BAB II
KAJIAN LITERATUR

2.1 *Artificial Intelligence*

Karena mereka memiliki pengetahuan dan pengalaman, manusia cerdas (pandai) dalam menyelesaikan masalah. Belajar adalah cara mendapatkan pengetahuan. Orang tentu lebih mampu menyelesaikan masalah dengan lebih banyak pengetahuan. Namun, pengetahuan saja tidak cukup; manusia juga diberi akal untuk melakukan penalaran dan membuat kesimpulan berdasarkan pengalaman dan pengetahuan mereka. Orang dengan banyak pengetahuan dan pengalaman tidak akan dapat menyelesaikan masalah dengan baik jika mereka tidak memiliki kemampuan menalar yang baik. Sebaliknya, orang yang memiliki kemampuan menalar yang sangat baik juga tidak akan dapat menyelesaikan masalah dengan baik jika mereka tidak memiliki pengetahuan dan pengalaman yang memadai[1].

Dengan cara yang sama, mesin harus diberi bekal pengetahuan agar mereka cerdas dan dapat menalar. Ada dua komponen utama yang sangat penting untuk membuat aplikasi kecerdasan buatan atau *Artificial Intelligence* :

1. Basis Pengetahuan (*knowledge base*), bersifat fakta-fakta, teori, pemikiran dan hubungan antar satu dengan yang lainnya.
2. Motor inferensi (*inference engine*), kemampuan menarik kesimpulan berdasarkan pengetahuan dan pengalaman.

Alan Turing, seorang ahli matematika dan pencipta AI pertama di Inggris, melakukan percobaan pada tahun 1950-an. Turing *(Turing Test*) adalah sebuah komputer yang ditempatkan pada jarak jauh melalui terminalnya. Ada dua terminal: satu dengan software *Artificial Intelligence* dan satu lagi dengan operator. Operator tidak tahu bahwa ada software *Artificial Intelligence* di sisi lain terminal. Mereka berkomunikasi antara satu sama lain melalui terminal di ujung yang memberikan tanggapan terhadap berbagai pertanyaan yang diajukan oleh operator. Operator tersebut percaya bahwa ia sedang berkomunikasi dengan operator lain di terminal lain. Turing berpendapat bahwa jika mesin memiliki kemampuan untuk berkomunikasi dengan orang lain, maka mesin tersebut dapat dianggap cerdas (seperti manusia)[1].

2.2 *Machine Learning*

Dalam bidang kecerdasan buatan atau *Artificial Intelligence*, *Machine Learning* berfungsi sebagai alat analisis dalam *data mining*[2]. Algoritme *Machine Learning* digunakan dalam ilmu pengembangan model dan algoritme secara statistik yang digunakan sistem komputer untuk menjalankan tugas tanpa instruksi eksplisit, mengandalkan pola dan inferensi sebagai gantinya. Algoritme ini digunakan oleh sistem komputer untuk memproses sejumlah besar data historis dan mengidentifikasi pola data, memungkinkan mereka untuk memprediksi hasil yang lebih akurat dari set data yang dimasukkan. Ilmuwan data, misalnya, dapat mengajarkan aplikasi medis untuk mendiagnosis kanker menggunakan gambar sinar-x dengan menyimpan jutaan gambar yang dipindai dan membuat diagnosis yang tepat[3].

Gambar 2.1 Gambar Tahapan *Machine Learning*

Machine Learning mendorong pertumbuhan perusahaan, menciptakan aliran pendapatan baru, dan menyelesaikan masalah sulit. Meskipun data sangat penting untuk pengambilan keputusan bisnis, perusahaan telah menggunakan berbagai sumber data, seperti umpan balik karyawan, pelanggan, dan keuangan, tetapi penelitian machine learning telah mengotomatiskan dan mengoptimalkan proses ini. Bisnis dapat mencapai hasil lebih cepat dengan menggunakan perangkat lunak yang dapat menganalisis data dengan kecepatan tinggi[3].

Hubungan matematis yang ada antara semua kombinasi data input dan output adalah inti dari pembelajaran mesin. Meskipun tidak mengetahui hubungan ini sebelumnya, model pembelajaran mesin dapat berpikir jika diberikan set data yang cukup. Ini berarti bahwa setiap algoritme pembelajaran mesin dibangun di sekitar fungsi matematika yang dapat diubah. Ini adalah prinsip dasar yang dapat dipahami[3]:

a. Kami “melatih” algoritme tersebut dengan memberinya kombinasi input/*output* (i,o) berikut – (2,10), (5,19), dan (9,31)
b. Algoritme tersebut mengomputasi hubungan antara input dan *output* menjadi: o=3*i+4

c. Selanjutnya, kami memberinya input 7 dan memintanya untuk memprediksi *output*. Algoritme tersebut dapat secara otomatis menentukan *output*-nya menjadi 25.

Meskipun ini adalah pemahaman dasar, pembelajaran mesin berkonsentrasi pada prinsip bahwa selama sistem komputer memiliki data dan daya komputasi yang cukup untuk memprosesnya, setiap poin data kompleks dapat dihubungkan secara matematis. Oleh karena itu, besarnya input secara langsung berkorelasi dengan keakuratan output.

Bergantung pada output yang diperkirakan dan tipe input, algoritma dapat dikategorikan ke dalam empat gaya belajar yang berbeda[3].

a. *Machine Learning* yang di awasi

Untuk menilai korelasi, ilmuwan data memberikan algoritme dengan data pelatihan yang diberi label dan ditentukan. Input dan output algoritme ditentukan oleh sampel data. Misalnya, anotasi ditambahkan ke gambar angka tulisan tangan untuk menunjukkan angka yang tepat. Jika diberikan contoh yang cukup, sistem pembelajaran yang diawasi dapat mengenali klaster piksel dan bentuk yang terkait dengan setiap angka. Pada akhirnya, sistem pembelajaran akan mengenali angka tulisan tangan dan dengan akurat membedakan angka 9, 4, 6 dan 8.

Pembelajaran yang diawasi sangat mudah dirancang dan mudah digunakan. Saat membagi data ke dalam kategori, memprediksi set hasil yang mungkin terbatas, atau menggabungkan hasil dari dua algoritme pembelajaran mesin lainnya, sistem pembelajaran ini sangat bermanfaat. Menjual jutaan set data tidak berlabel, bagaimanapun, adalah tantangan.

Pelabelan data adalah proses membagi data input dengan nilai output yang relevan. Untuk pembelajaran yang diawasi, data pelatihan berlabel diperlukan. Misalnya, jutaan foto apel dan pisang harus ditandai dengan kata "apel" atau "pisang". Kemudian, aplikasi pembelajaran mesin dapat menggunakan data pelatihan ini untuk menemukan nama buah saat diberi gambar buah. Melabeli jutaan data baru, bagaimanapun, dapat menjadi tugas yang menantang dan memakan waktu. Sampai batas tertentu, layanan crowd-working seperti Amazon Mechanical Turk dapat mengatasi keterbatasan algoritme pembelajaran yang diawasi ini. Mungkin lebih mudah untuk mendapatkan data karena layanan ini menyediakan akses ke kolam besar tenaga kerja murah yang tersebar di seluruh dunia[3].

b. *Machine Learning* tidak di awasi

Algoritme pembelajaran yang tidak diawasi digunakan untuk melatih data non-label. Algoritme ini memindai data baru untuk membuat hubungan yang signifikan antara

input dan output yang telah ditentukan. Pola dapat ditemukan dan digolongkan oleh algoritme ini. Misalnya, algoritme yang tidak diawasi dapat mengelompokkan berita dari berbagai situs web ke dalam kategori umum, seperti olahraga, kriminal, dll. Algoritme ini dapat menggunakan pemrosesan bahasa alami untuk memahami emosi dan makna yang terkandung dalam tulisan. Pembelajaran yang tidak diawasi dapat digunakan di toko ritel untuk mengidentifikasi pola pembelian konsumen dan memberikan hasil analisis data, seperti kemungkinan pembeli akan membeli roti jika mereka juga membeli mentega. Rekognisi pola, pengenalan anomali, dan mengelompokkan data secara otomatis ke dalam beberapa kategori adalah semua manfaat dari pembelajaran tidak diawasi. Sangat mudah untuk mengatur data pelatihan karena tidak memerlukan label. Selain itu, algoritme ini dapat digunakan untuk memproses dan membersihkan data untuk pemodelan lebih lanjut secara otomatis. Metode ini hanya dapat memberikan prediksi yang tidak akurat. Selain itu, teknik ini tidak dapat memilih sendiri hasil data tertentu[3].

c. Pembelajaran yang semi-diawasi

Metode ini menggabungkan pembelajaran yang diawasi dan yang tidak diawasi, seperti namanya. Teknik ini menggunakan sejumlah kecil data berlabel dan tidak berlabel untuk melatih sistem. Pertama, algoritme pembelajaran mesin secara parsial dilatih dengan data berlabel. Setelah itu, algoritme itu sendiri melabeli data yang tidak berlabel. Setelah itu, tanpa diprogram secara eksplisit, model dilatih kembali pada campuran data yang dibuat melalui proses yang dikenal sebagai pelabelan semu. Keuntungan dari teknik ini adalah bahwa Anda tidak perlu menyimpan jumlah data berlabel yang besar. Saat bekerja dengan data yang panjang, metode ini berguna[3].

d. *Machine Learning* penguatan

Pembelajaran penguatan adalah teknik di mana algoritme diberi nilai penghargaan untuk langkah-langkah yang berbeda yang harus dilalui. Oleh karena itu, tujuan model ini adalah untuk mengumpulkan sebanyak mungkin poin penghargaan sebelum mencapai tujuan akhir. Dalam sepuluh tahun terakhir, kebanyakan aplikasi pembelajaran penguatan telah muncul dalam gim video. Dalam gim klasik dan modern, algoritme pembelajaran penguatan yang canggih telah mencapai hasil yang mengesankan; mereka seringkali secara signifikan mengalahkan pasangan manusia mereka.

