BAB II

LANDASAN TEORI

2.1 SPK (Sistem Pendukung Keputusan)

Pada tahun 1971, Michael Scott Morton memperkenalkan istilah Manageme nt Decision System sebagai awal dari konsep Sistem Pendukung Keputusan (SPK) (Turban, 2001). Dalam perkembangannya, banyak lembaga riset, institusi pendidikan, dan perusahaan yang tertarik untuk meneliti serta mengembangkan konsep ini. Berdasarkan penelitian tersebut, SPK dirancang sebagai sistem berbasis komputer yang berfungsi untuk membantu mengambil keputusan dengan menggunakan data dan model yang spesifik dalam menyelesaikan masalah yang bersifat tidak terstruktur Nofriansyah & Defit (2017, h. 1).

Menurut Nofriansyah & Defit dalam Sumarno & Harahap (2020), Sistem Pendukung Keputusan (SPK) adalah sistem berbasis komputer yang tersusun atas tiga komponen utama yang saling terintegrasi. Komponen pertama adalah sistem bahasa, berfungsi sebagai media komunikasi antara pengguna dengan komponen lain di dalam SPK. Komponen kedua adalah sistem pengetahuan, berperan sebagai pusat penyimpanan informasi terkait domain permasalahan, baik dalam bentuk data maupun prosedur. Terakhir, sistem pemrosesan masalah yang berfungsi untuk mengintegrasikan kedua komponen sebelumnya serta menyediakan kemampuan pemecahan masalah guna membantu proses pengambilan keputusan.

Berdasarkan pendapat Fithri & Latifah yang dikutip dalam Sumarno & Harahap (2020), SPK adalah sistem interaktif yang menyediakan data, model, serta kemampuan manipulasi informasi. Sistem ini dirancang untuk membantu

pengambil keputusan dalam situasi semi-terstruktur, di mana solusi yang benar tidak dapat ditentukan dengan metode yang pasti.

2.2 Coffee Shop

Coffee shop merupakan salah satu dari dua puluh dua jenis restoran yang ada. Berdasarkan Kamus Besar Bahasa Indonesia (1998), kedai kopi atau *coffee shop* didefinisikan sebagai tempat yang menyediakan kopi espresso dan makanan ringan. Namun, seiring perkembangan waktu, *coffee shop* kini berkembang dengan menawarkan tidak hanya kopi sebagai produk unggulan, tetapi juga beragam makanan ringan dan berat. Istilah *coffee shop* kini lebih sering disebut dengan *cafe*, yang maknanya mengalami pergeseran. Secara etimologis, kata *café* berasal dari bahasa Prancis yang berarti kopi (Oldenburg, 1989:126).

Menurut Wachdijono & Yahya (2021) Pada zaman sekarang, *coffee shop* merupakan tempat yang menyajikan berbagai olahan kopi dan makanan ringan, dengan desain dan penataan yang menarik untuk memikat konsumen. Oleh karena itu, kebanyakan kedai kopi modern telah memiliki barista, yang berperan penting dalam menyajikan kopi berkualitas tinggi dan lezat kepada pelanggan.

Kemudian Laras Gemi Nestiti dkk. (2022) Menjelaskan bahwa Coffee shop kini tidak hanya dianggap sebagai tempat untuk konsumsi, melainkan juga sebagai simbol gaya hidup masyarakat modern, terutama bagi kalangan muda. Nilai estetika yang disesuaikan dengan selera anak muda menjadikan coffee shop sebagai "tempat nongkrong", yang di dalamnya muncul berbagai makna subjektif terkait aktivitas atau perilaku individu yang mengunjunginya.

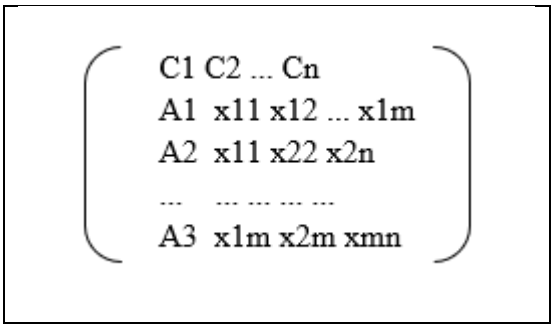
2.3 MABAC (Multi-Attribute Border Aproximation Criteria)

Menurut Ndruru dkk. (2020), Pamučar & Ćirović (2015) mengembangkan metode MABAC yang dipilih karena lebih stabil dan konsisten jika dibandingkan dengan metode multi-kriteria pengambilan keputusan lainnya, seperti SAW, COPRAS, MOORA, TOPSIS, dan VI-KOR. Metode ini dianggap dapat diandalkan dalam pengambilan keputusan yang rasional. Dasar metode ini berfokus pada pengukuran jarak fungsi kriteria dari setiap alternatif berdasarkan area perkiraan yang telah ditentukan.

