PENGEMBANGAN BACK-END DAN PERANCANGAN API DOCS WEBSITE THINK ACTION

Alvon Jovanus Chandra[#1], Robby Tan[*2]

[#]*Progam Studi Sarjana Teknik Informatika, Fakultas Teknologi Infomrasi, Universitas Kristen Maranatha*
Jln. Prof. Drg. Surya Sumantri No. 65, Sukawarna, Bandung, Indonesia
[1]2072011@maranatha.ac.id

[*]*Progam Studi Sarjana Teknik Informatika, Fakultas Teknologi Infomrasi, Universitas Kristen Maranatha*
Jln. Prof. Drg. Surya Sumantri No. 65, Sukawarna, Bandung, Indonesia
[2]robby.tan@it.maranatha.edu

Abstract — **This report discusses the back-end development and API documentation design for the goal tracking and employee attendance system. The goal tracking system is called Think Action, which can help users to track long-term and short-term goals automatically. The employee attendance system is called Pin Point, which can facilitate employees to take attendance anywhere using the map feature. This system is built using Node.js, Express, TypeScript, MongoDB, and Clean Architecture technologies. The functionality of the system was tested through unit and E2E testing, ensuring that every aspect of the system functions properly and as expected. With a focus on clarity and accessibility, the API documentation developed enables easy and effective integration for developers, especially in front-end development.**

Keywords— **API documentation, back-end development, employee attendance system, goal tracking system, testing**

I. PENDAHULUAN

Di era digital yang serba cepat dan penuh persaingan ini, mengatur dan mencapai tujuan/*goals* dalam hidup menjadi sangat penting. Ketika individu memiliki tujuan yang jelas, individu tersebut akan lebih termotivasi untuk bekerja keras dan mencapai hasil yang lebih baik. Misalnya, tujuan dalam bidang *fitness*, seperti mencapai berat badan ideal dalam waktu 6 bulan, dan tujuan dalam bidang *finance*, seperti menabung untuk membeli rumah dalam waktu 2 tahun. Hal ini akan berdampak positif pada produktivitas dan kinerja individu tersebut dalam meraih tujuannya. Namun, kesulitan sering muncul dalam melacak tujuan jangka panjang akibat penggunaan sistem masih manual dan kurang efisien. Misalnya, invididu yang ingin mencapai resolusi tahunannya perlu menulis dan mengingat sendiri apa yang telah dilakukan setiap minggu atau bulan. Hal ini tentu saja merepotkan dan berisiko membuat lupa atau kehilangan motivasi.

Untuk mengatasi permasalahan tersebut, dibangunlah sistem *tracking goals* yang dapat melacak tujuan jangka panjang dan jangka pendek secara otomatis. Sistem ini bernama Think Action, yaitu sebuah *website* yang memungkinkan pengguna untuk membuat *posts* mengenai tujuan jangka panjang, tujuan jangka pendek, dan tujuan yang telah selesai. Pengguna juga dapat menerima notifikasi mengenai deadline dari tujuan mereka melalui Google Calendar. Selain itu, pengguna juga dapat memberikan *like* dan *comment* pada *posts* orang lain, serta melakukan *support* atau *unsupport* terhadap pengguna lain. Dengan demikian, pengguna dapat melacak tujuan mereka dengan lebih mudah, efektif, dan interaktif.

Permasalahan berikutnya adalah mengenai presensi karyawan. Presensi merupakan salah satu kegiatan penting yang dilakukan oleh karyawan di setiap perusahaan. Presensi dapat digunakan sebagai bukti kehadiran, kinerja, dan tanggung jawab karyawan. Namun, presensi karyawan seringkali mengalami kendala, seperti kesulitan dalam mengakses sistem presensi, ketidaksesuaian lokasi presensi dengan lokasi kerja, atau kesalahan dalam pencatatan data presensi. Hal ini dapat berdampak negatif bagi karyawan maupun perusahaan, seperti terjadinya kesalahan dalam pembayaran, penilaian, atau pengawasan karyawan.

Solusi untuk masalah tersebut adalah membuat sebuah sistem yang dapat memfasilitasi karyawan untuk melakukan presensi dimanapun. Sistem tersebut bernama Pin Point, website yang dapat diakses melalui perangkat *mobile* maupun *desktop*. Pin Point menggunakan fitur map untuk menampilkan lokasi karyawan secara *real-time*. Karyawan dapat mencatatkan presensi dengan mengisi form yang tersedia pada *website* dan wajib mengirimkan foto sebagai bukti kehadiran. Setiap karyawan dapat melihat laporan presensi masing-masing. Dengan demikian, karyawan dapat melakukan

presensi dengan mudah dan efektif.

Pengembangan *back-end website* Pin Point dan Think Action dibuat dengan menggunakan teknologi yang stabil dan modern, yaitu Node.js, Express, TypeScript, MongoDB, dan Clean Architecture. Dengan menggunakan teknologi-teknologi tersebut website ini akan memberikan pengalaman pengguna yang baik dalam melakukan presensi. Seluruh autentikasi dan otorisasi pada API dibuat menggunakan JWT. Selain itu, website ini juga akan mengimplementasikan *unit testing* dan End-to-End *testing* untuk memastikan keamanan dan konsistensi fungsionalitasnya. Dengan melakukan tes secara teratur, pengembang dapat memastikan bahwa setiap fitur berjalan sesuai yang diharapkan.

Rumusan masalah laporan ini diambil dari kedua proyek tersebut, yaitu bagaimana merancang dokumentasi API yang baik dan sesuai untuk sistem *tracking goals* dan presensi karyawan, bagaimana cara menerapkan prinsip clean architecture dalam pengembangan *back-end*, dan bagaimana cara mengimplementasikan *unit testing* dan End-to-End *testing* dalam pengembangan *back-end*.

II. PROFIL PERUSAHAAN

Cataliz Indonesia merupakan perusahaan yang bergerak di bidang *coaching*, *training*, dan *mentoring*. Perusahaan ini bertujuan untuk membantu individu dan bisnis untuk mencapai tujuan dengan mengembangkan keterampilan yang diperlukan. Cataliz menyediakan program *Career Journey for IT Talents*, yaitu program magang yang dikhususkan untuk mahasiswa yang memiliki kompetensi di bidang IT untuk mewujudkan kemandirian berkarir menjadi profesional IT Talents. Program magang ini tersedia dalam 2 jalur yaitu *front-end* dan *back-end*. Seluruh peserta akan mendapatkan pelatihan *basic skill* seperti pelatihan mengenai *human development* dan *project based learning*. Setelah menyelesaikan pelatihan yang diberikan, peserta akan diberikan project industri dari *client*.

III. LANDASAN TEORI

A. API

Application Programming Interface atau yang biasa disingkat menjadi API merupakan sekumpulan aturan dan protokol yang memungkinkan berbagai perangkat lunak atau komponen sistem untuk berinteraksi satu sama lain. API menentukan cara bagaimana komponen-komponen ini dapat berkomunikasi, dan biasanya digunakan oleh pengembang perangkat lunak untuk mengakses fungsionalitas atau data dari aplikasi atau layanan lain, dengan kata lain API memungkinkan aplikasi untuk saling berbagi data, fitur, layanan, atau sistem operasi.

Keberhasilan API sangat bergantung pada dokumentasinya untuk memenuhi kebutuhan informasi bagi pengembang perangkat lunak [1]. Pembuatan dokumentasi API yang buruk akan menjadi kendala utama dalam pembelajaran API tersebut sehingga pengembang tidak dapat memenuhi kebutuhan informasi API. Dokumentasi API yang baik harus memenuhi kebutuhan pengembang, baik yang berorientasi pada konsep maupun kode, dengan menyediakan informasi tentang struktur API dan alasan di balik keputusan desainnya serta memungkinkan pengembang untuk mulai bekerja dengan kode secepat mungkin.