Metode ini jarang digunakan dalam lingkungan bisnis, meskipun paling efektif dalam lingkungan data yang tidak pasti dan kompleks. Untuk tugas yang terdefinisi dengan

baik, metode ini tidak efektif, dan bias developer dapat memengaruhi hasil. Ilmuwan data dapat mempengaruhi hasil karena mereka membuat penghargaan[3].

2.3 Chatbot

Chatbot adalah program komputer yang berkomunikasi dengan manusia melalui teks, suara atau visual. Ini adalah percakapan yang terjadi antara manusia dan sistem, yang sudah diprogram untuk mengikuti respons yang telah ditetapkan sebelumnya[4]. Chatbot menggunakan berbagai teknologi terbaru, seperti kecerdasan buatan atau *Artificial Intelligence* (AI), *Machine Learning*, *Deep Learning*, dan *Natural Language Processing* (NLP). Chatbot bekerja dengan memindai kata kunci yang paling cocok, atau dengan pola kata yang paling mirip dengan basis data tekstual. Chatbot akan memberikan jawaban yang sesuai berdasarkan kata kunci teks atau video[5].

Ada 3 macam pola kerja chatbot, yaitu:

a. *Pattern Matching*

Metode Matching Pattern memungkinkan bot untuk bekerja dengan menyesuaikan pola saat mengelompokkan teks dengan memindai kata kunci, kemudian memberikan jawaban yang sesuai dengan kata kunci tersebut. Jika ada permintaan yang tidak sesuai dengan pola atau kata kunci, bot tidak akan mampu merespon.

b. *Decision tree-based*

Metode ini dianggap kurang ramah pengguna karena pengguna diharuskan mengikuti urutan jawaban yang diprogramkan pada bot. Hal ini terjadi karena pengguna hanya dapat memilih pertanyaan yang tersedia pada sistem, dan pengguna diberikan pilihan berupa tombol yang berisi teks dari jawaban pertanyaan yang ditunjukkan pada widget.

c. *Contextual*

Metode ini unggul dari yang lain. Pengembang membutuhkan perencanaan yang matang sebelum membuat bot dengan metode contextual, yang memungkinkan bot untuk berbicara dengan cara yang terlihat natural. karena membutuhkan database yang cukup luas untuk memenuhi permintaan yang berbeda dari pelanggan.

Perkembangan Chatbot :

A. 1966, ELIZA

Eliza, yang diciptakan oleh ilmuwan komputer Joseph Weizenbaum pada tahun 1966, adalah induk dari semua chatbot modern. Joseph Weizenbaum mencoba membuat

mesin yang dapat lulus Tes Turing karena dia sangat tertarik dengan pekerjaan Alan Turing. Eliza hanya menggunakan dua ratus baris kode dan belajar alur percakapan dari terapis. Sistem operasi Eliza dirancang untuk mengenali kata atau kalimat kunci yang dimasukkan ke dalamnya, sehingga dapat membuat respons yang sesuai dengan yang sudah ada di program. Jika ada kata-kata seperti "hari ini aku akan pergi ke kantor", Eliza akan menemukan kata "kantor" dan menjawab, "ceritakan lebih banyak tentang kantormu." Dengan cara ini, pengguna akan memiliki perasaan bahwa dia sedang berinteraksi dengan orang yang benar-benar peduli dengannya[6].

Eliza dianggap sebagai obrolan pertama dalam sejarah ilmu komputer karena dia adalah mesin dengan kemampuan berbicara seperti manusia yang pertama. Namun, istilah Chatterbot pertama kali muncul pada tahun 1994 oleh Michael Mauldin. Chatterbot Eliza, juga disebut sebagai chatterbot, telah berubah dan terus berkembang dengan baik. Banyak robot muncul setelah Eliza. Misalnya, Parry muncul pada tahun 1972, Racter muncul pada tahun 1983, dan Alice muncul pada tahun 1995. MSN dan AOL juga menggunakan chatbot dalam sistem telepon dengan alur percakapan sederhana[6].

B. 1972, PARRY

Kenneth Colby membuat chatbot pertama, Parry, pada tahun 1972. Seorang psikiater merancang Parry untuk meniru pemikiran orang yang paranoid atau skizofrenia. Kenneth Colby menciptakan Parry karena ketidakpuasan terhadap psikoanalisis yang tidak mampu menghasilkan data yang akurat untuk kemajuan ilmu pengetahuan. Colby percaya bahwa penggabungan komputer dengan pikiran akan menghasilkan hasil yang lebih ilmiah dalam studi penyakit mental secara keseluruhan. Dengan menggunakan pengalamannya di bidang psikiatris, Colby merancang Parry. Parry bukanlah chatbot; dia lebih seperti AI[6].

C. 1995, ALICE

ALICE (*Artificial Linguistic Internet Computer Entity*) dibuat oleh Richard Wallace pada tahun 1995. Karena kemajuan teknologi saat ini, Alice sekarang dapat menggunakan pemrosesan bahasa alami, juga dikenal sebagai pemrosesan bahasa alami (NLP). Alice ditulis dalam bahasa pemrograman *AI Markup Language* (AIML), yang memungkinkannya memberikan respons yang lebih kompleks. Alice akan menyimpan obrolan yang dia terima dan memasukkannya ke dalam basis datanya. Alice dapat didownload dan diubah oleh siapa saja karena bersifat Open Source. Sekitar lebih dari 500 orang telah membantu pengembangan Alice dan telah berhasil membuat 100.000 baris AIML[6].

D. 2020, Masa Kini

Chatbot sekarang dapat dilihat ketika kita membeli sesuatu, di aplikasi smartphone, atau di website. Perusahaan besar seperti Apple dan Amazon juga sedang berusaha untuk membuat mesin yang dapat meniru percakapan manusia, bahkan tanpa mengetik, hanya dengan bicara. Siri dari Apple dan Alexa dari Amazon akan menjawab dengan cara yang sama seperti orang normal[6].

2.4 *Natural Language Processing*

Natural Language Processing (NLP) adalah gabungan dari bidang ilmu komputer dan kecerdasan buatan yang berkaitan dengan linguistik. Proses pengolahan bahasa manusia (NLP) adalah tentang bagaimana mesin memahami bahasa manusia untuk berinteraksi satu sama lain. Melalui pengolahan bahasa manusia, komputer memiliki kemampuan untuk belajar dan memahami bahasa manusia, yang memungkinkan mereka untuk berkomunikasi dengan manusia. Bahasa manusia unik karena dirancang untuk menyampaikan makna. Membuat komputer dapat mengerti bahasa manusia adalah tugas yang sulit karena strukturnya yang kompleks. Selain itu, setiap bahasa memiliki keunikannya sendiri dan mungkin memiliki makna ganda. Sebagai contoh dapat dilihat dari kalimat berikut, "*Look at the dog with one eye*", di mana kalimat tersebut dapat memiliki arti "melihat anjing dengan satu mata" atau "melihat anjing yang mempunyai mata satu"[7].

Syntactic Analysis (analisis sintaksis) dan *Semantic Analysis* (analisis semantik) adalah dua pendeketan utama untuk memahami proses pengolahan bahasa natural atau *Natural Language Processing*. Kedua pendekatan ini digunakan untuk memverifikasi struktur bahasa. *Syntactic Analysis* (analisis sintaksi) adalah teknik untuk mengatur kalimat sehingga memiliki bahasa yang tepat, sedangkan *Semantic Analysis* (analisis semantik) berfokus pada penafsiran. *Syntactic Analysis* (analisis sintaksi) melibatkan penentuan struktur kalimat seperti subjek, predikat, kata benda, kata kerja, kata ganti dan sebagainya. Sistem dapat membaca kalimat yang dimasukkan, memecahnya menjadi kata-kata, dan kemudian menghasilkan deskripsi yang terstruktur. Untuk membuat pencarian informasi lebih mudah, teknik ini dapat digunakan untuk menyederhanakan kalimat. *Syntactic Analysis* (analisis sintaksi) juga dapat membantu menemukan kata atau kalimat baru atau tidak biasa[7].