Pada bagian berikut, Hondro (2018) menuliskan menjelaskan tahapan penerapan metode MABAC (*Multi-Attributive Border Approximation area Comparison*), termasuk formulasi matematis yang digunakan. Asumsi utama dari metode ini berfokus pada pengukuran jarak kriteria dari setiap alternatif terhadap area perbatasan yang telah diperkirakan. Secara keseluruhan, implementasi metode MABAC mencakup enam tahapan utama, yang akan dijelaskan melalui formulasi matematis pada bagian berikutnya.

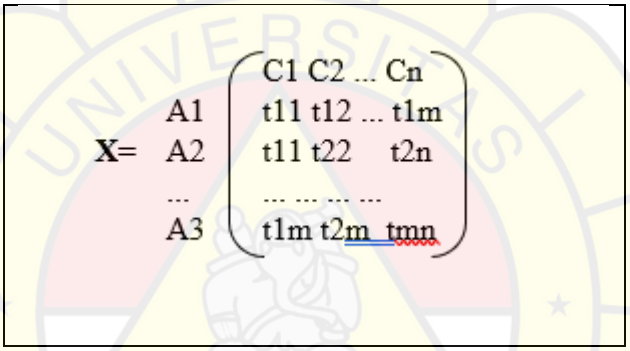
2.3.1 Membentuk Matriks Keputusan Awal (X)

Tahapan pertama metode ini adalah mengevaluasi **m** alternatif menggunakan **n** kriteria. Setiap alternatif dinyatakan dalam bentuk vektor vektor **Ai = (xi1, xi2, xi3, ..., xin)**, di mana elemen **xij** adalah nilai alternatif ke-***i*** pada kriteria ke-**j**. Indeks ***i*** berkisar antara 1 hingga **m**, sementara **j** berkisar antara 1 hingga **n**. Ilustrasi dari proses ini dapat dilihat pada Gambar 2.1, yang menggambarkan Matriks Keputusan Awal.



Gambar 2.1 Contoh Matriks Keputusan Awal (X) Hondro (2018)

2.3.2 Normalisasi Elemen Matriks Awal (X)



Gambar 2.2 Contoh Normalisasi Elemen Matriks Awal (X) Hondro (2018)

Elemen-elemen dalam matriks ternormalisasi (N) dihitung dengan menerapkan rumus normalisasi berdasarkan jenis kriteria. Rumus yang digunakan adalah sebagai berikut:

1. Untuk Kriteria Benefit (For benefit-type criteria)

Kriteria ini berlaku ketika nilai yang lebih besar dianggap lebih baik. Normalisasi dilakukan dengan rumus: $t_{ij} = \frac{x_{ij} - x_i^-}{x_{i+} - x_i^-}$

2. Untuk Kriteria Cost (For cost-type criteria)

Kriteria ini digunakan ketika nilai yang lebih kecil dianggap lebih baik. Rumus normalisasi untuk jenis kriteria ini adalah: $t_{ij} = \frac{x_{ij} - x_i^+}{x_i^- - x_i^+}$

Pada matriks keputusan awal (X), elemen-elemen yang digunakan antara lain xij, x_i^+ dan x_i^-. Nilai x_i^+ dan x_i^-. Memiliki definisi sebagai berikut:

- x_i^+= max (x1, x2, x3, xm) yang merupakan nilai tertinggi atau maksimum dari kriteria yang diamati di antara seluruh alternatif.
- x_i^- = min (x1, x2, x3, ..., xm) adalah nilai terendah atau minimum dari kriteria yang diamati pada semua alternatif.

2.3.3 Perhitungan Elemen Matriks Tertimbang (V)

$$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{bmatrix}$$

Gambar 2.3 Contoh Perhitungan Elemen Matriks Tertimbang (V)

Matriks Tertimbang (V) dibentuk dengan menghitung elemen-elemen pada setiap alternatif berdasarkan bobot kriteria yang telah ditentukan. Proses perhitungannya dilakukan menggunakan rumus berikut:

vij=(wi·tij)+wi

Keterangan:

vij : Nilai matriks tertimbang untuk alternatif ke-*i* pada kriteria ke-*j*.

wi: Bobot atau koefisien untuk setiap kriteria.

tij: Nilai elemen dari matriks keputusan yang telah dinormalisasi (N).

Dengan rumus di atas, nilai *tij* dari matriks normalisasi dikalikan dengan bobot kriteria *wi*, kemudian ditambahkan dengan nilai bobot *wi* itu sendiri. Proses ini memastikan setiap elemen dalam matriks tertimbang (V) telah mempertimbangkan prioritas bobot kriteria yang berbeda-beda.

