Implementasi REST API Pada Pengembangan Aplikasi Backend Untuk *Platform* Kursus Online (Growup)

1st Rizqy Eka Putra Rizaldy
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
rizqyep@student.telkomuniversity.ac.id

2nd Umar Ali Ahmad
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
umar@telkomuniversity.ac.id

3rd Burhanuddin Dirgantoro
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
burhanuddin@telkomuniversity.ac.id

Abstrak— Teknologi mengalami perkembangan yang sangat pesat dalam beberapa tahun terakhir ini, hal ini menyebabkan meningkatnya perusahaan rintisan yang mengembangkan produk berbasis digital juga perusahaan lainnya yang menambahkan produk berbasis digital. Hal ini tentunya juga diiringi dengan meningkatnya permintaan akan kebutuhan talenta digital di Indonesia. Namun sayangnya kapasitas jumlah dan kualitas talenta digital di Indonesia masih belum bisa memenuhi permintaan ini. Untuk mengatasi hal itu GrowUp hadir untuk menjembatani dan mengurangi ketimpangan antara permintaan dan kesediaan serta kualitas talenta digital yang ada di Indonesia. GrowUp akan menghadirkan produk dalam bentuk *website* kursus *online* yang didalamnya akan diterapkan metode FCP (Fundamental, *Conceptual* dan *Practical*) sehingga *user* dapat memperoleh pengetahuan dan juga kemampuan pemecahan masalah yang kuat serta memahami cara mengaplikasikan pengetahuan yang telah diperoleh untuk membantu ekosistem digital di Indonesia berkembang ke depannya. Untuk mengembangkan *platform* berbasis *website* ini maka dibutuhkan pengembangan aplikasi yang dapat menunjang kebutuhan pertukaran data dan juga transaksi yang terjadi di dalam platform. Penelitian yang dilakukan akan berfokus pada implementasi REST API dalam mengembangkan aplikasi *backend* pada platform GrowUp sehingga dapat menunjang kebutuhan pertukaran dan pemrosesan data yang terjadi di dalam platform.

Kata kunci— aplikasi *backend,* growup, REST API

I. PENDAHULUAN

GrowUp merupakan startup yang berdiri di bidang pengembangan digital skills pada bidang teknologi, desain dan bisinis yang bertujuan membantu masyarakat Indonesia dengan target pasar masyarakat usia produktif untuk memulai karir ataupun mengembangkan kompetensi di bidang digital skills melalui platform berbasis website dan juga pembelajaran secara tatap muka daring. Aplikasi GrowUp adalah aplikasi berbasis website yang akan menyediakan materi bagi user untuk belajar secara asinkron atau self-paced learning. GrowUp memiliki dua bagian website yaitu bagian tutor untuk mendaftarkan kursus dan memasukkan materi juga melakukan pembaharuan materi jika dibutuhkan, juga terdapat bagian student dashboard agar dapat dilihat progres belajar dan kursus yang diambil. Tutor yang bertindak sebagai pemilik kursus dapat berasal dari berbagai kalangan mulai dari professional di bidang terkait ataupun akademisi sesuai dengan target kursus yang dibawa. Ukuran talent pool atau kesediaan talenta digital yang memadai di Indonesia masih jauh dari kebutuhan. Oleh karena itu, platform GrowUp akan membantu untuk mengembangkan talent pool di Indonesia agar perusahaan-perusahaan di Indonesia tidak lagi kesulitan dalam mencari talenta yang kompetensinya sesuai dengan kebutuhan mereka untuk melakukan scale-up ataupun bagi perusahaan baru yang ingin mengembangkan produk berbasis digital.

Untuk membangun platform yang akan menyediakan kursus digital terutama dengan bentuk produk yang berbasis digital maka dibutuhkan aplikasi backend yang dapat menyediakan dan mengolah berbagai data dan transaksi yang akan terjadi pada platform GrowUp yang berbasis website. Oleh karena itu dikembangkanlah aplikasi backend dengan pengimplementasian REST API yang merupakan arsitektur resource-oriented agar kedepannya aplikasi backend dapat diakses dari berbagai client seperti website dan mobile application.