Semantic Analysis (analisis semantik) memproses teks untuk mengidentifikasi dan memahami topik yang dimaksud. *Semantic Analysis* juga mempelajari hubungan antara berbagai konsep dalam teks. Sebagai contoh, apabila sebuah teks terdapat kata "*money" dan "accounting*", maka topik yang sedang dibahas berkaitan dengan "*economy"* [7].

2.5 *Restaurant*

Restaurant adalah bisnis yang menyiapkan dan menyajikan makanan maupun minuman kepada pelanggan dengan imbalan uang. Makanan biasanya disajikan di tempat, tetapi banyak *restaurant* yang juga menawarkan layanan antar makanan dan pengiriman makanan. *Restaurant* sangat bervariasi dalam penampilan dan layanan, termasuk berbagai macam masakan dan model layanan, mulai dari jenis *restaurant* cepat saji dan kafetaria hingga *restaurant* keluarga dengan harga menengah, dan perusahaan mewah dengan harga tinggi[8].

Dalam memilih sistem informasi restoran untuk memenuhi kebutuhan bisnis saat ini dan masa depan, berikut beberapa hal yang perlu dipertimbangkan :

a. Standardisasi *Restaurant*

   Bisnis restoran dimulai dengan satu unit bisnis yang sukses yang memperoleh kepercayaan pasar. Faktor-faktor yang berkontribusi pada kesuksesan restoran, seperti resep masakan, suasana, tata letak, dan lainnya, telah menjadi standar. Oleh karena itu, pemilik bisnis restoran akan merasa perlu untuk mematuhi standar ini di setiap cabang bisnis mereka. Dari awal, restoran telah memiliki nilai unik di pasar yang harus dijaga oleh setiap divisinya. Baik penerapan maupun pemantauan standarisasi akan menjadi lebih mudah dengan modul ini.

b. Visibilitas Aktifitas Usaha *Restaurant*

   Dengan memiliki pemantauan secara real-time terhadap seluruh operasi restoran, manajer di seluruh rantai produksi memiliki lebih banyak pilihan yang dapat mereka buat. Dalam setiap perusahaan restoran, aktivitas yang dipantau mungkin berbeda. Alat pemantauan ini harus memiliki kemampuan untuk mengukur hal-hal seperti kecepatan pelayanan, kebersihan restoran, dan sebagainya. Untuk memudahkan pemantauan fisik, modul ini harus diintegrasikan dengan CCTV. Pada akhirnya, perusahaan akan menghasilkan peningkatan efisiensi dalam hal pengawasan.

c. Analisa dan Pelaporan Usaha *Restaurant*

   Dengan menyimpan dan mengelola semua data historis dengan cara terbaik, analisis dan pelaporan yang akurat dapat dilakukan. Pelaporan yang akurat akan menunjukkan keadaan sebenarnya, dan analisis yang akurat akan memberikan gagasan untuk meningkatkan pertumbuhan bisnis.

d. Ekosistem Digital Pada Sebuah Bisnis *Restaurant* Banyak Cabang

Kecepatan dan ketepatan sangat penting untuk memenuhi standardisasi restoran. Sistem informasi restoran yang kompleks memungkinkan tugas rutin dilakukan secara otomatis.

Gambar 2.2 Ekosistem Digital Bisnis *Restaurant* Multi Site

Pada restoran yang memiliki banyak cabang, ekosistem digital harus sedapat mungkin disederhanakan. Tujuannya adalah untuk menjaga kecepatan pengembangan dan kelancaran operasional. Dalam era digitalisasi, semua orang berlomba-lomba untuk membuat sistem lebih baik untuk memenuhi kebutuhan internal dan memenuhi kebutuhan pelanggan. Agar perusahaan dapat menerapkan ide dan inovasi untuk mendukung pertumbuhan mereka lebih cepat, infrastruktur TI harus solid untuk mendukung pola kerja DevOps.

Selain itu, perusahaan harus menentukan modul dan fitur apa yang diperlukan untuk sistem informasi restoran.

A. Kantor Pusat

i) Manajemen Outlet

Sistem informasi karyawan/HRD, aset tetap, dan lainnya akan tersedia dalam modul ini. Untuk membuatnya lebih standar, manajemen menu juga dapat ditambahkan. Selain itu, alat untuk memantau kinerja karyawan secara real-time harus ada di submodul manajemen karyawan. agar pusat dan cabang dapat bekerja sama untuk menilai kualitas pelayanan restoran.

ii) Manajemen Persediaan

Mulai dari pembelian hingga transfer saham Menarik data di cabang juga harus didukung oleh sistem manajemen persediaan ini, sehingga dapat digunakan untuk restoran sistem franchise.

iii) Manajemen Produksi

Modul ini terhubung langsung ke modul laporan keuangan, modul manajemen menu masakan (yang mencakup kategori standardisasi restoran) dan modul

persediaan. Dengan modul produksi, perhitungan harga pokok persediaan akan lebih akurat.

iv) Laporan Keuangan

Seluruh biaya dan pendapatan akan dikategorikan menurut masing-masing cabang dan pusat. Ini juga akan mencakup laporan laba rugi per cabang dan konsolidasi, serta memiliki cash flow. Untuk restoran franchise, pusat biaya dan keuntungan menjadi penting. Selain itu, modul pelaporan keuangan harus memiliki manajemen hutang dan piutang (AR/AP). Manajemen AR/AP akan mencakup umur hutang dan piutang dan penjadwalan jatuh tempo, dan modul AR/AP harus memiliki peringatan jatuh tempo. Selain itu, modul ini terhubung ke sistem informasi SDM untuk sistem gaji karyawan.

Kontrol posting adalah bagian penting dari sistem informasi akuntansi. Dalam situasi seperti ini, beberapa posting akan dilakukan secara otomatis, dan beberapa lainnya akan dilakukan secara manual. Yang terbaik adalah seluruh transaksi dapat dicatat secara otomatis, sehingga posting manual hanya diperlukan untuk penyesuaian. Pada akhirnya, laporan keuangan harus memenuhi standar akuntansi yang berlaku umum (SAK-ETAP/IFRS) dan akurat.

B. Cabang

Point of Sales dengan fitur :

i) Pemesanan

Proses pemesanan makanan dapat dilakukan di meja kasir. Namun akan lebih meningkatkan layanan jika pemesanan dapat dilakukan dari tempat duduk pelanggan. Hal ini memerlukan *Point of Sales* berupa aplikasi *mobile*, sehingga seluruh karyawan dapat menerima order. Jika kasir berlaku sekaligus penerima *order*, tentunya akan terlalu sibuk dan dapat meningkatkan kesalahan. Pelanggan harus dapat mengetahui apa yang mereka pesan dan berapa jumlah yang di pesan untuk menghindari kesalahan pemesanan. Seluruh pesanan akan diterima juga pada bagian kasir dan manajemen produksi (dapur). Fitur PoS ini juga dapat memberikan program keanggotaan yang dapat terintegrasi pada modul pemasaran melalui saluran *digital* (*digital marketing*).

ii) Pembayaran

Sebuah PoS akan dilengkapi dengan laci uang dan mesin pencetak nota pembayaran. Selain itu, juga ada fasilitas untuk pembayaran non tunai, seperti kartu debit, kupon dan sebagainya. Karyawan kasir akan lebih dekat dengan

bagian administrasi. Tugasnya menjaga transaksi pembayaran yang kemudian diserahkan ke bagian keuangan. Modul pembayaran pada PoS akan perlu terhubung dengan modul keuangan.

iii) Hak Akses Bertingkat

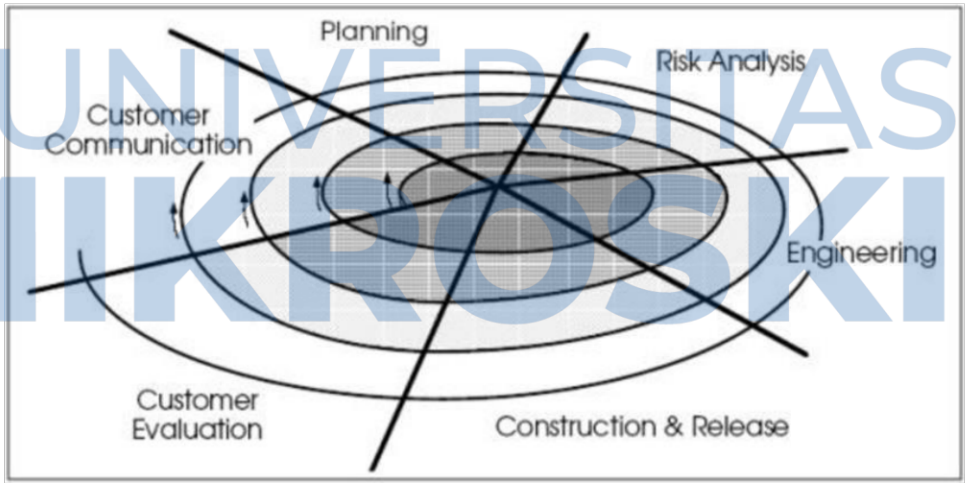
Ini adalah modul yang dimaksudkan untuk meningkatkan standar kepatuhan di ekosistem digital. Ini meningkatkan pengendalian dengan memberikan hak akses yang berbeda kepada manajemen, supervisor, dan karyawan kasir. Selain itu, modul ini akan terhubung ke audit trail untuk melacak tindakan perubahan.

iv) *Audit Trail*

Berfungsi untuk melacak aktivitas pada PoS yang memberikan informasi, kapan dan siapa yang melakukan aktivitas tersebut. *Audit trail* dibutuhkan untuk memenuhi unsur kepatuhan dan standar akuntabilitas.