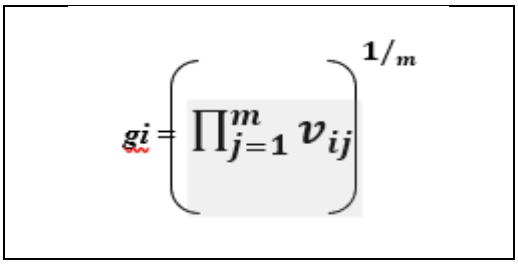
Selanjutnya, matriks tertimbang (V) yang diperoleh dapat divisualisasikan dalam bentuk gambar yang menjelaskan nilai tertimbang dari setiap alternatif dan kriteria. Gambar ini memberikan representasi jelas bagaimana setiap nilai dipengaruhi oleh bobot kriteria.

$$V = \begin{pmatrix} w1 * t11 + w1\ w2 * t11 + w2 \ldots wn * t1n + wn \\ w1 * t21 + w1\ w2 * t22 + w2\ wn * t2n + wn \\ \ldots \quad \ldots \quad \ldots \quad \ldots \\ w1 * tm1 + w1\ w2 * tm2 + w2 \ldots wn * tmn + wn \end{pmatrix}$$

Gambar 2.4 Contoh Elemen matriks tertimbang (V) Hondro (2018)

2.3.4 Penentuan matriks area perkiraan perbatasan (G)

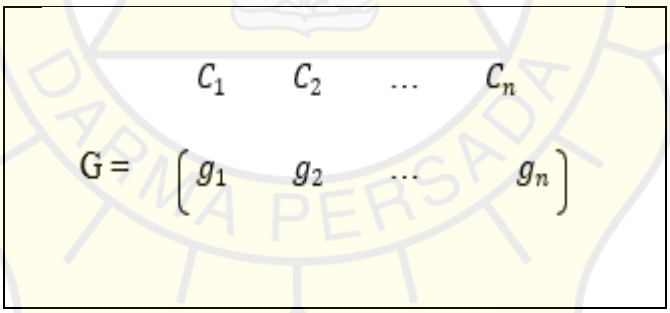
Perhitungan area batas ini penting untuk mengidentifikasi rentang nilai yang relevan bagi setiap kriteria dalam proses pengambilan keputusan. Rumus yang digunakan untuk menentukan area perkiraan batas setiap kriteria di sajikan pada gambar berikut:



$$gi = \left(\prod_{j=1}^{m} v_{ij} \right)^{1/m}$$

Gambar 2.5 Contoh Penentuan matriks area perkiraan perbatasan (G) Hondro (2018)

Dalam tahap ini **vij** menunjukkan elemen-elemen dari matriks berbobot (V), di mana **m** mewakili jumlah total alternatif yang dievaluasi. Setelah menghitung nilai **gi** berdasarkan masing-masing kriteria, hasil perhitungan ini akan membentuk matriks area perkiraan batasan (G), yang memiliki dimensi **n x 1**, di mana **n** adalah jumlah total kriteria yang digunakan dalam pemilihan alternatif yang tersedia. Representasi matematik nya ada di gambar berikut:

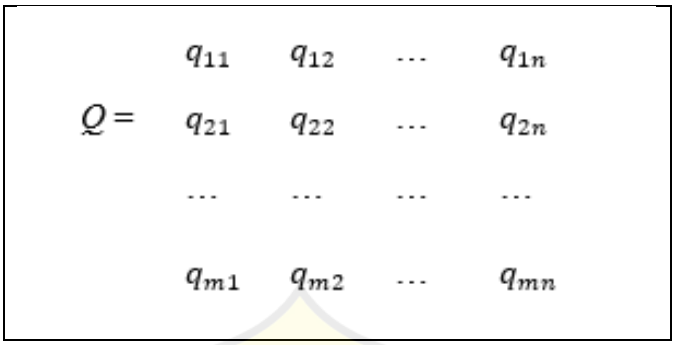


$$G = \begin{matrix} C_1 & C_2 & \cdots & C_n \\ \end{matrix} \begin{pmatrix} g_1 & g_2 & \cdots & g_n \end{pmatrix}$$

Gambar 2.6 Contoh Matriks area perkiraan perbatasan (G) Hondro (2018)

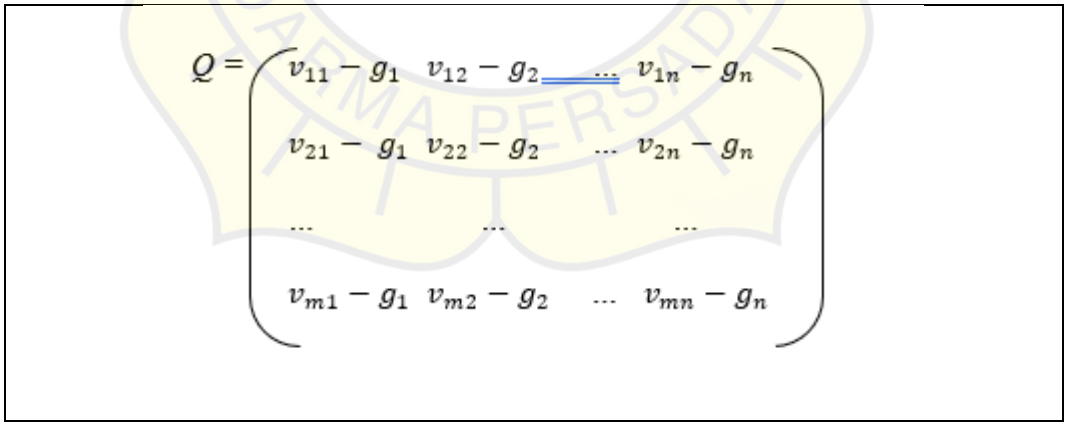
Matriks **G** ini menggambarkan hasil perhitungan untuk masing-masing kriteria yang digunakan dalam evaluasi alternatif. Hasil dari matriks ini akan digunakan dalam tahap-tahap selanjutnya untuk memutuskan alternatif terbaik berdasarkan area perkiraan batasan yang telah dihitung.

