BAB 2
TINJAUAN PUSTAKA

2.1 Extension berbasis Chromium

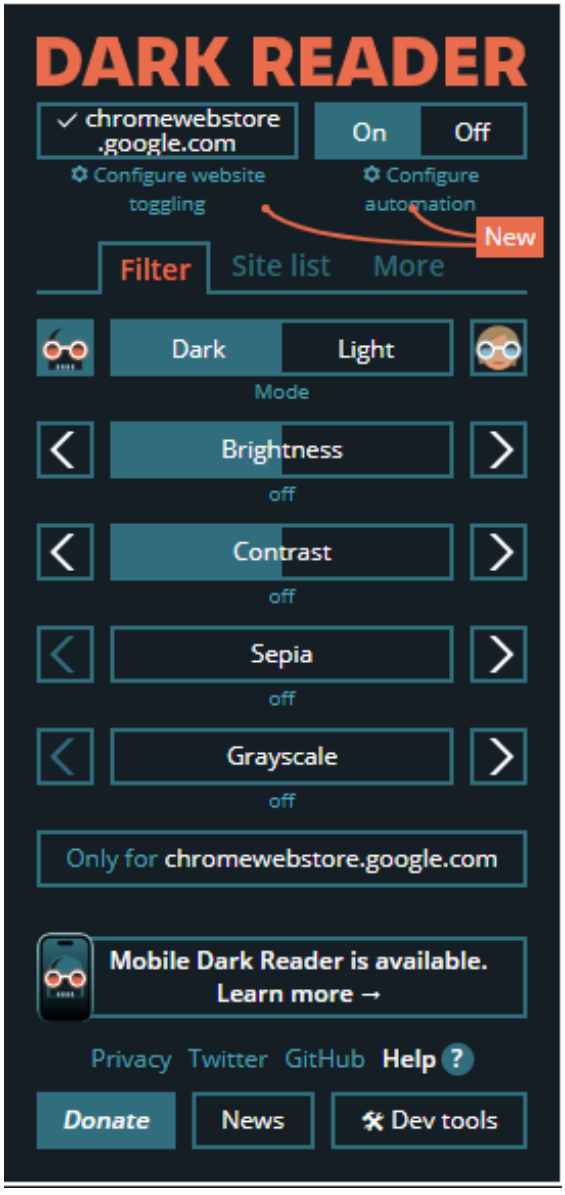
Extension berbasis *Chromium* atau *Chrome Extension* merupakan jenis ekstensi yang di kembangkan oleh *Google* untuk digunakan dengan browser web yang berbasis *Chromium*, seperti *Google Chrome* dan *Microsoft Edge*. Ekstensi ini menambah kemampuan browser dengan menambahkan fitur tambahan sesuai dengan kebutuhan pengguna dan juga mengubah cara browser untuk berinteraksi dengan halaman web yang sedang pengguna buka.

Jika dibandingkan dengan ekstensi pada browser yang tidak berbasis *Chromium*, ekstensi berbasis *Chromium* memiliki keunggulan dalam hal dukungan dan kompabilitas dengan berbagai fitur yang dapat digunakan dan *API* yang disediakan langsung oleh *Platform Chromium*. Hal ini membuat pengembangan aplikasi ekstensi pada browser berbasis *Chromium* menjadi lebih mudah dan memungkinkan pengembang untuk mengakses berbagai fungsi pada browser yang jarang disediakan secara langsung pada browser [10].

Dalam konteks pengembangan aplikasi ekstensi *Chrome* untuk mendeteksi E-mail penipuan, memanfaatkan *Platform Chromium* sebagai basis pengembangan dapat memberikan akses ke berbagai fitur yang diperlukan pengembang untuk mengimplementasikan fitur deteksi penipuan dengan lebih efektif. Dengan demikian, aplikasi ekstensi berbasis *Chromium* ini menjadi pilihan yang tepat untuk membangun aplikasi yang dapat memperkuat keamanan dan kenyamanan pengguna dalam menghadapi ancaman E-mail penipuan.

Untuk membangun aplikasi ekstensi berbasis *Chromium* ini hanya bisa menggunakan beberapa bahasa pemongraman saja yang memang sudah menjadi syarat tertulis dari *Google* selaku penyedia *platform*, yaitu Javascript, HTML5 dan CSS. Namun, pengembang masih bisa menggunakan *dependency* dalam pengembangan aplikasi, seperti menggunakan JQuery. Penulis menambahkan bahasa pemograman PHP yang digunakan sebagai *Backend* dari aplikasi ekstensi dan dalam bentuk API yang penulis buat sendiri. *Backend* ini menggunakan *framework* Laravel Sanctum yang dikhususkan untuk mengelola API saja.

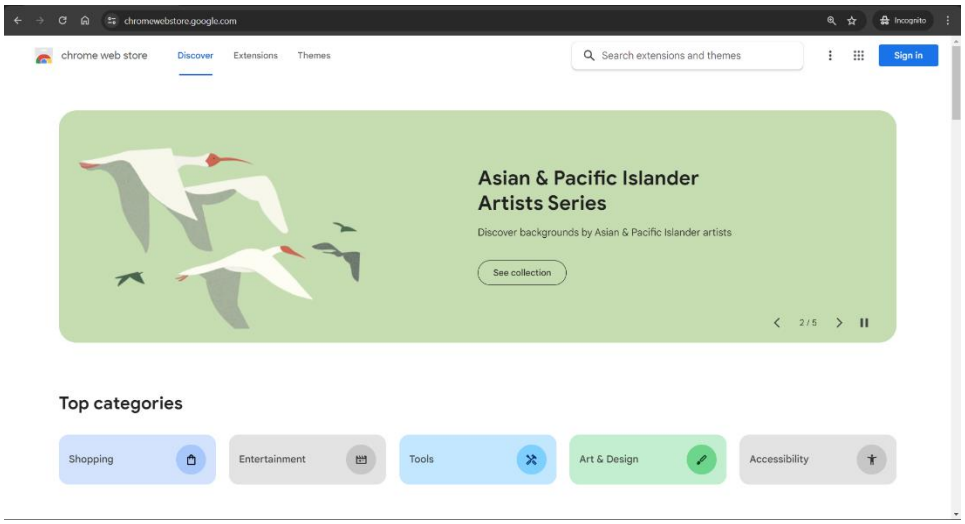
Aplikasi *Chrome Extension* bekerja dengan cara memasukan *code* pada halaman website tertentu, sehingga dapat memungkinkan ekstensi untuk berinteraksi lebih terhadap halawan website yang pengguna kunjungi. Aplikasi *chrome extension* juga banyak tersedia di *chrome extension web store* dengan berbagai macam kategori, mulai dari kebutuhan belanja, hiburan, alat, seni hingga aksesibilitas. Contoh aplikasi *chrome extension* yang termasuk kategori aksesibilitas dengan nama *Dark Reader* yang berfungsi untuk mengubah tampilan website menjadi *dark mode* bisa dilihat pada gambar 2.1.



Gambar 2.1 aplikasi chrome extension kategori aksesibilitas

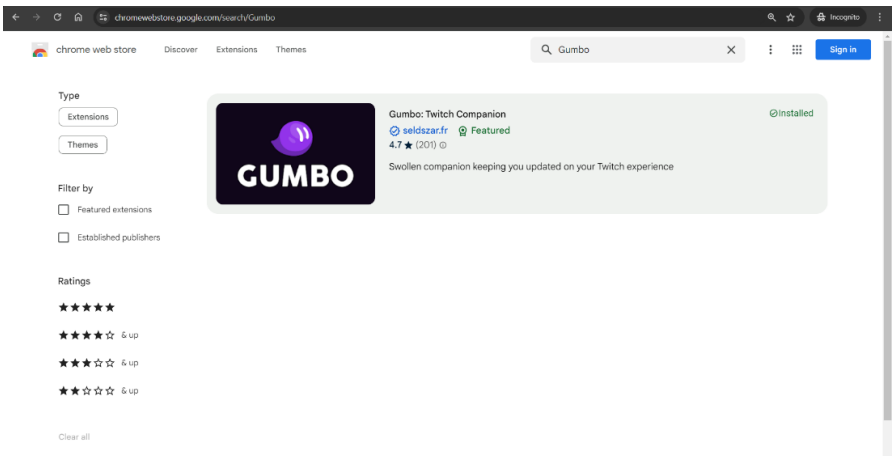
Kemudahan dalam instalasi aplikasi *chrome extension* juga menjadi bahan pertimbangan penulis untuk menggunakan *chrome extension* sebagai platform aplikasi. Dengan syarat utama pengguna diwajibkan untuk menggunakan browser yang berbasis *Chromium* seperti *Microsoft Edge, Chrome Browser, Brave,* dll. Berikut merupakan cara untuk menginstal aplikasi *chrome extension*:

1. Pengguna dapat mengakses *Chrome Web Store* yang tersedia di internet atau dengan mengakses URL https://chromewebstore.google.com .



