BAB 2
TINJAUAN PUSTAKA

Tinjauan pustaka merupakan peninjauan pustaka-pustaka yang berkaitan dengan pembangunan WhatsApp *Mesengger chatbot* untuk mengidentifikasi pesan penipuan dengan memanfaatkan *large language model*. Tinjauan pustaka ini akan menjadi dasar dari penelitian yang akan dilakukan.

2.1 Penipuan

Penipuan adalah tindakan atau praktik yang melibatkan penggunaan tipu daya atau manipulasi untuk memperoleh keuntungan atau merugikan pihak lain. Ini melibatkan upaya untuk memperoleh sesuatu dengan cara yang tidak jujur, seringkali dengan maksud mengelabui atau menipu orang lain. Penipuan dapat dilakukan dalam berbagai konteks, termasuk keuangan, bisnis, perdagangan, atau interaksi pribadi [21].

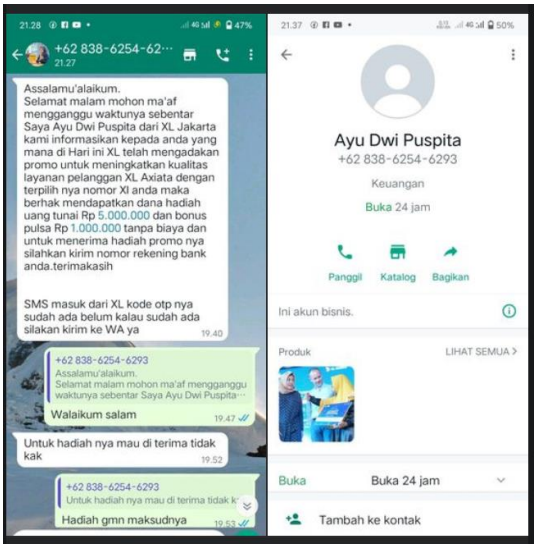
Adapun beberapa pemicu terjadinya tindak penipuan, diantaranya:

1. Ketidakwaspadan atau kurangnya pengetahuan
2. Tekanan keuangan
3. Kepercayaan berlebihan
4. Teknologi keamanan yang lemah
5. Ketidakmampuan untuk mengelola informasi pribadi

2.1.2 Pesan Penipuan

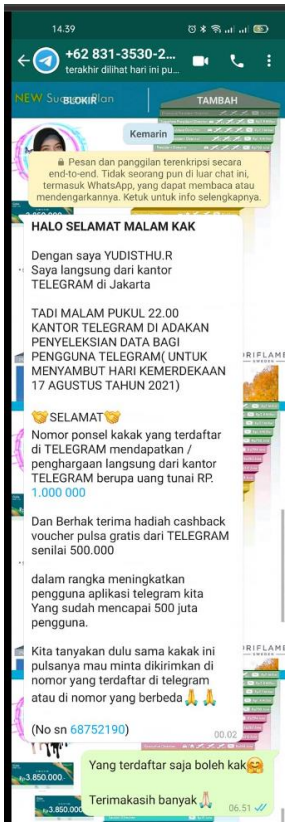
Pesan penipuan merupakan pesan yang dirancang untuk menipu atau memanipulasi penerima dengan tujuan tertentu, seperti mencuri informasi pribadi atau keuangan, menginstal malware, atau menipu agar melakukan tindakan tertentu yang merugikan. Pesan penipuan sering kali terlihat seperti pesan yang sah atau dari sumber tepercaya, namun sebenarnya bertujuan untuk merugikan penerima. Contoh umum dari pesan penipuan adalah modus undian berhadiah, modus tawaran pinjaman online, modus tawaran judi online, modus *auto transfer* pulsa, modus permintaan kode OTP (*One Time Password*). Berikut merupakan contoh pesan

penipuan yang peneliti temukan dari postingan Facebook masyarakat yang pernah menjadi koban penipuan.



Gambar 2.1 Contoh Pesan Penipuan 1

Sumber: https://www.facebook.com/share/p/aWJz2YQXakfoF4TX/



Gambar 2.2 Contoh Pesan Penipuan 2

Sumber: https://www.facebook.com/share/p/NiaivcBa7VyBAiiM/

2.2 *Chatbot*

Chatterbot (disebut juga *chatbot*) adalah sebuah program komputer yang dibuat dengan tujuan untuk mensimulasikan percakapan atau komunikasi interaktif dengan pengguna manusia [22]. *Chatbot* dapat beroperasi melalui berbagai saluran komunikasi, seperti teks, suara, atau elemen visual . Tujuan utama dari chatbot adalah memberikan respons dan menjawab pertanyaan pengguna atau menjalankan tugas tertentu berdasarkan input yang diterimanya. *Chatbot* umumnya digunakan untuk memberikan layanan pelanggan otomatis, bantuan informasi, atau interaksi pengguna pada berbagai platform, termasuk situs web, aplikasi pesan, dan lainnya.

Pada awalnya, program komputer (*chatbot*) ini diuji melalui *turing test*, yaitu dengan merahasiakan identitasnya sebagai mesin agar dapat mengecoh orang yang berbicara dengannya. Jika pengguna tidak dapat mengidentifikasi *chatbot* sebagai program komputer, maka *chatbot* tersebut dikategorikan sebagai kecerdasan buatan (*artificial intelligence*). *Chatbot* digunakan untuk berbagai tujuan praktis seperti bantuan online, layanan personal, atau akuisisi informasi, dan dalam konteks ini, berfungsi sebagai jenis agen percakapan (*conversational agent*). Perbedaan antara *chatbot* dan sistem pemrosesan bahasa alami (*Natural Language Processing System*) terletak pada kesederhanaan algoritma yang digunakan. Meskipun banyak chatbot tampaknya dapat memahami dan merespons input manusia, sebenarnya, chatbot tersebut hanya melakukan pemindaian kata kunci dalam input dan meresponsnya dengan kata kunci yang paling sesuai, atau pola kata-kata yang paling mirip dari basis data tekstual.

2.3 WhatsApp Platform

WhatsApp Platform adalah sebuah platform komunikasi pesan singkat seperti SMS yang terdapat pada *smartphone*, selain dari mengirimkan pesan teks. Aplikasi ini juga dapat mengirim pesan suara, file, gambar, dan video [23]. Whatsapp juga merupakan aplikasi pesan instan lintas platform gratis (*freeware*) yang memanfaatkan teknologi VoIP (*Voice over* IP). Dengan teknologi tersebut,

pengguna dapat saling mengirim pesan teks dan suara, melakukan panggilan suara dan video, berbagi video, gambar/foto, lokasi, dokumen, dan jenis format media lainnya.

Aplikasi Whatsapp dapat digunakan pada berbagai perangankat seperti Android, iOS, BlackBerry OS, Windows Phone, dan lainnya. Whatsapp juga dapat digunakan pada perangkat dekstop melalui *browser web* atau menggunakan WhatsApp Desktop. Salah satu aspek keamanan utama WhatsApp adalah penggunaan enkripsi *end-to-end*, yang melindungi privasi pengguna dengan memastikan bahwa hanya pengirim dan penerima yang dapat membaca isi pesan tersebut.

WhatsApp juga mendukung fitur grup, memungkinkan pengguna untuk berkomunikasi dalam konteks kelompok dengan beberapa orang sekaligus. Grup dapat memiliki admin yang bertanggung jawab atas pengaturan dan keanggotaan. Fitur tambahan termasuk pengaturan privasi seperti status terlihat terakhir, notifikasi, dan kemampuan untuk memblokir kontak. WhatsApp juga memiliki versi khusus untuk bisnis, WhatsApp *Business*, yang menyediakan fitur tambahan seperti profil bisnis, respons otomatis, dan statistik pesan.

