decision* | Description: Description: AD_Decision | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu |
| 2 | Status Awal |  | Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal |
| 3 | Aktivitas | Description: Description: AD_Aktivitas | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja |
| 4 | Penggabungan atau *Join* | Description: Description: AD_Join | Asosiasi penggabungan dimana lebih dari satu aktivitas yang digabungkan menjadi satu |
| 5 | Status Akhir |  | Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir |
| 6 | Swimlane | Description: Description: AD_Swimplane  Atau  Description: Description: AD_Swimplane | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi |

### **Sequence**

*Sequence diagram* menggambarkan objek-objek yang berperan serta dalam *use case* dan pesan-pesan yang dilewati oleh objek dari waktu ke waktu untuk sebuah *use case*. (Dennis, Alan. 2012). *Sequence diagram* merupakan model dinamis yang mendukung *dynamic view* untuk pengembangan sistem. *Sequence diagram* menunjukkan urutan yang eksplisit yang dilewati antar objek dalam suatu interaksi yang didefinisikan. Dikarenakan *sequence diagram* menekankan pada urutan kegiatan berbasis waktu, maka dapat sangat membantu dalam memahami spesifikasi secara *real-time* dan dapat digunakan untuk membantu memahami *use case* yang kompleks. Berikut ini adalah contoh *sequence diagram*:



Gambar 2.5 Contoh Sequence Diagram (Dennis, Allan. 2012)

Berikut ini adalah elemen-elemen dari sequence diagram:

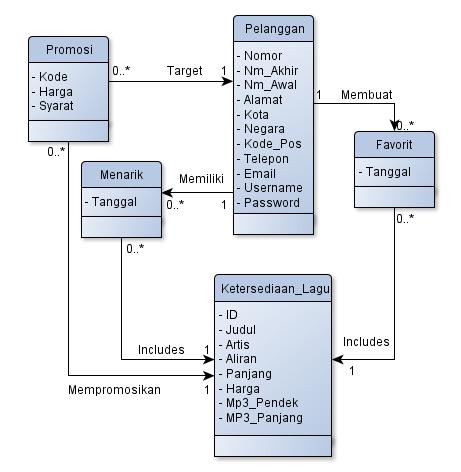
Tabel 2.3 Elemen-Elemen Sequence Diagram (Dennis, Allan. 2012)

| **No.** | **Nama Elemen** | **Simbol** | **Definisi** |
| --- | --- | --- | --- |
| 1 | Aktor |  | * Aktor adalah orang atau sistem yang memperoleh manfaat dari dan di luar sistem * Aktor berpartisipasi secara berurutan dengan mengirimkan atau menerima pesan * Aktor ditempatkan di bagian atas diagram |
| 2 | Object | Object | * Objek berpartisipasi secara berurutan dengan mengirimkan atau menerima *message* * Objek ditempatkan di bagian atas diagram |
| 3 | Lifeline |  | *Lifeline* menunjukkan masa dari sebuah objek dalam sebuah urutan *Lifeline* berisi X pada suatu poin dimana kelas atau objek tidak lagi berinteraksi |
| 4 | Focus of control |  | * *Focus of control* berupa persegi panjang yang tipis ditempatkan di atas *lifeline* * *Focus of control* ditunjuk ketika sebuah objek sedang mengirim atau menerima *message* |
| 5 | Message |  | *Message* menyampaikan informasi dari satu objek ke objek lain |
| 6 | Object destruction | **X** | *Object destruction* berupa simbol X ditempatkan pada akhir dari suatu objek *lifeline* untuk menunjukkan bahwa *lifeline* akan berakhir |

### **Class**

*Class diagram* merupakan model statis yang mendukung *static view* pada pengembangan sistem. *Class diagram* menampilkan kelas-kelas dan relasi kelas yang terdapat dalam sebuah sistem dalam waktu ke waktu, secara konstan. (Dennis, Alan. 2012).

Berikut ini adalah contoh class diagram:



Gambar 2.5 Contoh Class Diagram (Dennis, Allan. 2012)

Berikut ini adalah elemen-elemen dari *class diagram*:

Tabel 2.4 Elemen-Elemen Class Diagram (Dennis, Allan. 2012)

| **No.** | **Nama Elemen** | **Simbol** | **Definisi** |
| --- | --- | --- | --- |
| 1 | Class |  | * Class dapat berupa jenis orang, tempat atau mengenai suatu hal yang harus didapatkan sistem dan disimpan informasinya. * Class mempunyai nama class yang ditulis dengan huruf tebal dan berada di atas bagiannya (kompartemen). * Class mempunyai daftar atribut di tengah bagiannya * Class mempunyai daftar operasi di bagian paling bawah * Class tidak secara eksplisit menunjukkan operasi yang tersedia untuk semua class |
| 2 | Atribut | Nama Atribut | * Atribut menggambarkan properties yang menggambarkan kondisi dari suatu objek. * Atribut dapat berasal dari atribut lain, ditunjukkan dengan menempatkan garis miring sebelum nama atribut itu |
| 3 | Metode | Operation name () | * Metode menggambarkan tindakan atau fungsi yang dapat dilakukan sebuah class * Metode dapat diklasifikasikan sebagai konstruktor, query atau operasi update * Metode terdiri dari tanda kurung yang dapat mengandung parameter khusus atau informasi yang dibutuhkan untuk melakukan operasi |
| 4 | Asosiasi |  | * Asosiasi menggambarkan hubungan antara beberapa class, atau class dengan dirinya sendiri. * Asosiasi diberi label kata kerja atau nama peran, tergantung yang mana yang lebih baik yang menggambarkan relasi * Asosiasi terdapat di antara satu class atau lebih * Asosiasi berisi simbol multiplicity, yang mewakili jumlah minimum dan maksimum suatu class instance dapat dikaitkan dengan class intance yang lain |

## Algoritma Dijkstra

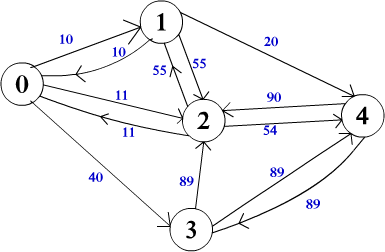
Sudah banyak algoritma yang bisa digunakan untuk tujuan pencarian rute terpendek. Tidak bisa di pungkiri Djikstra masih menjadi salah satu yang populer dari sekian banyak algoritma tersebut (Setiawan, 2015).

Algoritma Dijkstra, dinamakan sesuai dengan nama penemunya, seorang ilmuwan komputer berkebangsaan Belanda yang bernama Edsger Dijkstra, adalah algoritma yang digunakan untuk mencari lintasan terpendek pada sebuah graf berarah.

Cara kerja algoritma Dijkstra memakai stategi greedy, dimana pada setiap langkah dipilih sisi dengan bobot paling kecil yang menghubungkan sebuah simpul yang sudah terpilih dengan simpul lain yang belum terpilih.

Djikstra merupakan salah satu varian bentuk algoritma popular dalam pemecahan persoalan terkait masalah optimasi pencarian lintasan terpendek. Sebuah lintasan yang mempunyai panjang minimum dari verteks a ke z dalam *graph* berbobot. Bobot tersebut adalah bilangan positif jadi tidak dapat dilalui oleh node negatif. Namun jika terjadi demikian, maka penyelesaian yang diberikan adalah infiniti (Tak Hingga). Pada algoritma Dijkstra, node digunakan karena algoritma Dijkstra menggunakan graph berarah untuk penentuan rute listasan terpendek.

Algoritma Djikstra bertujuan untuk menemukan jalur terpendek berdasarkan bobot terkecil dari satu titik ke titik lainnya. Misalkan titik mengambarkan gedung dan garis menggambarkan jalan, maka algoritma Dijkstra melakukan kalkulasi terhadap semua kemungkinan bobot terkecil dari setiap titik.



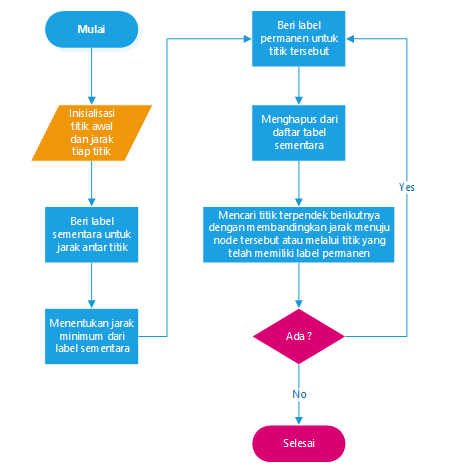
**Gambar 2.12** Contoh Ketersambungan Titik dalam Algoritma Dijkstra

Pertama-tama tentukan titik mana yang akan menjadi node awal, lalu beri bobot jarak pada node pertama ke node terdekat satu per satu, Dijkstra akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap. Inilah urutan logika dari algoritma Dijkstra (Setiawan, 2015):

* + - 1. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada node awal dan nilai tak hingga terhadap node lain (belum terisi)
      2. Set semua node “Belum terjamah” dan set node awal sebagai “Node keberangkatan”.
      3. Dari node keberangkatan, pertimbangkan node tetangga yang belum terjamah dan hitung jaraknya dari titik keberangkatan. Sebagai contoh, jika titik keberangkatan A ke B memiliki bobot jarak 6 dan dari B ke node C berja