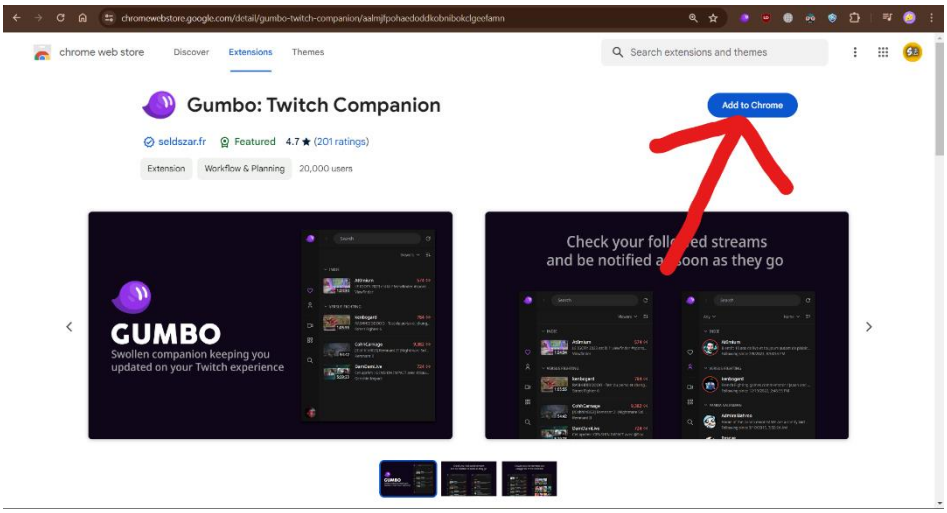
Gambar 2.2 Chrome Extenstion web store

2. Pengguna memilih aplikasi yang diinginkan dengan cara mencari atau dengan klik tombol kategori sesuai keinginan. Untuk contoh disini penulis mencoba mencari aplikasi bernam *Gumbo,* aplikasi ini merupakan aplikasi untuk *notification management.* Lalu, pilih aplikasi yang tersedia.



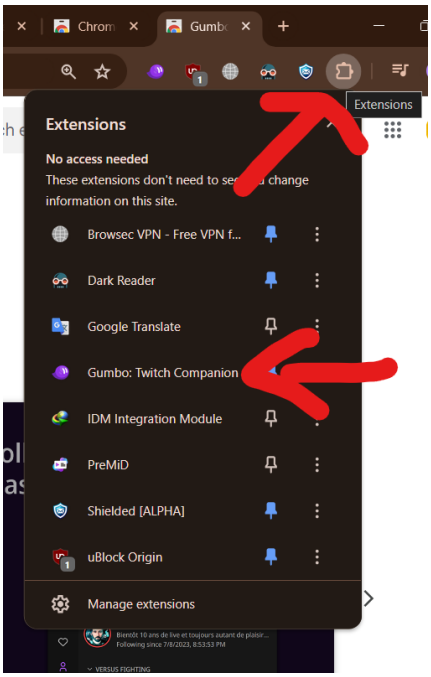
Gambar 2.3 pencarian aplikasi chrome extension

3. Klik tombol "*Add to Chrome*" atau "Tambahkan ke Chrome" lalu ikuti prosedur yang tersedia untuk memulai proses instalasi aplikasi *chrome extension* pada browser pengguna.



Gambar 2.4 Instalasi aplikasi chrome extension

4. Setelah proses instalasi selesai, pengguna dapat mengakses aplikasi *chrome extension* melalui *toolbar* yang berada di kanan atas pada browser pengguna, lalu klik aplikasi yang ingin pengguna gunakan.



Gambar 2.5 akses aplikasi chrome extension

Adapun masalah keamanan dan privasi yang menjadi faktor utama dalam pemilihan *chrome extension* sebagai platform dari aplikasi. Terdapat beberapa izin yang tersedia pada platform *chrome extension* yang harus menjadi perhatian penting untuk pengguna. Pada aplikasi yang penulis buat, terdapat beberapa izin yang digunakan untuk aplikasi bisa berjalan dan bekerja dengan baik, seperti:

1. *Storage*

   Izin *Storage* merupakan izin yang memungkinkan *developer* untuk menyipan data pada browser pengguna. Biasanya, izin ini digunakan untuk kepentingan *Session*, status aplikasi, dll.

2. *Alarms*

   Izin *Alarms* digunakan untuk ekseskusi logika yang dibutuhkan pada aplikasi secara berkala. Biasanya, izin ini diperlukan untuk memperbarui data secara otomatis.

3. *contextMenus*

   Izin ini digunakan untuk keperluan *Easy Access* untuk pengguna aplikasi. Biasanya, izin ini digunakan agar pengguna dapat meng-akses aplikasi melalui cara yang tidak biasa, seperti klik kanan atau melalui *hotkey* yang sudah pengguna atur atau secara *default.*

4. *HostPermission*

   Izin *HostPermission* memungkinkan *developer* untuk mengakses server dari luar aplikasi untuk kebutuhan khusus. Biasanya, izin ini digunakan jika *developer* menggunakan API dalam aplikasi mereka dan bisa juga untuk menyimpan data ke server diluar aplikasi.

Maka dari itu, perlu nya pengetahuan pengguna akan perizinan pada suatu aplikasi *chrome extension* sangatlah penting. Hal ini untuk menghindari eksploitasi data pribadi pengguna yang sensitif dengan sangat mudah untuk diakses oleh

developer “nakal”. Penulis sangat menjungjung tinggi akan keamanan dan privasi bagi pengguna dalam aplikasi pendeteksi E-mail penipuan, karena hanya menggunakan beberapa izin yang diperlukan agar aplikasi bisa berjalan dengan baik dan melakukan pembenaran (*justification*) secara tertulis mengenai perizinan yang digunakan demi kenyamanan dan kepercayaan pengguna.

Penggunaan *chrome extension* sebagai *platform* dari aplikasi pendeteksi E-mail penipuan sudah menjadi hal yang tepat karena memberikan pengalaman browsing yang lebih kaya dan juga dapat disesuaikan sesuai dengan kebutuhan pengguna.

2.2 VirusTotal API

Gambar 2.6 Logo VirusTotal

VirusTotal merupakan sebuah *platform* yang sering digunakan untuk menganalisis suatu berkas dan URL untuk mendeteksi adanya ancaman *malware* dan ancaman keamanan lainnya. API ini memungkinkan pengembang untuk mengirim *request* HTTP dan dibalas dengan *response* dalam bentuk JSON yang berisi hasil dari analisis berbagai mesin antivirus dan layanan keamanan yang terhubung ke VirusTotal [14].

Dengan menggunakan VirusTotal API, pengembang dapat mengintegrasikan fitur yang tersedia pada VirusTotal ke dalam aplikasi mereka. Misalnya, pengembang dapat melakukan scan berkas dengan cara mengunggah berkas yang sudah dalam bentuk ZIP ke VirusTotal untuk di Analisa, dan kemudian akan mendapat *response* berupa hasil analisis dalam bentuk JSON. *Response* yang diberikan oleh VirusTotal dapat diolah kembali oleh pengembang sehingga membantu pengguna dalam mengidentifikasi berkas yang berpotensi berbahaya dan mengambil tindakan yang sesuai untuk mencegah hal buruk terjadi.

VirusTotal juga menyediakan berbagai macam *endpoints,* mulai dari yang sifat nya gratis hingga berbayar. *Endpoints* dalam konteks pembangunan perangkat lunak atau aplikasi merujuk pada titik akhir dari suatu komunikasi antara aplikasi dan server yang digunakan, biasanya dalam bentuk *request* HTTP. *Endpoints* juga menentukan suatu permintaan dapat dikirim dan bagaimana aplikasi harus menanggapi atau me-*response* permintaan tersebut. Dalam konteks VirusTotal API, *endpoints* adalah URL spesifik yang biasa digunakan untuk berinteraksi antara aplikasi dengan API tersebut. Setiap *endpoints* mungkin memiliki fungsionalitas yang berbeda-beda tergantung dari jenis API yang digunakan [14].

Berikut merupakan beberapa *endpoints* yang populer dan sering digunakan pada layanan API VirusTotal.