2.3.5 Perhitungan elemen matriks jarak alternatif dari daerah perkiraan perbatasan (Q)



Gambar 2.7 Contoh Perhitungan elemen matriks jarak alternatif dari daerah perkiraan perbatasan (*Q*) Hondro (2018)

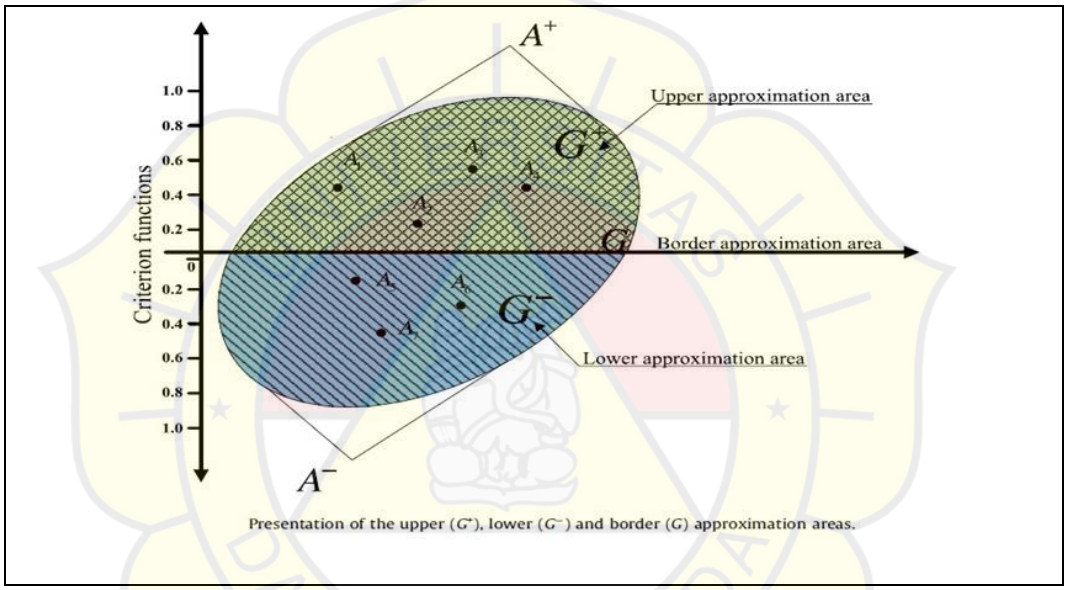
Jarak alternatif dari area perkiraan batasan (**qij**) dihitung dengan cara mengurangkan nilai elemen-elemen dalam matriks tertimbang **(V)** dengan nilai yang ada dalam area perkiraan batasan (G). Secara matematis, persamaan ini dapat dituliskan sebagai **Q = V – G**, yang juga dapat digambarkan dengan cara lain seperti pada gambar berikut.



Gambar 2. 8 Contoh Nilai daerah perkiraan perbatasan Hondro (2018)

Pada bagian ini, **gi** melambangkan area perkiraan batasan untuk kriteria **Ci**, sementara **vij** adalah elemen-elemen dari matriks berbobot (**V**). **n** menunjukkan

jumlah total kriteria, dan **m** adalah jumlah alternatif yang di evaluasi. dapat ditempatkan dalam salah satu dari tiga area berikut, area perkiraan batasan (G)**,** area perkiraan atas (G+)**,** atau area perkiraan bawah (G-)**.** Secara matematis, ini dapat ditulis sebagai **Ai** $\epsilon\{G\ v\ G^{+}v\ G^{-}\}$. Area perkiraan atas (**G+**) adalah area yang mencakup alternatif ideal (**A+**), aitu alternatif yang paling mendekati nilai terbaik berdasarkan semua kriteria. Sementara adalah area yang mencakup alternatif anti-ideal (A-), yaitu alternatif yang paling jauh dari nilai terbaik yang diinginkan.



Gambar 2.9 Gambar Presentasi G^+ dan G^- pada daerah perkiraan perbatasan *Q* Hondro (2018)

$$A_i \in \begin{cases} G^+ \ if \ q_{ij} > 0 \\ G \ if \ q_{ij} = 0 \\ G^- \ if \ q_{ij} < 0 \end{cases}$$

Gambar 2.10 Contoh A^i alternatif ke daerah perkiraan (G, G^+ atau G^-) Hondro (2018)

Untuk memilih alternatif terbaik dari sekumpulan alternatif, alternatif ***Ai*** harus berada dalam area perkiraan atas (G+) sebanyak mungkin kriteria. Sebagai contoh, jika alternatif ***Ai*** terletak dalam area perkiraan atas untuk 5 dari 6 kriteria yang ada, dan untuk satu kriteria lainnya berada **di** area perkiraan bawah (G-), maka ini menunjukkan bahwa berdasarkan 5 kriteria tersebut, alternatif ***Ai*** mendekati atau setara dengan alternatif ideal. Namun, berdasarkan satu kriteria yang berada di **G-**, ***Ai*** lebih mendekati atau setara dengan alternatif anti-ideal.