B. Git

Git adalah sebuah perangkat lunak yang digunakan untuk mengatur versi kode sumber program dengan cara menandai baris dan kode yang perlu diganti atau ditambahkan. Git diciptakan oleh Linus Torvalds pada tahun 2005 untuk mengembangkan kernel Linux. Git masuk dalam kategori Distributed Version Control System (DVCS), yang artinya setiap pengembang memiliki salinan lengkap dari repositori kode, termasuk riwayat perubahan.

Semua perangkat lunak yang digunakan untuk analisis harus didokumentasikan dengan baik, lebih baik lagi jika dibagikan secara terbuka dan dapat diakses langsung oleh orang lain [2]. Salah satu cara mendokumentasikannya yaitu menggunakan Git. Git semakin berkembang seiring munculnya perangkat lunak yang tersedia secara open-source dengan mendukung beberapa layanan penyimpanan kode seperti Github, SourceForge, Bitbucket, dan GitLab.

C. TypeScript

TypeScript adalah sebuah bahasa pemrograman yang didasarkan pada JavaScript, tetapi menambahkan fitur-fitur baru seperti strong-typing dan konsep pemrograman OOP (*class*, *interface*). TypeScript dibuat oleh Microsoft untuk mempermudah pengembangan aplikasi yang lebih besar dan kompleks. TypeScript disebut sebagai *super-set* dari JavaScript, yang artinya semua kode JavaScript juga bisa dijalankan sebagai kode TypeScript.

JavaScript sering menjadi bahasa yang kurang ideal untuk mengembangkan dan memelihara aplikasi besar, TypeScript bertujuan untuk mengatasi kekurangan ini [3]. Salah satu keuntungan utama TypeScript yaitu meningkatkan kualitas kode untuk proyek yang lebih besar dan kompleks, seperti meningkatkan keamanan kode, bantuan otomatisasi dari alat pengembangan, dan dokumentasi yang lebih baik. TypeScript juga memungkinkan pengembang untuk menggunakan fitur-fitur bahasa terbaru yang mungkin belum didukung oleh semua *web browser*, karena kode TypeScript akan dikompilasi

menjadi JavaScript yang kompatibel dengan berbagai *web browser*. Dengan ekosistem besar yang mendukungnya dan komunitas pengembang yang aktif, TypeScript telah menjadi pilihan yang populer dalam pengembangan aplikasi web dan aplikasi berbasis Node.js.

Pengembangan dengan TypeScript biasanya menggunakan kompilasi di mana kode TypeScript dikonversi menjadi kode JavaScript. TypeScript memanfaatkan file konfigurasi TypeScript untuk mengatur aturan kompilasi. Selain itu, TypeScript mendukung definisi tipe yang dapat digunakan untuk menggambarkan struktur data yang kompleks, sehingga dapat membantu dalam pengembangan kode yang lebih aman dan terstruktur [4].

Berikut ini merupakan contoh kode untuk menjumlahkan dua bilangan dalam bahasa Typescript:

```
1.    // Sum 2 numbers function
2.    function sum(a: number, b: number): number {
3.      return a + b;
4.    }
5.
6.    // Declare variables
7.    const num1: number = 10;
8.    const num2: number = 20;
9.
10.   // Save the result
11.   const result: number = num1 + num2;
12.
13.   // Log the result  console.log(result);
```

Kode Program 1 Contoh kode program dalam TypeScript

D. Node.js

Node.js adalah lingkungan runtime JavaScript yang bersifat *open source* dan berbasis pada mesin JavaScript V8 dari Google Chrome. Node.js memungkinkan pengembang untuk menjalankan kode JavaScript di luar browser. Dengan Node.js, pengembang dapat membuat aplikasi web dan *server-side* yang *real-time* dan *scalable*. Salah satu keunggulan utama Node.js adalah model pemrograman berbasis peristiwa (*event-driven*) dan *non-blocking* I/O, yang berarti kode tidak akan memblokir eksekusi saat menunggu operasi I/O selesai, sehingga membuat aplikasi menjadi ringan dan efisien [5].

Node.js memiliki berbagai modul internal yang dapat digunakan untuk berbagai tugas seperti mengelola *file*, melakukan permintaan HTTP, dan berkomunikasi dengan basis data. Selain itu, Node.js memiliki *package manager* bawaan yang disebut npm (Node Package Manager), yang memungkinkan pengembang untuk dengan mudah menginstal, mengelola, dan membagikan paket JavaScript, baik yang dibuat oleh sendiri maupun oleh komunitas.

Node.js menunjukkan performa yang jauh lebih baik daripada PHP dalam situasi *concurrency* tinggi, baik dalam tes benchmark maupun tes skenario [6]. PHP cocok untuk permintaan kecil tetapi mengalami kesulitan dengan permintaan besar. Node.js sangat cocok digunakan dalam situasi yang memerlukan I/O yang intensif, bukan situs yang memerlukan pemrosesan yang intensif. Dengan semua keunggulan ini, Node.js telah menjadi salah satu pilihan utama dalam pengembangan aplikasi web dan *server-side*, serta digunakan oleh banyak perusahaan di seluruh dunia.

E. Express

Express, atau yang sering disebut Express.js, adalah sebuah *framework web open source* yang dibuat dengan menggunakan bahasa pemrograman JavaScript. Express berjalan di atas Node.js, sebuah platform yang memungkinkan JavaScript untuk dijalankan di luar browser. Express dirancang secara fleksibel dan minimalis, sehingga pengembang dapat membuat aplikasi web dan *server-side* yang *real-time* dan *scalable*.

Salah satu fitur utama dari Express adalah pendekatan yang ringan dan minimalis dalam desainnya. Ini berarti Express memberi pengembang banyak kebebasan (*unopinionated*) untuk memilih alat dan modul tambahan yang akan digunakan dalam proyek [7]. Dengan cara ini, Express memungkinkan pengembang untuk membangun aplikasi web yang sesuai dengan kebutuhan tanpa harus membawa beban berlebihan dari fitur yang tidak digunakan.

Express sangat cocok untuk membangun berbagai jenis aplikasi web, termasuk aplikasi web dan REST API. Express menyediakan banyak fungsi tambahan di atas modul HTTP yang tidak perlu ditulis ulang dari awal untuk tugas umum dalam menangani *request*, *route*, dan *response*. Selain itu, Express juga memiliki dukungan yang kuat untuk middleware, yang memungkinkan pengembang untuk menambahkan fungsionalitas tambahan, seperti *authentication*, *authorization* dan *session management* [8].

Berikut ini merupakan contoh implementasi kode untuk membuat aplikasi menggunakan Node.js dan Express:

```
import express from 'express';

// Create express application
const app = express();

// Defining port with default port = 3000
const port = process.env.port || 3000;

// Create route
app.get('/', (req, res) => {
  res.send('Hello World!');
});

// Listening to server
export const server = app.listen(port, () => {
  console.log('Server is running on port', port);  });
```

Kode Program 2 Contoh aplikasi Node.js dan Express

F. MongoDB

MongoDB merupakan salah satu sistem manajemen basis data (*Database Management System*) yang dirancang menggunakan pendekatan NoSQL (*Not Only SQL*), di mana NoSQL adalah sistem pengelolaan database yang fleksibel dan tidak membutuhkan perintah atau *query* yang rumit. MongoDB ini bersifat dokumen dan dirancang untuk menyimpan, mengelola, dan mengakses data. Dalam MongoDB, data disimpan dalam format BSON (Binary JSON), yang merupakan format dokumen yang fleksibel dan mudah diakses [9].