2.6 Metodologi Spiral

Metode Spiral adalah model proses perangkat lunak evolusioner yang menghubungkan sifat iteratif prototipe melalui aspek kontrol dan sistem dari model hasil linier. Model ini memungkinkan pengembangan perangkat lunak versi lain dengan cepat. Rilis tambahan dapat berupa model kertas atau prototipe selama iterasi pertama. Pada iterasi berikutnya, versi yang lebih lengkap dari sistem teknik diproduksi[9].



Gambar 2.3 Tahapan Metode Spiral

Dalam model spiral memiliki tahapan-tahapan yaitu *customer communication*, *planning*, *risk analysist*, *engineering*, *contruction & release*, *customer evaluation.* Dalam penelitian sebelumnya menjelaskan sebagai berikut [9]:

1. *Customer Communication*

Pada tahap ini, pelanggan dan sistem berkomunikasi tentang permintaan mereka. Seperti pengumpulan data yang terdiri dari observasi dan wawancara, kebutuhan sistem dan kebutuhan pengguna, teknik pengumpulan data sangat penting bagi penelitian.

2. *Planning*

   Proses perencanaan yang menetapkan tujuan dan cara untuk mencapainya. Ini termasuk menentukan waktu pengerjaan, sumber daya dan informasi lainnya, seperti spesifikasi software dan hardware yang digunakan.

3. *Risk Analysist*

   Analisis risiko ini dilakukan untuk mengevaluasi risiko teknis pengelolaan dan teknologi. Diagram Relasi Entitas atau *Entity Relationship Diagram* (ERD) adalah model penjelas relasi dalam database yang didasarkan pada pemahaman kata objek. ERD digunakan dalam perancangan sistem.

4. *Engineering*

   Membangun satu atau lebih representasi dari aplikasi tersebut membutuhkan pekerjaan *engineering*. Jika pengguna menemukan bug atau fungsi *update* saat menggunakan sistem, maka *maintenance* akan dilakukan. Panel admin ini akan dibuat dengan PHP (h*ypertext preprocessor*) sebagai bahasa pemrograman, bootstrap dan laravel sebagai framework dan *Visual Studio Code* sebagai *code editor.* PHP (hypertext preprocessor) adalah sebuah bahasa pemrograman berbasis web yang digunakan untuk membuat aplikasi berbasis web, seperti website, blog, atau aplikasi web. Bootstrap adalah template untuk front-end situs web. *Visual Studio Code* adalah *code editor* yang dikembangkan oleh microsoft untuk W*indows*, L*inux* dan *MacOs* yang mendukung pemrograman dalam bahasa PHP. Laravel adalah satu-satunya framework yang memungkinkan Anda memaksimalkan penggunaan PHP selama proses pengembangan website.

5. *Contruction & Release*

   Aktivitas yang diperlukan untuk pembangunan perangkat lunak, pengujian, instalasi, dan penyediaan kepada pengguna atau dukungan pelanggan, seperti pelatihan penggunaan perangkat lunak dan dokumen seperti buku petunjuk perangkat lunak. Tes fungsionalitas seperti perangkat lunak, perangkat keras, dan pengujian kotak hitam dilakukan. Teknologi pengujian kotak hitam berfokus pada informasi perangkat lunak dan menghasilkan kasus uji dengan membagi masukan dan keluaran dari program, termasuk pengujian komprehensif.

6. *Customer Evaluation*

Evaluasi pengguna atau pelanggan selama presentasi perangkat lunak dalam fase rekayasa atau implementasi selama instalasi, serta fase konstruksi dan rilis, akan membantu Anda mendapatkan aktivitas yang diinginkan.

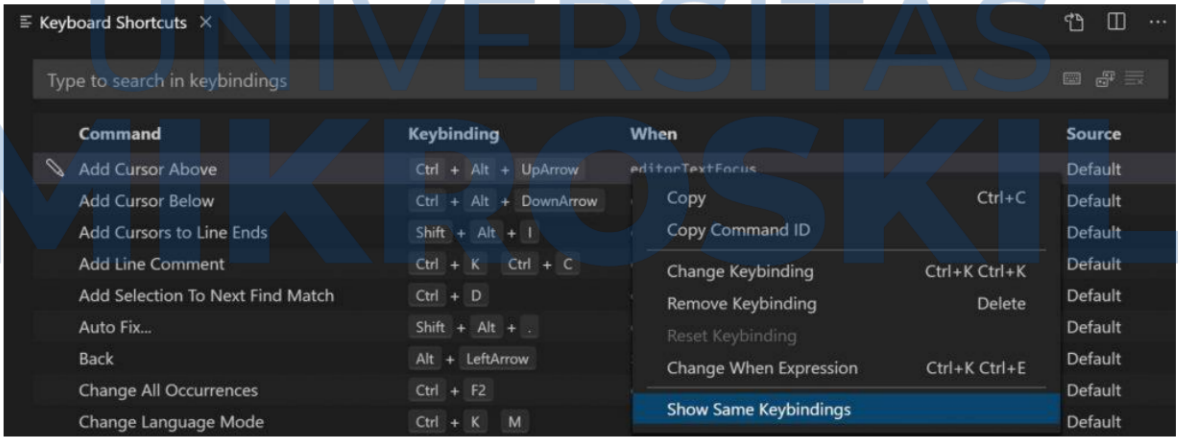
2.7 *Visual Studio Code*

Visual Studio Code adalah sebuah code editor gratis yang dikembangkan oleh salah satu raksasa teknologi dunia, *Microsoft*, dan dapat digunakan di perangkat desktop berbasis *Windows*, *Linux*, dan *MacOS*. Namun, *Visual Code* adalah program editor yang ringan meskipun memiliki kemampuan yang luar biasa. Dengan *Visual Code Studio*, Anda dapat membuat dan mengedit source code berbagai bahasa pemrograman, seperti JavaScript, TypeScript, dan Node.js. Bahkan, *Visual Code Studio* bekerja dengan bahasa dan runtime environment lain, seperti PHP, Python, Java, dan.NET. Hal ini dimungkinkan oleh ekosistemnya yang luas dan banyaknya extension yang tersedia[10].

Berikut ini beberapa fitur *Visual Code Studio* yang menjadikannya sebagai software editor paling banyak digunakan saat ini[10]:

a. *Basic Editing*

Kemampuan *Visual Code Studio* untuk coding tidak diragukan lagi, mengingat fungsinya sebagai editor kode. Ia memiliki semua yang diperlukan. Anda dapat memilih antara *Shortcuts Keyboard*, Pilihan Banyak, dan pilihan kolom.

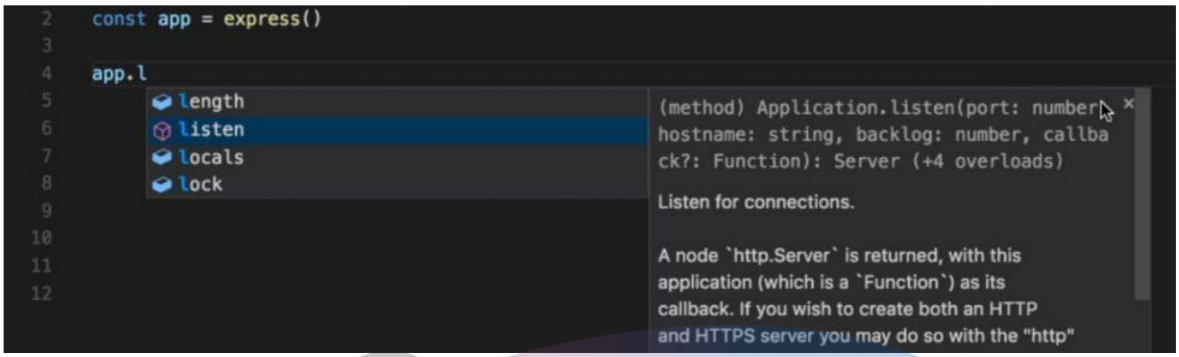


Gambar 2.4 Fitur *Basic Editing*

Bahkan, *Visual Code* memasukkan fitur *Auto Save* dan *Hot Exit*, yang membantu mencegah kejadian tidak diinginkan seperti lupa menyimpan *file*.