Secara umum, nilai yang lebih tinggi menunjukkan bahwa alternatif ***Ai*** lebih dekat dengan alternatif ideal, sedangkan nilai yang lebih rendah menunjukkan bahwa alternatif tersebut lebih mendekati alternatif anti-ideal.

2.3.6 Perankingan Alternatif (*Ranking Alternatives*)

Untuk mendapatkan nilai akhir dari fungsi kriteria suatu alternatif, jarak elemen-elemen dari area perkiraan batas (***qi***) dijumlahkan secara horizontal dalam matriks ***Q***. Dengan kata lain, semua nilai jarak untuk setiap kriteria pada alternatif tertentu diakumulasikan. Rumus perhitungan tersebut adalah:

$$(Si = \sum_{j=1}^{n} q_{ij}, j = 1, 2, \dots, n, i = 1,2, \dots, m)$$

Dalam rumus ini:

- *Si* merepresentasikan total nilai fungsi kriteria untuk alternatif ke-iii.
- *qij* adalah jarak antara alternatif ke-iii pada kriteria ke-*j*.
- *n* menyatakan jumlah kriteria, sedangkan mmm adalah total alternatif yang dipertimbangkan.

Dengan menjumlahkan nilai-nilai jarak ini, kita dapat menilai sejauh mana performa suatu alternatif dibandingkan dengan alternatif lainnya. Nilai akhir *Si*

yang lebih tinggi menunjukkan bahwa alternatif tersebut lebih dekat pada kriteria ideal.

2.4 Definisi Aplikasi

Dalam Tita Faulina dkk. (2021), Prahasta (2014) menyatakan bahwa aplikasi adalah tugas spesifik yang dijalankan secara otomatis dan semi-otomatis dalam sebuah perusahaan. Sementara itu, Pressman (2012) menyimpulkan bahwa perangkat lunak aplikasi merupakan program-program independen yang dirancang untuk memenuhi kebutuhan bisnis yang rinci.

Kemudian menurut Darmayuda (2014) dalam Tita Faulina dkk. (2021) bahwa aplikasi dapat dibagi menjadi dua kategori, yaitu aplikasi terkoneksi dan aplikasi terputus. Aplikasi terkoneksi adalah aplikasi yang memerlukan koneksi terus-menerus ke sebuah database selama aplikasi tersebut dijalankan. Sementara itu, aplikasi terputus adalah aplikasi di mana pengguna tidak terhubung secara konstan ke database, melainkan koneksi hanya dilakukan saat mengambil atau menyimpan data.

2.5 Definisi Android

Liyando & Kusbianto (2020) menjelaskan bahwa Sistem operasi Android dikembangkan menggunakan kernel Linux dan dirancang khusus untuk perangkat mobile. Sistem ini menawarkan platform terbuka bagi pengembang untuk membuat dan mengembangkan aplikasi. Pada awalnya, sistem operasi ini dikembangkan oleh Android Inc., sebuah perusahaan perangkat lunak yang kemudian diakuisisi oleh Google Inc. Untuk mendukung pengembangan ekosistem Android, Google membentuk Open Handset Alliance, sebuah kerjasama yang melibatkan 34

perusahaan terkemuka di bidang teknologi. Beberapa perusahaan yang tergabung di dalamnya antara lain HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia. Konsorsium ini memungkinkan Android berkembang pesat sebagai sistem operasi utama di industri smartphone.

2.5.1 Visual Studio Code

Salamah (2021) menyebut *Visual Studio Code* (VS Code) yang di dikembangkan oleh Microsoft ini sebagai teks editor modern yang efisien dan ringan. Editor ini mendukung Linux, Mac, dan Windows sebagai sistem operasi utamanya. Secara langsung, VS Code menyediakan dukungan untuk bahasa pemrograman populer seperti JavaScript, TypeScript, dan Node.js. Jika pengembang ingin menggunakan bahasa lain seperti C++, C#, Python, Go, atau Java, mereka dapat memanfaatkan plugin yang tersedia di marketplace Visual Studio Code untuk memperluas fungsionalitas editor ini.

2.5.2 Flutter

Raharjo (2019) dalam buku nya menjelaskan bahwa Flutter merupakan framework open-source sekaligus SDK yang dikembangkan oleh Google untuk mempermudah pembuatan aplikasi dengan performa tinggi. Dengan Flutter, pengembang dapat membuat aplikasi yang berjalan di platform Android dan iOS hanya menggunakan satu codebase saja. Di lengkapi dengan fitur hot reload yang dapat melihat perubahan kode tanpa perlu melakukan kompilasi atau build ulang. Flutter menggunakan bahasa pemrograman Dart, yang memiliki kemiripan dengan Java atau JavaScript.