1. Get an IP address report

Endpoint ini memiliki link URL dengan *method* GET sebagai berikut:

```
https:// www.virustotal.com/api/v3/ip_addresses/{ip}
```

Endpoint ini berguna untuk melakukan cek pada domain namun dalam bentuk IP. Terdapat beberapa informasi yang didapat dan berguna seperti nama pemilik domain, tipe *encryption,* dan hasil vote dari semua layanan antivirus yang bekerja sama dengan VirusTotal. Berikut merupakan *source code* untuk implementasi *endpoint* ini dalam bahasa pemograman Javascript:

```
const options = {
  method: 'GET',
  headers: {
    accept: 'application/json',
    'x-apikey': ApiKey
  }
};

const ip = 89.213.211.215;
fetch(`https://www.virustotal.com/api/v3/ip_addresses/${ip}`, options)
  .then(response => response.json())
  .then(response => console.log(response))
  .catch(err => console.error(err));
```

Jika implementasi sudah berhasil, maka *endpoint* ini akan memberikan *response* sebagai berikut:

```
{
  "data": {
    "id": "89.213.211.215",
    "type": "ip_address",
```

```
    "links": {
      "self": "https://www.virustotal.com/api/v3/ip_addresses/89.213.211.215"
    },
    "attributes": {
      "last_analysis_date": 1719585206,
      "last_https_certificate_date": 1719585402,
      "last_analysis_stats": {
        "malicious": 0,
        "suspicious": 0,
        "undetected": 93,
        "harmless": 0,
        "timeout": 0
      },
…
```


2. Get a domain report

Endpoint ini memiliki link URL dengan *method* GET sebagai berikut:

```
https://www.virustotal.com/api/v3/domains/{domain}
```

Endpoint ini sama hal nya seperti *endpoint* sebelum nya yaitu get an ip address report, namun pada *endpoint* ini berbentuk domain. Kegunaan dan hasil *response* dalam *endpoint* ini pun hampir sama. Berikut merupakan *source code* untuk implementasi *endpoint* ini dalam bahasa pemograman Javascript:

```
const options = {
  method: 'GET',
  headers: {
    accept: 'application/json',
    'x-apikey': ApiKey
  }
};
const domain = resaka.my.id;
fetch(`https://www.virustotal.com/api/v3/domains/${domain}`, options)
  .then(response => response.json())
  .then(response => console.log(response))
  .catch(err => console.error(err));
```

Jika implementasi sudah berhasil, maka *endpoint* ini akan memberikan *response* sebagai berikut:


```
{
  "data": {
    "id": "resaka.my.id",
    "type": "domain",
    "links": {
      "self": "https://www.virustotal.com/api/v3/domains/resaka.my.id"
    },
    "attributes": {
      "tags": [],
      "categories": {},
     "last_analysis_results": {…},
      "whois_date": 1711782458,
      "last_https_certificate": {
        "cert_signature": {
          "signature_algorithm": "sha256RSA",
      "last_analysis_stats": {
```

```
        "malicious": 0,
        "suspicious": 0,
        "undetected": 93,
        "harmless": 0,
        "timeout": 0
      },
…
```

3. Upload a file

Endpoint ini memiliki link URL dengan *method* GET sebagi berikut:

```
https://www.virustotal.com/api/v3/files
```

endpoint ini memiliki kemampuan untuk menganalisa suatu file yang sudah dikemas dalam bentuk zip. File zip tersebut lalu di ubah kembali kedalam format string base64 sehingga dapat dikirim dalam bentuk request dari server-side. Berikut merupakan impelentasi *source code* dalam bahasa pemograman Javascript:

```
const form = new FormData();
form.append('file', `data:application/x-zip-compressed;name=chrome-
extension.zip;base64, ${base64}`)
const options = {
  method: 'POST',
  headers: {
    accept: 'application/json',
    'x-apikey': ApiKey
  }
};
options.body = form;
fetch('https://www.virustotal.com/api/v3/files', options)
  .then(response => response.json())
  .then(response => console.log(response))
  .catch(err => console.error(err));
```

Jika implementasi sudah berhasil, maka *endpoint* ini akan memberikan *response* seperti berikut:

```
{
  "data": {
    "type": "analysis",
    "id": "ODU5MGU0ZWYzMGE1MmU3YWNlZDdkOWFjYTc3MDVmOWM6MTcxOTczOTgxOA==",
    "links": {
      "self":
"https://www.virustotal.com/api/v3/analyses/ODU5MGU0ZWYzMGE1MmU3YWNlZDdkOWFjYTc3
MDVmOWM6MTcxOTczOTgxOA=="
    }
  }
}
```

Id dalam *response* yang diberikan akan digunakan pada *endpoint* lainnya yang berguna untuk melakukan *retrive* data dari hasil analisa.

4. Scan URL

Endpoint ini memiliki link URL dengan *method* POST sebagai berikut:

```
https://www.virustotal.com/api/v3/urls
```

Endpoint ini berguna jika ingin melakukan analisa dari suatu URL. *Endpoint* ini memberikan *response* dari hasil analisa URL yang di berikan saat suatu server-side melakukan request ke *endpoint* ini. Berikut merupakan *source code* dalam implementasi *endpoint* ini dalam bahasa pemograman Javascript:

```
const options = {
  method: 'POST',
  headers: {
    accept: 'application/json',
    'x-apikey': ApiKey,
    'content-type': 'application/x-www-form-urlencoded'
  },
  body: new URLSearchParams({url: 'https://www.resaka.my.id/'})
};
fetch('https://www.virustotal.com/api/v3/urls', options)
  .then(response => response.json())
  .then(response => console.log(response))
  .catch(err => console.error(err));
```

Jika implementasi sudah berhasil, maka *endpoint* ini akan memberikan *response* sebagai berikut:

```
{
  "data": {
    "type": "analysis",
    "id": "u-97843778984db69f472ac13bd11713701a2d7e445b41572e48e96f53aa2af797-
1719745181",
    "links": {
      "self": "https://www.virustotal.com/api/v3/analyses/u-
97843778984db69f472ac13bd11713701a2d7e445b41572e48e96f53aa2af797-1719745181"
    }
  }
}
```

5. Get a URL / file analysis

Endpoint ini memiliki link URL dengan *method* GET sebagai berikut:

```
https://www.virustotal.com/api/v3/analyses/{id}
```

Endpoint ini berfungsi untuk mendapatkan hasil dari analisa suatu *endpoint* yang tidak memberikan hasil analisa secara langsung, seperti pada *endpoint* “Scan a file”. Dengan menggunakan Id yang di dapatkan, maka kita akan mendapatkan hasil dari analisa pada *endpoint* ini. Untuk implementasi dalam bahasa pemograman Javascript sebagai berikut:

```
const options = {method: 'GET', headers: {accept: 'application/json'}};
const id = 'ODU5MGU0ZWYzMGE1MmU3YWNlZDdkOWFjYTc3MDVmOWM6MTcxOTczOTgxOA==';
fetch(`https://www.virustotal.com/api/v3/analyses/${id}`, options)
  .then(response => response.json())
  .then(response => console.log(response))
```

```
  .catch(err => console.error(err));
```

Jika implementasi sudah berhasil, maka *endpoint* ini akan memberikan *response* dalam bentuk JSON berisi hasil analisa dari id yang diberikan.

Integrasi VirusTotal API pada aplikasi ekstensi deteksi E-mail penipuan dapat memberikan tambahan lapisan keamanan yang lebih kuat lagi. Dengan demikian, VirusTotal API menjadi komponen penting dalam mengamankan E-mail pengguna dari ancaman keamanan yang berpotensi merugikan.

2.3 Google Translate API

Gambar 2.7 Logo Google Translate

Google Translation API merupakan layanan yang disediakan oleh *Google* untuk menerjemahkan teks dari suatu bahasa ke bahasa yang lain. API ini memungkinkan pengembang untuk mengintegrasikan fungsi terjemahan otomatis ke dalam aplikasi [11]. Dikarenakan mayoritas API yang penulis gunakan menerima request dalam bahasa Indonesia atau bahasa Inggris, sehingga untuk melakukan Analisa text pada *body* E-mail dibutuhkan penyesuaian bahasa terlebih dahulu. Dengan menggunakan *Google Translate API,* pengembang dapat mengubah teks pada *body* E-mail yang bukan bahasa Indonesia menjadi bahasa Indonesia maupun text yang bukan bahasa Inggris menjadi bahasa Inggris, sehingga memudahkan pengembang untuk melakukan *request* ke suatu API.

Google Translate API menggunakan teknologi *Aritificial Intelligence* (AI) atau kecerdasan buatan untuk melakukan penerjemahan teks dari suatu bahasa ke bahasa lain. AI dalam *Google Translate* bekerja dengan mempelajari pola dan struktur dari sebuah teks untuk memahami makna dan konteks dari teks yang akan

diterjemahkan [11]. Melalui teknologi *Deep Learning* seperti *Neural Machine Translation* (NMT), *Google Translate API* dapat menerjemah suatu teks pada suatu bahasa dengan akurat dan alami jika dibandingkan dengan metode terjemahan konvensional. AI dalam *Google Translate API* juga akan terus diperbarui dan juga ditingkatkan melalui pembaruan berlanjut yang berdasarkan dari data baru dan *feedback* dari pengguna, sehingga hal ini dapat terus meningkatkan kemampuan dalam menerjemah teks menjadi lebih baik dari waktu ke waktu.

Penulis menggunakan *Google Translate API* untuk menerjemahkan teks pada *body* E-mail yang pengguna buka. Dikarenakan bahsa pada teks E-mail tidak mungkin selalu bahasa Indonesia, sedangkan *Text Processing* yang dilakukan hanya mampu untuk meng-analisa teks bahasa Indonesia. Maka dari itu, penggunaan *Google Translate API* merupakan solusi yang tepat, mengigat tingkat akurasi dan kemampuan dari *Google Translate API* untuk mendalami konteks dari suatu teks. Sehingga teks hasil terjemahan tidak akan merubah arti dan makna dari teks orisinal nya.