MongoDB memiliki kelebihan seperti kemampuan untuk mengelola volume data besar dengan fleksibilitas yang tinggi [10]. MongoDB juga menyediakan beragam fitur yang mendukung pengembangan website dan aplikasi, seperti integrasi dengan berbagai teknologi populer seperti Node.js, kemampuan mengelola query yang kuat dengan fitur agregasi, serta pencarian dan pemrosesan data yang efisien.

Berikut ini merupakan contoh kode program mengintegrasikan MongoDB dengan Node.js:

```
import { MongoClient } from 'mongodb';
// Link URI MongoDB
const uri =
'mongodb+srv://USERNAME:PASSWORD@CLUSTERURL';
// Create new MongoDB Client
const client = new MongoClient(uri,
    { useNewUrlParser: true }
);

// Create async function to run MongoDB
async function run() {
  try {
    // Connect to MongoDB Client
    await client.connect();

    // Access Database
    const database = client.db('database');
    // Access Collection
    const collection = database.collection('collection');

    // Perform database operations here
  } finally {
    await client.close();
  }
}

// Run the function  run()
```

Kode Program 3 Contoh integrasi MongoDB dan Node.js

G. Clean Architecture

Clean Architecture adalah sebuah pendekatan untuk membuat perangkat lunak agar kode mudah dipahami, diuji, dan dipelihara. Clean Architecture memisahkan berbagai lapisan fungsional dalam aplikasi dan menjaga agar setiap lapisan memiliki tanggung jawab yang jelas dan terdefinisi dengan baik. Tujuan utama dari pendekatan ini adalah untuk membuat sistem yang independen dari UI, *database*, *framework*, dan hal eksternal lainnya [11].

Dalam Clean Architecture, aplikasi dibagi menjadi beberapa lapisan atau komponen yang saling terkait. Lapisan-lapisan tersebut biasanya meliputi lapisan presentasi, lapisan bisnis, dan lapisan data. Lapisan presentasi adalah yang paling dekat dengan pengguna dan bertanggung jawab untuk mengelola antarmuka pengguna. Lapisan bisnis berisi logika bisnis utama aplikasi seperti model dan validasi, sedangkan lapisan data mengurus akses ke sumber data eksternal, seperti basis data.

Clean Architecture mempunyai beberapa manfaat seperti memberikan fleksibilitas untuk mengganti atau modifikasi *tools*, *framework*, dan *database* tanpa mempengaruhi logika bisnis [12]. Struktur pada Clean Architecture memungkinkan sistem berkembang seiring dengan kompleksitasnya. Fitur-fitur baru dapat ditambahkan dengan dampak minim pada kode yang sudah ada. Lapisan-lapisan terdapat pada Clean Architecture mempermudah melakukan testing pada aplikasi karena komponen sudah terbagi sesuai fungsinya.

H. Pengujian

Pengujian/*testing* dalam pemrograman adalah proses yang dilakukan untuk memeriksa apakah aplikasi sudah sesuai dengan persyaratan yang diharapkan atau belum. Proses ini melibatkan pemeriksaan komponen dalam sistem perangkat lunak menggunakan alat manual atau otomatis. Tujuan utama dari testing adalah untuk menemukan kesalahan atau masalah potensial dalam kode sebelum perangkat lunak tersebut digunakan secara aktif atau diterapkan dalam lingkungan produksi. Terdapat beberapa jenis pengujian seperti *unit testing* dan End-to-End *testing* (E2E *testing*).

Unit testing adalah jenis pengujian perangkat lunak yang dilakukan untuk menguji suatu bagian atau komponen perangkat lunak. Unit ini bisa berupa kode, fungsi, metode, prosedur, modul, atau objek tersendiri. Satu unit mengacu pada bagian terkecil dari kode yang dapat diuji secara terisolasi. Pengujian unit adalah faktor penting dalam pengembangan perangkat lunak karena bertujuan untuk memastikan setiap unit bekerja dengan benar dalam isolasi [13].

E2E *testing* adalah teknik pengujian yang menguji seluruh produk atau keseluruhan suatu perangkat lunak. E2E *testing* berfokus pada pengujian aplikasi dari awal hingga akhir dalam skenario yang meniru pengalaman pengguna [14]. Ini mencakup semua komponen aplikasi dan interaksi antara komponen-komponen tersebut, termasuk antarmuka pengguna, server, basis data, dan layanan lainnya.

I. JWT

JSON Web Token (JWT) adalah salah satu mekanisme untuk melakukan autentikasi dan otorasi. JWT bertindak sebagai pembawa token yang memungkinkan pengguna untuk membuktikan identitas mereka dan mengakses sistem atau layanan tertentu. Setelah pengguna berhasil masuk, server akan menghasilkan JWT yang berisi informasi tertentu tentang pengguna (seperti ID pengguna) dan mengirimkannya kembali ke pengguna. Pengguna kemudian akan menyertakan JWT ini dalam setiap permintaan berikutnya ke server, memungkinkan server untuk memverifikasi identitas pengguna dan memberikan akses yang sesuai.

Bagian penting dari JWT adalah tanda tangan digitalnya. Tanda tangan ini dihasilkan oleh server saat menciptakan JWT, menggunakan algoritma hash khusus yang menggabungkan *header* dan *payload* token dengan *secret key* yang hanya diketahui oleh *server* [15]. Ketika *server* menerima permintaan dengan JWT, server dapat memverifikasi tanda tangan ini untuk memastikan bahwa *token* tidak diubah sejak dibuat. Hal ini memungkinkan *server* untuk memverifikasi *token* (seperti ID pengguna), tanpa perlu menyimpan informasi sesi di *server* atau melakukan *query database* untuk setiap permintaan. Ini membuat JWT sangat berguna dalam skenario autentikasi berbasis *token*, seperti autentikasi API.

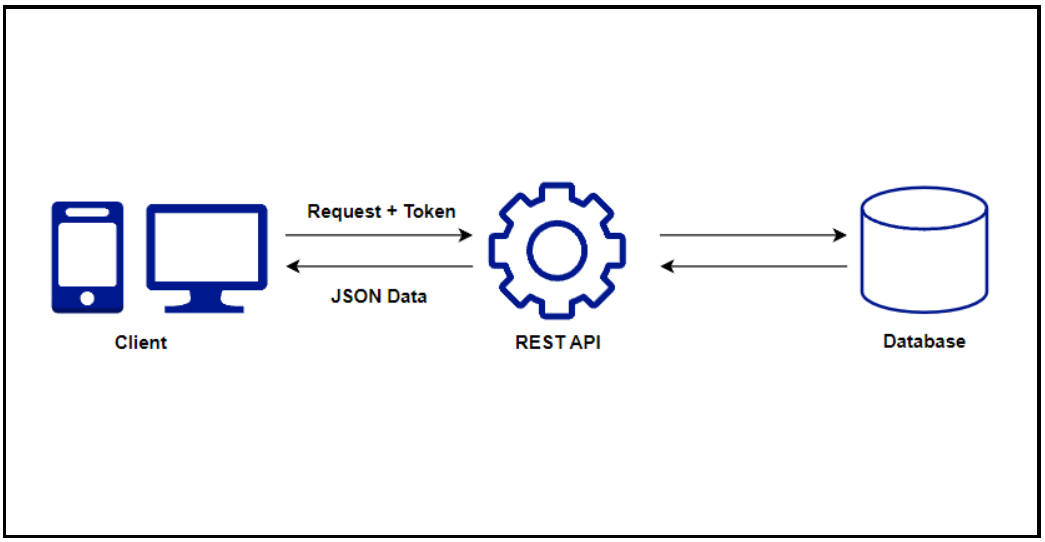
IV. HASIL PEKERJAAN

A. Hasil Project Pin Point

Pin Point adalah project pertama yang diberikan oleh perusahaan untuk dikerjakan. Project ini bertujuan memudahkan karyawan dalam melakukan presensi, baik di kantor maupun di luar kantor. Pembuatan bagian back end akan menggunakan teknologi-teknologi yang telah dipelajari. Berikut ini adalah penjelasan project Pin Point.