WhatsApp juga menyediakan fasilitas WhatsApp *Web* dimana penggunanya dapat menautkan akun yang sama dengan perangkat yang berbeda. Pesan yang dikirm kan dan diterima akan disinkronkan antar perangkat, dan pengguna dapat melihatnya di kedua perangkat.

2.4 API (Application Programing Interface)

Application Programing Interface (API) merupakan perangkat lunak yang menghubungkan dua atau lebih dari aplikasi yang berbeda agar dapat saling terhubung serta berkomunikasi satu sama lain [24]. API juga merupakan antarmuka virtual antara dua fungsi perangkat lunak yang berinteraksi, seperti sebuah *word processor* dengan *spreadsheet*. Dengan menggunakan API, pengembang dapat menggunakan fungsi aplikasi lain yang sudah ada, sehingga tidak perlu membuat ulang semuanya dari awal [25].

API tersedia untuk sistem jaringan, sistem *file*, sistem *windowing*, dan sistem basis data. Perkembangan teknologi API dimulai dari pembuatan subrutin

sederhana yang memberikan kemampuan interoperabilitas dan modifikasi sistem untuk mendukung pertukaran data antara beberapa aplikasi. Subrutin tersebut awalnya hanya dapat melakukan perhitungan matematis sederhana, tetapi seiring waktu muncul sebuah API pustaka perhitungan yang hampir selalu tersedia dalam setiap bahasa pemrograman. Dari subrutin sederhana ini, timbul berbagai gagasan tentang bagaimana API harus dikembangkan, terutama sejalan dengan evolusi paradigma pemrograman berorientasi objek. Hal ini mengakibatkan pengelompokan sejumlah subrutin serupa menjadi sebuah kelas pembungkus yang menyatukan fungsi-fungsi tersebut.

Pada pengembangan sistem yang lebih luas, seringkali sebuah API dapat berfungsi sebagai *middleware*, yaitu sebuah subsistem mandiri yang menyediakan fungsi-fungsi yang bermanfaat bagi subsistem lain. Untuk mengakses fungsi-fungsi tersebut, diperlukan koneksi tersendiri terhadap *middleware* tersebut. Umunya, koneksi ke *middleware* dilakukan menggunakan *socket*.

Karena infromasi yang digunakan diambil dari hasil *generate* dari model AI pada OpenAI, maka sebuah API diperlukan untuk memfasilitasi pertukaran data atau informasi pada aplikasi yang dihasilkan.

2.5 OpenAI

OpenAI merupakan sebuah perusahaan riset kecerdasan buatan (AI) yang didirikan dengan tujuan untuk mengembangkan dan memajukan teknologi kecerdasan buatan dengan pendekatan yang aman dan bermanfaat bagi masyarakat [26]. OpenAI didirikan pada tahun 2015 dengan tujuan utama untuk memastikan bahwa kecerdasan buatan dikembangkan dengan cara yang bermanfaat dan aman bagi semua orang. OpenAI dikelas karena mengembangkan model-model kecerdasan buatan canggih. Salah satu model yang terkenal adalah GPT (*Generative Pre-trained Transformer*).

OpenAI terlibat dalam berbagai proyek dan inisiatif untuk memajukan kecerdasan buatan. Mereka juga merilis sejumlah besar penelitian dan model kedalam domain publik. Pada Agustus 2023 OpenAI meluncurkan *Fine-tuning* untuk GPT-3.5 Turbo, *fine-tuning* memungkinkan bagi para pengembang untuk menyesuaikan model sesuai dengan kebutuhan khusus mereka, dan

menjalankannya dalam skala besar. Hasil uji awal menunjukkan bahwa versi GPT-3.5 Turbo yang telah di *fine-tuning* dapat menyamai, atau bahkan mengungguli, kemampuan dasar GPT-4 pada tugas-tugas tertentu yang lebih spesifik.

OpenAI memberikan fasilitas bagi para pengembang berupa API untuk menggunakan hasil dari penelitiannya antaralain:

1. *Text Generation*
2. *Embeddings*
3. *Image Generation*
4. *Text to speech*
5. *Prompt engineering*
6. *Speech to text*
7. *Fine-tuning*

2.5.2 Fine-tuning

Fine-tuning adalah proses penyesuaian atau pelatihan ulang pada model kecerdasan buatan yang sudah dilatih sebelumnya, biasanya menggunakan *dataset* yang lebih kecil dan spesifik untuk tugas tertentu [27]. Tujuan *fine-tuning* adalah meningkatkan performa model pada tugas atau domain khusus, dengan memanfaatkan pengetahuan yang telah dimilikinya dari pelatihan awal.

Secara umum, langkah-langkah *fine-tuning* melibatkan:

1. *Model Pre-trained*

   Mulai dengan model kecerdasan buatan yang usdah dilatih sebelumya dengan *dataset* besar dan beragam. Model *pre-tained* ini biasanya telah memahami struktur bahasa atau fitur lainnya secara umum.

2. *Dataset* khusus

   Sediakan *dataset* yang sesuai dengan tugas atau domain spesifik yang ingin diatasi. *Dataset* ini lebih kecil dan berfokus untuk menyesuaikan representasinya dengan data spesifik yang diberikan.

3. Pelatihan Ulang

   Gunakan *dataset* khusus untuk melatih ulang model. Proses pelatihan ulang ini memungkinkan model untuk menyesuaikan representasinya dengan data spesifik yang diberikan.

4. Pengaturan Hiperparameter

   Sesuaikan hiperparameter model seperti tingkat pembelajaran, jumlah iterasi, dan parameter lainya agar model dapat beradaptasi secara optimal dengan *dataset* khusus.

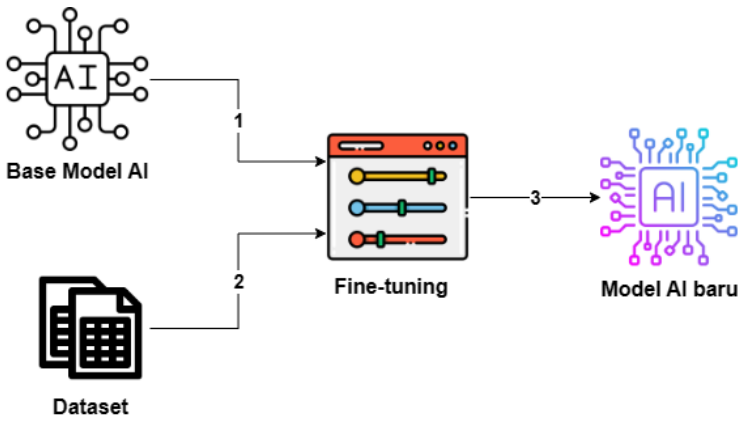
5. Evaluasi dan Penyesuaian

   Evaluasi performa model pada *dataset* validasi atrau uji untuk memastikan bahwa *fine-tuning* memberikan hasil yang diinginkan. Jika diperlukan, lakukan penyesuaian iterasi lebih lanjut.

Fine-tuning memberikan fleksibilitas kepada pengembang untuk mengoptimalkan model yang sudah ada agar lebih sesuai dengan kebutuhan tugas atau domain tertentu. Proses ini sangat berguna ketika ingin menggunakan model kecerdasan buatan yang sudah ada untuk tugas spesifik yang tidak tercakup dalam pelatihan awal model tersebut. Fine-tuning pada penelitian ini berfungsi sebagai penyesuaian model AI GPT 3.5 dari open AI agar dapat mengidentifikasi sebuah pesan penipuan atau bukan.