b. *IntelliSense*

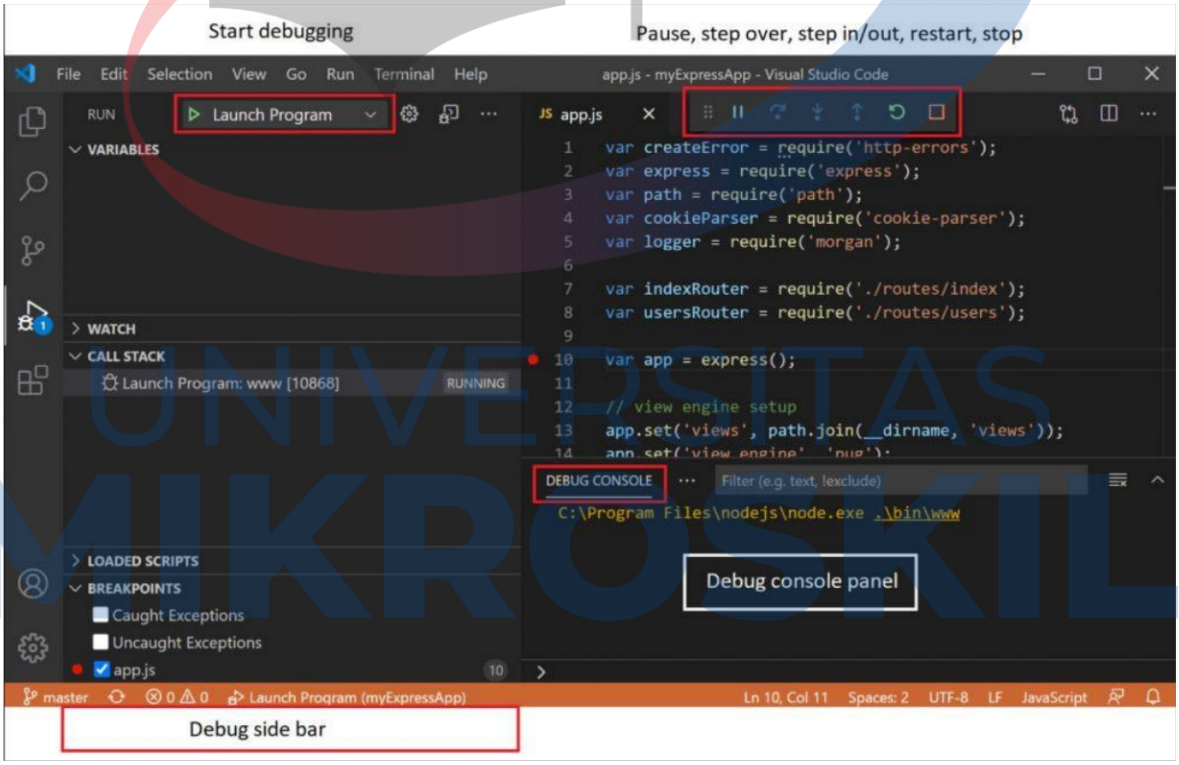
IntelliSense adalah fitur *Visual Studio* yang memungkinkan coding menjadi lebih mudah. Caranya mirip dengan Autocomplete, menyarankan kata-kata lengkap berdasarkan apa yang Anda ketik.



Gambar 2.5 Fitur *IntelliSense*

c. *Debugging*

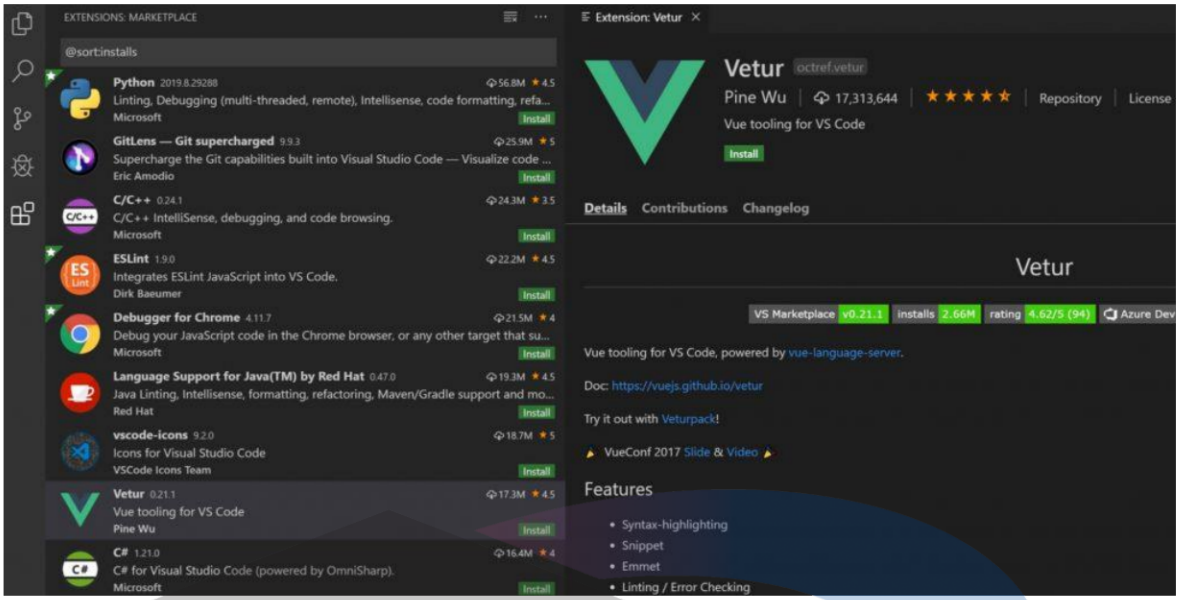
Debugging, fitur penting lainnya di *Visual Code*, membantu Anda mengedit, meng-*compile*, dan mengeksekusi kode berulang kali (*looping*).



Gambar 2.6 Fitur *Debugging*

d. *Extension Marketplace*

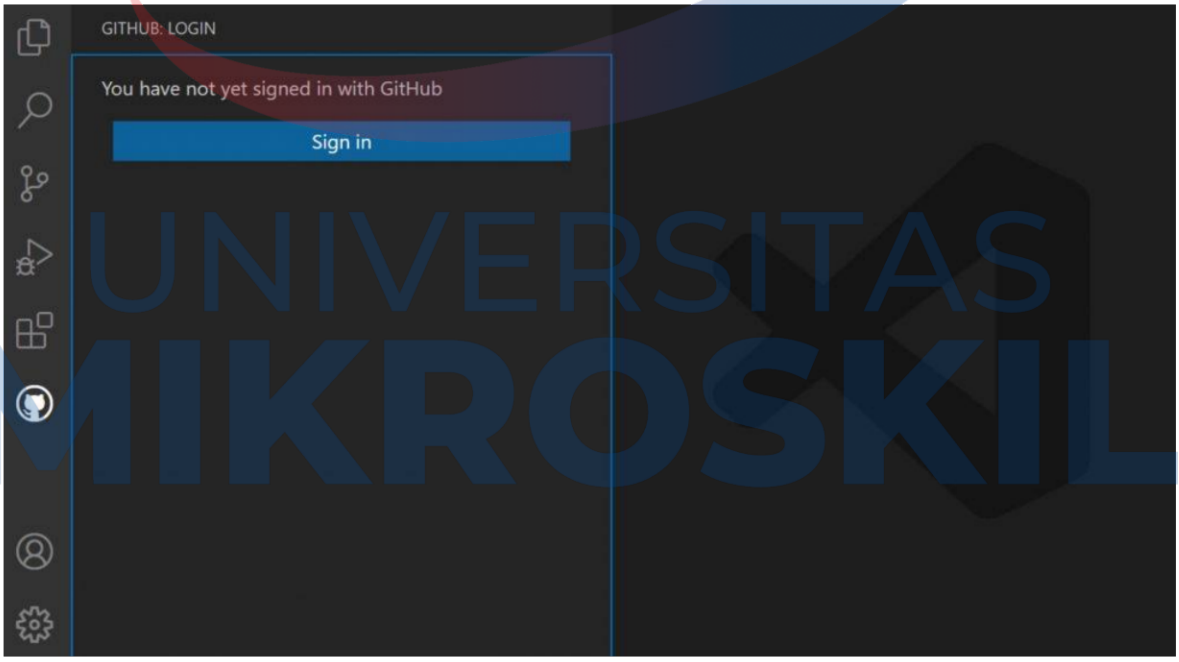
Fitur yang membuat *Visual Code Studio* unggul jauh dari pesaingnya adalah *Extension Marketplace*. Dengan *Extension*, Anda dapat menginstal alat, debuggers, dan bahkan bahasa pemrograman tambahan dengan mudah.