Project Pin Point bertujuan untuk memberikan kemudahan kepada karyawan dalam melakukan presensi di mana pun mereka berada. Proses presensi dapat dilakukan setelah karyawan diundang ke dalam grup oleh admin. Saat melakukan presensi, karyawan harus mengirimkan lokasi menggunakan fitur Google Map dan juga mengunggah foto sebagai bukti kehadiran. Karyawan hanya dapat melihat presensi mereka sendiri dan tidak memiliki akses untuk melihat presensi karyawan lain.

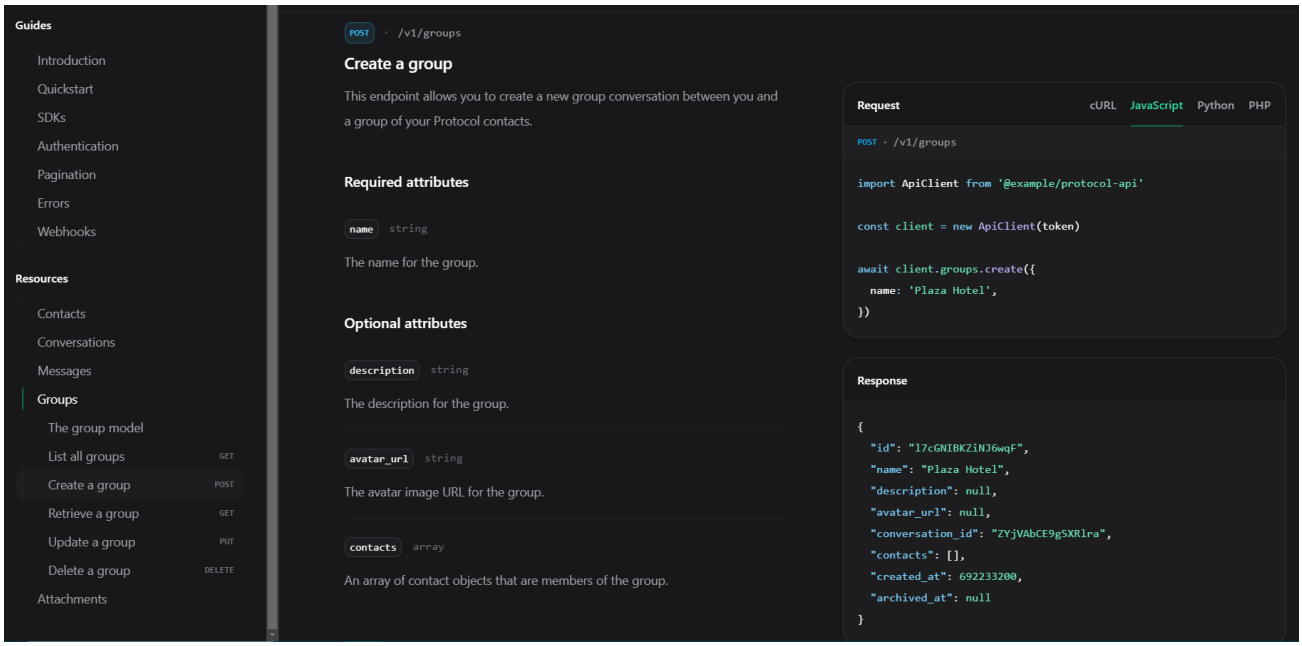
Pin Point tidak hanya bermanfaat bagi karyawan, tetapi juga bagi admin yang bertugas mengelola data presensi karyawan. Admin dapat mengundang dan menghapus karyawan yang terdaftar pada website, membuat dan mengubah data grup, serta melihat laporan presensi karyawan berdasarkan berbagai kriteria, seperti nama dan tanggal.



Gambar 1. Desain arsitektur sistem Pin Point

Gambar 1 merupakan desain arsitektur sistem untuk project Pin Point. Pada bagian pertama, *client* (dapat berupa *mobile* atau komputer) menggunakan Vue.js untuk *front-end* dan mengirimkan *request* ke REST API. Bagian kedua adalah REST API yang berfungsi sebagai penghubung antara *client* dan *database*, memproses permintaan dari *client* dan berinteraksi dengan *database*. Bagian ketiga adalah database MongoDB yang menyimpan dan mengambil data sesuai permintaan dari REST API. Alur komunikasi dua arah ditunjukkan oleh panah bolak-balik antara setiap elemen sistem. Ini menunjukkan bahwa setiap elemen dapat berkomunikasi dan bertukar informasi dengan elemen lainnya.

Gambaran atau contoh mengenai API Docs yang telah dibuat terdapat pada gambar berikut:



Gambar 2. Gambaran API Docs pada project

Gambar 2 merupakan gambaran mengenai API Docs yang dibuat pada project Pin Point dan Think Action. API Docs tersebut memiliki deskripsi lengkap dari setiap endpoint seperti metode HTTP yang diperlukan beserta fungsi dari endpoint

tersebut. Setiap endpoint memiliki penjelasan mengenai atribut yang diperlukan, serta contoh permintaan dalam bahasa JavaScript. Respons yang dihasilkan diilustrasikan dalam bentuk JSON untuk memberikan pengguna gambaran mengenai apa yang dihasilkan dari masing-masing endpoint.

Berikut ini merupakan rincian model dan endpoint dalam API Docs yang telah dibuat:

- Users

Entitas ini menjadi dasar untuk autentikasi dan manajemen pengguna dalam sistem. Terdapat beberapa endpoint yang telah diimplementasikan, seperti:

1. Update Current User: Endpoint ini berguna untuk mengubah informasi authenticated user.
2. Get All Users: Endpoint ini hanya dapat diakses oleh admin dan berguna untuk mendapatkan semua pengguna dengan opsi limit dan halaman.
3. Get One User: Endpoint ini hanya dapat diakses oleh admin dan berguna untuk mendapatkan informasi lengkap tentang satu pengguna berdasarkan id.
4. Update One User: Endpoint ini hanya dapat diakses oleh admin dan berguna untuk memperbarui informasi pengguna tertentu.
5. Delete One User: Endpoint ini hanya dapat diakses oleh admin dan berguna untuk menghapus pengguna tertentu berdasarkan id.

- Attendances

Entitas ini digunakan untuk melacak kehadiran pengguna dalam suatu grup. Endpoint-endpoint yang telah diimplementasikan adalah:

1. Create One Attendance: Endpoint ini berguna untuk membuat catatan kehadiran baru.
2. Get Current User Attendances: Endpoint ini berguna untuk mendapatkan riwayat kehadiran authenticated user dengan opsi limit, halaman.
3. Get All Attendances: Endpoint ini hanya dapat diakses oleh admin dan berguna untuk melihat semua kehadiran pengguna dengan opsi limit, halaman, rentang waktu, dan id user.
4. Get One Attendance: Endpoint ini hanya dapat diakses oleh admin dapat mengakses detail kehadiran tertentu berdasarkan id attendance.
5. Delete One Attendance: Endpoint ini hanya dapat diakses oleh admin dan berguna untuk menghapus catatan kehadiran tertentu berdasarkan attendance.