Sebelum menggunakan *package* ini, diharuskan untuk lakukan peng-installan *package* terlebih dahulu dengan menggunakan composer. Composer bisa didapatkan di link berikut:

```
https://getcomposer.org/download/
```

Jika composer sudah berhasil diinstal, lakukan installasi *package* melalui terminal yang mengarah langsung ke *directory* dari *project*. Berikut merupakan *command* untuk installasi *package* Google Translate PHP:

```
composer require stichoza/google-translate-php
```

Jika proses installasi *packge* sudah sukses, maka *package* sudah bisa digunakan dalam *project*.

Untuk memulai menggunakan *package* ini, diharuskan untuk memanggil *object* terlebih dahulu dan melakukan inisialisasi seperti pada potongan *source code* berikut:

```
use Stichoza\GoogleTranslate\GoogleTranslate;
$tr = new GoogleTranslate('en'); // Translates into English
```

Atau bisa juga dengan memutuskan untuk menggunakan *auto detect language* yang dapat di implementasikan seperti pada potongan *source code* berikut:

```
$tr = new GoogleTranslate(); // Translates to 'en' from auto-detected language
by default
$tr->setSource('en'); // Translate from English
$tr->setSource(); // Detect language automatically
$tr->setTarget('id'); // Translate to Indonesian
```

Setelah itu, bisa dilanjutkan dengan *syntax* untuk men-translate suatu kalimat yang bisa di implementasikan seperti pada potongan *source code* dibawah ini:

```
echo $tr->translate('Hello World!');
```

Atau bisa juga dengan metode *chain* seperti pada potongan *source code* berikut:

```
echo $tr->setSource('en')->setTarget('id')->translate('Goodbye');
```

Atau bisa juga dengan metode *static* seperti pada potongan *source code* berikut:

```
echo GoogleTranslate::trans('Hello again', 'id', 'en');
```

2.4 OpenAI Fine-Tuning

Gambar 2.8 Logo OpenAI

OpenAI Fine-Tuning merupakan proses melatih model kecerdasan buatan yang sudah ada dengan memberikan data contoh yang spesifik untuk menyesuaikan kinerja dari model tersebut pada suatu tugas yang diberikan. Model yang digunakan dalam fine-tuning sudah dilatih sebelumnya dengan data yang sangat banyak dan beragam, sehingga model yang dipakai sudah memiliki *basic* pengetahuan dan kemampuan untuk memahami dan memberikan output berupa text yang alami. Namun, agar model yang dipakai bisa memberikan output yang lebih spesifik dan sesuai dengan kebutuhan, model tersebut perlu dilatih terlebih dahulu dengan menggunakan *dataset* yang memiliki format sesuai dengan dokumentasi yang diberikan oleh OpenAI.

Dalam aplikasi *chrome extension* pendeteksi E-mail penipuan, menggunakan model GPT-3.5-Turbo-0125, salah satu model yang dikembangkan oleh OpenAI. Model ini memiliki kemampuan dasar yang sangat kuat dalam memahami bahasa alami dan memberikan output berupa teks yang relevan. Melalui proses fine-tuning, model ini dilatih kembali dengan *dataset* yang berisi contoh-contoh teks yang spesifik, seperti E-mail yang memiliki kemungkinan penipuan. Proses ini memungkinkan model untuk mengenali pola-pola tertentu dari sebuah teks yang diberikan dan menunjukan adanya kemungkinan penipuan dalam E-mail, sehingga model dapat memberikan output berupa prediksi yang lebih akurat saat digunakan dalam aplikasi.

Proses fine-tuning memerlukan beberapa tahapan proses, termasuk pengumpulan *sample* yang dikemas dalam bentuk *dataset* dan sesuai dengan format yang terdapat dalam dokumentasi. *Dataset* yang digunakan memiliki isi berupa pasangan teks input dan output yang diinginkan, hal ini akan membantu model untuk belajar dari contoh-contoh yang diberikan. Dan *platform OpenAI* sendiri memberikan rekomendasi untuk memberikan minimal 10 contoh *dataset,* semakin banyak contoh *dataset* yang diberikan akan semakin baik respons dari model yang dilatih. Dengan demikian, fine-tuning dapat membantu model untuk lebih terfokus pada konteks spesifik dan meningkatkan akurasi dalam mengerjakan suatu tugas, dan fine-tuning penulis kira sangat cocok dan memiliki kemampuan untuk membantu mendeteksi E-mail penipuan. Berikut merupakan contoh *dataset* yang penulis gunakan dalam format JSONL *(Javascript Object Notation Lines)* dapat dilihat pada potongan *source code* berikut.

```
{
    "messages": [
        {
            "role": "system",
"content": "Shielded is an app for analyzing which email body is indicated as
phising or not, user will sent text content of body email in English that you
will be analyze and you have to decided this email is phising or not by answer
in Indonesian"
        },
        {
            "role": "user",
"content": "Relax and have fun with poker, blackjack, roulette, progressive
video slots at your own leisure from your couch. Our safe, secure games will get
you smiling when you start seeing dollars pouring in. We're serious about fun.
When YOU WIN, we win! We Work You WIN We Pay! You Play!
http://casinojackpotsdeals.com"
```

```
        },
        {
            "role": "assistant",
"content": "Skema E-mail seperti ini sebelumnya telah terdeteksi percobaan
Phising, Jangan lakukan aksi apapun sebelum melakukan analisa total!"
        }
    ]
```

Dapat terlihat dalam *dataset* terdapat beberapa *role* seperti “system”, “user”, dan “assistant”, berikut merupakan penjelasan dari *role* tersebut:

1. *Role system*

Content dalam role ini bertujuan untuk memberikan intruksi atau konteks terhadap model yang akan dilatih mengenai bagaimana model tersebut harus menanggapi atau me-*response* selama sesi interaksi terjadi. Dari potongan *dataset* diatas, nilai dari role *system* adalah sebagai berikut:

```
"Shielded is an app for analyzing which email body is indicated as phising or not, user will sent text content of body email in English that you will be analyze and you have to decided this email is phising or not by answer in Indonesian"
```

Intruksi ini memberitahu model bahwa aplikasi *Shielded* merupakan aplikasi untuk analisa *body* dari E-mail yang terindikasi *Phising*, juga memberikan konteks bahwa *role user* akan memberikan input berupa isi dari *body* E-mail yang akan di analisa dan model harus menentukan bahwa E-mail tersebut termasuk *Phising* atau tidak dan memberikan jawaban dalam bahasa Indonesia.

2. *Role user*

Content dari *role user* merupakan *input* dari pengguna, yang mana *input* dari pengguna merupakan *body* dari E-mail yang pengguna ingin analisa. Dikarenakan model dilatih pada umum nya dalam bahasa Inggris, maka *body* dari E-mail yang pengguna kirim harus di *translated* terlebih dahulu ke bahasa Inggris. Sehingga apapun bahasa yang terkandung dalam E-mail pengguna, pada akhir nya akan menjadi bahasa Inggris dan dapat menyesuaikan dengan kebutuhan model untuk melakukan analisa suatu *body* dari E-mail. Jika berdasarkan contoh *dataset* diatas, *role user* memiliki *content* sebagai berikut:

```
"Relax and have fun with poker, blackjack, roulette, progressive video slots at your own leisure from your couch. Our safe, secure games will get you smiling when you start seeing dollars pouring in. We're serious about fun. When YOU WIN, we win! We Work You WIN We Pay! You Play! http://casinojackpotsdeals.com"
```

3. *Role assistant*

Content dari *role assistant* menunjukan respons dari model yang diharapkan. *Role* ini yang nantinya akan menjadi pedoman bagi model ketika memberika respons dari setiap *request* yang diberikan. Sehingga respons dari model tidak akan berbeda jauh dari isi *content role assistant.* Jika berdasarkan contoh *dataset* diatas, *content* dari *role assistant* sebagai berikut:

```
"Skema E-mail seperti ini sebelumnya telah terdeteksi percobaan Phising, Jangan lakukan aksi apapun sebelum melakukan analisa total!"
```

Untuk memulai proses melatih model dari *platform* OpenAI Fine-Tuning, berikut merupakan langkah-langkah nya:

1. Login

   Masuk ke OpenAI Dashboard dengan cara meng-akses link berikut:

```
https://platform.openai.com/
```

2. Navigate

   Pergi ke menu Fine-Tuning yang berada di navigasi sebelah kiri.

3. Upload *dataset* JSONL

   Klik tombol “create” dan unggah file JSONL yang sudah dibuat.

4. Configure

   Pilih model GPT-3.5-Turbo-0125 karena model ini merupakan model paling stabil yang OpenAI punya setidaknya saat penulis membuat skripsi ini. Masukan juga nama *suffix*, ini hanya untuk penanda khusus untuk model yang sudah di fine-tuning dan nantinya model dengan nama *suffix* inilah yang akan dipakai selama ber-interaksi.

5. Start

   Klik Start Fine-Tuning dan tunggu proses selesai.

6. Evaluate

   Uji model di dashboard setelah proses selesai.

7. Deploy

Dapatkan endpoint API dan juga gunakan model yang sudah dilatih dengan menggunakan nama *suffix* yang sebelum nya sudah di atur.