II. KAJIAN TEORI

A. Aplikasi Backend

Aplikasi *Backend* merupakan aplikasi yang berfokus pada sisi bagaimana sebuah *website* bekerja. kode pada aplikasi *backend* berfokus pada fungsionalitas dan logika yang dibutuhkan oleh sebuah *website* [1]. Aplikasi backend utamanya berfungsi untuk menerima *request* dari client dan memberikan *response* sesuai dengan *request* tersebut. Aplikasi *backend* juga merupakan bagian perangkat lunak yang bertindak untuk menghubungkan *client* dengan *database* agar dapat terjadi proses pertukaran dan manipulasi data.

B. REST API

REST adalah singkatan dari Representational State Transfer adalah sebuah terminologi yang dikemukakan oleh Roy Fielding dalam disertasinya. REST dalah sebuah arsitektur yang didefinisikan untuk membantu membuat

sistem yang terorganisir dan terdistribusi. [2]. Sedangkan REST API adalah sebuah API yang menerapkan architectural pattern REST. Pada REST API, operasi seperti CRUD (Create, Read, Update, Delete) akan dilakukan dengan cara memetakan HTTP Verbs seperti GET, POST, PUT, DELETE ke setiap endpoint sesuai dengan deskripsi tugas dari masing-masing endpoint.
Aristektur REST menjelaskan enam batasan, adapan keenam batasan arsitektur REST adalah sebagai berikut [3]:

a. Uniform Interface
Uniform interface adalah penyederhanaan dan pemisahaan arsitektur antara client dan server, hal ini memungkinkan setiap bagian dapat dikembangkan secara indepenen. Batasan ini adalah fundamental dari desain RESTful. RESTful menggunakan HTTP method (GET, POST, PUT, DELETE) untuk menjelaskan metode request, URI (Uniform Resource Identifier) untuk mengidentifikasi sumber, dan HTTP response yang didalamnya terdapat data status dan body untuk menjelaskan informasi yang dikembalikan server.

b. Client-Server
Batasan client-server akan menjelaskan pemisahan antara client dan server, sehingga dapat berkembang secara independent. Dengan menerapkan pemisahan client dan server, portablitas antarmuka pada berbagai platform akan meningkat, begitu juga dengan skalabilitas melalui penyederhanaan komponen server.

c. Stateless
Batasan ini mengharuskan sebuah request dari client ke server harus memiliki informasi yang lengkap dan utuh. Server tidak boleh memanfaatkan data ataupun konteks yang sudah di simpan pada server. Mengacu pada batasan ini, maka seluruh state dan session harus disimpan pada client.

d. Layered System
Batasan Layered System memungkinkan sebuah arsitektur yang terdiri dari hierarchial layers dengan membatasi perilaku dari setiap komponen.

e. Code on Demand
REST juga memungkinkan fungsionalitas client untuk disesuaikan oleh server secara sementara dengan cara transfer logic. Dengan batasan ini, client dapat langsung mengeksekusi code yang telah disediakan oleh server, contohnya seperti applet pada bahasa pemrograman Java.

C. Node.js
Node.js adalah sebuah runtime environment cross-platform yang bersifat *opensource* yang digunakan untuk mengembangkan aplikasi web pada bagian *server-side*. Aplikasi Node.js ditulis menggunakan bahasa JavaScirpt dan dapat dijalankan di berbagai sistem operasi [4].

Node.js dibuat berdasarkan arsitektur *event-driven* dan *non-blocking Input/Output* API yang didesain untuk mengoptimasi skalabilitas dan throughput pada aplikasi web *real-time* [4].

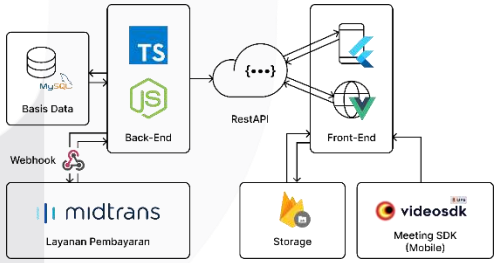
D. TypeScript
TypeScript adalah sebuah bahasa pemrograman opensource yang dikembangkan dan oleh Microsoft. TypeScript adalah superset dari Javascript, artinya TypeScript memiliki semua fitur yang ada pada JavaScript dengan tambahan beberapa fitur. Berbeda dengan bermacam subset dari JavaScript dan juga linting tools yang bertujuan untuk membuat bahasa yang lebih ringan [5].