- Groups

Entitas ini merupakan grup untuk admin dan pengguna. Endpoint-endpoint yang telah diimplementasikan meliputi:

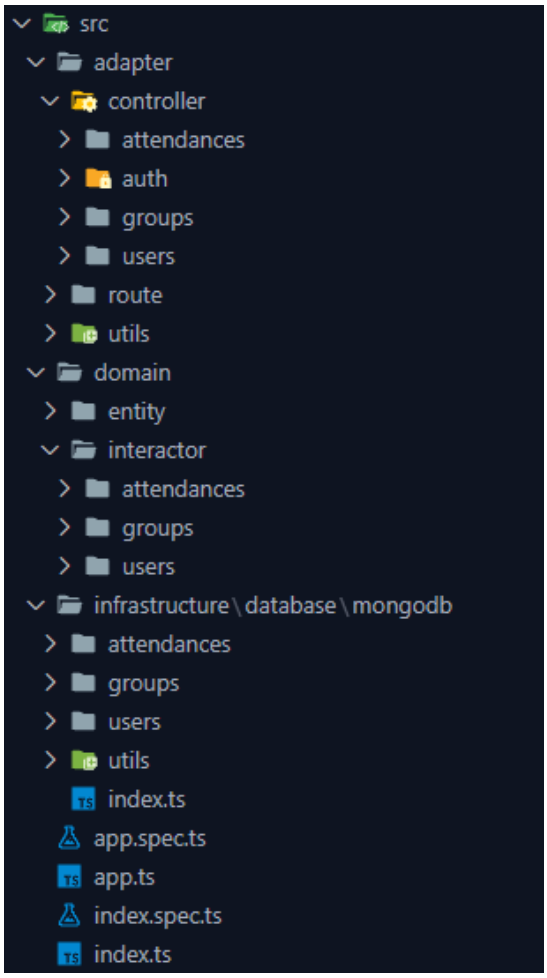
1. Get Groups that Invited Current User: Endpoint ini berguna untuk melihat grup yang mengundang authenticated user.
2. Update One Group: Endpoint ini hanya dapat diakses oleh admin dan berguna untuk memperbarui informasi grup.
3. Get All Groups: Endpoint ini hanya dapat diakses oleh admin dan berguna untuk mendapatkan semua grup dengan opsi limit dan halaman.
4. Create One Group: Endpoint ini hanya dapat diakses oleh admin dan berguna untuk membuat grup baru.
5. Get One Group: Endpoint ini hanya dapat diakses oleh admin dan berguna untuk mendapat detail grup tertentu berdasarkan id grup.
6. Delete One Group: Endpoint ini hanya dapat diakses oleh admin dan berguna untuk menghapus grup tertentu berdasarkan id grup.

- Authentication

Endpoint-endpoint ini terkait dengan proses authentication dan authorization. Beberapa endpoint yang telah diimplementasikan adalah:

1. Register: Endpoint ini berguna untuk mendaftar dengan menyediakan email, username, phone, dan password.
2. Login: Endpoint ini berguna untuk melakukan login dengan email dan password.
3. Logout: Endpoint ini berguna untuk logout dari sistem.
4. Update Current User Password: Endpoint ini berguna untuk memperbarui kata sandi authenticated user dengan menyediakan kata sandi saat ini dan kata sandi baru.

e-ISSN : 2443-2229



Gambar 3 Struktur folder project Pin Point

Gambar 3 adalah struktur folder yang terdapat pada project Pin Point. Direktori src adalah tempat di mana kode sumber aplikasi tersimpan. Semua kode yang terdapat dalam direktori src menggunakan bahasa Typescript, maka dari itu sebelum dijalankan, kode wajib di *compile* terlebih dahulu menjadi bahasa Javascript. Kode yang telah di *compile* akan masuk ke dalam folder dist yang nantinya akan menjadi runtime yang siap untuk dijalankan. Subdirektori yang terdapat pada src ada 3 yaitu adapter, domain, dan infrastructure. Berikut ini masing-masing penjelasan dari subdirektori tersebut:

- Adapter

Lapisan ini merupakan penghubung antara lapisan eksternal dengan aplikasi. Pada aplikasi ini, adapter berupa express.js. Subdirektori yang terdapat pada lapisan ini yaitu controller, route, dan utils. Berikut ini masing-masing penjelasan dari subdirektori tersebut:

  - Controller

    Direktori ini berisi kode yang mengendalikan aliran data antara pengguna dan sistem seperti *request* dan *response* API. Ada beberapa subdirektori seperti attendances, auth, groups, dan users yang masing-masing menangani entitas tertentu dalam aplikasi.
  - Route

    Direktori ini berisi kode yang menentukan rute atau jalur yang dapat diakses oleh pengguna.
  - Utils

    Direktori ini berisi kode utilitas yang digunakan di lapisan adapter.

- Domain

Lapisan ini merupakan inti dari aplikasi pin point, di mana aturan bisnis dan logika aplikasi didefinisikan. Subdirektori yang terdapat pada lapisan ini yaitu entity dan interactor. Berikut ini masing-masing penjelasan dari subdirektori tersebut:

  - Entity

    Direktori ini berisi entitas untuk logika bisnis dalam seluruh aplikasi. Entitas ini terdiri dari user, attendances, dan groups. Fungsi dari entitas ini adalah untuk memvalidasi seluruh *property* yang dimiliki.

o Interactor

Direktori ini berisi kode yang menangani logika bisnis aplikasi. Ada beberapa subdirektori di sini seperti attendances, groups, dan users yang masing-masing menangani fungsi tertentu dalam aplikasi.

- Infrastructure

Lapisan ini mengelola aspek teknis seperti *database* dan komunikasi eksternal. Subdirektori yang terdapat pada lapisan ini yaitu database. Pada aplikasi ini, *database* yang digunakan yaitu mongodb. Subdirektori mongodb yang terdapat pada *database* ini berisi kode yang berinteraksi dengan *database* MongoDB. Ada beberapa subdirektori di sini seperti attendances, groups, users, dan utils yang masing-masing menangani fungsi tertentu dalam aplikasi.

Pengujian yang dilakukan pada proyek Pin Point adalah pengujian unit dan pengujian E2E yang dilakukan pada setiap endpoint API. Setiap pengujian E2E dimulai dengan melakukan proses login salah satu pengguna untuk mendapatkan token autentikasi. Token ini kemudian digunakan untuk membuat permintaan ke aplikasi. Setelah permintaan selesai, respons diperiksa untuk memastikan bahwa aplikasi berperilaku seperti yang diharapkan.

Berikut ini merupakan contoh pengujian E2E pada project Pin Point:

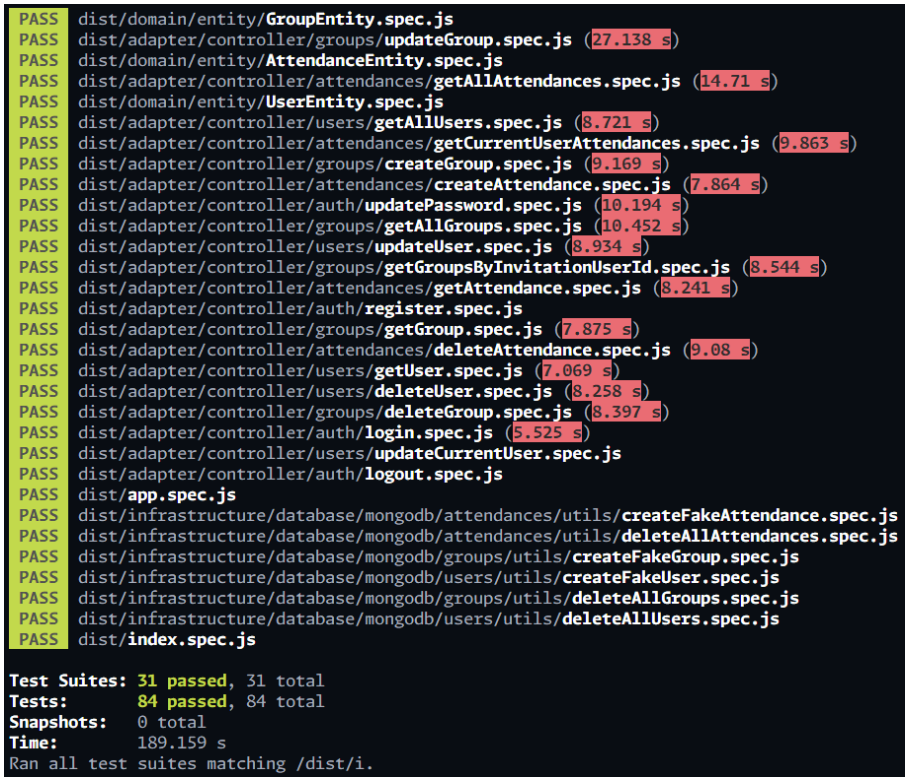
```
1.      describe('delete user example', () => {
2.        let app;
3.
4.        beforeEach(async () => {
5.          jest.setTimeout(20000);
6.          await deleteAllUsers();
7.        });
8.
9.        beforeAll(async () => {
10.         app = createApp();
11.       });
12.
13.       it('should be able delete user (admin only)', async () => {
14.         const data = await createFakeUser();
15.
16.         const authResponse = await request(app).post(`/users/login`).send({ email: data[0].email, password: '
    12345678' });
17.         const { token } = authResponse.body;
18.
19.         const response = await request(app)
20.           .delete(`/users/${data[1]._id.toString()}`)
21.           .set('Authorization', `Bearer ${token}`);
22.
23.         // expect http response
24.         expect(response.statusCode).toEqual(204);
25.
26.         // expect response json
27.         expect(response.body).toStrictEqual({});
28.       });
29.
30.       it('should thrown not found error (404) if not able to find user (admin only)', async () => {
31.         const data = await createFakeUser();
32.
33.         const authResponse = await request(app).post(`/users/login`).send({ email: data[0].email, password: '
    12345678' });
34.         const { token } = authResponse.body;
35.
36.         const response = await request(app)
37.           .delete(`/users/123456bd60bb49fe10c7b719`)
38.           .set('Authorization', `Bearer ${token}`);
39.
40.         // expect http response
41.         expect(response.statusCode).toEqual(404);  });
```

Kode Program 4 Pengujian Delete One User Pin Point

Kode Program 4 berisi kode untuk melakukan pengujian penghapusan user yang dilakukan pada API untuk menghapus satu user. Seluruh pengujian ini mengirimkan DELETE request ke rute /users/:id dan menyertakan id user pada rute tersebut. Berikut ini merupakan penjelasan mengenai kode diatas dalam bentuk tabel.

TABEL 1
PENJELASAN PENGUJIAN DELETE ONE USER PIN POINT

Pengujian	Deskripsi	Baris	Hasil Response
Menghapus satu user	Pengujian untuk memverifikasi bahwa aplikasi dapat menghapus user dan mengembalikan respons yang benar	13 – 28	204 No Content
Mengembalikan error jika id tidak ditemukan	Pengujian untuk memverifikasi bahwa aplikasi mengembalikan error ketika mencoba menghapus user yang tidak ada	30 – 42	404 Not Found



Gambar 4. Hasil pengujian project Pin Point

Gambar 4 merupakan hasil seluruh pengujian yang terdapat pada project pin point baik *unit testing* maupun E2E *testing*. Ada total 31 *test suites* yang dijalankan. Semua *test suites* tersebut berhasil (*passed*), yang menunjukkan bahwa setiap kumpulan *test suites* tersebut berjalan dengan sukses tanpa menemukan masalah. Dari total 84 pengujian yang terbagi dalam 31 *test suites*, seluruh pengujian berhasil (*passed*). Ini menunjukkan bahwa setiap kasus uji individual dalam suite tes tersebut berhasil memenuhi ekspektasi yang ditetapkan.

B. Hasil Project Think Action

Think Action adalah project kedua yang diberikan oleh perusahaan untuk dikerjakan. *Website* ini berguna untuk membantu pengguna dalam mencatat *goals* yang diinginkan. *Website* ini dirancang untuk melakukan *tracking goals* seperti *yearly goals*, *weekly goals*, dan *complete goals*. *Yearly goals* adalah impian jangka panjang yang ingin diwujudkan oleh

pengguna dalam satu tahun. *Weekly goals* adalah impian jangka pendek yang ingin diwujudkan oleh pengguna dalam satu minggu. *Complete goals* adalah impian yang telah tercapai dan sudah berhasil diwujudkan oleh pengguna.

Website Think Action memiliki beberapa fitur yang membuatnya berbeda dari sistem *tracking goals* lainnya. Pertama, *website* ini dapat melakukan *tracking goals* secara otomatis melalui Google Calendar berdasarkan tahun maupun bulan. Pengguna tidak perlu mencatat perkembangan impian mereka setiap tahun secara manual. Kedua, website ini memiliki fitur supporting, yaitu fitur yang memungkinkan pengguna untuk memberikan dukungan kepada pengguna lain. Pengguna juga dapat menyukai dan berkomentar pada postingan pengguna lain.

*Websit*e Think Action merupakan sebuah inovasi yang bermanfaat untuk membantu pengguna dalam mewujudkan impian mereka. Dengan menggunakan *website* ini, pengguna dapat mengelola dan memantau impian mereka secara mudah dan efektif. *Website* ini juga dapat meningkatkan motivasi dan keterlibatan pengguna dalam mewujudkan impian mereka, karena adanya fitur *supporting* yang memungkinkan pengguna untuk saling memberi dukungan dan semangat.

Berikut ini merupakan rincian model dan endpoint dalam API Docs yang telah dibuat:

- Users

Entitas ini menjadi dasar untuk autentikasi dan manajemen pengguna dalam sistem. Terdapat beberapa endpoint yang telah diimplementasikan, seperti:

1. Get One User: Endpoint ini berguna untuk mendapatkan informasi lengkap tentang satu pengguna berdasarkan id.
2. Get All Supporter: Endpoint ini berguna untuk mendapatkan seluruh detail supporter satu pengguna berdasarkan id.
3. Get All Supporting: Endpoint ini berguna untuk mendapatkan seluruh detail supporting satu pengguna berdasarkan id.
4. Get Current User Request: Endpoint ini berguna untuk mendapatkan seluruh detail request yang dimiliki oleh authenticated user.
5. Get Current User Notification: Endpoint ini berguna untuk mendapatkan seluruh detail notification yang dimiliki oleh authenticated user.
6. Update Current User Profile: Endpoint ini berguna untuk memperbarui data yang dimiliki oleh authenticated user.
7. Support Another User: Endpoint ini berguna untuk mendukung pengguna lain.
8. Unsupport Another User: Endpoint ini bergguna untuk tidak lagi mendukung pengguna lain.
9. Search User Account: Endpoint ini berguna untuk melakukan pencarian pengguna lain berdasarkan username.

- Comments

Entitas ini digunakan untuk menyimpan data komentar pada sebuah post. Endpoint-endpoint yang telah diimplementasikan adalah:

1. Get All Comments: Endpoint ini berguna mendapatkan seluruh komentar pada sebuah post.
2. Get All Reply: Endpoint ini berguna mendapatkan seluruh balasan komentar pada sebuah komentar.
3. Create One Comment: Endpoint ini berguna untuk membuat komentar pada satu post.
4. Create One Reply: Endpoint ini berguna untuk membuat balasan komentar pada satu komentar.
5. Update One Comment: Endpoint ini berguna untuk memperbarui komentar yang telah dibuat.
6. Delete One Comment: Endpoint ini berguna untuk menghapus komentar yang telah dibuat.