Jika proses melatih model sudah berhasil, maka model yang sudah di sesuaikan dengan OpenAI Fine-Tuning sudah dapat digunakan. Berikut merupakan implementasi *endpoint* OpenAI Fine-Tuning dengan menggunakan ajax:

```
var data = {
    model: 'ft:gpt-3.5-turbo-0125:indonesia-computer-
university:analyzephising:9gTMFjDF', //Suffix yang digunakan
    messages: [
        {
            "role": "system",
            "content": "Shielded is an app for analyzing which email body is
indicated as phising or not, user will sent text content of body email in
English that you will be analyze and you have to decided this email is phising
or not by answer in Indonesian"
        },
        {
            "role": "user",
            "content": inputText //Input dari pengguna
        }
    ],
    max_tokens: 256 //Maksimal token yang akan digunakan
};
$.ajax({
    url: 'https://api.openai.com/v1/chat/completions',
    type: 'POST',
    headers: {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer ' + OpenAIKey
    },
    data: JSON.stringify(data),
    success: function (response) {
        console.log(response);
    }
});
```

2.5 Laravel Sanctum

Gambar 2.9 Logo Laravel Sanctum

Laravel Sanctum menyediakan sistem autentikasi untuk SPAs (*Single Page Applications*), aplikasi mobile, dan simple dengan *basic* APIs (*Application*

Programming Interfaces). Sanctum membuat setiap pengguna dalam aplikasi dapat men-*generate* beberapa token API untuk akun mereka. Token ini dapat memberikan kemampuan atau ruang lingkup dengan aksi spesifik yang di perbolehkan oleh aplikasi untuk di eksekusi.

Laravel Sanctum hadir untuk memecahkan dua masalah yang berbeda, seperti:

1. API Tokens

   Sanctum merupakan *package* praktis yang dapat digunakan untuk Masalah API token bagi pengguna tanpa melalui proses autentikasi. Fitur ini terinspirasi dari Github dan beberapa aplikasi yang memiliki masalah "*personal access token*". Sebagai contoh, bayangkan "Pengaturan Akun" pada suatu aplikasi dapat men-*generate* sebuah API token untuk suatu akun. Sanctum juga dapat men-*generate* dan mengatur token tersebut. Token tersebut biasa nya memiliki jangka kadaluwarsa yang panjang (mungkin hingga bertahun-tahun), namun mungkin juga dapat dihapus secara manual oleh pengguna.

2. SPA Authentication

Sanctum hadir dengan menawarkan proses autentikasi yang mudah dalam Suatu halaman aplikasi yang butuh untuk berkomunikasi dengan Laravel yang ditenagain oleh API. SPAs ini mungkin hadir dalam *repository* yang sama dengan aplikasi Laravel atau mungkin juga terpisah secara keseluruhan dengan *repository*, seperti SPA yang dibuat menggunakan Vue CLI, Next.js atau framework lainnya.

Dalam fitur ini, Sanctum tidak menggunakan token yang sama seperti yang sudah dijelaskan. Namun, Sanctum menggunakan Laravel's Build-in Cookie yang berbasis dalam *Authentication Services.* Secara tipikal, Sanctum menggunakan *web authentication guard* untuk dapat berjalan. Hal ini memberikan keuntungan dari *CSRF Protection*, sesi autentikasi, dan juga menjaga dari kebocoran autentikasi *credentials* melalui XSS.

Sanctum hanya melakukan autentikasi menggunkan *cookies* saat *request* datang melalui aplikasi *front-end*. Saat Sanctum menerima HTTP *request*, Sanctum

akan cek autentikasi dari *cookies* dan, jika tidak ada, Sanctum akan melakukan cek pada *header "Authorization"* dari API token yang valid.

Untuk dapat menggunakan Laravel Sanctum, mesin diharuskan untuk ter-install *composer* terlebih dahulu. Lakukan download dan install *Composer*, dapatkan file install nya dengan meng-akses link official *Composer* berikut:

```
https://getcomposer.org/download/
```

Setelah berhasil installasi Composer, lalu install *package* dari Laravel Sanctum dengan cara memasukan *command* berikut ke dalam terminal yang sudah mengarah ke folder *project* Laravel:

```
composer require laravel/sanctum
```

Jika sudah berhasil, lakukan *publish* konfigurasi dari Sanctum dan file migrasi dengan menggunkan *command artisan* vendor:publish. Konfigurasi Sanctum secara otomatis akan tersedia dalam folder "*configuraion*" dalam apliksi Laravel.

```
php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"
```

Langkah terakhir, jalankan migrasi database. Sanctum akan membuat satu tabel dalam database untuk *store* API token.

```
php artisan migrate
```

Selanjutnya, jika ingin menggunakan Sanctum untuk autentikasi SPAs, diharuskan untuk memasukan Sanctum's Middleware ke dalam grup API Middleware dalam aplikasi Laravel yang terletak di App/Http/Kernel.php:

```
'api' => [
\Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
\Illuminate\Routing\Middleware\ThrottleRequests::class.':api',
\Illuminate\Routing\Middleware\SubstituteBindings::class,
],
```

Untuk melakukan autentikasi SPA, halaman login pada *front-end* diharuskan untuk melakukan *request* ke *endpoint "*/sanctum/csrf-cookie*"* untuk inisialisasi CSRF Protection untuk aplikasi *front-end:*

```
axios.get('/sanctum/csrf-cookie').then(response => {
// Login...
});
```

Selama *request* berlangsung, Laravel akan membuat cookie XSRF-TOKEN yang berisi nilai dari CSRF-TOKEN saat itu. Token ini harus diberikan melalui header

X-XSRF-TOKEN dalam *subsequent header*, yang mana dalam beberapa *HTTP client library* seperti Axios dan *Angular HTTPClient* secara otomatis akan membuat header tersebut. Tetapi, jika menggunakan Javascript, diharuskan untuk memasukan *header* X-XSRF-TOKEN secara manual untuk memastikan memiliki nilai yang sama dengan cookie XSRF-TOKEN yang sudah diatur dalam *route*.

Untuk menjaga *routes* sehingga *request* yang datang diharuskan untuk melewati proses autentikasi terlebih dahulu, diharuskan untuk memasukan Sanctum *Authentication Guard* ke dalam API *routes* yang terletak di "routes/api.php". Langkah ini untuk memastikan setiap *request* yang datang sudah ter-autentikasi dan memiliki isi API token header yang valid jika *request* datang dari pihak ketiga. Dapat di implementasikan dengan potongan *source code* seperti berikut:

```
use Illuminate\Http\Request;
Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});
```

Jika *front-end* membutuhkan autentikasi dengan *private / presence broadcast channel*, diharuskan untuk memasukan *method* Broadcast::routes yang terdapat pada "routes/api.php".

```
Broadcast::routes(['middleware' => ['auth:sanctum']]);
```

Selanjutnya, agar Pusher's Authorization *request* berhasil, diharuskan untuk menyediakan Custom Pusher's Authorizer saat inisialisasi Laravel Echo. Hal ini membuat aplikasi agar melakukan konfigurasi Pusher untuk menggunakan *Instance Axios* yang sudah di konfigurasi untuk *cross-domain request.*

```
window.Echo = new Echo({
   broadcaster: "pusher",
   cluster: import.meta.env.VITE_PUSHER_APP_CLUSTER,
   encrypted: true,
   key: import.meta.env.VITE_PUSHER_APP_KEY,
   authorizer: (channel, options) => {
       return {
           authorize: (socketId, callback) => {
               axios.post('/api/broadcasting/auth', {
                   socket_id: socketId,
                   channel_name: channel.name
               })
               .then(response => {
                   callback(false, response.data);
               })
               .catch(error => {
                   callback(true, error);
               });
           }
```

```
        };
    },
})
```

Jika ingin menggunakan token sebagai metode autentikasi, maka diharuskan untuk membuat route yang menerima email / username dan password dari pengguna, yang nantinya akan ditukar dengan credentials untuk token Sanctum baru. Untuk implementasi seperti pada potongan source code berikut:

```
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\ValidationException;
Route::post('/sanctum/token', function (Request $request) {
    $request->validate([
        'email' => 'required|email',
        'password' => 'required'
    ]);
    $user = User::where('email', $request->email)->first();
    if (! $user || ! Hash::check($request->password, $user->password)) {
        throw ValidationException::withMessages([
            'email' => ['The provided credentials are incorrect.'],
        ]);
    }
    return $user->createToken($request->email)->plainTextToken;
});
```

Dengan *method* ini, pengguna akan mendapatkan token yang nantinya akan digunakan setiap melakukan *request*. Dan token ini harus diberikan pada *header* bernama “Authorization”.

Dengan menerapkan Laravel Sanctum sebagai server-side pada aplikasi yang akan dibangun, diharapkan dapat mempercepat *response* dari server-side dan juga mengurangi *spike peformance* pada client-side karena tugas analisa semua diserahkan pada server-side. Dengan penggunaan hosting VPS (*Virtual Private Server*) dengan server berada di dalam negri, mendapatkan *response* sekitar 14.195 ms (*milisecond*) [20].