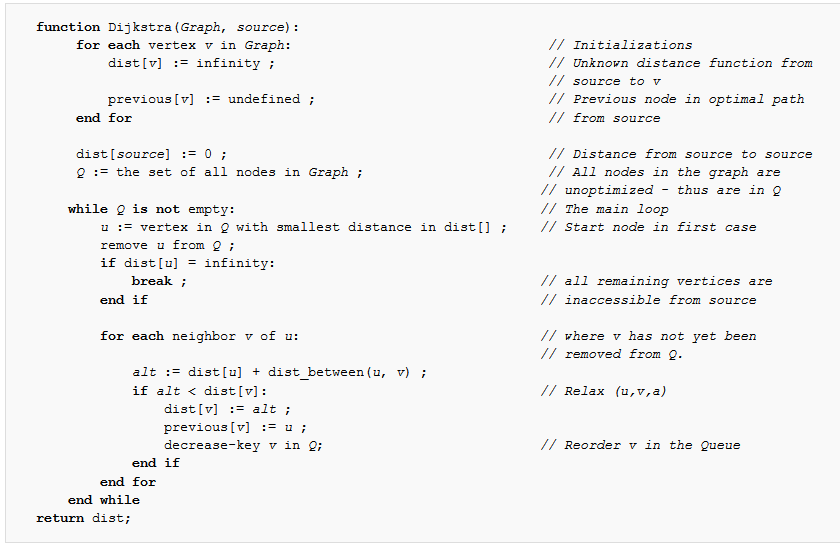
rak 2, maka jarak ke C melewati B menjadi 6+2=8. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.

* + - 1. Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah terjamah sebagai “Node terjamah”. Node terjamah tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
      2. Set “Node belum terjamah” dengan jarak terkecil (dari node keberangkatan) sebagai “Node Keberangkatan” selanjutnya dan lanjutkan dengan kembali ke step 3.



Flowchart Algoritma Dijkstra

Berikut adalah *pseudo-code* algoritma Djikstra



## Tools Yang Digunakan

### **Java**

Menurut (Jubille Enterprise 2015 : 1) Java merupakan bahasa pemrograman berorientasi objek yang dikembangkan oleh Sun Microsystem yang dimulai oleh James Gosling dan dirilis pada tahun 1995, saat ini Sun Microsystem telah di akuisisi oleh Oracle Corporation.

Java bersifat Write Once, Run Anywhere (program yang dituliskn satu kali dan dapat berjalan pada banyak platform).

Menurut (Bambang Hariyanto 2014 : 1) Bahasa Java memberi harapan mejadi perekat universal yang mengkoneksi pemakai dengan informasi dari web server, basis data, penyedia informasi dan sumber – sumber lain. Bahasa Java memiliki fitur keamanan built-in. Bahasa Java juga mempermudah pemrograman aplikasi multithreading.

Java adalah sebuah bahasa pemrograman berorientasi obyek murni. Jadi program- program Java berada dalam sebuah struktur kelas-kelas dan obyek-obyek. Pada dasarnya sintaks pada bahasa Java mirip dengan sintaks pada bahasa C atau C++. Java bertipe kuat (strongly-typed). Ini berarti semua tipe data terikat secara statis atau dengan kata lain setiap nama variabel diasosiasikan dengan sebuah tipe data tunggal yang dikenali pada saat kompilasi.

### **Hypertext Markup Language (HTML)**

Hypertext Markup Language (HTML) adalah bahasa standard yang digunakan untuk menampilkan halaman web. [Hidayatullah, Priyanto. 2015].

Yang bisa dilakukan dengan HTML yaitu :

* + - 1. Mengatur tampilan dari halaman *web* dan isinya.
      2. Membuat table dalam halaman web.
      3. Mempublikasikan halaman web secara online.
      4. Membuat form yang bisa digunakan untuk menangani registrasi dan transaksi via web.
      5. Menambahkan objek-objek seperti citra, audio, video, animasi, java applet dalam halaman web.
      6. Menampilkan area gambar (*canvas*) di browser.

Berikut ini adalah elemen-elemen yang digunakan pada HTML:

Tabel 2.5 Elemen-Elemen HTML [Suehring, Steve. 2013]

| **Elemen** | **Keterangan** |
| --- | --- |
| <a> | Mendefinisikan sebuah anchor, tetapi lebih tepat jika diartikan sebagai tautan dikarenakan tautan ini digunakan untuk saling menautkan antara satu dokumen HTML ke dokumen HTML yang lain |
| <body> | Mendefinisikan *body* atau isi dokumen HTML, berfungsi untuk menentukan bagaimana isi suatu dokumen ditampilkan di web browsernya. Isi dokumen tersebut dapat berupa teks, gambar, animas, link dan seterusnya |
| <br /> | Memberi baris baru atau pindah baris |
| <col /> | Mendefinisikan nilai atribut dari satu kolom atau lebih dalam sebuah tabel |
| <colgroup> | Menentukan kelompok dari satu kolom atau lebih dalam sebuah tabel untuk performatan |
| <div> | Mendefinisikan sebuah section dalam dokumen |
| <font> | Mendefinisikan jenis font, warna dan ukuran untuk teks |
| <form> | Mendefinisikan sebuah form HTML untuk input form |
| <frame /> | Mendefinisikan frame dalam fremeset |
| <frameset> | Mendefinisikan satu set frame |
| <h1> to <h6> | Digunakan untuk menunjukkan awal dari suatu *header* atau judul dari dokumen HTML tersebut. |
| <head> | Digunakan untuk memberikan informasi tentang dokumen tersebut |
| <html> | Mendefinisikan root dari suatu dokumen HTML |
| <img /> | Berfungsi untuk menampilkan gambar pada dokumen HTML |
| <input /> | Mendefinisikan input field pada form |
| <li> | Digunakan untuk menampilkan informasi dalam bentuk item daftar |
| <link /> | Mendefinisikan hubungan antara dokumen dan sumber eksternalnya |
| <object> | Meletakkan embed sebuah objek |
| <ol> | Mendefinisikan daftar dalam format penomoran |
| <option> | Menampilkan beberapa pilihan yang berbentuk dalam sebuah daftar drop-down |
| <p> | Membuat sebuah paragraf |
| <param /> | Mendefinisikan sebuah parameter untuk objek |
| <pre> | Membuat teks dengan ukuran huruf yang sama |
| <script> | Mendefinisikan client-side script |
| <span> | Mendefinisikan sebuah section dalam dokumen |
| <table> | Membuat table |
| <td> | Mendefinisikan sel di dalam sebuah tabel |
| <textarea> | Mendefinisikan sebuah kontrol input multiline |
| <th> | Mendefinisikan sel header di dalam  sebuah tabel |
| <title> | Membuat judul untuk dokumen HTML |
| <tr> | Membuat baris di dalam sebuah tabel |
| <u> | Membuat teks bergaris bawah, fungsi tag ini sama dengan tag <ins> tetapi tag <u> tidak dianjurkan untuk kategori HTML text formatting melainkan termasuk kategori HTML Style |
| <ul> | Mendefinisikan daftar dalam format bullet |
| <var> | Mendefinisikan sebuah variabel |

### **DBMS dan MySQL**

*Database Management System* (DBMS) adalah aplikasi yang dipakai untuk mengelola basis data. DBMS biasanya menawarkan beberapa kemampuan yang terintegrasi seperti:

1. Membuat, menghapus, menambah dan memodifikasi basisdata.
2. Pada beberapa DBMS pengelolaannya berbasis windows (bentuknya seperti jendela-jendela) sehingga mudah digunakan.
3. Tidak semua orang bisa mengakses basisdata yang ada sehingga memberikan keamanan bagi data.
4. Kemampuan berkomunikasi dengan program aplikasi yang lain. Misalnya dimungkinkan untuk mengakses basis data MySQL menggunakan aplikasi yang dibuat menggunakan PHP.
5. Kemampuan pengaksesan melalui komunikasi antar computer (client server).

MySQL adalah salah satu aplikasi DBMS yang sudah sangat banyak digunakan oleh para pemrogram aplikasi web. Contoh DBMS lainnya adalah : PostgreSQL (freeware), SQL Server, MS Access dari Microsoft, DB2 dari IBM, Oracle, Foxpro dan sebagainya.

Kelebihan dari MySQL adalah gratis, handal, selalu di-update dan banyak forum yang memfasilitasi para penggunak jika memiliki kendala. MySQL juga menjadi DBMS yang sering dibundling dengan web server sehingga instalasinya jadi lebih mudah. [Hidayatullah, Priyanto. 2015]

## Whitebox

White Box Testing adalah salah satu cara untuk menguji suatu aplikasi atau software dengan cara melihat modul untuk dapat meneliti dan menganalisa kode dari program yang di buat ada yang salah atau tidak. Kalau modul yang telah dan sudah di hasilkan berupa output yang tidak sesuai dengan yang di harapkan maka akan dikompilasi ulang dan di cek kembali kode-kode tersebut hingga sesuai dengan yang diharapkan (Nidhra and Dondetti, 2012).

Kasus yang sering menggunakan white box testing akan di uji dengan beberapa tahapan yaitu:

1. Pengujian seluruh keputusan yang menggunakan logikal.

2. Pengujian keseluruh loop yang ada sesuai batasan-batasannya.

1. Pengujian seluruh keputusan yang menggunakan logical.
2. Pengujian keseluruhan loop yang ada sesuai batasan-batasannya.
3. Pengujian pada struktur data yang sifatnya internal dan yang terjamin validitasnya.