Beberapa fitur tambahan pada TypeScript adalah dukungan pemrograman berorientasi objek dan tipe statis secara opsional. TypeScript ini dikembangkan dengan tujuan untuk membuat aplikasi dalam skala besar dan akan dikompilasi ke bahasa JavaScript.

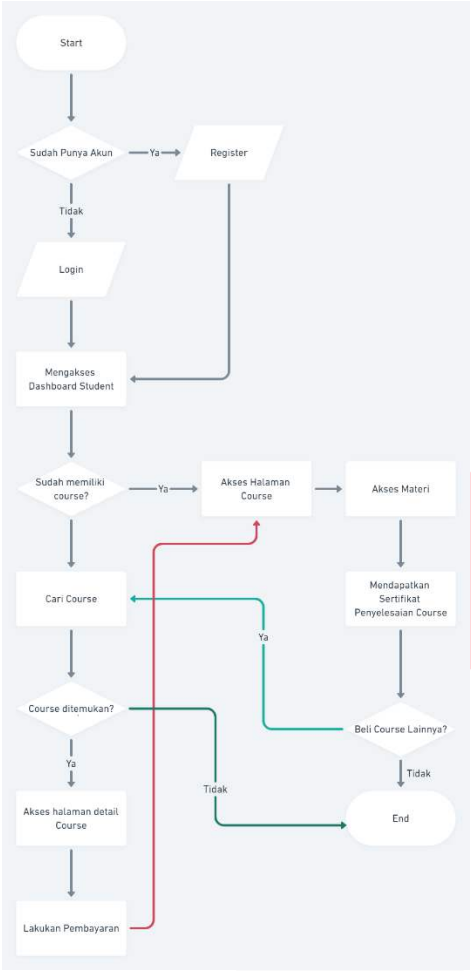
E. Unit Testing
Unit Testing adalah pengujian terautomasi yang dilakukan dengan cara melakukan isolasi terhadap suatu program atau aplikasi menjadi unit kecil untuk menguji kesesuaian data masukan dan keluaran agar sesuai dengan hasil yang diharapkan [6]. Melakukan isolasi program menjadi unit-unit membuat pengembang dapat memastikan bahwa program dapat berjalan sesuai dengan rancangan dan juga dapat membuat pengembang dapat mengidentifikasi banyak kemungkinan ataupun *edge cases* sehingga memungkinkan penanganan lebih awal terhadap permasalahan yang mungkin muncul dalam pengembangan program atau aplikasi.

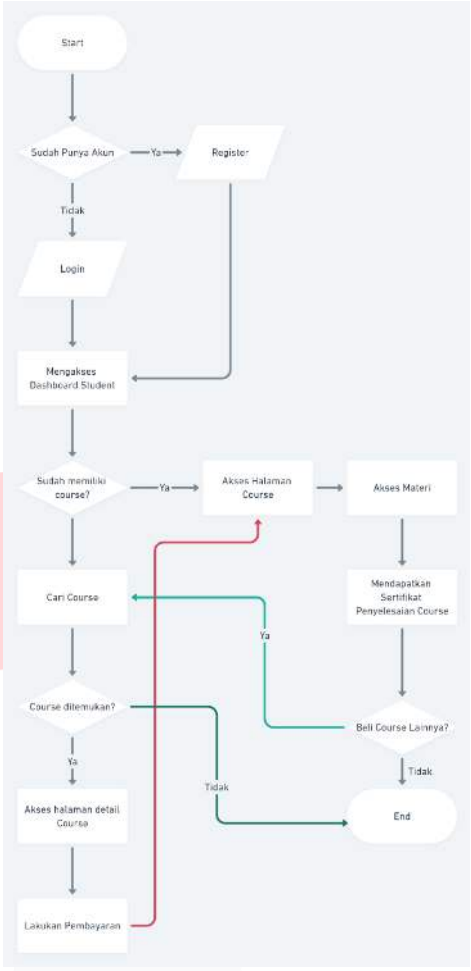
III. METODE

A. Arsitektur *Platform* GrowUp
Komponen utama aplikasi GrowUp terdiri dari aplikasi *backend, mobile* dan *website*¸ selain itu juga terdapat beberapa integrasi dengan pihak ketiga seperti *Firebase Storage* sebagai tempat penyimpanan media, midtrans untuk memproses pembayaran dan videosdk yang dimanfaatkan dalam fitur *live mentorship* sebagai media pertemuan *online* antara *user* dan mentor. Arsitektur lengkap platform GrowUp dapat dilihat pada Gambar berikut