2.6 Local Storage

Local Storage merupakan fitur dari browser yang memungkinkan pengembang untuk menyimpan data sementara secara local dan berbentuk *String*. Dengan menggunakan *Local Storage*, pengembang dapat menyimpan informasi dari pengguna seperti preferensi pengguna, status aplikasi atau data lain yang diperlukan untuk berjalan nya *session* pada aplikasi [11]. *Local Storage* bersifat *persist*, yang artinya data akan tersimpan secara permanen dalam browser pengguna

hingga pengguna sendiri menghapusnya atau melakukan pembersihan *cache & cookies* pada browser.

Penggunaan *Local Storage* dalam aplikasi ekstensi dapat mempermudah dalam hal mengelola data tanpa perlu mengirim *request* pada server. Hal ini dapat menunjang kinerja aplikasi dan meningkatkan pengalaman kepada pengguna. Namun, *Local Storage* memiliki Batasan ukuran yang bergantung pada browser yang pengguna pakai, sehingga *Local Storage* hanya bisa menyimpan data dalam ukuran kecil saja.

2.7 JSON

JSON (*Javascript Object Notation*) merupakan format data ringan yang digunkanan untuk pertukaran data antara aplikasi dengan suatu server. JSON sering digunakan dalam pengembangan aplikasi ekstensi untuk mengirim data dari browser ke server maupun dari server ke browser. JSON memiliki dua struktur utama, yaitu:

1. JSON Array

   JSON *Array* adalah kumpulan nilai dipisah oleh koma (,) dan diapit oleh tanda kurung siku ([]). Nilai dalam *array* berupa *varchar*. Contohnya seperti:

```
[ 1, 2, 3, "hai”, {"variabel": "nilai"}, null ]
```

2. JSON Object

   JSON *Object* adalah kumpulan nilai berpasangan yang dipisahkan oleh koma (,) dan diapit oleh tanda kurung kurawal ({}). Setiap pasangan nilai terdiri dari nama nilai (dalam bentuk *String*) dan nilai itu sendiri. Contohnya seperti:

```
{ "nama": "John", "umur": "23", "mahasiswa_aktif": true, "alamat": { "jalan":
                  "Jl. Lombok No. 4", "kota": "Bandung"} }
```

JSON memungkinkan data agar terstruktur secara rapih dan mudah untuk dibaca oleh manusia ataupun komputer. Penggunaan JSON pun sudah sangat umum digunakan dalam pengembangan aplikasi baik web ataupun ekstensi karena

kesederhanaannya dan kemampuannya dalam mempresentasikan berbagai jenis data dengan baik.

2.8 Framework Chrome Extension

Framework Chrome Extension merupakan kumpulan alat dan aturan yang membantu pengembang dalam membuat suatu aplikasi berbasis *Chrome Extension* secara efisien. *Framework* ini memberikan struktur dasar yang dapat pengembang gunakan untuk meng-akses API pada browser yang pengguna pakai [10]. *Framework* ini juga memiliki kemampuan untuk mengelola tampilan antarmuka pengguna, menyimpan data, dan melakukan berbagai macam tugas lain yang diperlukan dalam pengembangan ekstensi. Dengan mengandalkan *Framework Chrome Extension,* pengembang dapat mengurangi waktu dan usaha untuk mengembangkan aplikasi *Chrome Extension,* karena banyak fungsi dasar yang disediakan pada *framework* ini. Berikut ini merupakan struktur utama dari *framework*:

1. Manifest File

   Manifest File dalam pengembangan aplikasi *chrome extension* bisa dianggap sebagai inti dari konfigurasi awal suatu aplikasi seperti nama aplikasi, versi, ikon, izin, dll.

2. Background Script

   Background Script dapat diimplementasikan menggunakan OOP, dimana pengembang dapat membuat *class* untuk mengatur logika yang berjalan di latar belakang, memantau aksi yang dilakukan oleh pengguna maupun secara otomatis dan mengelola status.

3. Content Script

   Content Script digunakan untuk memanipulasi halaman web yang pengguna akses dan mengelola interaksi dengan halaman.

4. Popup atau Option Pages

Popup atau *Option Pages* digunakan untuk menambahkan antarmuka untuk pengguna dan terdapat logika di baliknya. *Option Pages* pun biasa digunakan pengguna untuk *easy access*.

Selain itu, *Framework Chrome Extension* juga memberikan kemudahan dalam mengakses fitur pada browser yang tidak secara langsung diberikan ke pengguna, seperti mengelola tab, menyimpan data secara lokal dan melakukan komunikasi ke server diluar browser [10]. Keuntungan menggunakan *Framework Chrome Extension* ini adalah mempercepat proses pengembangan pada suatu aplikasi *Chrome Extension,* mengurangi tingkat kerumitan pada kode, dan menjaga konsistensi dan kepatuhan terhadap standar yang sudah diberikan oleh *Google* selaku penyedia *Framework* ini. Dengan memiliki struktur yang tertata, pengembang dapat lebih fokus dalam pengembangan fitur pada ekstensi yang unik dan meningkatkan produktifitas secara keseluruhan.

2.9 Definisi E-mail Penipuan

E-mail penipuan, sering disebut sebagai *phishing*, adalah metode penipuan yang dilakukan melalui email dengan tujuan mendapatkan informasi sensitif dari penerima. Pelaku *phishing* biasanya menyamar sebagai entitas yang sah, seperti bank, perusahaan teknologi, atau layanan online terpercaya, untuk memancing korban agar memberikan informasi pribadi, seperti kata sandi, nomor kartu kredit, atau detail akun lainnya. E-mail penipuan biasanya dirancang agar tampak meyakinkan, menggunakan logo, bahasa, dan format yang menyerupai email asli dari organisasi yang mereka klaim sebagai pengirim. Tindakan ini dapat mengarah pada pencurian identitas, akses tidak sah ke akun, atau kerugian finansial yang signifikan bagi korban.

E-mail penipuan memiliki beberapa ciri khas yang dapat diidentifikasi, baik dari teks, tautan URL, domain pengirim, maupun file lampiran dalam email tersebut:

1. Karakteristik Penulisan

Email penipuan sering kali menggunakan bahasa yang mendesak, seperti "Akun Anda akan ditutup dalam 24 jam!" atau "Tindakan segera diperlukan."

Taktik ini dimaksudkan untuk memancing kepanikan pada penerima sehingga mereka segera merespons tanpa berpikir panjang. Selain itu, Email penipuan mungkin menawarkan hadiah besar atau kesempatan finansial yang luar biasa, seperti "Anda memenangkan undian senilai $1.000.000!" Tawaran ini biasanya disertai dengan permintaan untuk mengklik tautan atau memberikan informasi pribadi untuk mengklaim hadiah tersebut.

2. Tautan URL yang mencurigakan

Email penipuan sering kali menyertakan tautan yang tampaknya sah tetapi sebenarnya mengarahkan penerima ke situs web phishing. Situs web ini dirancang untuk meniru situs resmi dan meminta korban untuk memasukkan informasi pribadi. Tautan ini mungkin terlihat asli, tetapi jika diperiksa lebih lanjut, tautan tersebut bisa memiliki perbedaan kecil atau menggunakan domain yang mencurigakan.

3. Domain pengirim termasuk dalam list *phishing*

Domain pengirim dalam email penipuan mungkin mirip dengan domain perusahaan yang sah tetapi dengan perbedaan kecil, seperti penggunaan huruf yang hampir mirip (misalnya, "paypa1.com" bukannya "paypal.com"). Selain itu, aplikasi ini memeriksa apakah domain pengirim terdaftar dalam database phishing seperti PhishTank, yang berisi daftar domain yang diketahui digunakan untuk phishing.

4. *File attachment* yang berbahaya

Email penipuan sering kali menyertakan file attachment yang mengandung *malware* atau *ransomware*. File ini mungkin diberi nama yang terlihat sah, seperti "Invoice_12345.pdf", tetapi jika dibuka, file tersebut dapat menginfeksi perangkat dengan program berbahaya yang mencuri data atau mengenkripsi file korban untuk mendapatkan tebusan.

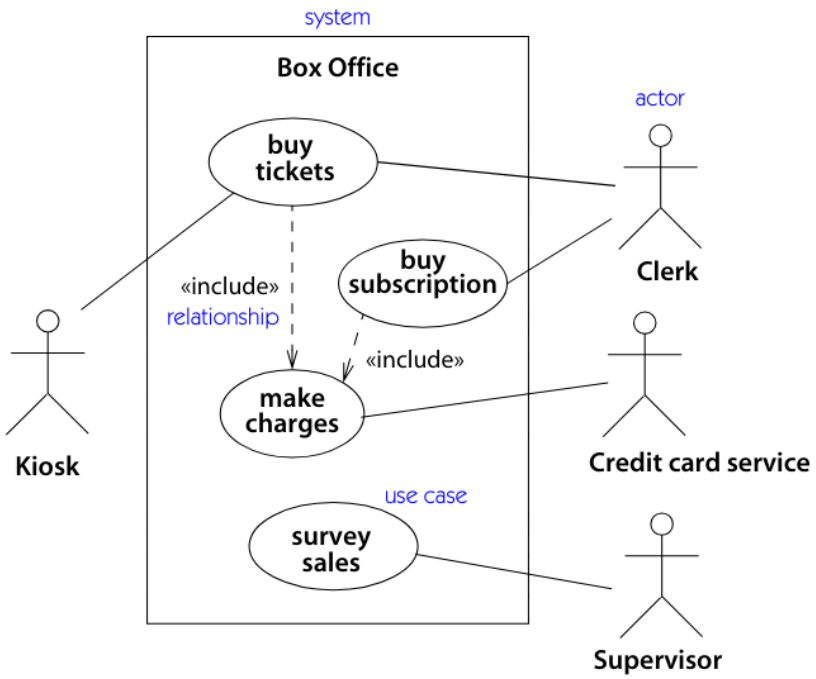
2.10 UML (Unified Modeling Language)

UML merupakan sebuah bahasa visual yang digunakan untuk mendesain dan memodelkan system aplikasi yang berbasis objek. UML memberikan standar dalam menggambarkan struktur, perilaku, dan interaksi antar komponen dalam

sebuah perangkat lunak atau aplikasi. UML juga dapat digunakan sebagai alat transfer ilmu mengenai sebuah pernagkat lunak yang akan atau sedang dikembangkan oleh satu developer ke developer yang lainnya.UML sangat penting untuk Sebagian orang karena berfungsi sebagai *bridge* atau jembatan penerjemah antara pengembang system dengan pengguna perangkat lunak atau aplikasi [13].