Kelebihan White Box Testing antara lain (Nidhra and Dondetti, 2012) :

1. Kesalahan Logika

Menggunakan sintax ‘if’ dan sintax pengulangan. Langkah selanjutnya metode white box testing ini akan mencari dan mendeteksi segala kondisi yang di percaya tidak sesuai dan mencari kapan suatu proses perulangan di akhiri.

1. Ketidaksesuaian Asumsi

Menampilkan dan memonitor beberapa asumsi yang diyakini tidak sesuai dengan yang diharapkan atau yang akan diwujudkan, untuk selanjutnya akan dianalisa kembali dan kemudian diperbaiki.

1. Kesalahan Pengetikan

Mendeteksi dan mencaribahasa-bahasa pemograman yang di anggap bersifat case sensitif.

Kelemahan White Box Testing adalah pada perangkat lunak yang jenisnya besar, metode white box testing ini dianggap boros karena melibatkan banyak sumberdaya untuk melakukannya. (Nidhra and Dondetti, 2012)

# METODE PENELITIAN

## Metode Penelitian

Dalam penyusunan Laporan Tugas Akhir ini, metode penelitian yang digunakan penulis yaitu dengan cara mengamati mengenai keadaan secara langsung dari atau tepatnya yang menjadi objek penelitian. Teknik pengumpulan data yang penulis lakukan dalam mencari dan mengumpulkan data serta mengolah informasi yang diperlukan menggunakan beberapa metode sebagai berikut :

**3.1.1 Metode Pengamatan (Observasi Research)**

Penulis mengadakan pengamatan langsung mahasiswa, metode ini dilakukan untuk mengumpulkan data sebagai sumber informasi yang sangat penting yang dapat membantu dalam analisa dan untuk langkah selanjutnya dalam rangka pembangunan sistem tersebut.

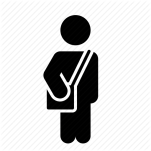
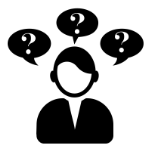
### **Metode Wawancara (Interview Research)**

Metode ini dilakukan melalui proses tanya jawab dengan seorang atau beberapa narasumber di tempat atau lokasi dimana objek penelitian dilakukan. Proses tanya jawab ini dilakukan langsung kepada mahasiswa dan dosen.

### **Metode Pustaka (Library Research)**

Yaitu memperoleh data dengan cara membaca dan mempelajari buku-buku dan literature-literature yang berhubungan dengan teori dan laporan penelitian ini.

## Analisa Sistem Berjalan

Gambar 3.2 Rich Diagram

Universitas Mercubuana saat ini memiliki 46 tempat yang bisa didatangi, namun saat ini belum ada aplikasi yang dapat memberikan rute yang dapat dilalui para mahasiswa untuk mendapatkan informasi menuju ruangan atau gedung yang akan dituju.

Saat ini biasanya, akan ada beberapa skenario untuk mencari suatu gedung ataupun ruangan yang akan dituju oleh mahasiswa, namun pada umumnya akan terjadi skenario seperti :

1. Mahasiswa berangkat dari tempat tinggalnya

2. Mahasiswa sampai di Universitas Mercubuana

3. Setelah masuk kedalam Universitas Mercubuana, pastilah mahasiswa tersebut akan menuju ke suatu tempat, misalnya : ruangan kelas, ruang dosen atau kantin. Namun jika mahasiswa tersebut tidak mengetahui rute mana yang harus dilalui untuk sampai ke tempat tujuannya.

4. Setelah itu pastilah mahasiswa tersebut bertanya kepada mahasiswa / dosen / orang disekitar mahasiswa tersebut berada kemana arah untuk dapat menuju ke tempat tujuan yang ingin mahasiswa tersebut tuju.

5. Setelah bertanya barulah mahasiswa tersebut mencari tempat berdasarkan petunjuk yang diberikan.

6. Terakhir, sampailah mahasiswa tersebut ke tempat yang dituju.

Oleh karena itu dibutuhkan suatu aplikasi agar mahasiswa ataupun dosen dapat mendapatkan informasi tempat dan rute untuk dapat menuju dari posisinya sekarang ke tempat yang akan dituju.

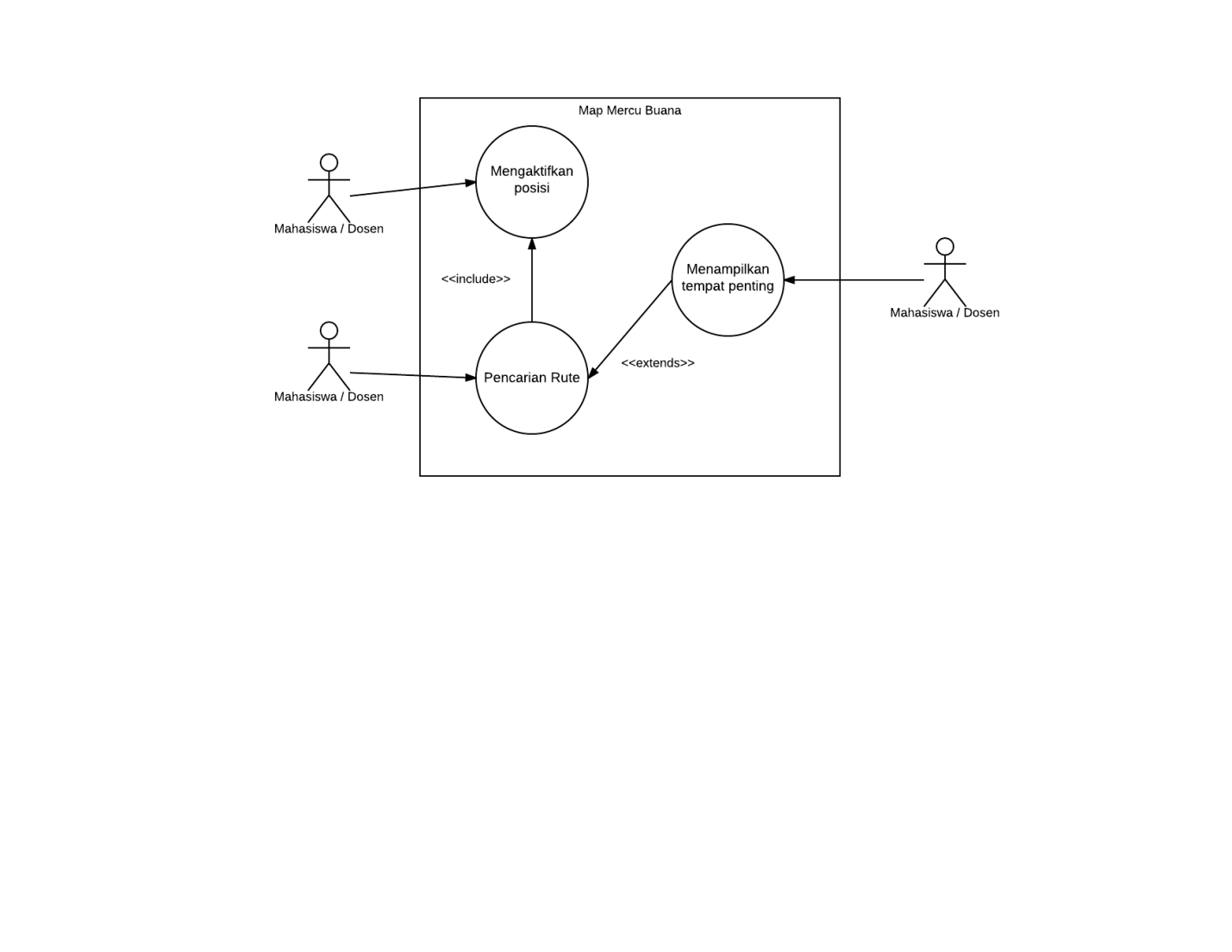
**BAB 4. PERANCANGAN**

## Perancangan Sistem

Berdasarkan analisa kebutuhan sistem yang telah dijelaskan sebelumnya perancangan sistem yang diusulkan adalah sebagai berikut:

### **Use Case Diagram**

Pada *design use case* digambarkan interaksi antara actor terhadap sistem web sinaptika berikut ini adalah gambar 3.3 *use case diagram*