2.5.2.2 Proses Penggunaan *Fine-Tuning* OpenAI

Proses penggunaan *Fine-tuning* Open AI dapat dilihat pada gambar 2.3.



Gambar 2.3 Alur Penggunaan *Fine-Tuning* OpenAI

Langkah pertama sebelum melakukan proses *fine-tuning* yakni penyiapan *dataset. Dataset* pada penelitian ini memakai data sebanyak 61 data, terdiri dari 34 data pesan penipuan dan 27 data pesan promosi. Data penipuan didapatkan dari pencarian Google, masyarakat yang mendapatkan pesan penipuan, dan dari peneliti pribadi. Data pesan promosi didapatkan dari pesan iklan dari akun terverifikasi.

Dataset diperlukan untuk proses penyesuaian model AI agar model AI yang dilatih dapat melakukan tugasnya lebih spesifik sesuai *dataset* yang diberikan. *Dataset* yang diperlukan untuk proses *fine-tuning* adalah file berformat jsonl, dengan format isi data seperti berikut. Format *dataset* JSONL dapat dilihat pada gambar 2.4, dan contoh dataset yang dibuat dapat dilihat pada gambar 2.5.

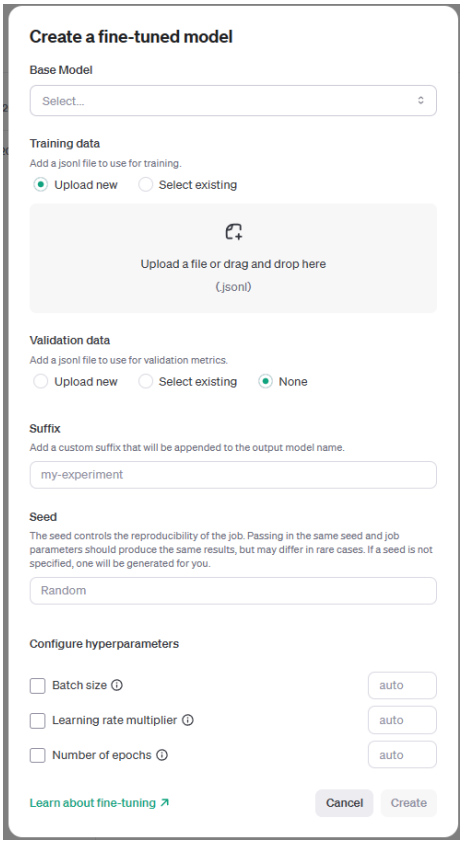
```
{
    "messages":[
        {
            "role":"system",
            "content":"prompt"},
        {
            "role":"user",
            "content":"inputan user"},
        {
            "role":"assistant",
            "content":"outputan assistant"}
    ]
}
```

Gambar 2.4 Format *Dataset* JSONL

```
{
    "messages":[
        {
            "role":"system",
            "content":"Sebuah chatbot yang dapat mengidentifikasi dan mengklasifikasi sebuah pesan atau chat penipuan."},
        {
            "role":"user",
            "content":  "Nomor Anda di Daftarkan Auto TP Oleh 082353276868 Anda Akan Mengirimkan Pulsa rp.10.00
                        Setiap tgl 1 U/b erhenti ketik: TP 10 082353276868 Kirim Ke 858 Gratis "},
        {
            "role":"assistant",
            "content":  "Pesan ini  teridentifikasi penipuan, pesan seperti ini merupakan pesan penipuan agar anda mentransfer pulsa anda
                        ke nomor yang dituju, saran : abaikan saja pesan seperti ini, dan jangan ikuti perintah dari pesan tersebut"}
    ]
}
```

Gambar 2.5 Contoh *Dataset*

Setelah penyiapan *dataset* selesai, selanjutnya pemilihan basis model AI yang akan dilatih, basis model yang dapat digunakan dari Open AI antara lain babbage-002, davinci-002, GPT 3.5 Turbo, dan model yang sudah dilakukan penyesuaian atau pelatihan sebelumnya. Selanjutnya setelah pemilihan basis model yakni mengunggah file *dataset* yang sudah disiapkan, kemudian lakukan proses *fine-tuning*. Tampilan konfigurasi *fine-tuning* dapat dilihat pada gambar 2.6.



Gambar 2.6 Tampilan Konfigurasi *Fine-Tuning*

Pada penelitian ini model AI yang dipilih adalah model GPT 3.5 Turbo, model ini dipilih karena model ini merupakan model yang paling canggih di antara kedua model lainnya. GPT-3.5 Turbo memiliki kemampuan pemahaman dan generasi teks yang sangat baik, mampu menangani berbagai tugas dengan akurasi dan kefasihan yang tinggi. GPT-3.5 Turbo juga dapat digunakan untuk berbagai macam aplikasi, mulai dari *chatbot* yang sangat responsif dan interaktif, pembuatan konten kreatif, analisis data tingkat lanjut, hingga aplikasi yang memerlukan pemahaman bahasa yang sangat mendalam

2.5.3 Text Generation

OpenAI *text generation model* atau sering disebut *Generative Pre-trained Transformers* (GPT) atau *Large Language Models* (LLM) yang dilatih untuk memahami bahasa alami, kode, dan gambar. Model-model ini, seperti GPT-4, mampu menghasilkan teks sebagai respons terhadap masukan yang diberikan, yang disebut juga sebagai "*prompt*". Merancang *prompt* adalah cara penggunanya

"memprogram" model ini, biasanya dengan memberikan instruksi atau beberapa contoh tentang bagaimana menyelesaikan tugas.

Menggunakan model generasi teks OpenAI, penggunanya bisa membangun aplikasi untuk berbagai tujuan, seperti:

1. Menyusun Dokumen: Membantu dalam penulisan esai, artikel, laporan, dan berbagai jenis dokumen lainnya.
2. Menulis Kode Komputer: Menghasilkan kode untuk berbagai bahasa pemrograman, debug kode, atau menyediakan solusi pemrograman.
3. Menjawab Pertanyaan: Menyediakan jawaban atas pertanyaan berdasarkan basis pengetahuan tertentu.
4. Menganalisis Teks: Melakukan analisis teks untuk menemukan pola, sentimen, atau informasi penting.
5. Memberikan Antarmuka Bahasa Alami pada Perangkat Lunak: Memungkinkan pengguna untuk berinteraksi dengan perangkat lunak menggunakan bahasa alami.
6. Tutor dalam Berbagai Subjek: Memberikan bimbingan atau penjelasan dalam berbagai bidang ilmu pengetahuan.
7. Menerjemahkan Bahasa: Mengonversi teks dari satu bahasa ke bahasa lain.
8. Mensimulasikan Karakter dalam Permainan: Menciptakan dialog atau perilaku karakter dalam permainan.

Untuk menggunakan model ini melalui API OpenAI, diharuskan mengirim permintaan ke *Chat Completions* API dengan masukan dan API *Key*, dan menerima respons yang berisi *output* dari model. *Chat models* mengambil daftar pesan sebagai *input*. Setiap pesan dalam daftar ini biasanya memiliki struktur berikut:

a. *Role*: Menunjukkan peran pengirim pesan. Peran ini bisa berupa "*system*" (untuk memberikan instruksi atau konteks awal), "*user*" (untuk *input* dari pengguna), atau "*assistant*" (untuk *output* dari model).
b. *Content*: Berisi teks pesan itu sendiri.