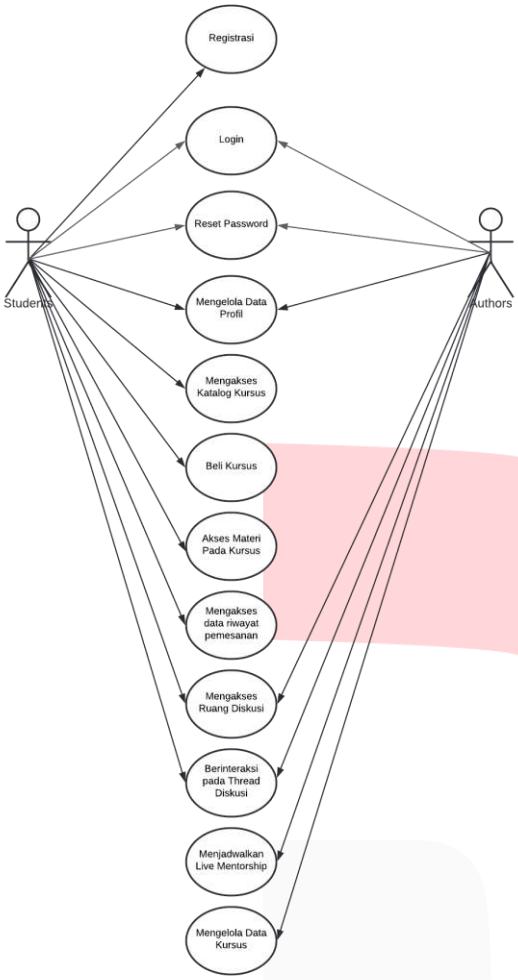
B. FlowChart





Gambar Flowchart pertama adalah alur pengguna dengan *role author/mentor* untuk membuat sebuah kursus, pengguna akan masuk ke dashboard dan melakukan beberapa tahapan yaitu, membuat kursus, menambahkan modul materi, dan menambahkan materi pada setiap modul. Sedangkan Gambar Flowchart kedua merupakan alur pengguna dengan *role student* untuk mendapatkan akses kedalam sebuah kursus, pengguna akan memulai dari proses pencarian kursus, lalu masuk ke halaman detail kursus dan melakukan pembelian untuk sebuah kursus. Pengguna akan memperoleh akses ke sebuah kursus jika pembayaran sudah di proses dan diverifikasi oleh *payment-gateway*.

C. Use Case Diagram



Pada diagram terdapat dua aktor yang terlibat pada sistem yaitu student dan author. Gambar diatas memiliki 12 use-case yaitu :

a. UC-1: Registrasi
b. UC-2: Login
c. UC-3: Reset Password
d. UC-4: Mengelola Data Profil
e. UC-5: Mengakses Katalog Kursus
f. UC-6: Beli Kursus
g. UC-7: Akses Materi Pada Kursus
h. UC-8: Mengakses Riwayat Pesanan
i. UC-9: Mengakses Ruang Diskusi
j. UC-10: Berinteraksi pada Thread Diskusi
k. UC-11: Menjadwalkan Live Mentorship
l. UC 12: Mengelola Data Kursus

IV. HASIL DAN PEMBAHASAN

Pengujian pada aplikasi backend GrowUp dilakukan dengan metode Unit Testing menggunakan framework mocha dan chai assertion library untuk melakukan assertion pada kode respons HTTP dan assertion pada object data response.

Adapun cakupan pengujian akan dilakukan pada use case 1 – 11 yang sudah dijelaskan pada bab sebelumnya, yang mana fungsionalitas pada use case 1 – 11 adalah fungsionalitas utama pada platform GrowUp saat ini,

Skenario pengujian akan ¬menggunakan metode equivalent partitioning yang membagi inputan valid dan invalid.

```
Use Case 1 : Registrasi - POST /v1/user/register
  Checks Register Endpoint for Wrong Data Input(s)
    ✓ TC1 - Should return 400 if full name is empty
    ✓ TC2 - Should return 400 if Email is empty
    ✓ TC3 - Should return 400 if occupation is empty
    ✓ TC4 - Should return 400 if date of birth is empty
    ✓ TC5 - Should return 400 if email is mal-formed
  Test Register Endpoint with Normal Data
    ✓ TC6 - Should Return  400 if registered with email that already existed
    ✓ TC7 - Should Return  201 if registered with clean data (288ms)
```