<<include>>

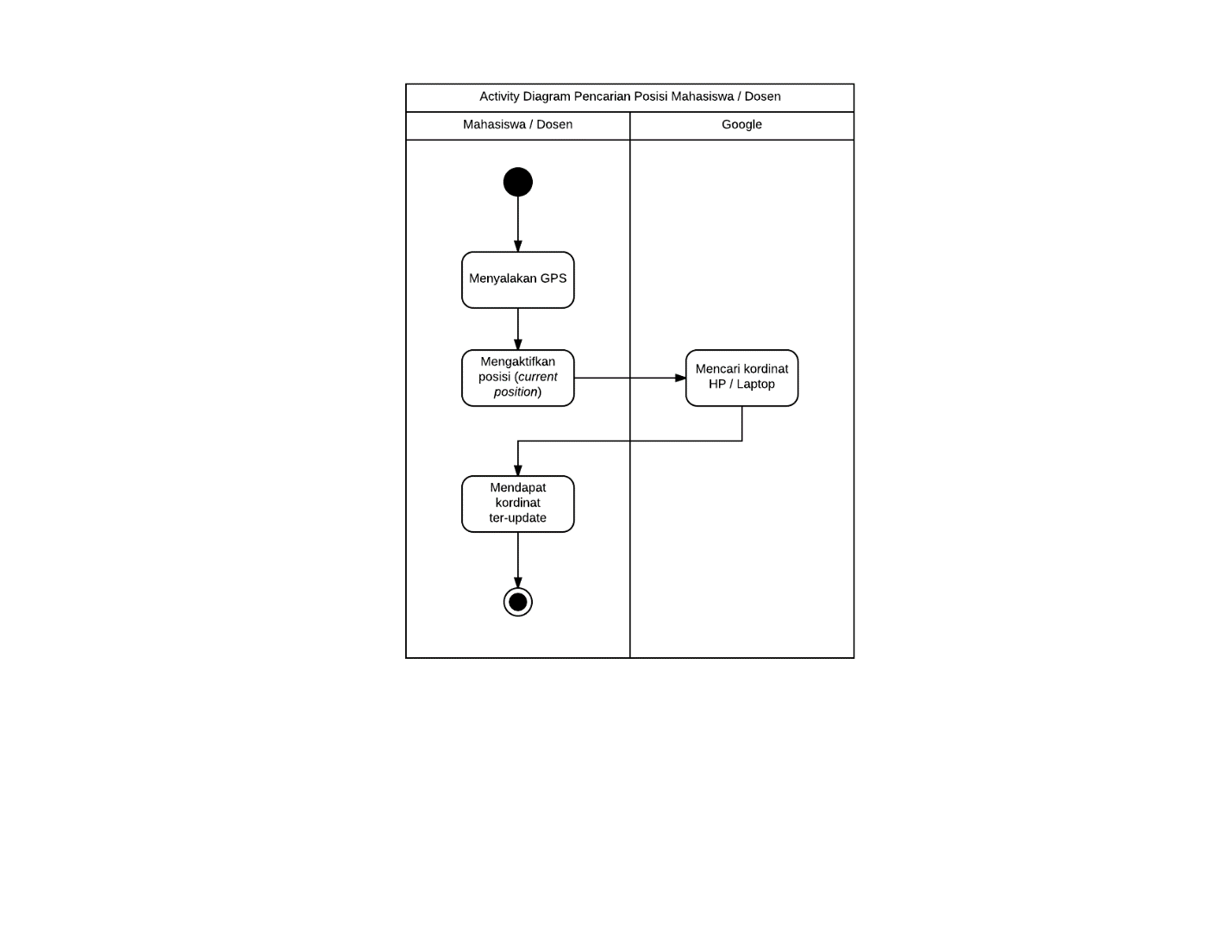
|  |  |  |
| --- | --- | --- |
| Use Case Name | Mencari Rute | |
| Aktor | Mahasiswa / Dosen | |
| Description | *Use case* mencari rute berfungsi untuk melihat jalur yang dapat dilalui oleh mahasiswa / dosen untuk menuju suatu tempat. Pada *use case* tersebut juga menampilkan keterangan-keterangan pembantu yang dapat memberikan informasi kepada mahasiswa / dosen untuk dapat menuju tempat tujuan disamping rute yang harus dilalui. | |
| Basic Flow | Aktor | Sistem |
|  | 1. Melihat peta Universitas Mercu Buana | 1. Menampilkan peta Universitas Mercu Buana |
|  | 1. Melihat rute yang dapat dilalui dari posisi mahasiswa ke posisi gedung yang dituju | 1. Mencari dan menampilkan rute yang harus dilalui untuk menuju dari posisi mahasiswa ke posisi gedung yang dituju. |
|  | 1. Melihat keterangan kemana mahasiswa/dosen harus pergi untuk bisa ke tempat tujuan (jika ruangan terdapat diatas lantai 1). | 1. Menampilkan keterangan untuk dapat menuju ke tempat tujuan. |
| Pre-Condition | Membuka aplikasi dan memilih gedung / ruangan mana yang akan dituju. | |
| Post-Condition | Mendapatkan informasi dari aplikasi rute pencarian Universitas Mercu Buana | |

|  |  |  |
| --- | --- | --- |
| Use Case Name | Mengaktifkan Posisi | |
| Aktor | Mahasiswa / Dosen | |
| Description | *Use case* mengaktifkan rute berfungsi untuk mendapatkan *longitude* dan *latitude* dan kemudian mengupdate lokasi aktor pada device yang digunakannya. | |
| Basic Flow | Aktor | Sistem |
|  | 1. Mengupdate posisi (*current position*) | 1. Menampilkan lokasi ter*update* pada device aktor |
| Pre-Condition | Mengaktifkan GPS | |
| Post-Condition | Membuka aplikasi dan mengaktifkan posisi untuk mendapatkan *update position*. | |

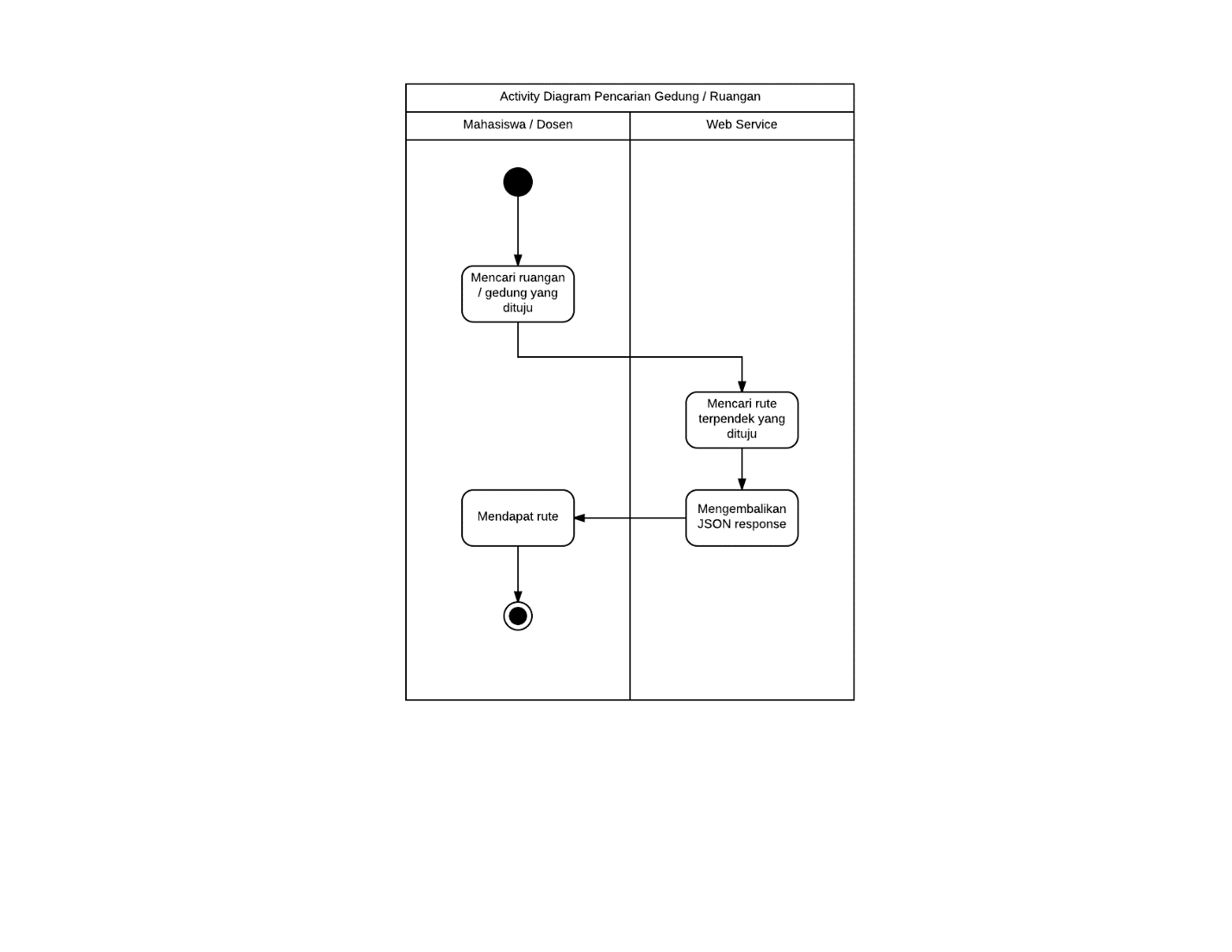
|  |  |  |
| --- | --- | --- |
| Use Case Name | Menampilkan Tempat Penting | |
| Aktor | Mahasiswa / Dosen | |
| Description | *Use case* menampilkan tempat penting berfungsi sebagai *Quick Access* bagi actor untuk dapat menuju ke tempat yang sering dikunjungi oleh mahasiswa dengan hanya memilih secara langsung *list* tempat yang sering dikunjungi oleh mahasiswa tanpa harus mengetik lokasi tempat yang akan dituju. | |
| Basic Flow | Aktor | Sistem |
|  | 1. Memilih *list* tempat yang sering dikunjungi, seperti : Kantin, Mushala, Ruang Dosen, dll. | 1. Menampilkan rute ketempat yang akan dituju oleh aktor. |
| Pre-Condition | Membuka menu *list* tempat yang sering dituju. | |
| Post-Condition | Mendapat *list* menu tempat yang sering dituju. | |