- Posts

Entitas ini digunakan untuk menyimpan data post seperti resolution, weekly goals, dan complete goals. Endpoint-endpoint yang telah diimplementasikan meliputi:

1. Get All Posts: Endpoint ini berguna untuk melihat seluruh post yang ada.
2. Get All Like: Endpoint ini berguna untuk melihat detail like pada satu post.
3. Create One Resolution: Endpoint ini berguna untuk membuat satu resolution.
4. Create One Weekly Goal: Endpoint ini berguna untuk membuat satu weekly goal.
5. Create One Complete Goal: Endpoint ini berguna untuk membuat satu complete goal.
6. Update One Resolution: Endpoint ini berguna untuk memperbarui resolution yang telah dibuat.
7. Update One Weekly Goal: Endpoint ini berguna untuk memperbarui weekly goal yang telah dibuat.
8. Update One Complete Goal: Endpoint ini berguna untuk memperbarui complete goal yang telah dibuat.
9. Like Post: Endpoint ini berguna untuk menambahkan like pada satu post.
10. Unlike Post: Endpoint ini berguna untuk menghilangkan like pada satu post.
11. Delete One Post: Endpoint ini berguna untuk menghapus satu post.

- Authentication

Endpoint-endpoint ini terkait dengan proses authentication dan authorization. Beberapa endpoint yang telah diimplementasikan adalah:

1. Register: Endpoint ini berguna untuk mendaftar dengan menyediakan email, username, dan password.
2. Login: Endpoint ini berguna untuk melakukan login dengan email dan password.
3. Logout: Endpoint ini berguna untuk logout dari sistem.
4. Update Current User Password: Endpoint ini berguna untuk memperbarui kata sandi authenticated user dengan menyediakan kata sandi saat ini dan kata sandi baru.

Pengujian yang dilakukan pada proyek Think Action adalah pengujian unit dan pengujian E2E yang dilakukan pada setiap endpoint API. Setiap pengujian E2E dimulai dengan melakukan proses login salah satu pengguna untuk mendapatkan token autentikasi. Token ini kemudian digunakan untuk membuat permintaan ke aplikasi. Setelah permintaan selesai, respons diperiksa untuk memastikan bahwa aplikasi berperilaku seperti yang diharapkan.

Berikut ini merupakan contoh pengujian unit dan pengujian E2E yang terdapat pada aplikasi Think Action:

```
1.      let mongoServer;
2.      let client;
3.
4.      beforeAll(async () => {
5.        mongoServer = new MongoMemoryServer();
6.        const mongoUri = await mongoServer.getUri();
7.        client = new MongoClient(mongoUri, { useNewUrlParser: true, useUnifiedTopology: true });
8.        await client.connect();
9.      });
10.
11.     afterAll(async () => {
12.       await client.close();
13.       await mongoServer.stop();
14.     });
15.
16.     describe('deleteAllUsers', () => {
17.       it('should delete all users', async () => {
18.         const db = client.db('think-action');
19.         const users = db.collection('users');
20.
21.         // Menambahkan beberapa users
22.         await users.insertMany([{ email: 'test1@email.com' }, { email: 'test2@email.com' }]);
23.
24.         // Memastikan user telah ditambahkan
25.         let userCount = await users.countDocuments({});
26.         expect(userCount).toBe(2);
27.
28.         // Menghapus semua user
29.         await deleteAllUsers();
30.
31.         // Memastikan semua user telah dihapus
32.         userCount = await users.countDocuments({});
33.         expect(userCount).toBe(0);
34.       });  });
```

Kode Program 5 Pengujian Unit Delete All Users Think Action

Kode Program 5 merupakan pengujian unit untuk menghapus semua data pada collection users. Berikut ini merupakan penjelasan mengenai kode diatas dalam bentuk tabel.

TABEL II
PENJELASAN PENGUJIAN UNIT DELETE ALL USERS THINK ACTION

Pengujian	Deskripsi	Baris
Menghapus semua data pada collection users di mongodb	Pengujian untuk memverifikasi bahwa fungsi dapat menghapus seluruh data pada collection users yang terdapat pada database mongodb.	16 – 34

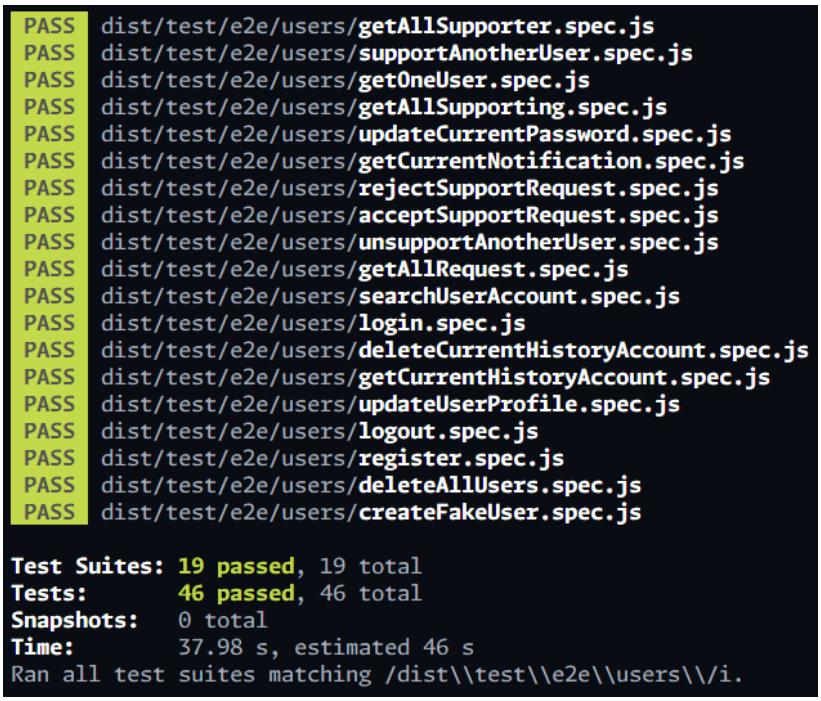
```
1.      describe('search user account example', () => {
2.        let app;
3.
4.        afterAll(async () => {
5.          await deleteAllUsers();
6.        });
7.
8.        beforeEach(async () => {
9.          jest.setTimeout(20000);
10.         await deleteAllUsers();
11.       });
12.
13.       beforeAll(async () => {
14.         app = createApp();
15.       });
16.
17.       it('should be able search user account', async () => {
18.         const data = await createFakeUser();
19.
20.         const authResponse = await request(app)
21.           .post(`/v1/users/login`)
22.           .send({ email: data[0].email, password: '12345678' });
23.         const { token } = authResponse.body;
24.
25.         const response = await request(app)
26.           .get('/v1/users/search')
27.           .query({ username: 'user1' })
28.           .set('Authorization', `Bearer ${token}`);
29.
30.         // expect http response
31.         expect(response.statusCode).toEqual(200);
32.         expect(response.body.results).toEqual(1);
33.
34.         // expect response json
35.         expect(response.body.data[0]._id.toString()).toEqual(data[1]._id.toString());
36.         expect(response.body.data[0].supportedBy).toBeDefined();
37.
38.
39.         // after search, check if history account is inserted
40.         const response2 = await request(app).get('/v1/users/history').set('Authorization', `Bearer ${token}`)
    ;
41.         expect(response2.body.data[0]._id).toEqual(data[1]._id.toString());
42.         expect(response2.body.data[0].supportedBy).toBeDefined();
43.
44.       });
45.     });
```

Kode Program 6 Pengujian Search User Think Action

Kode Program 6 berisi pengujian untuk melakukan pencarian user lain yang dilakukan pada API searchAnotherUser. Seluruh pengujian ini mengirimkan GET request ke rute /v1/users/search dan mengirimkan username pada query. Berikut ini merupakan penjelasan mengenai kode diatas dalam bentuk tabel.