Contoh Struktur pesan

```
[
  {"role": "system", "content": "You are a helpful assistant."},
  {"role": "user", "content": "What is the capital of France?"}
```

```
4. ]
```

Model akan memproses daftar pesan yang diberikan sebagai *input* dan menghasilkan pesan sebagai *output*. Pesan yang dihasilkan ini biasanya memiliki peran *assistant*.

Contoh *output* yang dihasilkan:

```
1. {"role": "assistant", "content": "The capital of France is Paris."}
```

2.6 *Dataset*

Dataset adalah kumpulan data yang terorganisir, biasanya dalam bentuk tabel atau *file*, yang digunakan untuk analisis atau penelitian dalam berbagai bidang seperti ilmu komputer, statistik, dan ilmu sosial. *Dataset* bisa berisi berbagai jenis informasi, mulai dari teks, gambar, audio, hingga data numerik, dan sering kali digunakan untuk melatih algoritma kecerdasan buatan atau untuk mengambil wawasan yang berguna melalui analisis data.

Pada penelitian ini dataset merujuk kepada kumpulan data yang digunakan untuk melatih model kecerdasan buatan. *Dataset* ini berisi sekumpulan informasi atau contoh yang telah dikumpulkan untuk dilakukan pelatihan model AI untuk mengidentifikasi pesan penipuan. Pembuatan *dataset* pada penelitian ini menggunakan total 60 data, dengan 30 data berisi pesan yang terindikasi penipuan yang berasal korban yang pernah mengalami, riset peneliti pribadi, dan dari hasil kuesioner. 30 data lainnya berisi iklan promosi yang berasal dari akun-akun terverifikasi. Format *file dataset* yang dibutuhkan adalah JSONL dengan isi format sebagai berikut:

```
1. {"messages": [
2.         {
3.             "role": "system",
4.             "content": "Konteks Awal"},
5.         {
6.             "role": "user",
7.             "content": "input user"},
8.         {
9.             "role": "assistant",
10.            "content": "output assistant"},
11.        }
12.    ]
13. }
```

Contoh *dataset* yang dibuat adalah sebagai berikut:

```
1. {
2.     "messages":[
3.         {
4.             "role":"system",
```

```
                    "content":"Sebuah chatbot yang dapat mengidentifikasi dan mengklasifikasi sebuah pesan atau chat penipuan."
        },
        {
            "role":"user",
             "content":"PEMBERITAHUAN RESMI Anda Terdaftar Sebagai Penerima DANA BANTUAN Rp.50jt dari BPJS KESEHATAN PUSAT untuk Prosedur/Cara KLAIM DANA BANTUANNYA Silahkan KETIK:(KLAIM BANTUAN BPJS) KIRIM Via WA ke No +6011-72206138"
        },
        {
            "role":"assistant",
           "content":"Pesan ini teridentifikasi penipuan, alasannya Pesan tersebut menggunakan tata bahasa yang kurang resmi, yang tidak sesuai dengan umumnya komunikasi resmi dari instansi pemerintah atau lembaga kesehatan. Nomor telepon yang diberikan tidak sesuai dengan format nomor telepon Indonesia. Nomor internasional seperti +6011 tidak konsisten dengan nomor telepon Indonesia.Instansi resmi seperti BPJS Kesehatan biasanya tidak menggunakan pesan teks atau WhatsApp untuk menginformasikan atau mengelola klaim dana bantuan sebesar itu. Mereka lebih cenderung menggunakan saluran komunikasi resmi seperti surat resmi atau panggilan telepon langsung.Jika Anda tidak memiliki riwayat atau hubungan dengan BPJS Kesehatan dan mendapatkan pesan seperti ini tanpa permintaan sebelumnya, itu bisa menjadi indikasi penipuan. Saran : harap untuk tidak memberikan informasi apapaun terkait data diri anda, serta untuk memastikan kebenaran pesan seperti ini ebaiknya hubungi langsung BPJS Kesehatan melalui saluran komunikasi resmi mereka dan jangan pernah memberikan informasi pribadi atau melakukan tindakan tertentu tanpa verifikasi yang jelas."
        }
    ]
}

```

2.7 OpenAI API

API *Key* dari OpenAI adalah sebuah kunci otentikasi yang digunakan untuk mengakses layanan API dari OpenAI. Pengguna dapat berinteraksi denga API melalui *request* HTTP dengan bahasa pemrograman apapun, melalui Phyton *bindings* resmi dari OpenAI, *Library* Node.js resmi dari Open AI, atau library yang dikelola oleh komunitas. Untuk mendapatkan API *key* dari OpenAI dilakukan dengan cara berikut:

1. Daftar akun pada OpenAI pada situs berikut: https://openai.com/
2. Setelah mendaftar dan masuk, buka bagian API dan buat API Key.

Karena pada penelitian ini sistem yang dibangun menggunkan Node.js, maka untuk menggunakan API key ini perlu memasang *Libray* Node.js dari OpenAI pada direktori projek dengan perintah berikut:

```
npm install openai@^4.0.0
```

Setelah *library* terpasang, *library* OpenAI dan API *key* bisa digunakan dengan contoh perintah sebagai berikut:

```
import OpenAI from "openai";

const openai = new OpenAI({
    apiKey: process.env.OPENAI_API_KEY,
```

```
});

const chatCompletion = await openai.chat.completions.create({
    messages: [{ role: "user", content: "Say this is a test" }],
    model: "gpt-3.5-turbo",
});
```

Respon dari API OpenAI berupa objek JSON yang berisi hasil dari permintaan, berikut contoh respon dari perintah sebelumnya:

```
{
  "id": "chatcmpl-123",
  "object": "chat.completion",
  "created": 1677652288,
  "model": "gpt-3.5-turbo-0125",
  "system_fingerprint": "fp_44709d6fcb",
  "choices": [{
    "index": 0,
    "message": {
      "role": "assistant",
      "content": "\n\nThis is a tesyt",
    },
    "logprobs": null,
    "finish_reason": "stop"
  }],
  "usage": {
    "prompt_tokens": 9,
    "completion_tokens": 12,
    "total_tokens": 21
  }
}
```

Untuk penggunaan *Chat Completions* API dapat menggunakan contoh perintah berikut:

```
import OpenAI from "openai";

const openai = new OpenAI();

async function main() {
  const completion = await openai.chat.completions.create({
        messages: [{"role": "system", "content": "Sebuah Asisten yang sangat membantu."},
        {"role": "user", "content": "halo, siapakah kah anda?"}],
    model: "gpt-3.5-turbo",
  });

  console.log(completion.choices[0]);
}
main();
```

Respon *Chat Completions* API dari perintah diatas berupa JSON seperti berikut:

```
{
  "choices": [
    {
      "finish_reason": "stop",
      "index": 0,
      "message": {
        "content": "Halo, saya adalah assten yang siap membantu anda.",
        "role": "assistant"
      },
      "logprobs": null
```

```
    }
  ],
  "created": 1677664795,
  "id": "chatcmpl-7QyqpwdfhqwajicIEznoc6Q47XAyW",
  "model": "gpt-3.5-turbo-0613",
  "object": "chat.completion",
  "usage": {
    "completion_tokens": 17,
    "prompt_tokens": 57,
    "total_tokens": 74
  }
}
```

2.8 Baileys

Baileys adalah sebuah *library* yang dibuat dalam bahasa TypeScript/Javascript yang berfungsi sebagai API untuk WhatsApp Web. *Library* ini diciptakan dengan tujuan untuk memungkinkan pengguna untuk berinteraksi dengan WhatsApp Web secara langsung melalui *WebSocket*, tanpa memerlukan Selenium atau *browser* lainnya. Dengan menggunakan Baileys, pengguna dapat mengakses dan mengontrol versi WhatsApp Web dan WhatsApp *multi-device*.