### **Activity Diagram**

Pada *Activity Diagram* digambarkan aktivitas ataupun alur dari sebuah proses system dan actor dengan menggunakan *swimeline* berikut adalah gambar 3.4 *activity diagram*

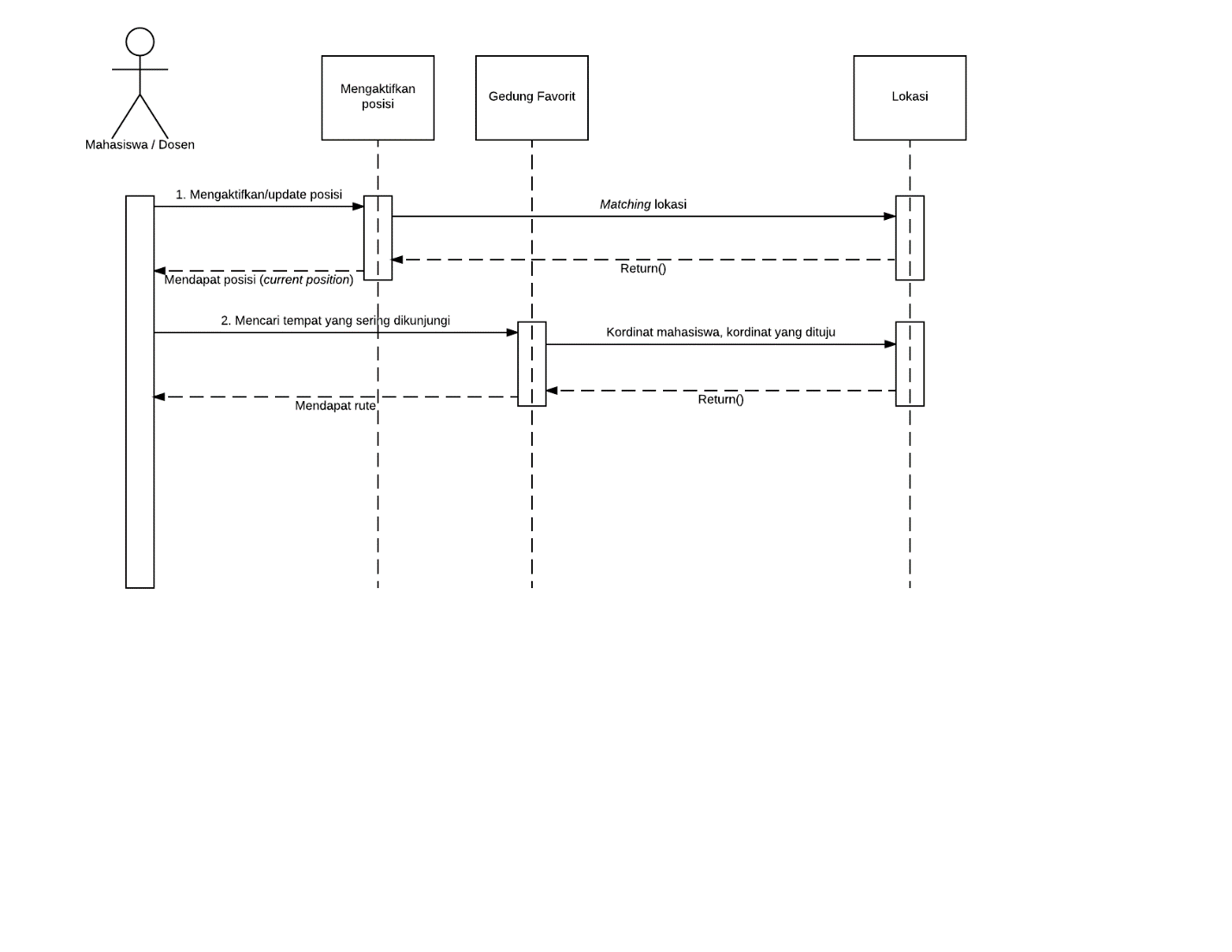


Pada gambar diatas mahasiswa / dosen harus menyalakan GPS terlebih dahulu sebelum menggunakan aplikasi, setelah itu mahasiswa / dosen harus mengupdate posisi dimana dia berada sekarang agar posisi yang ditampilkan oleh *device* mahasiswa / dosen lebih akurat, setelah itu mahasiswa / dosen dapat mencari gedung / ruangan yang akan dituju dan akan mendapatkan rute yang dapat dilalui untuk dapat mencapai ke gedung / ruangan yang dituju tersebut, seperti yang ditunjukan pada gambar 3.X. Aktivity Diagram.



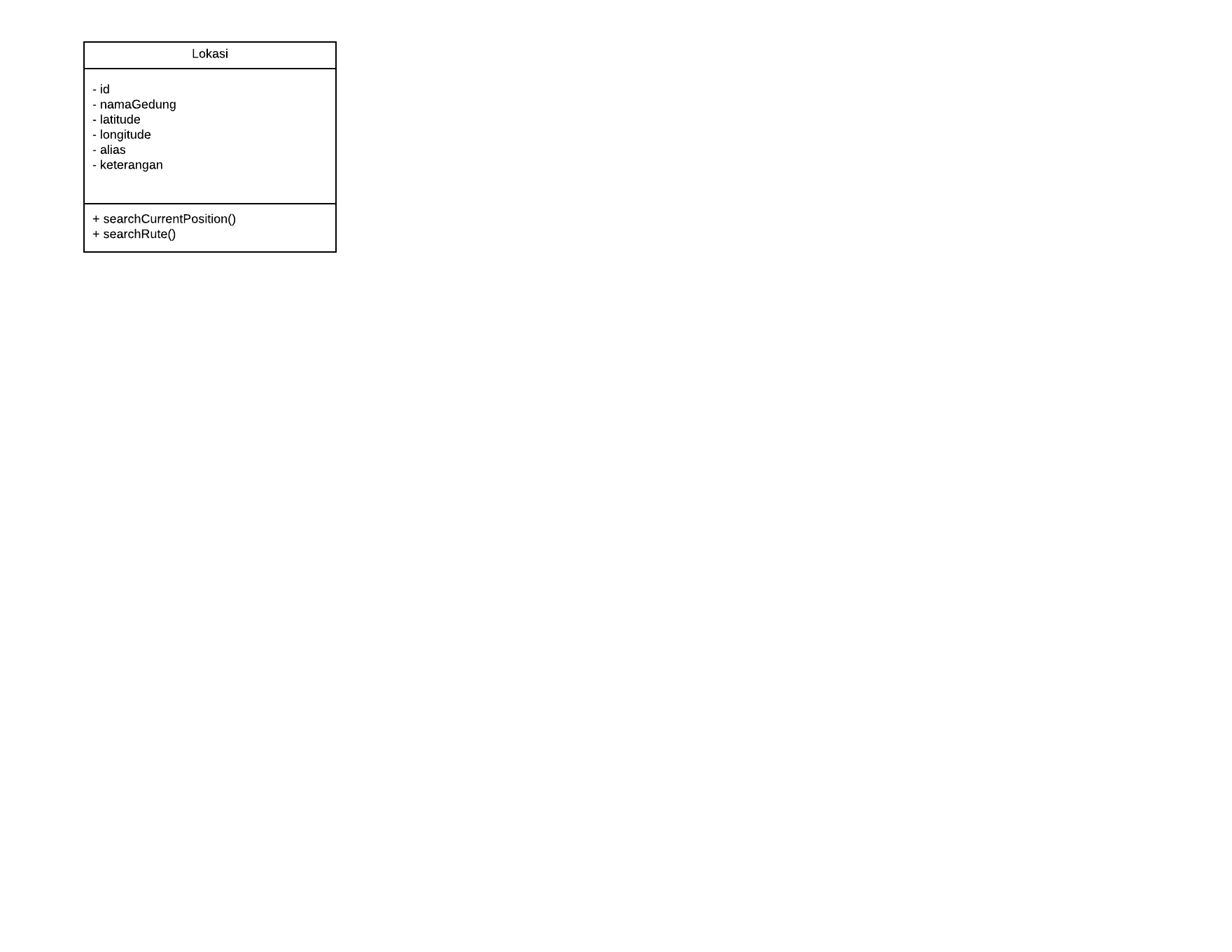
### **Sequence Diagram**

*Sequence diagram* menggambarkan tahapan-tahapan yang harus dilakukan oleh setiap aktor atau objek pada sistem aplikasi*. Sequence diagram* memperjelas bagaimana sebuah aplikasi berjalan hingga proses *database*. Pada pembangunan aplikasi ini terdapat beberapa *sequence diagram* yang dirancang, berikut ini adalah *sequence diagram* dari aplikasi ini:



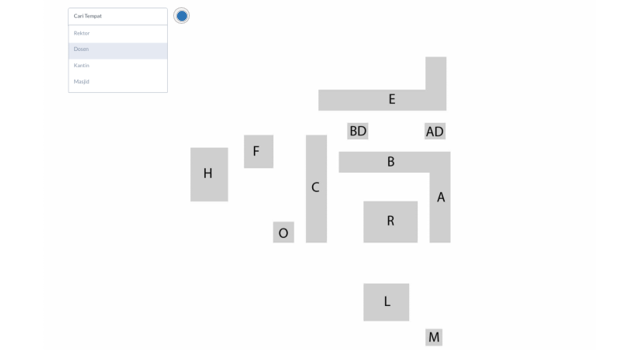
### **Class Diagram**

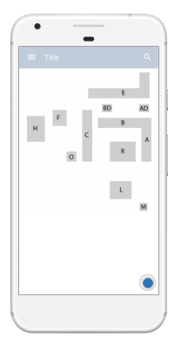
Untuk mendukung tampilan data yang ada pada aplikasi, maka disusunlah *class diagram*. *Class diagram* menggambarkan class-class yang saling berelasi sehingga dapat memberikan informasi sesuai kebutuhan. Berikut ini adalah rancangan *class diagram* pada aplikasi yang dibangun:



## Perancangan Antarmuka

Perancangan antarmuka digunakan untuk menciptakan tampilan agar mudah digunakan dan mudah dipahami oleh pengguna yang akan menggunakan aplikasi pencarian rute terpendek ini. Pada rancangan aplikasi, terdiri dari 1 tampilan map Universitas Mercubuana Meruya dan 1 menu yang memuat sekitar 13 tombol untuk dapat memilih tempat yang akan dikunjungi. Rancangan antarmuka utama pada versi *mobile* dan web penelitian ini sebagai berikut :

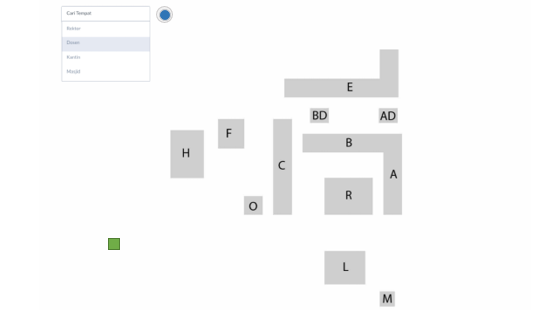


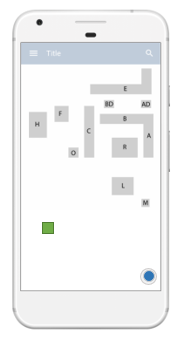


Terlihat pada gambar X.X terdapat layout yang menggambarkan map dari Universitas Mercubuana Meruya. Pada versi web, terdapat sebuah tombol berbentuk lingkaran berwarna biru disebelah input pencarian dan pada versi mobile berada disebelah pojok kanan bawah, dimana tombol ini berfungsi untuk mengupdate lokasi tempat kita berada sekarang. Sebelum menjalankan fungsi dari tombol ini, diharapkan pengguna untuk mengaktifkan GPS agar hasil yang digunakan bersifat akurat, namun jikapun tidak mengaktifkan GPS, aplikasi dapat tetap berjalan dengan menjadikan default posisi pengguna berada menjadi dipintu masuk Universitas Mercubuana Meruya. Pada aplikasi ini, terdapat 3 buah macam gambar yang berbeda-beda warnanya untuk mengambarkan posisi start, rute dan end dari sebuah rute. Gambar tersebut dapat dideskripsikan sebagai berikut :

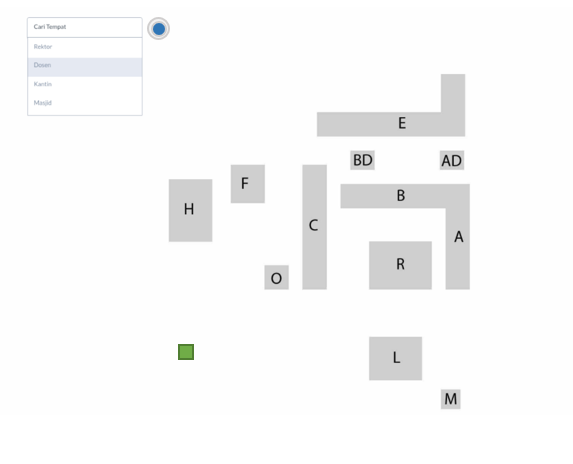
* + - 1. = Start
      2. = End
      3. = Rute yang dapat dilewati

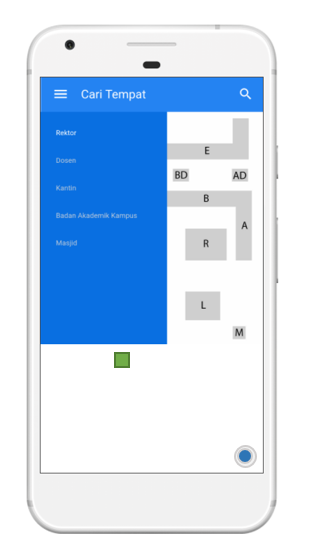
Berikut tampilan saat aplikasi saat pengguna mengupdate posisinya :



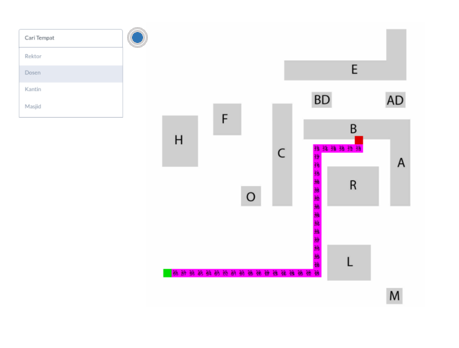


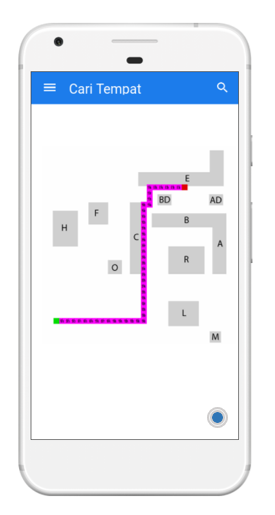
Kemudian pojok kiri atas, terdapat text input auto complete untuk mencari tempat yang akan dituju sesuai apa yang diketikkan oleh pemakai aplikasi. Pada menu tersebut pula terdapat tombol browse yang dapat memunculkan menu quick access dimana pada menu tersebut terdapat list seluruh gedung atau tempat yang dapat dipilih dengan cara menekan tombol berdasarkan tempat yang akan dikunjungi. Berikut adalah tampilan layout saat kita menekan tombol browse :





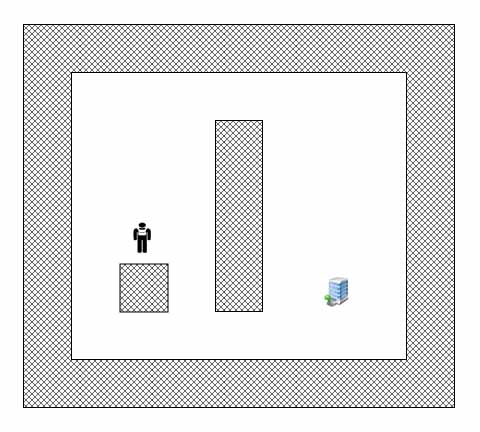
Lalu setelah kita menekan salah satu tombol yang terdapat dimenu tersebut, aplikasi akan secara otomati mencari rute terpendek berdasarkan lokasi pengguna aplikasi menuju ke tempat yang dipilih oleh pengguna aplikasi. Berikut adalah rancangan antarmuka saat aplikasi memperlihatkan rute yang dapat dilewati :





## Perancangan Algoritma

Bayangkan kita memiliki sebuah peta beserta pengguna aplikasi dan sebuah gedung yang dituju.



Sayangnya, pengguna aplikasi tidak bisa berjalan lurus secara langsung ke tempat yang dituju, karena terdapat tembok yang menghalangi jalan.

Dan disini pengguna aplikasi harus bergegas untuk mencapai gedung yang dituju karena harus menghadiri meeting jurusan.

Tetapi bagaimana cara menulis algoritma untuk menentukan jalan yang harus diambil oleh user ? Disinilah A\* bekerja.

* + 1. Menyederhakan Area Pencarian

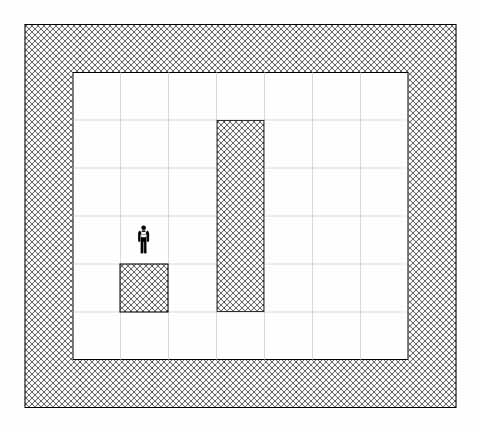
Langkah pertama dalam pencarian rute adalah menyederhanakan area pencarian kedalam sesuatu yang mudah untuk dikelola.

Cara untuk mengimplementasikannya tergantung dari kebutuhan. Contohnya, kita dapat membagi area pencarian menjadi pixels, tapi menyebabkan terlalu banyaknya node yang harus dibuat.

Jadi, kita dapat menggunakan sebuah bentuk persegi sebagai node untuk algoritma pencarian rute ini. Varian dengan jenis bentuk lainnya dapat memungkinkan (seperti segitiga atau segienam), namun persegi adalah yang paling sesuai untuk kebutuhan penulis dan juga yang paling sederhana.