2.5.3 Dart

Raharjo (2019) menyebutkan bahwa Dart merupakan bahasa pemrograman serbaguna yang dibuat dan dikembangkan oleh Google untuk berbagai keperluan pengembangan, seperti aplikasi Android, web front-end, sistem IoT**,** backend (CLI), hingga pembuatan game. Dart termasuk ke dalam bahasa pemrograman bertipe dinamis, yang membuatnya fleksibel dan efisien digunakan dalam pengembangan aplikasi modern. Selain itu, Dart memiliki kinerja yang tinggi serta mendukung penggunaan sebelum proses kompilasi.

2.5.4 Bloc Pattern

Menurut Saputra & Wiraganda (2019) Bloc Pattern adalah salah satu sistem manajemen state dalam Flutter yang dikembangkan oleh Google. Sistem ini sangat direkomendasikan untuk pengembang aplikasi mobile berbasis Flutter karena mempermudah pengelolaan data secara terpusat, sehingga dapat diakses oleh widget lain dalam proyek. Seperti platform Android dan iOS yang menerapkan arsitektur MVC, MVP, atau MVVP, Flutter juga memiliki pendekatan design pattern sendiri yang berfokus pada state management.

2.5.5 MapBox

Dalam jurnalnya, Dwi Oktavianto dkk. (2022) menyebut Mapbox sebagai salah satu layanan API Peta yang menyediakan fitur serupa dengan Google Maps API. Sejak diperkenalkan pada tahun 2010, Mapbox terus berkembang untuk memperbaiki keterbatasan layanan peta konvensional, termasuk yang ditawarkan oleh Google Maps. Platform ini banyak digunakan oleh situs-situs terkenal seperti Pinterest**,** Foursquare, Financial Times, Evernote, dan layanan transportasi Uber, menjadikannya salah satu penyedia peta kustom terbesar di dunia. Selain itu,

Mapbox juga berperan sebagai pengembang maupun kontributor utama bagi sejumlah pustaka dan aplikasi peta bebas yang terkenal.

Kemudian menurut Hadi (2015) dalam Hidayatulloh & Airlangga (2022) Salah satu keunggulan Mapbox adalah kemampuannya membangun peta dengan menggunakan perangkat lunak berbasis open source. Mapbox menyediakan lisensi kepada pengembang agar peta tersebut dapat diintegrasikan ke dalam aplikasi. Selain itu, platform ini mengombinasikan data dari berbagai sumber, termasuk instansi pemerintah seperti U.S Geological Survey dan NASA, serta komunitas open source seperti OpenStreetMap.

2.5.6 Firebase

Stonehem (2016) Mengklaim bahwa Firebase adalah layanan cloud dan backend yang memudahkan pengelolaan data untuk aplikasi mobile. Layanan ini sangat penting karena hampir semua aplikasi mobile masa kini memerlukan verifikasi pengguna dan pembaruan data. Firebase dirancang agar mudah digunakan, memungkinkan pembacaan dan penulisan data yang cepat, bahkan untuk pemula. Firebase tidak hanya bisa digunakan untuk membuat aplikasi iOS dan Android, tetapi juga aplikasi berbasis web dengan data real-time dan penyimpanan, serta menyediakan berbagai produk lain yang bisa dimanfaatkan oleh pengembang perangkat lunak.

2.5.7 *Cloud Firestore*

Menurut Mallik dkk. (2020) *Cloud Firestore* adalah database berorientasi dokumen berbasis NoSQL, dengan data yang disimpan dalam format JavaScript Object Notation (JSON).

Kemudian menurut Banane dkk. (2018) Database NoSQL dirancang untuk mengatasi tantangan dalam pengolahan data dalam jumlah besar, berasal dari berbagai sumber, dan dalam berbagai format, seperti yang sering terjadi pada situasi big data. Database NoSQL bersifat non-relasional, dapat diskalakan secara horizontal, dan berjalan pada mesin dengan konfigurasi standar.

2.5.8 *Firebase Authentication*

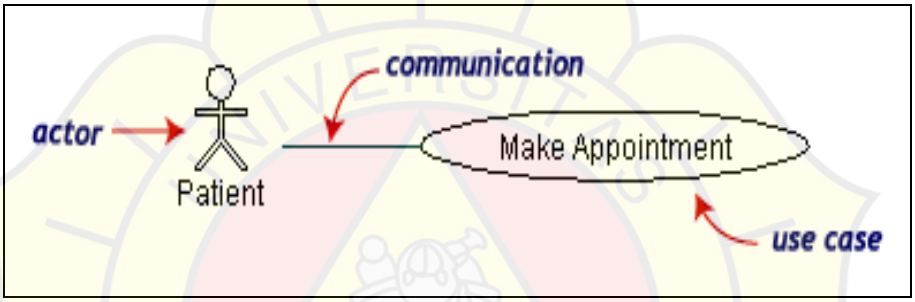
Menurut Moroney (2017) dalam Pramono & Javista (2021) *Firebase* dikembangkan sebagai layanan SDK berbasis cloud yang dapat digunakan untuk membangun aplikasi autentikasi kustom. Layanan ini mendukung integrasi identitas dengan berbagai platform online seperti GitHub, Twitter, Facebook, dan Google, atau melalui autentikasi menggunakan email dan kata sandi. Fitur Cloud Functions pada Firebase memungkinkan pengembang menulis kode program yang merespons suatu peristiwa dalam elemen Firebase, termasuk autentikasi.