HASIL PENGUJIAN USE CASE 1

```
Use Case 2 : Log In - POST /v1/user/login
  Checks Login Endpoint for Wrong Data Input(s)
    ✓ TC8 - Should Return  400 if Email is empty
    ✓ TC9 - Should Return  400 if email is mal-formed
    ✓ TC10 - Should Return  400 if Password is empty
  Test Login Endpoint with Normal Data
    ✓ TC11 - Should Return  401 with message Unauthorized if login with incorrect credential (242ms)
    ✓ TC12 - Should Return  200 with user and token response body if login with Correct Credential (245ms)
```

HASIL PENGUJIAN USE CASE 2

```
Use Case 3: Reset Password - POST /v1/user/forgot-password
  Checks Forgot Password Request Endpoint for Wrong Data Input(s)
    ✓ TC13 - Should Return  400 if Email is not present
    ✓ TC14 - Should Return  400 if email is mal-formed
  Test Forgot Password Endpoint with Normal Data
    ✓ TC15 - Should Return  201 if user with presented email existed
    ✓ TC16 - Should Return  400 if user with presented email does not exist

Use Case 3: Reset Password - GET /v1/user/forgot-password/:token
  Checks Forgot Password Request Endpoint for Invalid and Used Token Input(s)
    ✓ TC17 - Should Return  400 if token is not valid
  Test Forgot Password Endpoint with Valid Token
    ✓ TC18 - Should Return  400 if token is valid but already usedd
    ✓ TC19 - Should Return  200 if token is valid and not used

Use Case 3: Reset Password - PUT /v1/user/reset-password
  Checks Reset Password Endpoint for Wrong Data Input(s)
    ✓ TC20 - Should Return  400 if Token is empty
    ✓ TC21 - Should Return  400 if Password is empty
    ✓ TC22 - Should Return  400 if Password Confirmation is empty
    ✓ TC23 - Should Return  400 if Password and Password Confirmation mismatched
  Test Reset Password Endpoint with Normal Data
    ✓ TC24 - Should Return  400 if token is already used
    ✓ TC25 - Should Return  201 if data is clean and token is valid (password successfully reset) (245ms)
```

HASIL PENGUJIAN USE CASE 3

```
Use Case 4: Kelola Data Profil - GET /v1/user/profile
  ✓ TC26 - Should Return  401 if given no authorization token
  ✓ TC27 - Should Return  401 and unauthorized message if token is invalid
  ✓ TC28 - Should Return  200 and profile data if token is valid

Use Case 4: Kelola Data Profil - PUT /v1/user/profile
  Checks Update Profile Endpoint for Wrong Data Input(s)
    ✓ TC29 - Should Return  400 if Full Name is not present
    ✓ TC30 - Should Return  400 if Occupation is not present
    ✓ TC31 - Should Return  400 if Date of Birth is not present
  TC32 Test Update Profile Endpoint with Normal Data
    ✓ - Should Return  401 if presented with complete data but token is invalid
    ✓ TC33 - Should Return  401 if presented with complete data but token is not present
    ✓ TC34 - Should Return  200 if presented with complete data (profile updated successfully)
```

HASIL PENGUJIAN USE CASE 4

```
Use Case 5: Akses Katalog Kursus - GET /v1/courses
  Check Course Endpoint
    ✓ TC35 - Should Return  200 and courses data (no errors in query and server)

Use Case 5: Akses Katalog Kursus
  Get Course by Slug Endpoint (/v1/courses/:slug)
    ✓ TC36 - Should Return  200 and course detail data (no errors in query and server)
    ✓ TC37 - Should Return  404 if given non-exist slug
  Get Course's Lessons by Course ID (/v1/courses/lessons/:courseId)
    ✓ TC38 - Should Return  200 and lessons data (no errors in query and server)
```