2.10.1 Use Case Diagram

Use case diagram merupakan bagian dari diagram UML yang menggambarkan kegiatan atau interaksi dari aktor (pengguna) dengan sistem dari perangkat lunak atau aplikasi. Diagram ini membantu untuk memahami fungsionalitas sistem namun dalam bahasa yang ringan. dan melalui sudut pandang pengguna. Dalam aplikasi *chrome extension* pendeteksi E-mail penipuan, use case diagram dapat menggambarkan berbagai aksi dan scenario yang dapat dilakukan oleh pengguna saat menggunakan aplikasi *chrome extension* pendeteksi E-mail penipuan. Seperti, Use case diagram dapat menggambarkan langkah-langkah atau interaksi dari pengguna disaat pengguna melakukan aksi analisa *file attachment* yang terdapat pada *body* E-mail yang pengguna buka.

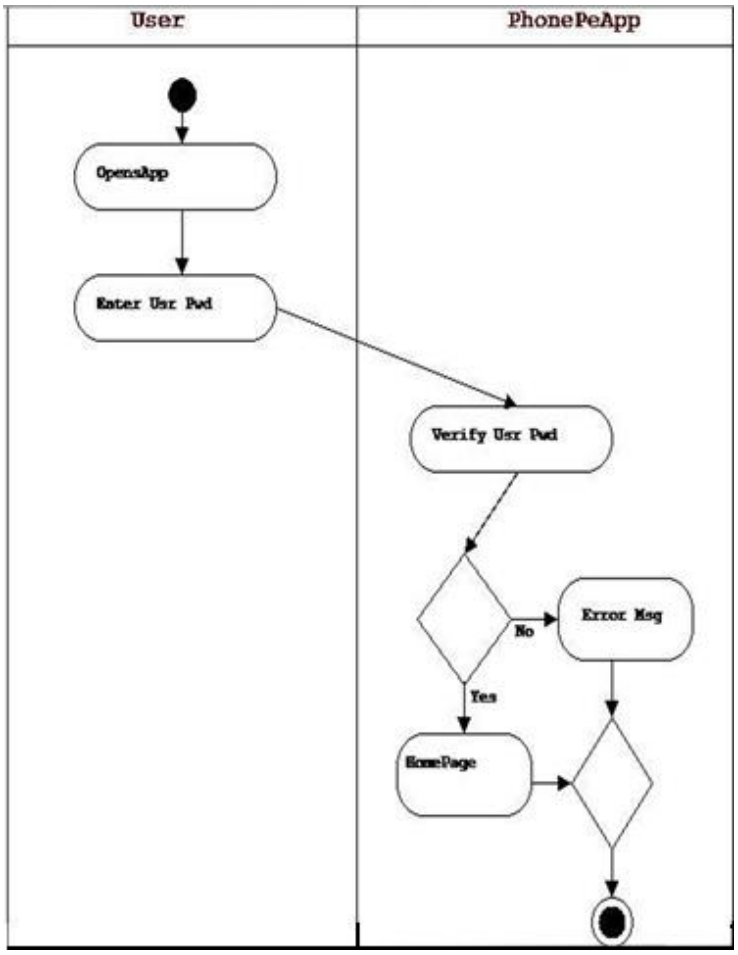


Sumber: *The unified modeling language reference manual* [16].

Gambar 2.10 Contoh Use Case Diagram

2.10.2 Activity Diagram

Activity diagram merupakan salah satu berbagai jenis diagram UML yang berguna sebagai penggambaran alur kerja atau aktifitas yang terjadi dalam suatu proses atau fungsi dalam perangkat lunak atau aplikasi. Diagram ini berguna untuk memodelkan proses bisnis, algoritma, atau logika program yang kompleks. Dalam aplikasi *chrome extension* pendeteksi E-mail penipuan, activity diagram dapat menggamabrkan langkah-langkah proses yang terjadi ketika aplikasi menerima *response* dari suatu API seperti VirusTotal API atau Google Translate API. Diagram ini dapat menunjukan bagaimana data diterima oleh sistem, bagaimana sistem mengirim data ke server, hingga bagaimana sistem mengolah data yang diterima sehingga menjadi informasi yang bersifat *user friendly*.

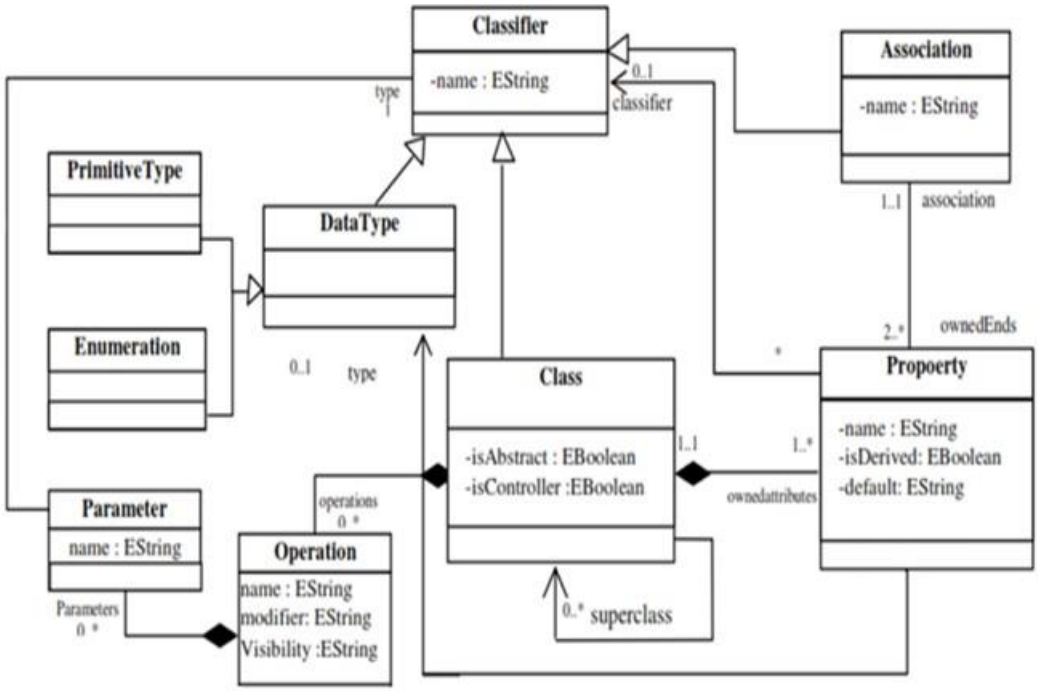


Sumber*: Novel approach to transform UML Sequence diagram to Activity diagram* [17].

Gambar 2.11 Contoh Activity Diagram

2.10.3 Class Diagram

Class Diagram merupakan salah satu dari berbagai jenis diagram UML yang berguna sebagai penggambaran struktur yang bersifat statis dari sistem, termasuk class-class yang terdapat pada sistem, relasi antar class, atribut class, dan metode yang dimiliki oleh class tersebut. Diagram ini sangat membantu dalam memahami struktur data dan hubungan antar kelas dalam sistem. Dalam aplikasi *chrome extension* pendeteksi E-mail penipuan, class diagram dapat menggambarkan struktur class yang terdapat dalam sistem. Seperti class untuk mengatur *session* aplikasi, class untuk mengatur komunikasi antara sistem dengan server API, dan class untuk mengatur sistem melakukan analisa terhadap suatu E-mail.