Setalah itu , area pencarian dapat digambarkan secara sederhana oleh array dua dimensi untuk menggambarkan sebuah map. Jadi jika map yang digambarkan menggunakan 25x25 node, area pencarian akan menjadi array dari 625 node.

Beginilah contoh map yang penulis bikin dan dibuat untuk menggambarkan area pencarian dengan 7x6 node = 42 total node



* + 1. Buka dan Tutup List

Sekarang penulis telah memiliki area pencarian yang sederhana, kemudian bagaimana algoritma A\* bekerja :

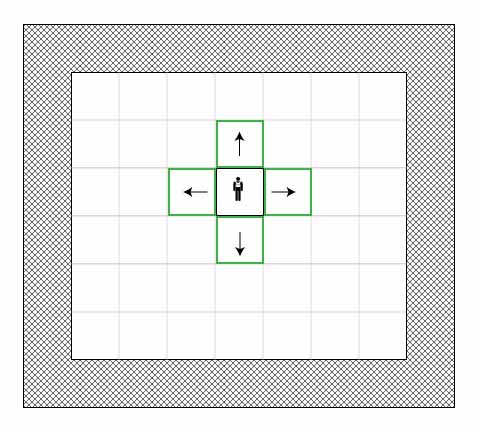
Karena pada dasarnya aplikasi ditujukan kepada user yang tidak memiliki pengetahuan akan keberadaan gedung di Universitas Mercubuana, jadi kita membutuhkan 2 daftar :

1. Menuliskan semua node yang sedang dipertimbangkan untuk menemukan jalur yang dapat dilewati(disebut jalur terbuka)

2. Menuliskan semua node sebagai jalur yang tidak dapat dilewati (disebut jalur tertutup )

Posisi pengguna dimulai dari posisi dimana pengguna berada saat ini(posisi awal) ke daftar jalur tertutup. Kemudian menambahkan semua jalur yang terbuka / dapat dilewati (Posisi pengguna dapat diibaratkan sebagai “A”) ke daftar jalur terbuka.

Beginilah contoh bagamana jika user berada dijalur yang dapat dilewati(digambarkan dengan warna hijau).



Sekarang user perlu menetukan pilihan mana yang dapat dilewati sebagai jalur yang paling pendek, tetapi bagaimana cara user memilih ? Pada algoritma A\*, hal ini dilakukan dengan memberi skor pada masing-masing node, yang disebut path scoring.

* + 1. Path Scoring

Untuk setiap node, dapat diberikan sebuah skor G + H dimana :

1. G adalah biaya pergerakan dari titik awal A ke node selanjutnya. Jadi untuk node yang berdekatan dengan titik A, nilai akan menjadi 1, tapi akan meningkat saat melangkah lebih jauh dari titik awal.

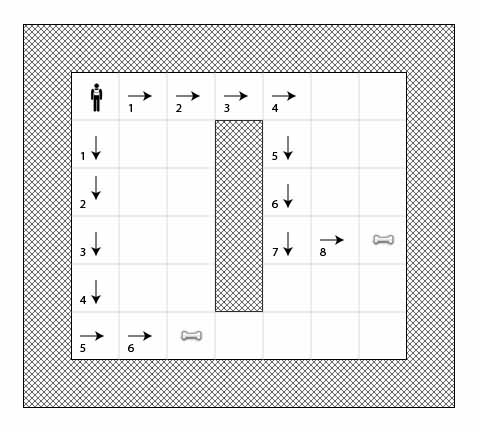
2. H adalah perkiraan jumlah pergerakan dari posisi saat ini ke titik tujuan (Kita bisa menggambarkan ini sebagai gedung yang dituju). Ini biasanya disebut sebagai heuristic karena sebenarnya kita tidak mengetahui berapa sesungguhnya total jumlahnya – ini semua hanya sebuah estimasi

Jadi apa yang dimaksud dengan biaya pergerakan ? Didalam aplikasi pencarian rute terdekat ini, itu hanyalah jumlah dari node.

* + 1. Lebih dala tentang G

Ingat bahwa G adalah biaya pergerakan dari titik awal A ke titik selanjutnya. Untuk menghitung G, kita perlu mengambil induk (titik dimana kita berasal) dan menambahkan ke dalamnya. Oleh karena itu, G dari setiap node akan mewakili total biaya jalur yang dihasilkan dari titik A sampai node.

Untuk contoh, diagram dibawah ini menunjukan 2 jalur ke 2 gedung berbeda, dengan skor G untuk setiap node.



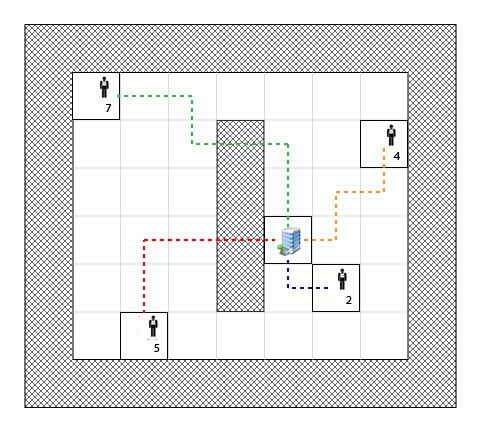
* + 1. Lebih dalam tentang H

Ingat bahwa H adalah perkiraan biaya pergerakan (dalam jumlah node) dari titik keberadaan sekarang menuju titik tujuan.

Semakin dekat perkiraan biaya pergerakan ke biaya sesungguhnya, semakin akurat jalur terakhirnya.

Untuk mempermudah, kita akan menggunakan “Manhattan distance method” ( juga dikenal sebagai “city block distance”) yang hanya menghitung jumlah horizontal dan vertical dari sisa node untuk mencapai point B tanpa memperhitungkan hambatan apapun atau perbedaan tanah.

Sebagai contoh, berikut adalah diagram yang menunjukan penggunaan “jarak blok” untuk memperkirakan H(digambarkan dengan kotak berwarna hitam)dari berbagai permulaan dan tujuan :



* + 1. A\* algoritma

Sekarang kita telah mengetahui bagaimana menghitung skor dari setiap node (kita akan memanggil ini dengan “F”, yang merupakan hasil dari G + H), sekarang kita akan membahas bagaimana algoritma A\* bekerja.

Pengguna akan menemukan rute terpendek dengan mengulangi langkah-langkah berikut :

1. Dapatkan node pada daftar yang terbuka yang memiliki nilai terendah ( S )

2. Hapus S dari daftar terbuka dan tambahkan S ke daftar tertutup.

3. Untuk setiap node T di S node yang dapat dilewati :

- Jika T adalah daftar tertutup: Abaikan

- Jika T tidak dalam daftar terbuka : Tambahkan dan hitung nilainya.

- Jika T sudah dalam daftar terbuka. Periksa apakah nilai F lebih rendah saat kita menggukan jalur yang dihasilkan saan ini untuk sampai kesana. Jika sudah, perbaharui skornya dan update induknya juga.

* + 1. Jalur Pengguna

Jalur yang dapat dilewati oleh pengguna aplikasi dapat dicontohkan sebagai berikut:

Pada diagram dibawah, penulis telah mencantumkan nilai untuk F = G + H sesuai dengan berikut :

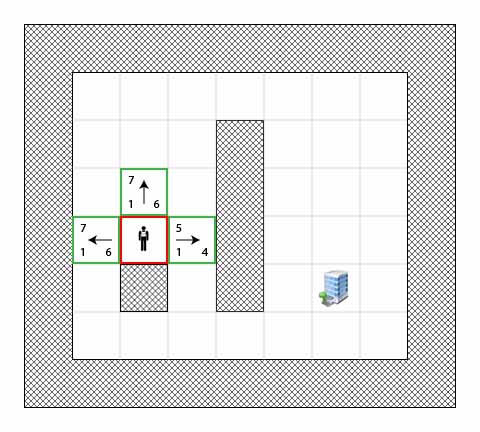
* F (Skor untuk node): Pojok kiri atas
* G(Biaya dari A ke node selanjutnya): Pojok kiri bawah
* H(Estimasi biaya dari node ke B(tujuan)): pojok kanan bawah

Selain itu, panah menunjukan arah gerakan untuk mencapai titik tujuan.

Akhirnya pada setiap langkah node merah menunjukan daftar tertutup dan kotak hijau menunjukan daftar terbuka.

Step 1

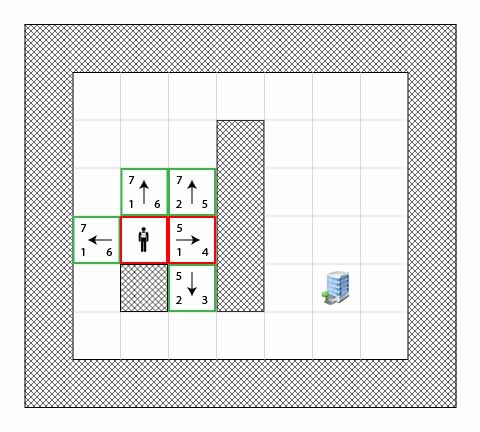
Pada langkah pertama, pengguna menentukan kotak yang berdekatan dengan posisi awal (titik A), menghitung nilai F dan menambahkan ke daftar terbuka:



Dapat kita lihat bahwa nilai H tercantum untuk setiap kotak (dua memiliki 6 dan satu memiliki 4).