Gambar 2.7 Fitur *Extension Marketplace*

e. *Github Integration*

Integrasi dengan *Github*, platform manajemen proyek terpopuler di dunia, merupakan fitur unggulan tambahan dari *Visual Code*. Anda dapat berkolaborasi dengan rekan kerja dan berbagi kode di sini tanpa berpindah *software*.



Gambar 2.8 Fitur *Github Integration*

Berikut ini beberapa kelebihan *Visual Studio Code* [10] :

a. Tersedia di Banyak *Platform*

Dengan menginstal *Visual Code* pada platform *Linux*, *MacOS*, dan *Windows*, tidak ada lagi kendala dukungan terhadap perangkat yang Anda gunakan.

b. Fitur yang Lengkap

Keunggulan utama Visual Studio adalah kemampuan untuk melengkapi fiturnya. Ini dapat dicapai dengan adanya *Extension Marketplace*, yang memungkinkan Anda menambah fiturnya dengan bebas.

c. Performa Secepat Kilat

Meskipun *Visual Code Studio* memiliki banyak *extension*, kinerjanya tetap stabil. Ini karena extension yang tersedia telah dioptimalkan sehingga tidak mempengaruhi kinerja kode editor ini.

d. Dukungan Arsitektur Terbaik

Berbagai teknologi terbaik di dalam *Visual Studio* digunakan, seperti *Electron* untuk pengembangan JavaScript dan Node.js, Monaco Cloud Editor untuk HTML, Roslyn untuk.NET, dan lainnya.

2.8 *Entity Relationship Diagram* (ERD)

ERD adalah gambar atau diagram yang menunjukkan hubungan antara entitas atau objek dalam sebuah database. ERD menunjukkan entitas sebagai kotak yang memiliki atribut yang terkait dengannya. Tanda panah atau garis yang menghubungkan dua entitas menunjukkan hubungan mereka satu sama lain. ERD memungkinkan pengembang database melihat struktur database dan memahami bagaimana entitas saling terkait. Bentuknya mirip dengan diagram yang menjelaskan hubungan antar objek data. Untuk menggambarkannya dibutuhkan[11]:

1. Notasi ialah seperangkat lambing yang menggambarkan data.
2. Simbol sebagai lambang sebagai penanda.
3. Bagan merupakan rancangan atau skema untuk mempermudah penafsiran.

Kegunaan ERD yaitu, sebagai berikut [11]:

a. Merencanaan Struktur Database

Dengan memvisualisasikan entitas, atribut, dan hubungan antara entitas, ERD membantu pengembang merencanakan dan merancang struktur database yang efektif.

b. Identifikasi Ketergantungan dan Hubungan

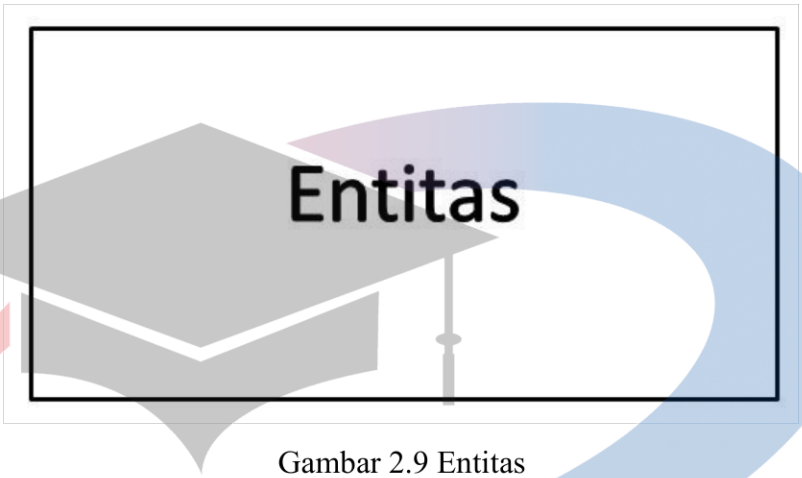
ERD membantu pengembang mengidentifikasi hubungan dan ketergantungan antara entitas dan entitas di database. Ini juga membantu dalam menentukan organisasi data dan cara akses ke data akan dilakukan.

c. Menyederhanakan Komunikasi dengan stakeholder

ERD memberikan pemangku kepentingan gambaran visual yang jelas tentang struktur database. Dengan menggunakan ERD, pengembang dapat dengan mudah menjelaskan dan berkomunikasi dengan pemangku kepentingan tentang bagaimana data akan dihubungkan dan diorganisasikan dalam database.

Entity Relationship Diagram terdiri dari tiga komponen utama, yang disebut sebagai penyusun atau notasi[11].

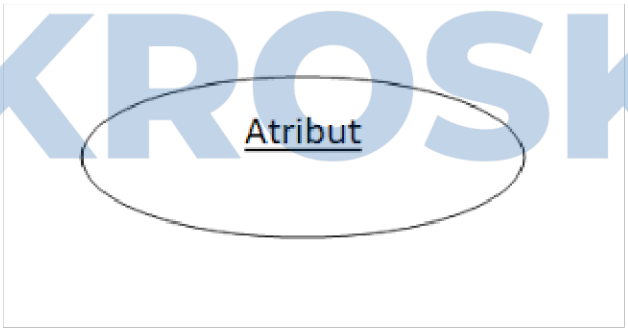
1. Entitas (*Entity*)



Gambar 2.9 Entitas

Entitas adalah benda nyata yang dapat dibedakan dari benda lain. Objektifnya dapat nyata atau abstrak. Data abstrak tidak berwujud, sedangkan data konkret adalah sesuatu yang benar-benar ada atau dapat dirasakan oleh alat indra. Orang, buku, karyawan, dan perusahaan adalah contoh entitas konkret. Berbeda dengan mata kuliah, kejadian dan pekerjaan tidak ada[11].

2. Atribut (*field*)

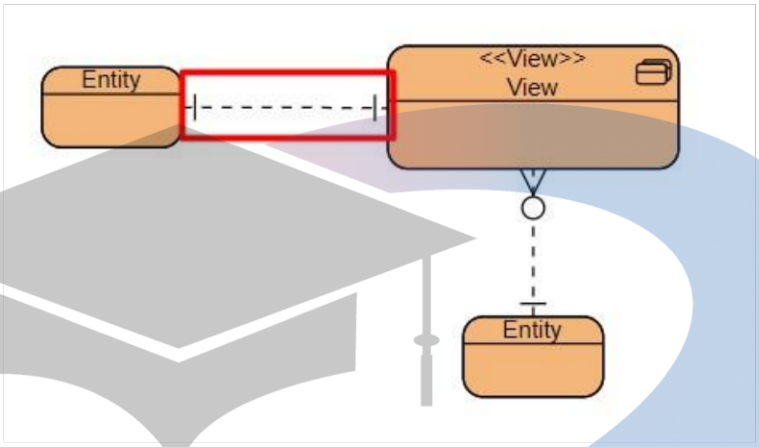


Gambar 2.10 Atribut

Pengertian ERD yang kedua adalah field atau atribut. Setiap entitas memiliki ciri-ciri yang menjelaskan karakteristiknya. Dibagi menjadi beberapa kategori berdasarkan jenisnya, misalnya. Atribut key, atribut yang unik dan berbeda. Misalnya, Nomor pokok mahasiswa (NPM), NIM dan nomor pokok lainnya.

Atribut Composite, atribut yang terdiri dari beberapa sub atribut yang memiliki arti tertentu. Contohnya, nama lengkap yang dipecah menjadi nama depan, tengah, dan belakang. Dan atribut deviratif, yang dihasilkan dari atribut atau relasi lain. Jenis atribut ini tidak wajib ditulis dalam diagram ER atau pun disimpan dalam database. Sebagai contoh deriative attribute adalah usia, kelas, selisih harga, dan lain-lain[11].

3. Relasi (*Relation*)



Gambar 2.11 Relasi

Relasi, atau hubungan antar entitas, yang menunjukkan adanya hubungan di antara berbagai entitas dari himpunan entitas yang berbeda. Misalnya, hubungan antara mahasiswa dan mata kuliah dalam sistem akademik adalah "mengambil". Mahasiswa mengambil mata kuliah[11].

Kardinalitas relasi, juga dikenal sebagai rasio kardinalitas, digunakan dalam ERD untuk menggambarkan bagaimana data berhubungan satu sama lain. Kardinalitas relasi ini terbagi menjadi empat kategori,
yaitu[11]:

Relasi Pertama, *One to One* (1:1). Apa arti dari semua ini? Misalnya, ada entitas A dan B. Setiap entitas dalam himpunan entitas A memiliki hubungan paling banyak dengan satu entitas dalam himpunan entitas B, dan sebaliknya. Oleh karena itu, setiap anggota dari himpunan entitas A hanya dapat berhubungan dengan satu anggota dari himpunan entitas B. Sebagai contoh, satu siswa (1) memiliki satu nomor siswa, dan sebaliknya[11].