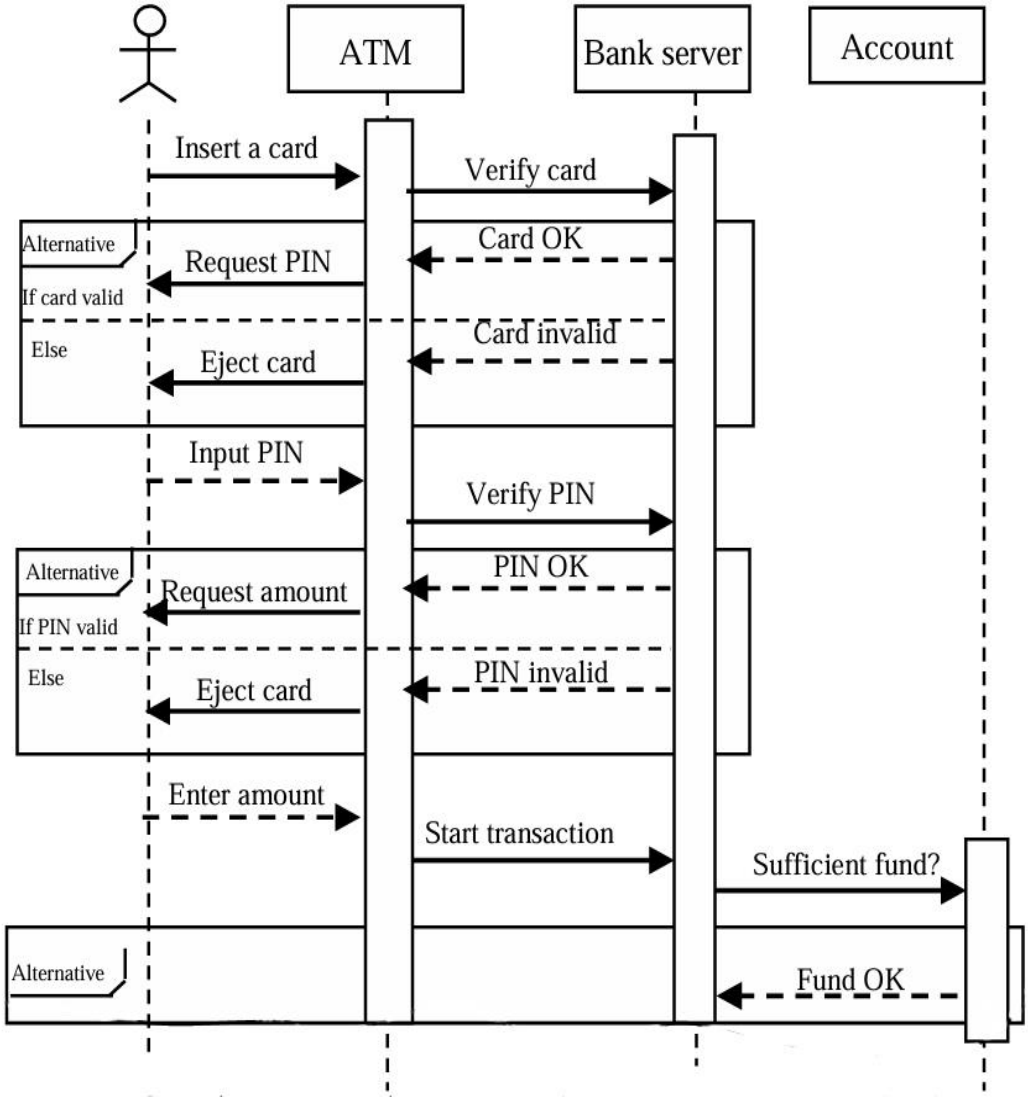
Sumber: *The unified modeling language reference manual* [16].

Gambar 2.12 Contoh Class Diagram

2.10.4 Sequence Diagram

Sequence Diagram merupakan salah satu dari berbagai jenis diagram UML yang berguna untuk menggambarkan urutan pesan atau interaksi antar objek dalam sistem selama satu proses atau dalam skenario tertentu. Diagram ini berguna untuk memodelkan bagaiman objek-objek dalam sistem berinteraksi satu sama lain dalam

suatu proses tertentu. Dalam aplikasi *chrome extension* pendeteksi E-mail penipuan, sequence diagram dapat meggambarkan interaksi antara komponen-komponen utama dalam perangkat lunak. Seperti, saat *file attachment* yang terdapat pada E-mail melakukan interaksi dengan VirusTotal API dan teks pada *body* E-mail melakukan interaksi dengan Google Translate API yang nanti nya akan diteruskan dengan *Text Mining.*



Sumber: *UML Sequence Diagram: An Alternative Model* [18].

Gambar 2.13 Contoh Sequence Diagram

2.11 Pengujian

Dalam konteks pengujian aplikasi *chrome extension* pendeteksi E-mail penipuan, pengujian Alpha dan pengujian Beta akan dilakukan untuk mengetahui

jika aplikasi yang dibangun memang memenuhi syarat dari sengi fungsionalitas dan juga untuk mengetahui tingkat kepuasan pengguna dalam menggunakan aplikasi yang akan dibangun.

Blackbox pada pengujian Alpha tidak kalah penting dalam menguji suatu aplikasi. Blackbox testing dilakukan tanpa diharuskan untuk melihat struktur dari aplikasi, melainkan hanya berfokus pada fungsionalitas pada aplikasi [15]. Pengujian ini hanya berfokus pada keberhasilan suatu fungsionalitas untuk mendeteksi E-mail penipuan tanpa harus memperhatikan implementasi lainnya seperti *text processing* atau integrasi ke API.

Pada pengujian beta, skala Likert digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang tentang fenomena sosial. Dalam penelitian, fenomena sosial ini telah ditetapkan secara spesifik oleh peneliti, yang selanjutnya disebut sebagai variable penelitian. Dengan skala Likert, maka variable yang akan diukur dijabarkan menjadi indikator variable. Kemudian indikator tersebut dijadikan sebagai titik tolak untuk menyusun item-item instrumen yang daptat berupa pernyataan atau pertanyaan [19].

Jawaban dari setiap item instrumen yang menggunakan skala Likert mempunyai gradasi dari sangat positif sampai sangat negatif, yang dapat berupa kata-kata antara lain:

a. Sangat Setuju
b. Setuju
c. Ragu-ragu
d. Tidak setuju
e. Sangat tidak setuju

Untuk keperluan kuantitatif, maka jawaban itu dapat diberi skor, misalnya:

a.	Sangat setuju/selalu/sangat positif diberi skor	5
b.	Setuju/sering/positif diberi skor	4
c.	Ragu-ragu/kadang-kadang/netral diberi skor	3
d.	Tidak setuju/hampir tidak pernah/negatif diberi skor	2
e.	Sangat tidak setuju/tidak pernah diberi skor	1

Kemudian dengan teknik pengumpulan data angket, maka instrumen tersebut misalnya diberikan kepada 100 orang karyawan yang diambil secara acak. Dari 100 orang pegawai setelah dilakukan analisis misalnya:

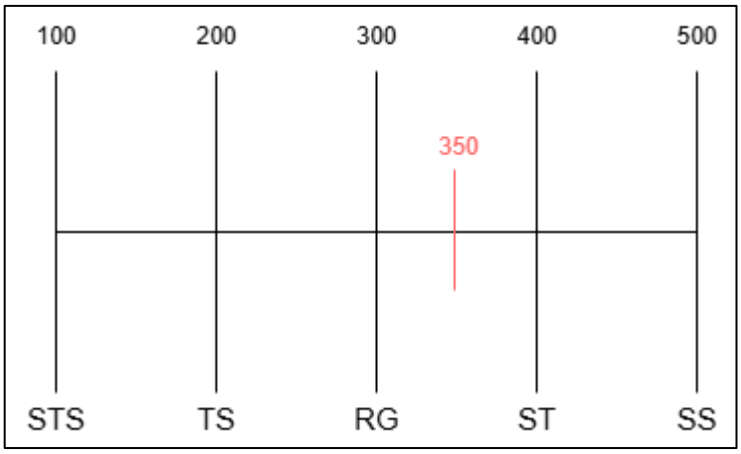
25	Orang menjawab	SS (Sangat Setuju)
40	Orang menjawab	ST (Setuju)
5	Orang menjawab	RG (Ragu-ragu)
20	Orang menjawab	TS (Tidak setuju)
10	Orang menjawab	STS (Sangat tidak setuju)

Data interval tersebut dapat dianalisis dengan menghitung rata-rata jawaban berdasarkan skoring setiap jawaban dari responden. Berdasarkan skor yang telah ditetapkan dapat dihitung sebagai berikut.

Jumlah skor untuk 25 orang yang menjawab SS	=	25 x 5	=	125
Jumlah skor untuk 40 orang yang menjawab ST	=	40 x 4	=	160
Jumlah skor untuk 5 orang yang menjawab RG	=	5 x 3	=	15
Jumlah skor untuk 20 orang yang menjawab TS	=	20 x 2	=	20
Jumlah skor untuk 10 orang yang menjawab STS	=	10 x 1	=	10
Jumlah total			=	350

Jumlah skor ideal (kriterium) untuk seluruh item = 5 x 100 = 500 (seandainya semua menjawab SS). Jumlah skor yang diperoleh dari penelitian = 350. Jadi berdasarkan data itu maka tingkat persetujuan terhadap metode kerja baru itu = (350 : 500) x 100% = 70% dari yang diharapkan (100%).

Secara kontinum dapat digambarkan seperti berikut:



Gambar 2.14 Contoh interval rating scale