2.8.1 Koneksi Baileys

Agar sistem yang dibangun dapat mengakses dan mengotrol WhatsApp Web sebagai *chatbot* Whatsapp melalui Baileys, maka Baileys perlu diautentikasikan sebagai klien Whatsapp Web dengan memindai QR. Berikut merupakan contoh program agar sistem yang dibangun dapat mengakses dan mengotrol WhatsApp Web sebagai *chatbot* Whatsapp melalui Baileys.

```
import makeWASocket, { DisconnectReason } from '@whiskeysockets/baileys'
import { Boom } from '@hapi/boom'

async function connectToWhatsApp () {
    const sock = makeWASocket({
        // can provide additional config here
        printQRInTerminal: true
    })
    sock.ev.on('connection.update', (update) => {
        const { connection, lastDisconnect } = update
        if(connection === 'close') {
                            const shouldReconnect = (lastDisconnect.error as Boom)?.output?.statusCode !== DisconnectReason.loggedOut
                console.log('connection closed due to ', lastDisconnect.error, ', reconnecting ', shouldReconnect)
            // reconnect if not logged out
            if(shouldReconnect) {
                connectToWhatsApp()
            }
        } else if(connection === 'open') {
            console.log('opened connection')
        }
    })
```

```
connectToWhatsApp()
```

2.8.1.1 Konfigurasi Koneksi

Konfigurasi yang diperlukan untuk menghubungkan ke WhatsApp Web menggunakan Baileys API dapat dengan menggunakan objek dari `*SocketConfig*` berikut merupakan seluruh struktur dari `*SocketConfig*`.

```
type SocketConfig = {
    /** the WS url to connect to WA */
    waWebSocketUrl: string | URL
    /** Fails the connection if the socket times out in this interval */
      connectTimeoutMs: number
    /** Default timeout for queries, undefined for no timeout */
    defaultQueryTimeoutMs: number | undefined
    /** ping-pong interval for WS connection */
    keepAliveIntervalMs: number
    /** proxy agent */
      agent?: Agent
    /** pino logger */
      logger: Logger
    /** version to connect with */
    version: WAVersion
    /** override browser config */
      browser: WABrowserDescription
      /** agent used for fetch requests -- uploading/downloading media */
      fetchAgent?: Agent
    /** should the QR be printed in the terminal */
    printQRInTerminal: boolean
    /** should events be emitted for actions done by this socket connection */
    emitOwnEvents: boolean
    /** provide a cache to store media, so does not have to be re-uploaded */
    mediaCache?: NodeCache
    /** custom upload hosts to upload media to */
    customUploadHosts: MediaConnInfo['hosts']
    /** time to wait between sending new retry requests */
    retryRequestDelayMs: number
    /** max msg retry count */
    maxMsgRetryCount: number
    /** time to wait for the generation of the next QR in ms */
    qrTimeout?: number;
    /** provide an auth state object to maintain the auth state */
    auth: AuthenticationState
      /** manage history processing with this control; by default will sync up
everything */
      shouldSyncHistoryMessage: (msg: proto.Message.IHistorySyncNotification) =>
boolean
    /** transaction capability options for SignalKeyStore */
    transactionOpts: TransactionCapabilityOptions
    /** provide a cache to store a user's device list */
    userDevicesCache?: NodeCache
    /** marks the client as online whenever the socket successfully connects */
    markOnlineOnConnect: boolean
    /**
     * map to store the retry counts for failed messages;
     * used to determine whether to retry a message or not */
    msgRetryCounterMap?: MessageRetryMap
    /** width for link preview images */
    linkPreviewImageThumbnailWidth: number
    /** Should Baileys ask the phone for full history, will be received async */
    syncFullHistory: boolean
    /** Should baileys fire init queries automatically, default true */
```

```
53.     fireInitQueries: boolean
54.     /**
55.      * generate a high quality link preview,
56.      * entails uploading the jpegThumbnail to WA
57.      * */
58.     generateHighQualityLinkPreview: boolean
59.
60.     /** options for axios */
61.     options: AxiosRequestConfig<any>
62.     /**
63.      * fetch a message from your store
64.       * implement this so that messages failed to send (solves the "this message
can take a while" issue) can be retried
65.      * */
66.     getMessage: (key: proto.IMessageKey) => Promise<proto.IMessage | undefined>
67. }
```

2.8.2 Menyimpan dan Memulihkan Sesi

Agar pengguna tidak perlu terus-menerus memindai kode QR setiap kali ingin terhubung dengan baileys maka diperlukan sebuah sesi. Berikut merupakan kredensial untuk menyimpan dan memulihkan sesi.

```
 1.    import    makeWASocket,    {    BufferJSON,    useMultiFileAuthState    }    from
'@whiskeysockets/baileys'
 2. import * as fs from 'fs'
 3.
 4. // utility function to help save the auth state in a single folder
 5. // this function serves as a good guide to help write auth & key states for SQL/no-
SQL databases, which I would recommend in any production grade system
 6. const { state, saveCreds } = await useMultiFileAuthState('auth_info_baileys')
 7. // will use the given state to connect
 8. // so if valid credentials are available -- it'll connect without QR
 9. const conn = makeWASocket({ auth: state })
10. // this will be called as soon as the credentials are updated
11. conn.ev.on ('creds.update', saveCreds)
```

2.8.3 Handling Event

Berikut merupakan semua *event* yang terdapat dalam baileys.

```
 1. export type BaileysEventMap = {
 2.      /** connection state has been updated -- WS closed, opened, connecting etc.
*/
 3.       'connection.update': Partial<ConnectionState>
 4.     /** credentials updated -- some metadata, keys or something */
 5.     'creds.update': Partial<AuthenticationCreds>
 6.     /** history sync, everything is reverse chronologically sorted */
 7.     'messaging-history.set': {
 8.         chats: Chat[]
 9.         contacts: Contact[]
10.         messages: WAMessage[]
11.         isLatest: boolean
12.     }
13.     /** upsert chats */
14.     'chats.upsert': Chat[]
15.     /** update the given chats */
16.     'chats.update': Partial<Chat>[]
17.     /** delete chats with given ID */
18.     'chats.delete': string[]
19.     'labels.association': LabelAssociation
20.     'labels.edit': Label
```

```
21.     /** presence of contact in a chat updated */
22.         'presence.update': { id: string, presences: { [participant: string]:
PresenceData } }
23.
24.     'contacts.upsert': Contact[]
25.     'contacts.update': Partial<Contact>[]
26.
27.     'messages.delete': { keys: WAMessageKey[] } | { jid: string, all: true }
28.     'messages.update': WAMessageUpdate[]
29.    'messages.media-update': { key: WAMessageKey, media?: { ciphertext: Uint8Array,
iv: Uint8Array }, error?: Boom }[]
30.     /**
31.      * add/update the given messages. If they were received while the connection
was online,
32.      * the update will have type: "notify"
33.      *  */
34.     'messages.upsert': { messages: WAMessage[], type: MessageUpsertType }
35.     /** message was reacted to. If reaction was removed -- then "reaction.text"
will be falsey */
36.     'messages.reaction': { key: WAMessageKey, reaction: proto.IReaction }[]
37.
38.     'message-receipt.update': MessageUserReceiptUpdate[]
39.
40.     'groups.upsert': GroupMetadata[]
41.     'groups.update': Partial<GroupMetadata>[]
42.     /** apply an action to participants in a group */
43.      'group-participants.update': { id: string, participants: string[], action:
ParticipantAction }
44.
45.     'blocklist.set': { blocklist: string[] }
46.     'blocklist.update': { blocklist: string[], type: 'add' | 'remove' }
47.    /** Receive an update on a call, including when the call was received, rejected,
accepted */
48.     'call': WACallEvent[]
49. }
```