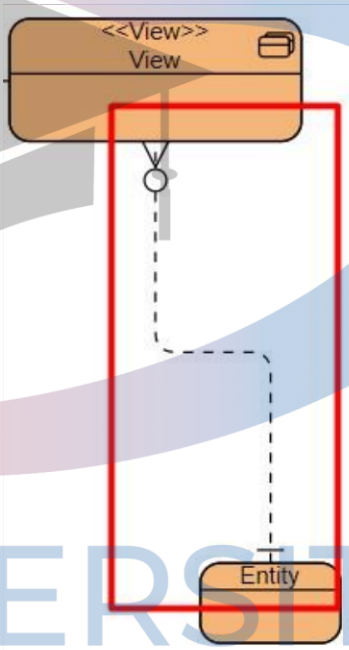
Relasi kedua, *One to Many* (1:M). Satu dari banyak ini berarti bahwa setiap entitas dalam himpunan entitas A memiliki kemampuan untuk berhubungan dengan banyak entitas dalam himpunan entitas B; dengan kata lain, setiap anggota entitas A memiliki kemampuan untuk berhubungan dengan lebih dari satu anggota entitas B; namun, tidak benar sebaliknya.

Satu kelas (1) memiliki banyak siswa (M), atau banyak siswa mengikuti ekstrakurikuler. Ini adalah contoh dari relasi One to Many ini[11].

Relasi Ketiga, *Many to One* (M:1), adalah kebalikan dari dua relasi sebelumnya. Misalnya, yaitu banyak pekerja (M) bekerja dalam satu departemen (1) atau banyak guru mengajar dalam satu mata kuliah[11].

Relasi keempat, *Many to Many* (M:N). Dalam kumpulan data entitas A, banyak entitas dapat berhubungan dengan banyak entitas pada kumpulan data entitas B. Misalnya, banyak siswa (M) mempelajari banyak pelajaran (N), dan sebaliknya, banyak siswa (M) mempelajari banyak pelajaran (N)[11].

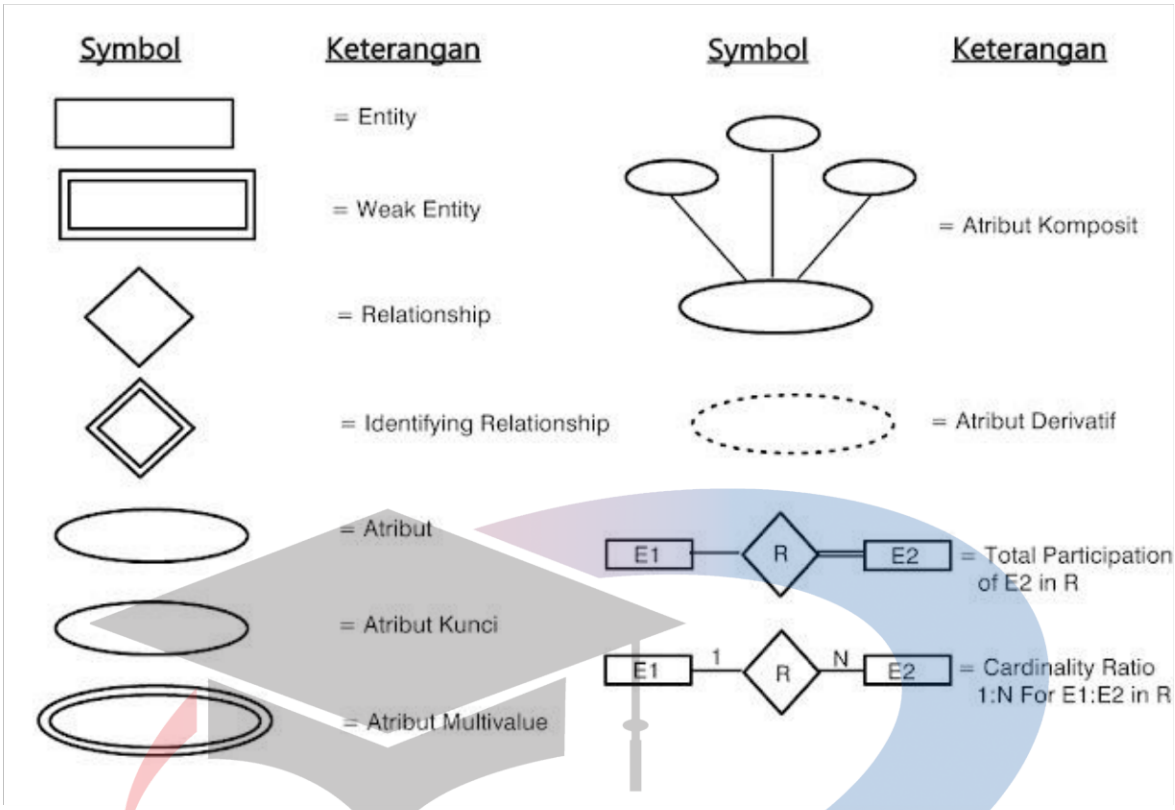
4. Garis



Gambar 2.12 Garis

Garis ini tidak hanya berfungsi untuk menghubungkan himpunan relasi dengan himpunan entitas, tetapi juga untuk menghubungkan himpunan entitas dengan atributnya. Untuk membuat awal dan akhir ERD jelas, garis dapat membantu pengguna melihat dan memahami alurnya[11].

Contoh notasi yang ada dalam *Entity Relationship* Diagram ditunjukkan di bawah ini. Individu diwakili dengan persegi panjang. Attrait dapat digambarkan dalam bentuk elips atau oval. Untuk jenis atribut lainnya, seperti atribut tombol, ditandai dengan garis di dalamnya[11].



Gambar 2.13 Macam-macam notasi pada ERD

Tidak sama dengan karakteristik komposit, yang digambarkan dengan lingkaran dan lingkaran tambahan yang terhubung ke garis, yang menunjukkan bahwa karakteristik ini terdiri dari banyak karakteristik kecil. Sebaliknya, karakteristik derivative digambarkan dengan lingkaran bergaris putus-putus. Simbol berbentuk belah ketupat atau safir biasanya digunakan untuk menggambarkan hubungan[11].

Berikut adalah tahap-tahap dalam membuat ERD[11]:

a. Tahap 1 - Menentukan entitas yang akan terlibat atau menentukan tabel
   Contoh masalah berasal dari SIAKAD Perkuliahan. Dalam sistem SIAKAD suatu perguruan tinggi, mahasiswa dapat melakukan input KRS, melihat KHS, mengambil banyak mata kuliah, dan mata kuliah diambil oleh para mahasiswa.Sebagai contoh, untuk penentuan entitasnya bisa dijadikan sebagai Mahasiswa, Dosen, Ruang, dan Mata_Kuliah[11].
b. Tahap 2 - Menentukan atribut-atribut key dari masing-masing himpunan entitas.
   Di tahap penentuan atributnya maka akan menjadi seperti ini[11]:
   1. Mahasiswa: nim, nama_mhs, almt_mhs
   2. Dosen: nip, nama_dsn, almt_dsn
   3. Mata kuliah: kd_mk, nama_mk
   4. Ruang: kd_ruang, lokasi, kapasitas

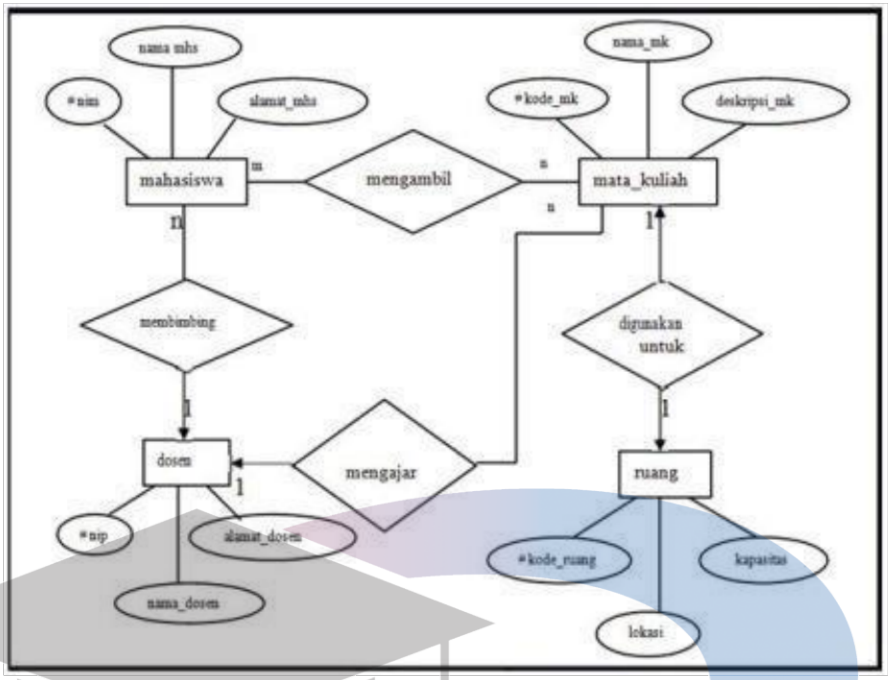
c. Tahap 3 - Menetapkan seluruh himpunan relasi di antara himpunan entitas yang ada beserta foreign key-nya dan kardinalitas relasi.