2.6 UML

Dalam buku yang berjudul *"Buku Ajar Rekayasa Perangkat Lunak"*, Hasanah & Untari (2020, h. 64) mengungkapkan bahwa UML (Unified Modeling Language) merupakan bahasa standar yang banyak digunakan di industri perangkat lunak untuk keperluan analisis, desain, dan pendefinisian kebutuhan sistem berbasis objek. Kemunculan UML merupakan jawaban atas tuntutan industri akan alat bantu pemodelan visual yang mampu mendukung aktivitas pengembangan perangkat lunak mulai dari perencanaan hingga dokumentasi.

2.6.1 Use Case Diagram

Menurut Hasanah & Untari (2020, h. 71) diagram ini berfungsi sebagai alat dokumentasi kebutuhan fungsional yang menunjukkan fungsi-fungsi utama sistem. Diagram ini memusatkan perhatian pada *"apa yang dilakukan"* sistem tanpa membahas cara kerjanya. Dalam *use case*, interaksi antara aktor (baik manusia maupun mesin) dan sistem divisualisasikan sebagai tugas-tugas spesifik, seperti proses login, pembuatan daftar belanja, dan aktivitas lain yang berhubungan langsung dengan pengguna sistem.

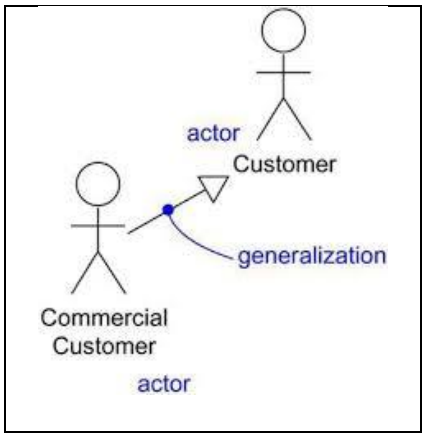


Gambar 2.10 Contoh *Use Case* Diagram Hasanah & Untari (2020)

Komponen pembentukan *use case* diagram adalah:

A. Aktor

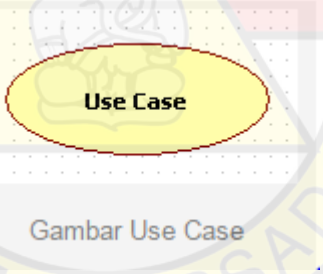
Hasanah & Untari (2020, h. 73) menyatakan bahwa secara prinsip, aktor tidak termasuk dalam elemen use case diagram. Namun, untuk membangun use case diagram, aktor tetap diperlukan karena mereka mewakili individu atau entitas lain, baik itu perangkat maupun sistem lain, yang berinteraksi dengan sistem tersebut.



Gambar 2.11 Contoh Aktor Hasanah & Untari (2020)

B. Use Case

Dalam pandangan Hasanah & Untari (2020, h. 74), *use case* menggambarkan fungsionalitas sistem untuk memudahkan pengguna atau pelanggan memahami sistem yang dikembangkan.



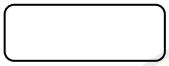
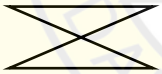
Gambar 2.12 Contoh *Use Case* Hasanah & Untari (2020)

2.6.2 Activity Diagram

Menurut Hasanah & Untari (2020, h. 79), *activity diagram* adalah bentuk modifikasi dari *state diagram*, dengan sebagian besar state yang menggambarkan aksi dan transisi antar state yang terjadi setelah proses internal selesai. *Activity diagram* tidak bertujuan untuk menggambarkan secara detail perilaku internal atau

interaksi antar subsistem, melainkan memberikan gambaran umum mengenai jalur aktivitas dan proses pada tingkat yang lebih tinggi.