Untuk mendengarkan semua *event* tersebut dapat dengan mendefinisikan *event listener* sebagai contoh berikut.

```
1. const sock = makeWASocket()
2. sock.ev.on('messages.upsert', ({ messages }) => {
3.     console.log('got messages', messages)
4. })
```

2.8.4 Mengirim Pesan

Untuk mengirim berbagai jenis pesan pada baileys hanya menggunakan sebuah fungsi `*sock.sendMessage()*`. Berikut jenis-jenis pesan yang dapat dikirim menggunakan baileys.

```
 1. import { MessageType, MessageOptions, Mimetype } from '@whiskeysockets/baileys'
 2.
 3. const id = 'abcd@s.whatsapp.net' // Id WhatsApp penerima pesan
 4. // Pesan Singkat
 5. const sentMsg  = await sock.sendMessage(id, { text: 'oh hello there' })
 6. // mengirim pesan balasan
 7. const sentMsg  = await sock.sendMessage(id, { text: 'oh hello there' }, { quoted:
message })
 8. // menandai pesan
 9.  const sentMsg  = await sock.sendMessage(id, { text: '@12345678901', mentions:
['12345678901@s.whatsapp.net'] })
```

```
// mengirim lokasi
const sentMsg  = await sock.sendMessage(
    id,
    { location: { degreesLatitude: 24.121231, degreesLongitude: 55.1121221 } }
)
// mengirim kontak
const vcard = 'BEGIN:VCARD\n'
            + 'VERSION:3.0\n'
            + 'FN:Jeff Singh\n'
            + 'ORG:Ashoka Uni;\n'
            + 'TEL;type=CELL;type=VOICE;waid=911234567890:+91 12345 67890\n'
            + 'END:VCARD'
const sentMsg  = await sock.sendMessage(
    id,
    {
        contacts: {
            displayName: 'Jeff',
            contacts: [{ vcard }]
        }
    }
)

const reactionMessage = {
    react: {
        text: "💖",
        key: message.key
    }
}

const sendMsg = await sock.sendMessage(id, reactionMessage)
```

2.9 Node.JS

Node.js adalah perangkat lunak yang dirancang untuk mengembangkan aplikasi web dan menggunakan sintaks bahasa pemrograman JavaScript. Dibuat dengan bahasa JavaScript, Node.js dapat dijalankan di Windows, Mac OS X, dan Linux tanpa memerlukan perubahan pada kode program [28]. Node.js memiliki pustaka server HTTP sendiri, sehingga memungkinkan untuk menjalankan *server web* tanpa perlu menggunakan program *server web* seperti Apache atau Lighttpd .

Node.js menggunakan mesin V8 dari Google Chrome untuk mengeksekusi kode JavaScript, dan menyediakan berbagai fitur serta modul bawaan yang mempermudah pengembangan *server-side applications*. Node.js menggunakan model pemrograman *non-blocking, event-driven* yang memungkinkan server menangani banyak koneksi secara bersamaan tanpa menunggu operasi selesai (misalnya membaca file atau mengakses *database*). Ini berbeda dengan *model blocking* yang lebih umum digunakan pada server tradisional.

Node.js disertai dengan NPM (*Node Package Manager*), yang merupakan ekosistem pustaka terbesar di dunia. NPM memungkinkan pengembang untuk

dengan mudah mengunduh dan mengelola paket-paket yang dibutuhkan untuk proyek mereka. NPM juga mempermudah berbagi kode dengan komunitas.

Adapun keunggulan dari Node.JS diantaranya:

1. *Asynchronou*s dan *Event-Driven***:** Node.js menangani operasi I/O dengan *model non-blocking*, yang berarti operasi dapat berjalan secara bersamaan tanpa menunggu satu sama lain.
2. Skalabilitas**:** Karena sifat *non-blocking*, Node.js sangat efisien dalam menangani aplikasi dengan permintaan tinggi.
3. Satu Bahasa**:** Penggunaan JavaScript baik di sisi server maupun klien memungkinkan konsistensi dalam pengembangan aplikasi.
4. Komunitas Besar dan Ekosistem**:** Dukungan komunitas yang besar dan ekosistem paket yang luas melalui NPM.

Ada juga kekuarangan dari Node.JS diantaranya:

1. *Model Asynchronous***:** Meski menjadi keunggulan, model ini juga bisa menjadi tantangan bagi pengembang yang tidak terbiasa, terutama dalam hal penanganan callback yang rumit.
2. Tidak Optimal untuk CPU-Intensive Tasks**:** Node.js kurang cocok untuk tugas-tugas yang membutuhkan banyak CPU karena sifatnya yang *single-threaded*.
3. *Callback Hell***:** Penggunaan callback yang berlebihan dapat membuat kode sulit dibaca dan dipelihara, meskipun masalah ini bisa diatasi dengan menggunakan async/await atau modul manajemen kontrol aliran lainnya.

2.10 OOP

Pemrograman berorientasi objek (OOP) merupakan paradigma pemrograman yang menggunakan konsep “objek” sebagai unit dasra untuk merancang dan mengorganisir kode. Dalam paradigma ini, program dibangun dengan cara yang mencerminkan struktur dan perilaku objek dalam dunia nyata. Objek adalah instansi konkret daru kelas, dan kelas sendiri adalah cetak biru yang mendefinisikan atribut (data) dan metode (fungsi) yang dimiliki oleh objek.

Beberapa konsep utama dalam pemrograman berorientasi objek meliputi:

1. Kelas

Kelas merupakan suatu entitas yang mendifinisikan atribut (variabel) dan metode (fungsi) yang akan dimiliki oleh objek yang dibuat berdasarkan kelas tersebut. Kelas dapat dianggap sebagai template untuk menciptakan objek.

2. Objek

   Objek merupakan instansi konkret dari suatu kelas. Objek memiliki atribut yang mendefinisikan karakteristik atau data objek tersebut, serta metode yang menentukan perilaku atau tindakan yang dapat dilakukan oleh objek.

3. Enkaptulasi

   Enkaptulasi adalah konsep yang menggabungkan data dan metode keadalam satu unit yang disebut objek. Enkaptulasi memungkinkan kontrol akses terhadap data dan metode, sehingga dapat menghindari modifikasi yang tidak diinginkan.

4. Pewarisan (*Inheritance*)

   Pewarisan memungkinkan suatu kelas (kelas turunan atau *subclass*) mewarisi atribut dan metode dari kelas lain (kelas dasar atau *superclass*). Konsep ini mendukung hirearki kelas dan pengunaan kembali kode.

5. Polimorfisme

   Polimorfisme memungkinkan suatu metode memiliki banyak bentuk atau perilaku. Dalam konteks OOP, polimorfisme dapat terjadi melalui konsep *overloading* (pembebanan) dan *overriding* (penggantian).

OOP memfasilitasi pemeliharaan kode, pengembangan sistem yangt kompleks, dan pengorganisasian yang lebih baik dari pada paradigma pemrograman lainnya. Beberapa bahasa pemrograman yang mendukung OOP antara lain Java, C++, Pythoion, PHP, dan banyak lagi.