TABEL III
PENJELASAN PENGUJIAN SEARCH USER THINK ACTION

Pengujian	Deskripsi	Baris	Hasil Response
Mendapatkan user berdasarkan username	Pengujian untuk melakukan pencarian user laindan memverifikasi bahwa aplikasi dapat menemukan data dan mengembalikan respons yang benar.	17 – 45	200 OK



Gambar 5 Hasil pengujian project Think Action

Gambar 5 merupakan hasil seluruh pengujian endpoint users yang terdapat pada project Think Action . Ada total *19 test suites* yang dijalankan. Semua *test suites* tersebut berhasil (*passed*), yang menunjukkan bahwa setiap kumpulan *test suites* tersebut berjalan dengan sukses tanpa menemukan masalah. Dari total 46 tes pengujian yang terbagi dalam 19 *test suites*, seluruh pengujian berhasil (*passed*). Ini menunjukkan bahwa setiap kasus uji individual dalam suite tes tersebut berhasil memenuhi ekspektasi yang ditetapkan.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Pembuatan dokumentasi API dibuat sesuai dengan pengembangan sistem *tracking goals*. Hal ini menggambarkan proses pembuatan dokumentasi yang jelas, konsisten, dan mudah diakses yang memungkinkan pengembang untuk mengintegrasikan berbagai fitur untuk *tracking goals* dan presensi. Dokumentasi tersebut dirancang untuk memastikan bahwa pengembang, terutama yang berfokus pada *front*-end, dapat menggunakan API dengan baik dengan menyediakan spesifikasi teknis yang lengkap seperti contoh permintaan dan hasil respons yang terstruktur dengan baik. Oleh karena itu, API yang dikembangkan tidak hanya mendukung kebutuhan saat ini tetapi juga memiliki ruang untuk adaptasi dan perluasan di masa depan untuk memastikan keberlanjutan sistem.

Pengujian back-end yang komprehensif juga dilakukan untuk memastikan fungsionalitas sistem. Pengujian unit dan E2E dilakukan secara menyeluruh untuk setiap fitur, memastikan bahwa semua skenario dan kasus yang mungkin terjadi telah

tercakup dan berfungsi sesuai harapan. Penggunaan teknologi seperti Node.js, Express, TypeScript, dan penerapan prinsip-prinsip Clean Architecture memungkinkan pembuatan kode yang modular dan mudah diuji. Hal ini memastikan bahwa sistem yang dikembangkan tidak hanya memenuhi kriteria teknis tetapi juga dapat diandalkan dan efisien dalam pengoperasiannya. Hasil pengujian yang telah berhasil (*passed*) menandakan bahwa sistem sudah berjalan dengan baik dan sesuai dengan apa yang diinginkan.

B. Saran

Saran bagi pembaca yang ingin mengembangkan back-end dan dokumentasi API dengan baik yaitu memastikan dahulu penguasaan terhadap bahasa pemrograman seperti JavaScript, PHP, Python, serta *framework* seperti Express.js. Pengetahuan mendalam tentang HTTP, REST API, Database SQL/NoSQL, dan konsep keamanan aplikasi web termasuk autentikasi dan otorisasi juga diperlukan. Dalam mendokumentasikan API, kejelasan dan kelengkapan informasi harus diutamakan seperti deskripsi dari masing-masing endpoint, contoh permintaan, dan contoh hasil respons. Pengujian API dapat dilakukan dengan alat seperti Postman. Terakhir, kemahiran dalam pengujian unit, integrasi, dan E2E dengan tools seperti Jest dan Supertest diperlukan untuk memastikan bahwa sistem bekerja sesuai harapan.

Saran bagi perusahaan mengenai sistem *tracking goals* yaitu untuk meningkatkan kemajuan sistem *tracking goals* dengan menambahkan fitur gamifikasi untuk meningkatkan keterlibatan pengguna, dimana pencapaian tujuan tertentu dapat dikaitkan dengan hadiah atau pencapaian dalam aplikasi, mendorong pengguna untuk tetap termotivasi dan berkomitmen pada tujuan mereka.

DAFTAR PUSTAKA

[1] M. Meng, S. Steinhardt, and A. Schubert, “Application programming interface documentation: What do software developers want?,” *Journal of Technical Writing and Communication*, vol. 48, no. 3, pp. 295–330, Jul. 2018, doi: 10.1177/0047281617721853.
[2] Y. Perez-Riverol *et al.*, “Ten Simple Rules for Taking Advantage of Git and GitHub,” *PLoS Computational Biology*, vol. 12, no. 7. Public Library of Science, Jul. 01, 2016. doi: 10.1371/journal.pcbi.1004947.
[3] G. Bierman, M. Abadi, and M. Torgersen, “Understanding TypeScript,” in *ECOOP 2014 – Object-Oriented Programming*, R. Jones, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 257–281.
[4] S. Fenton, *Pro TypeScript*. Apress, 2018. doi: 10.1007/978-1-4842-3249-1.
[5] S. L. Bangare, S. Gupta, M. Dalal, and A. Inamdar, “Using Node.Js to Build High Speed and Scalable Backend Database Server,” *International Journal of Research in Advent Technology (E-ISSN: 2321-9637) Special Issue National Conference “NCPCI-2016,”* 2016, [Online]. Available: www.ijrat.org
[6] K. Lei, Y. Ma, and Z. Tan, “Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js,” in *2014 IEEE 17th International Conference on Computational Science and Engineering*, IEEE, Dec. 2014, pp. 661–668. doi: 10.1109/CSE.2014.142.
[7] StrongLoop, “Express. Fast, unopinionated, minimalist web framework for Node.js.” Accessed: Oct. 24, 2023. [Online]. Available: https://expressjs.com/
[8] Christian Peters, “Building Rich Internet Applications with Node.js and Express.js,” Germany, 2016. [Online]. Available: www.uni-oldenburg.de/svs
[9] “MongoDB Documentation.” Accessed: Oct. 30, 2023. [Online]. Available: https://www.mongodb.com/docs/
[10] H. Matallah, G. Belalem, and K. Bouamrane, “Comparative Study Between the MySQL Relational Database and the MongoDB NoSQL Database,” *International Journal of Software Science and Computational Intelligence*, vol. 13, no. 3, pp. 38–63, Jul. 2021, doi: 10.4018/ijssci.2021070104.
[11] P. Aguilar and L. B. Figueira, “Clean Architecture is not only about business logic,” *I Workshop de Tecnologia da Fatec Ribeirão Preto*, vol. 1, no. 1, 2020.
[12] T. Hombergs, *Get Your Hands Dirty on Clean Architecture: A hands-on guide to creating clean web applications with code examples in Java*. 2019.
[13] E. Daka and G. Fraser, “A Survey on Unit Testing Practices and Problems,” Sheffield, Nov. 2014. Accessed: Oct. 30, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6982627/
[14] W. T. Tsai, X. Bai, R. Paul, W. Shao, and V. Agarwal, “End-to-end integration testing design,” *Proceedings - IEEE Computer Society’s International Computer Software and Applications Conference*, pp. 166–171, 2001, doi: 10.1109/CMPSAC.2001.960613.
[15] A. Bucko, K. Vishi, B. Krasniqi, and B. Rexha, “Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History,” *Computers*, vol. 12, no. 4, Apr. 2023, doi: 10.3390/computers12040078.