Setelah menentukan atribut-atributnya, maka langkah selanjutnya menetukan relasi. Inilah contoh relasi beserta kardinalitasnya[11].

1. Mahasiswa (M) mengambil Mata_Kuliah (N), yaitu banyak mahasiswa mengambil banyak mata kuliah.
2. Dosen (1) membimbing Mahasiswa (N), yaitu satu dosen membimbing banyak mahasiswa.
3. Dosen (1) mengajar Mata_Kuliah (N), yaitu satu dosen mengajar banyak mata kuliah.
4. Ruang (1) digunakan untuk Mata_Kuliah (1), yaitu satu ruang dapat digunakan untuk satu mata kuliah saja.

d. Tahap 4 - Membuat model Entity Relationship Diagram.

Setelah menetapkan hubungan dan rasio kardinalitas, selanjutnya membuat diagram ER. Dalam tahap ini hanya menggabungkan seluruh himpunan dan relasi dari tahap ketiga. Dimulai dari mahasiswa. Mahasiswa mengambil Mata_Kuliah, Dosen membimbing Mahasiswa, Dosen mengajar Mata_Kuliah, dan Ruang untuk Mata_Kuliah[11].



Gambar 2.14 Contoh ERD pada SIAKAD perkuliahan

Selanjutnya menentukan *primary key*, key yang dapat digunakan untuk atribut entitas lain. Jika entitas mahasiswa mengambil Mata_Kuliah, *primary key*nya terletak pada relasi mengambil, yaitu nim dan kd_mk. Jika entitas mahasiswa membimbing dosen, *foreign key* negerinya terletak pada nim. Mahasiswa juga dapat melihat KHS, atau kumpulan nilai, selama belajar di SIAKAD. Bukan entitas yang memiliki atribut nilai, tetapi hubungan Mengambil. Jika fitur ini terletak pada entitas mahasiswa, maka setiap mata kuliah yang diambil oleh satu mahasiswa akan memiliki nilai yang sama. Sebaliknya, jika aribut nilai terletak pada entitas mata kuliah, maka setiap mata kuliah yang diambil oleh mahasiswa tersebut akan memiliki nilai yang sama. Keduanya tidak dapat dilaksanakan. Maka dengan adanya atribut Nilai pada relasi Mengambil, maka seorang mahasiswa tertentu yang mengambil mata kuliah tertentu akan mendapatkan nilai yang tertentu pula. Sebenarnya relasi tidaklah memiliki atribut. Namun, jika diantara dua entitas relasinya Many to Many, maka akan membentuk entitas baru. Nantinya di dalam entitas baru tersebut terdapat primary key dari entitas Mahasiswa dan Mata_Kuliah untuk dijadikan *foreign key* pada entitas baru. Entitas baru tersebut bernama Mhs_ambil_MK dengan atribut password, nim, kd_mk, dan nilai. Dari semua atributnya, password akademiklah yang dijadikan sebagai *foreign key*nya. Karena satu mahasiswa memiliki password yang berbeda dengan mahasiswa lain. Maka hasil Entity Relationship Diagram akan seperti berikut. Mahasiswa mengambil Mhs_ambil_MK diambil Mata_Kuliah, kemudian disambungkan dengan Ruang dengan relasi digunakan untuk. Sambungkan entitas Dosen membimbing Mahasiswa dan Dosen

mengajar Mata_Kuliah. Yang berbeda dari bentuk ERD sebelumnya hanya terletak pada "Mhs_ambil_MK" saja agar nilai mata kuliah mahasiswa menjadi realistis[11].

2.9 *Unifled Modeling Language* (UML)

Bahasa pemodelan terintegrasi (UML) adalah singkatan dari *Unifled Modeling Language* , yang berarti bahasa pemodelan standar. UML adalah sekumpulan teknik diagram standar yang memungkinkan implementasi dan analisis setiap pengembangan sistem proyek untuk memberikan representasi grafis yang cukup kaya untuk model. Sebagian besar sistem saat ini berorientasi objek analisis, dan desain pendekatan menggunakan UML untuk menggambarkan sistem yang sedang berkembang. UML menampilkan berbagai perspektif dari sistem berkembang dalam satu set diagram. Diagram ini termasuk dalam dua kategori umum [14]:

a. Struktur (*structure*)

   Diagram struktur terdiri dari *class*, *object*, *package*, *deployment*, *component*, dan *composite structure* diagram.

b. Perilaku (*behavior*)

   Diagram perilaku terdiri dari *activity*, *sequence*, *communication*, *interaction overview*, *timing*, *state machine behavior*, *protocol state machine*, dan *Use Case* diagram.

2.10 *Use Case* Diagram

Use Case diagram menunjukkan hubungan antara *Actors* dan *Use Cases*. Digunakan untuk analisis dan desain sistem. Berikut ini adalah bagian dari sebuah *use case* diagram [14]:

a. *Use Cases*

   Use Cases menjelaskan tentang Tindakan/aksi yang dilakukan oleh *actors*. *Use Case* digambarkan dalam bentuk *elips* yang horizontal.

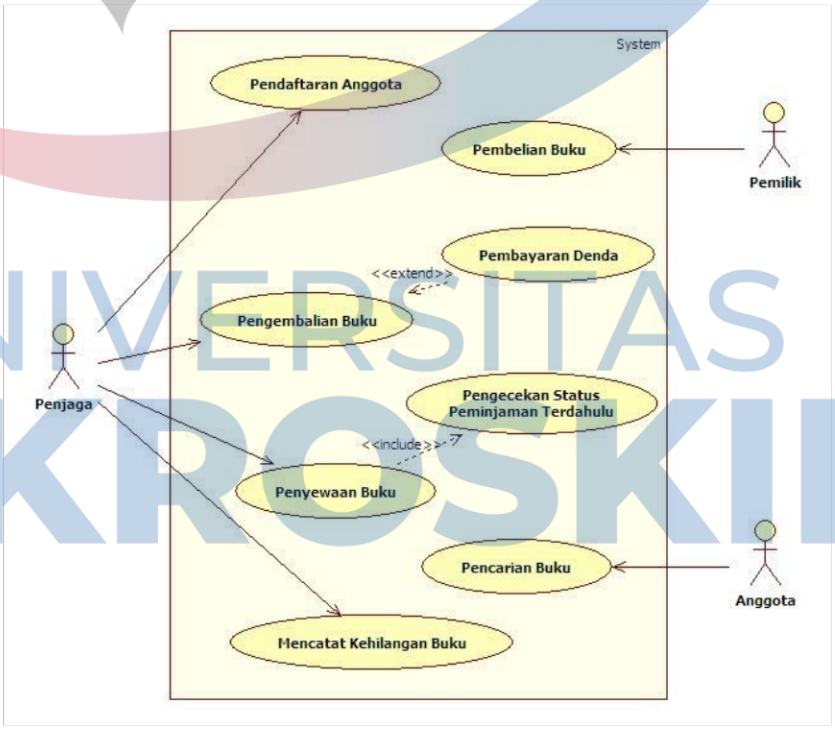
b. *Actors*

   Actors adalah seorang peran yang berinteraksi dengan sistem. *Actors* meliputi baik manusia maupun organisasi yang saling bertukar informasi.

c. *Relationship*

   Relationship adalah hubungan antara *use cases* dengan *actors*. *Relationship* dalam *use case* diagram meliputi [14]:

1. Asosiasi antara *actor* dan *use case*. Hubungan antara *actor* dan *use case* yang terjadi karena adanya interaksi antara kedua belah pihak. Asosiasi tipe ini menggunakan garis lurus dari *actor* menuju *use case* baik dengan menggunakan mata panah terbuka ataupun tidak.
2. Asosiasi antara 2 *use case*.Hubungan antara *use* case yang satu dan *use case* lainnya yang terjadi karena adanya interaksi antara kedua belah pihak. Asosiasi tipe ini menggunakan garis putus-putus/garis lurus dengan mata panah terbuka di ujungnya.
3. Generalisasi antara 2 *actor*. Hubungan *inheritance*(pewarisan) yang melibatkan *actor* yang satu (*the child*) dengan *actor* lainnya (*the parent*). Generalisasi tipe ini menggunakan garis lurus dengan mata panah tertutup diujungnya.
4. Generalisasi antara 2 *use case*. Hubungan *inheritance*(pewarisan) yang melibatkan *use case* satu (*the child*) dengan *use case* lainnya(*the parent*). Generalisasi tipe ini menggunakan garis lurus dengan mata panah tertutup diujungnya.



Gambar 2.15 *Use Case* Diagram Perpustakaan