HASIL PENGUJIAN USE CASE 5

```
Use Case 6: Beli Kursus -  POST /v1/order
  Test Order Endpoint with Incorrect Input(s)
    ✓ TC39 - Should Return  400 if itemId is not present
    ✓ TC40 - Should Return  400 if total is not present
    ✓ TC41 - Should Return  400 if orderType is not present
    ✓ TC42 - Should Return  400 if orderType is not a valid value
  Test Order Endpoint with Correct Request Body
    ✓ TC43 - Should Return  401 if given no authorization token
    ✓ TC44 - Should Return  401 if token is invalid
    ✓ TC45 - Should Return  200 if token is valid (308ms)
```

HASIL PENGUJIAN USE CASE 6

```
Use Case 7: Akses Materi Kursus - GET /v1/user/course-enrollments
  ✓ TC46 - Should Return  401 if given no authorization token
  ✓ TC47 - Should Return  401 if token is invalid
  ✓ TC48 - Should Return  200 if token is valid

Use Case 7: Akses Materi Kursus - GET /v1/user/course-enrollments/check/:courseId
  ✓ TC49 - Should Return  401 if given no authorization token
  ✓ TC50 - Should Return  401 if token is invalid
  ✓ TC51 - Should Return  200 if token is valid

Use Case 7: Akses Materi Kursus - GET /v1/course/inclass/lessons/:lessonSlug
  Test In Class Course Lesson Endpoint With Different Cases
    ✓ TC52 - Should Return  401 if given no authorization token
    ✓ TC53 - Should Return  401 if token is invalid
    ✓ TC54 - Should Return  401 if token is valid and user is not enrolled in course
    ✓ TC55 - Should Return  200 if token is valid and user in enrolled in course
  Test In Class Course Materials Endpoint With Different Cases
    ✓ TC56 - Should Return  401 if given no authorization token
    ✓ TC57 - Should Return  401 if token is invalid
    ✓ TC58 - Should Return  200 and material details data if token is valid
```

HASIL PENGUJIAN USE CASE 7

```
Use Case 8: Mengakses Riwayat Pembelian -  GET /v1/order
  Test Order History Endpoint with Different Cases
    ✓ TC59 - Should Return  401 if given no authorization token
    ✓ TC60 - Should Return  401 if token is invalid
    ✓ TC61 - Should Return  200 and orderhistories data if token is valid

Use Case 8: Mengakses Riwayat Pembelian -  GET /v1/order/:uuid
  Test Order History Endpoint with Different Cases
    ✓ TC62- Should Return  401 if given no authorization token
    ✓ TC63 - Should Return  401 if token is invalid
    ✓ TC64 - Should Return  401 if token is valid but order does not belong to current user
    ✓ TC65 - Should Return  404 if token is valid but order uuid does not exist
    ✓ TC66 - Should Return  200 and order history detail data if token is valid (282ms)
```

HASIL PENGUJIAN USE CASE 8

```
Use Case 9: Mengakses Ruang Diskusi - GET /v1/spaces
  Test Spaces Endpoint
    ✓ TC67 - Should Return  200 and spaces data (no errors in query and server)

Use Case 9: Berinteraksi Pada Ruang Diskusi - GET /v1/spaces/:id/threads
  Test Thread Endpoint
    ✓ TC68 - Should Return 404 if space id does not exist
    ✓ TC69 - Should Return 200 and threads data (no errors in query and server)
```