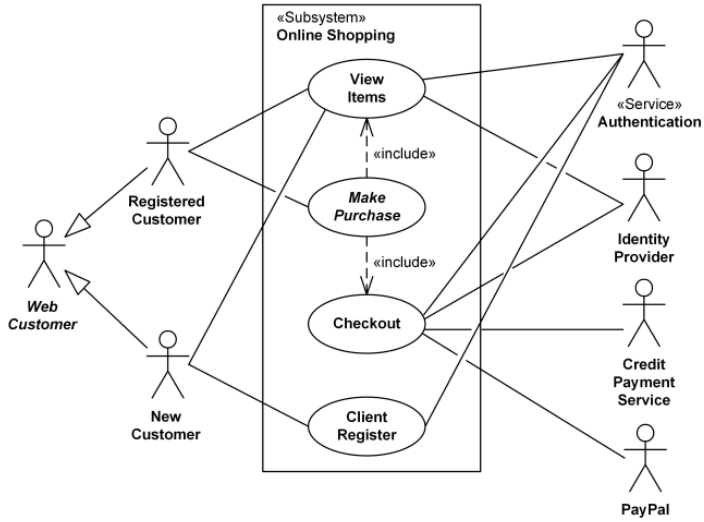
2.11 UML (*Unified Modeling Language*)

UML (*Unified Modeling Language*) merupakan suatu standar bahasa yang banyak digunakan pada dunia industri untuk mendefinisikan kebutuhan/*requirement*, analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. UML muncul dikarenakan adanya kebutuhan pemodelan visual untuk menspesifikasikan, manggambarkan, membangun, dan mengdokumentasikan sistem perangkat lunak [29].

Beberapa diagram UML yang sering digunakan pemodelan sebuah sistem berorientasi objek diantaranya:

1. *Use Case Diagram*

   Use case diagram merupakan salah satu jenis diagram dalam UML yang menvisualisasikan interkasi antara sistem dengan aktor. *Use case diagram* dapat menjelaskan jenis interaksi antara pengguna sistem dengan sistem itu sendiri [30]. Contoh *use case diagram* dapat dilihat pada gambar 2.7.

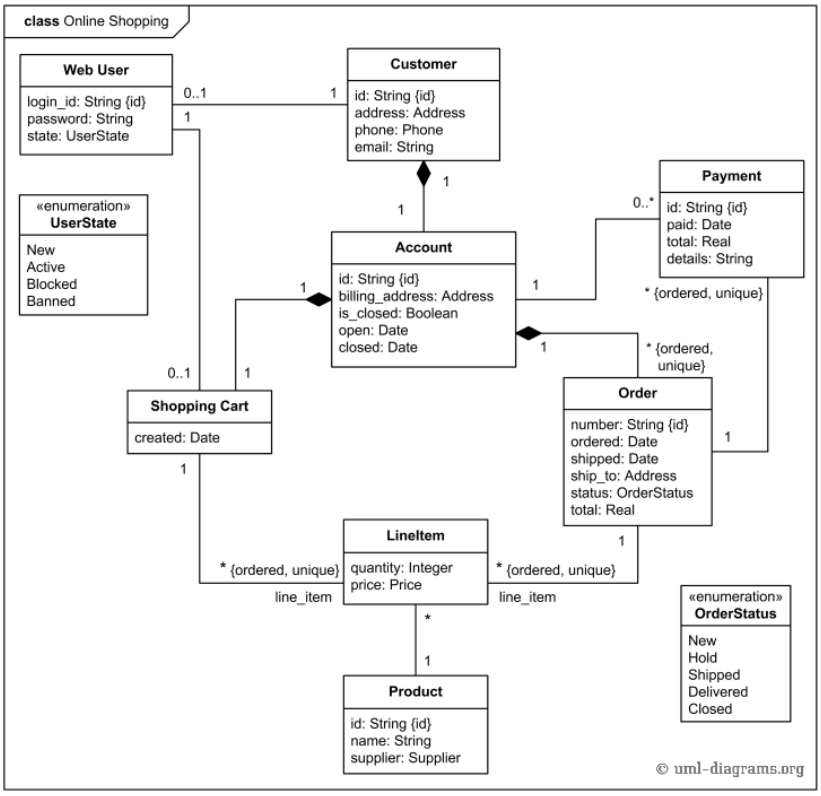


Gambar 2.7 Contoh *Use Case Diagram*

Sumber: https://www.uml-diagrams.org/examples/use-case-example-online-shopping.png

2. *Class Diagram*

   Class diagram adalah jenis diagram dalam *Unified Modeling Language* (UML) yang digunakan dalam rekayasa perangkat lunak untuk secara visual merepresentasikan struktur dan hubungan antar kelas dalam suatu sistem. Dalam class diagram, kelas-kelas dan hubungan antar kelas digambarkan dengan notasi grafis. Kelas merepresentasikan entitas atau objek dalam sistem, dan hubungan antar kelas menggambarkan bagaimana objek-objek tersebut berinteraksi satu sama lain. Contoh class diagram dapat dilihat pada gambar 2.8.

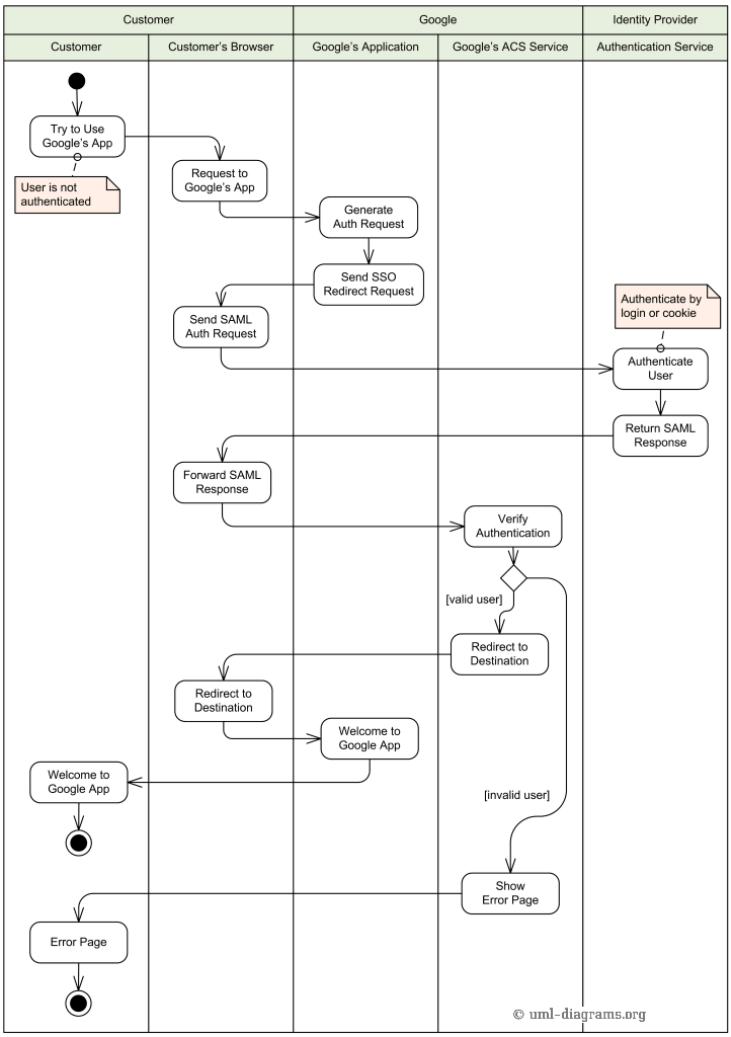


Gambar 2.8 Contoh *Class Diagram*

Sumber: https://www.uml-diagrams.org/examples/class-example-online-shopping-domain.png

3. *Acitivty Diagram*

Activity Diagram adalah salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk mengilustrasikan alur kontrol dalam suatu sistem dan merujuk pada langkah-langkah yang terlibat dalam eksekusi suatu use case. Fokus utama dari *activity diagram* adalah pada kondisi alur dan urutan di mana aktivitas-aktivitas tersebut terjadi. *Activity diagram* digunakan untuk memodelkan berbagai proses yang terjadi dalam suatu sistem. Diagram ini mampu menggambarkan bagaimana suatu aktivitas dimulai, keputusan yang mungkin terjadi, dan bagaimana aktivitas tersebut berakhir. *Activity diagram* dapat digunakan untuk menggambarkan alur kerja suatu sistem atau proses bisnis [29]. Contoh *activity diagram* dapat dilihat pada gambar 2.9.