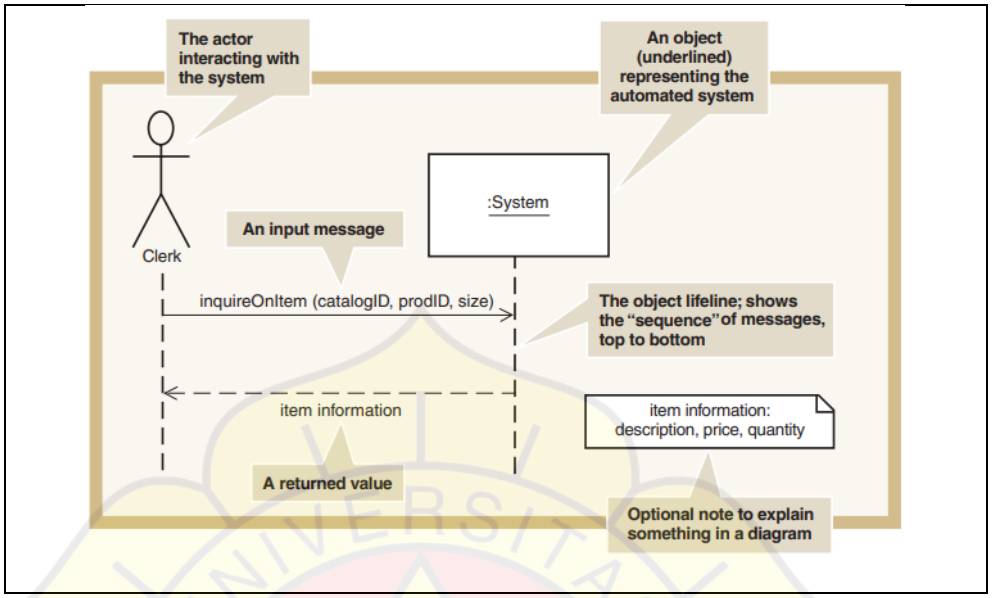
Tabel 2.1 Simbol-simbol dalam *Activity diagram* Hasanah & Untari (2020)

NO	SIMBOL	KETERANGAN
1		Titik awal
2		Menggambarkan sebuah kegitan atau tugas yang perlu dilakukan(activity)
3		Menggambarkan sasaran yang mengawali kegiatan
4		Menggambarkan proses yang dilakukan secara paralel, atau untuk menggabungkan dua aktivitas paralel menjadi satu
5		Tanda waktu
6		Pilihan untuk pengambilan keputusan
7		Titik akhir

2.6.3 *Sequence Diagram*

Dalam penjelasan Satzinger dkk. (2015, h. 252), *Sequence Diagram* berfungsi untuk menggambarkan alur informasi yang masuk dan keluar dari sistem otomatis, mencatat input dan output, serta mengidentifikasi interaksi antara aktor dan sistem.

Diagram ini termasuk dalam jenis *interaction diagram*, dan dalam praktik industri, istilah *interaction* dan *message* kerap digunakan secara bergantian.



Gambar 2.13 Contoh *Sequence Diagram* Satzinger dkk. (2015)

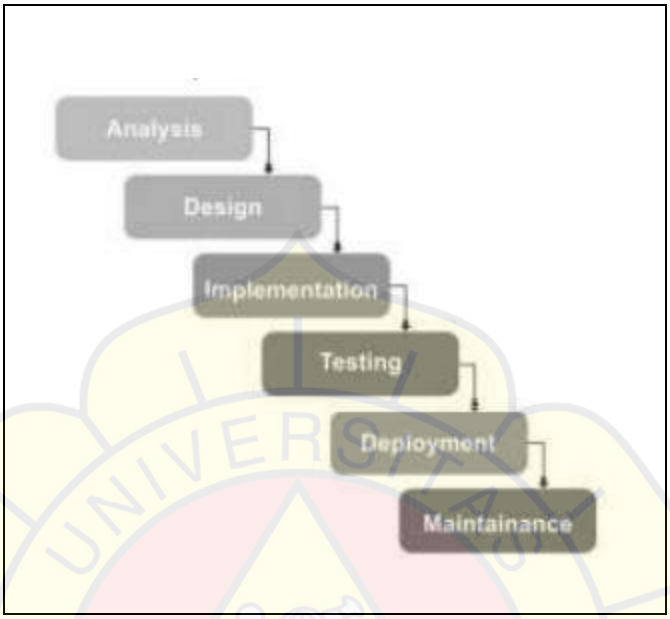
2.7 *Waterfall*

Model pengembangan perangkat lunak ini, yang diperkenalkan oleh Royce pada tahun 1970-an, Model ini diadaptasi dari metode pengembangan perangkat keras, karena pada masa itu belum tersedia metodologi lain untuk pengembangan perangkat lunak. Pendekatan yang sangat terstruktur ini memiliki kelemahan berupa risiko kerugian besar jika terjadi kesalahan pada tahap sebelumnya, karena sering kali mengakibatkan biaya pengembangan ulang yang tinggi.

Menurut Hasanah & Untari (2020, h. 21) Model *waterfall* merupakan model dengan pendekatan tradisional dalam pengembangan perangkat lunak yang menggunakan alur linier dan berurutan. Model ini melibatkan lima hingga tujuh tahap, dengan masing-masing tahap memiliki tugas dan tujuan tertentu. Keseluruhan tahap merepresentasikan siklus hidup perangkat lunak hingga proses

pengiriman. Setelah satu tahap selesai, pengembangan berlanjut ke tahap berikutnya, dengan hasil dari tahap sebelumnya mengalir ke tahap berikutnya.

Berikut merupakan gambar dari tahapan-tahapan model *Waterfall*.



Gambar 2.14 Model *Waterfall* Hasanah & Untari (2020)