Step 2

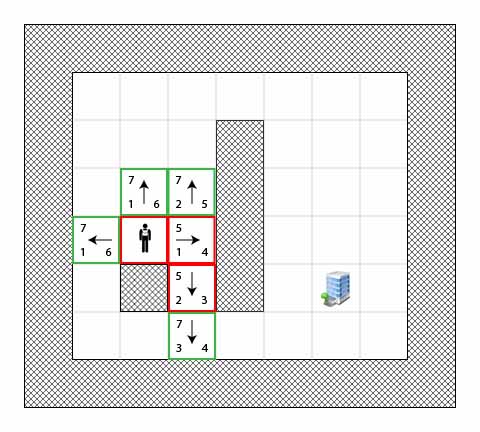
Pada langkah selanjutnya, pengguna memilih kotak dengan nilai F terendah, menambahkannya ke daftar tertutup dan mengambil kotak yang berdekatan.



Jadi kita akan melihat bahwa kotak dengan biaya terendah adalah biaya yang dimiliki F sebagi 4.Proses ini mencoba manambah kotak apapun yang berdekatan dengan daftar yang terbuka (dan menghitung nilainya). Kecuali harus diperhatikan bahwa kita tidak dapat menambah kotak posisi pengguna saat ini (karena itu sudah berada dalam daftar tertutup ) atau kotak tembok (karena tembok tidak dapat dilewati).

Step 3

Sekali lagi, kita memilih kotak dengan skor F terendah(5) dan terus berulang.

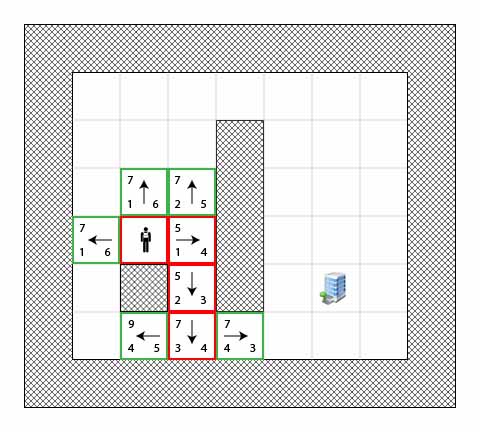


Dalam kasus ini, hanya ada satu kotak yang mungkin ditambahkan ke daftar terbuka, karena yang satu sudah masuk dalam daftar tertutup dan dua dua lainnya adalah tembok.

Step 4

Sekarang kita memiliki kasus yang menarik. Seperti yang bisa kita lihat di digambar sebelumnya, ada 4 kotak dengan skor F yang sama dengan 7, lalu bagaimana kita menentukan harus mengambil jalur / kotak yang terdekat ?

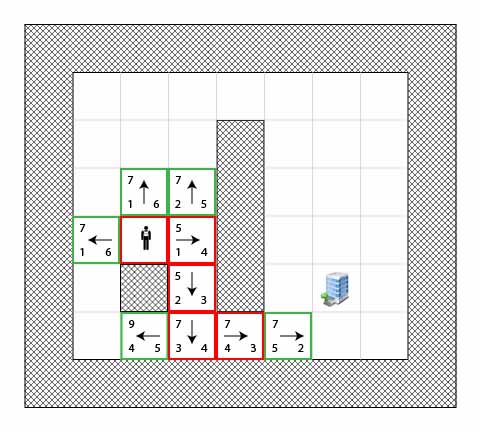
Ada berbagai solusi yang dapat digunakan, tapi satu cara yang sederhana ( dan cepat) adalah mengikuti kotak yang baru saja ditambahkan ke daftar terbuka.



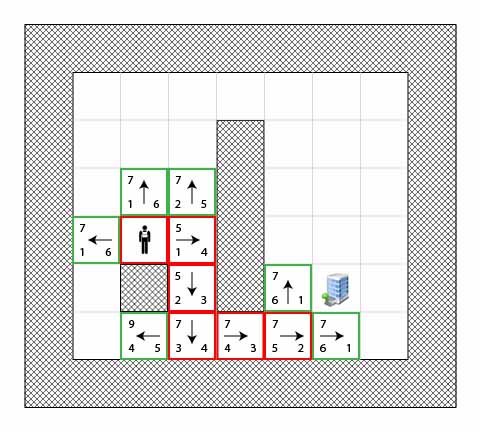
Kali ini dua kotak berdekatan dan dapat dilewati, hitunglah total skor seperti biasa

Step 5

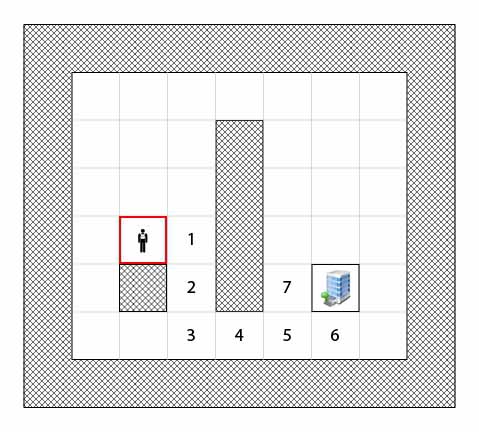
Sekali lagi kita memilih kotak dengan skor terendah (7)



Step 6



Kita hamper sampai, tetapi kali ini kita bisa melihat bahwa sebenarnya ada dua jalur terpendek yang bisa kita pilih diantara :



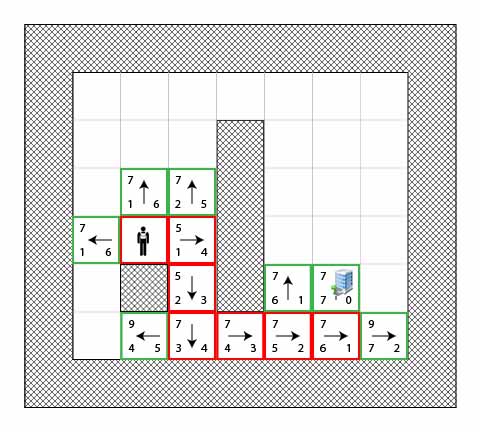
Pada contoh diatas terdapat 2 perbedaan jarak terpendek :

* 1-2-3-4-5-6
* 1-2-3-4-5-7

Sebenarnya tidak terlalu masalah kita akan memilih yang mana, ini dapat diatasi dengan penerapan sebenarnya didalam kode.

Step 7

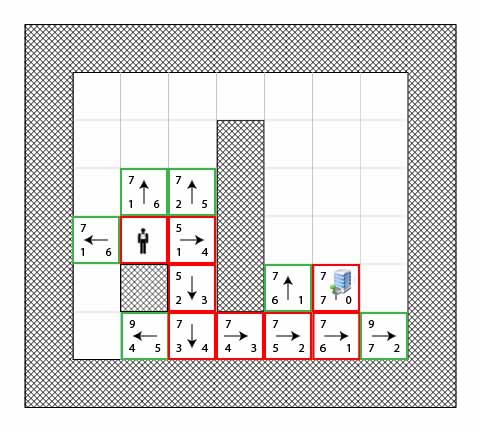
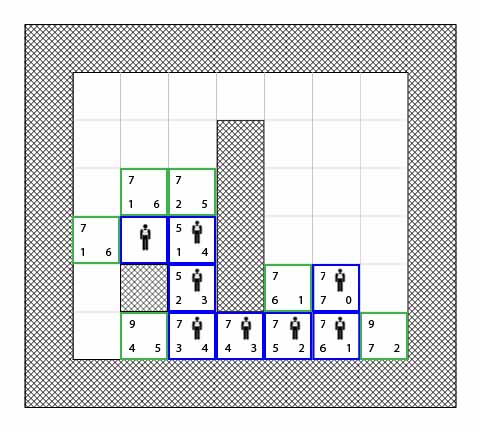
Lakukan pengulangan sekali lagi



Dan sekarang gedung yang dituju masuk kedalam daftar terbuka.

Step 8

Jika kotak target ada didaftar terbuka, algoritma menambahkannya ke daftar tertutup:



Maka kemudian semua yang harus dilakukan oleh algoritma adalah mundur untuk mencari tahu jalan terakhir.

**Daftar Pustaka**

Amalina, N., Sabri, M., Samad, A., Basari, H., & Husin, B. (2015). The Utilisation of Dijkstra ’ s Algorithm to Assist Evacuation Route in Higher and Close Building. https://doi.org/10.3844/jcssp.2015.330.336

Ardana, D., & Saputra, R. (2016). Penerapan Algoritma Dijkstra pada Aplikasi Pencarian Rute Bus Trans Semarang, (Snik), 299–306.

Dennis, A. (2012). *System Analysis & Design 5th Edition*.

Informatika, J. T. (2008). PENERAPAN ALGORITMA DIJKSTRA DALAM PENCARIAN SOLUSI, (10), 1–5.

Orlando, T. M., & Teknik, S. (2015). Penerapan Algoritma Dijkstra Untuk Menentukan Rute Wisata Minimum Kota Semarang, (13513038).

Roger S, P. (2015). Software.Engineering.A.Practitioners.Approach.8th.edition.pdf.

Sommerville, I. (2011). *Software Engineering 9th Edition*.

Wazlawick, R. S. (2013). *Object-Oriented Analysis and Design for Information Systems Modeling with UML , OCL , IFML*.