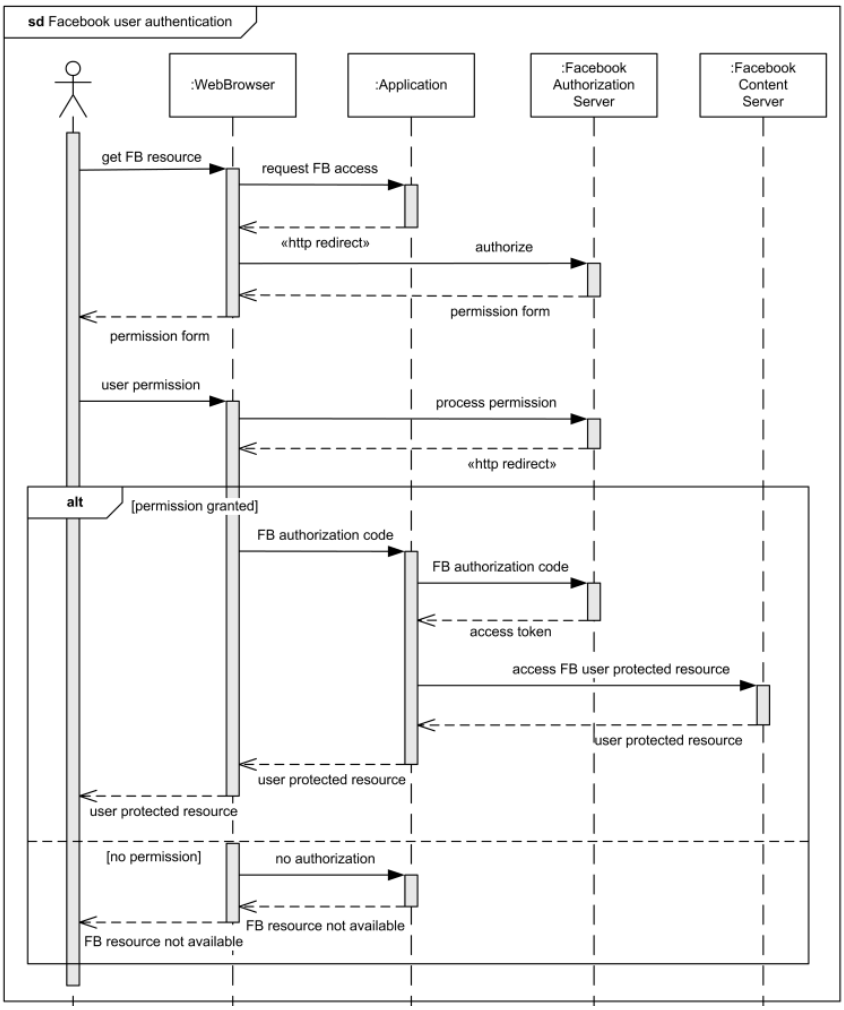


Gambar 2.9 Contoh *Activity Diagram*

Sumber: https://www.uml-diagrams.org/examples/activity-example-google-sso.png

4. *Sequence Diagram*

   Sequence diagram adalah salah satu jenis diagram dalam UML. Fokus utama dari diagram ini adalah pertukaran pesan antara objek yang terlibat dalam suatu interaksi yang berdasarkan urutan waktu. Dengan menggunakan sequence diagram, pemodelan interkasi antar objek dapat dijelaskan secara visual, termasuk langkah-langkah atau tahapan yang perlu dilakukan untuk mencapai sutu tujuan, seperti yang tergambarkan pada *use case diagram*. Contoh seqeunce diagram dapat dilihat pada gambar 2.10.



Gambar 2.10 Contoh *Sequence Diagram*

Sumber: https://www.uml-diagrams.org/examples/sequence-example-facebook-authentication.png

2.12 *Blackbox Testing*

Pengujian *black box* adalah salah satu teknik pengujian perangkat lunak di mana penguji tidak memiliki pengetahuan tentang struktur internal atau implementasi dari perangkat lunak yang diuji. Fokus utama dari pengujian ini adalah pada input dan output perangkat lunak, serta bagaimana perangkat lunak tersebut berfungsi sesuai dengan spesifikasi yang telah ditetapkan [31]. Beberapa poin utama tentang pengujian *black box* adalah:

1. Pengujian Fungsional: *Black box* testing berfokus pada pengujian fungsionalitas perangkat lunak berdasarkan persyaratan atau spesifikasi yang telah ditentukan.

Penguji memeriksa apakah perangkat lunak bekerja sesuai dengan yang diharapkan tanpa memperhatikan bagaimana perangkat lunak tersebut dibangun.

2. Pendekatan Berbasis Data: Pengujian ini biasanya melibatkan memberikan sejumlah input ke perangkat lunak dan mengamati keluaran yang dihasilkan. Jika output sesuai dengan yang diharapkan, maka perangkat lunak dianggap lulus pengujian.
3. Pengujian Antarmuka: Karena pengujian ini hanya berfokus pada bagian luar perangkat lunak, aspek seperti antarmuka pengguna (UI), interaksi pengguna, dan respons sistem menjadi bagian penting dari pengujian.

2.13 Skala Likert

Skala Likert adalah metode pengukuran yang umum digunakan untuk mengukur persepsi, sikap, atau pendapat seseorang atau kelompok terhadap suatu peristiwa atau fenomena sosial. Skala ini dinamai setelah penciptanya, Rensis Likert, dan sering digunakan dalam survei dan penelitian sosial [32]. Pertanyaan yang dirumuskan dalam bentuk positif untuk mengukur sikap positif. Respon terhadap pertanyaan ini diberi skor seperti 5 untuk sangat setuju, 4 untuk setuju, 3 untuk netral, 2 untuk tidak setuju, dan 1 untuk sangat tidak setuju. Pertanyaan yang dirumuskan dalam bentuk negatif untuk mengukur sikap negatif. Respon terhadap pertanyaan ini diberi skor kebalikan dari yang positif, yaitu 1 untuk sangat setuju, 2 untuk setuju, 3 untuk netral, 4 untuk tidak setuju, dan 5 untuk sangat tidak setuju. Skala likert memntukan presentase skor dari setiap pertayaan, berikut rumus perhitungan untuk menentukan persentase pada skala likert.

$$\text{Indeks\%} = \frac{Total\ nilai\ responden}{maximum\ skor} \times 100$$

Total nilai responden = bobot × frekuensi jawaban

Adapun interval penlinaian skala likert sebagai berikut:

Indeks 0% – 19,99% : Sangat Tidak Setuju

Indeks 20% – 39,99% : Tidak Setuju

Indeks 40% – 59,99% : Kurang Setuju

Indeks 60% – 79,99% : Setuju

Indeks 80% – 100% : Sangat Setuju