HASIL PENGUJIAN USE CASE 9

```
Use Case 10: Berinteraksi Pada Thread Diskusi - POST /v1/spaces/threads/
  Test Create Thread Endpoint with incorrect Input(s)
    ✓ TC72 - Should Return 400 and if title is not present on request body
    ✓ TC73 - Should Return 400 if spaceId is not present on request body
    ✓ TC74 - Should Return 400 if content is not present on request body
  Test Create Thread Endpoint with correct Input(s) and different case(s)
    ✓ TC75 - Should Return 401 and if token is not present
    ✓ TC76 - Should Return 401 and if token is present but invalid
    ✓ TC77 - Should Return 201 if token is present and valid

Use Case 10: Berinteraksi Pada Thread Diskusi - PUT /v1/spaces/threads/:threadId
  Test Update Thread Endpoint with incorrect Input(s)
    ✓ TC78 - Should Return 400 and if title is not present on request body
    ✓ TC79 - Should Return 400 if content is not present on request body
  Test Update Thread Endpoint with correct Input(s) and different case(s)
    ✓ TC80 - Should Return 401 and if token is not present
    ✓ TC81 - Should Return 401 and if token is present but invalid
    ✓ TC82 - Should Return 404 and if token is present and valid but thread id does not exist
    ✓ TC83 - Should Return 401 and if token is present and valid but thread does not belong to current user
    ✓ TC84 - Should Return 200 and if token is present and valid also thread is owned by current user

Use Case 10: Berinteraksi Pada Thread Diskusi - DELETE /v1/spaces/threads/:threadId
  Test Delete Thread Endpoint with different case(s)
    ✓ TC85 - Should Return 401 and if token is not present
    ✓ TC86 - Should Return 401 and if token is present but invalid
    ✓ TC87 - Should Return 404 and if token is present and valid but thread id does not exist
    ✓ TC88 - Should Return 401 and if token is present and valid but thread does not belong to current user
    ✓ TC89 - Should Return 200 and if token is present and valid also thread is owned by current user
```

```
Use Case 10: Berinteraksi Pada Thread Diskusi - GET /v1/spaces/threads/:threadId/answers
  Check Thread Answers by Thread ID Endpoint with different cases
    ✓ TC90 - Should Return 404  when given non-existed thread Id
    ✓ TC91 - Should Return 200 and thread answers data, when given existed slug (no errors in query and server)

Use Case 10: Berinteraksi Pada Thread Diskusi - POST /v1/spaces/threads/:threadId/answers
  Test Create Thread Answer Endpoint with incorrect Input(s)
    ✓ TC92 - Should Return 400 and if content is not present on request body
    ✓ TC93 - Should Return 400 if threadId is not present on request body
  Test Create Thread Answer Endpoint with correct Input(s) and different case(s)
    ✓ TC94 - Should Return 401 and if token is not present
    ✓ TC95 - Should Return 401 and if token is present but invalid
    ✓ TC96 - Should Return 404 if token is present and valid but thread id does not exist
    ✓ TC97 - Should Return 201 if token is present and valid
```

HASIL PENGUJIAN USE CASE 10

```
Use Case 11: Menjadwalkan Live Mentorship - POST /v1/author/liveMentorhsip
  Test Create Live Mentorship Endpoint with incorrect Input(s)
    ✓ TC98 - Should Return 400 if courseId is not present on request body
    ✓ TC99 - Should Return 400 if startTime is not present on request body
    ✓ TC100 - Should Return 400 if title is not present on request body
  Test Create Live Mentorship Endpoint with correct Input(s)
    ✓ TC101 - Should Return 401 if token is not present
    ✓ TC102 - Should Return 401 if token is present but not valid
    ✓ TC103 - Should Return 201 if token is present but

Use Case 11: Menjadwalkan Live Mentorship - GET /v1/author/liveMentorhsip
  Test Create Live Mentorship Endpoint with correct Input(s)
    ✓ TC104 - Should Return 401 if token is not present
    ✓ TC105 - Should Return 401 if token is present but not valid
    ✓ TC106 - Should Return 201 and live mentorship data if token is present
```

HASIL PENGUJIAN USE CASE 11

Dari hasil pengamatan pada pengujian setiap use case, aplikasi backend GrowUp sudah dapat menerima inputan dan juga mengembalikan response yang sesuai pada setiap endpoint yang akan diakseshal ini dapat dilihat dari pengujian yang sudah dilakukan terhadap inputan yang valid dan tidak valid pada request body, juga pada inputan request header untuk pengecekan autorisasi user dengan token yang tidak valid ataupun token yang valid namun role ataupun user yang terdapat didalam token tidak memiliki akses terhadap resource yang akan diakses.

Berdasarkan hasil pengujian juga diperoleh dari 106 test cases yang diuji terhadap semua endpoint pada use case 1-11 jumlah test case yang berhasil dilewati adalah 106 atau 100% berhasil melewati unit testing, menunjukkan bahwa aplikasi backend GrowUp dapat diandalkan untuk memproses dan menyediakan data untuk kebutuhan platform GrowUp.

V. KESIMPULAN

Berdasarkan hasil tahapan implementasi REST API dan juga pengujian yang telah dilakukan dalam mengembangkan aplikasi backend GrowUp dapat ditarik kesimpulan bahwa

1. Pengembangan aplikasi backend GrowUp dengan implementasi REST API menghasilkan RESTful API yang dapat digunakan untuk memenuhi kebutuhan produk GrowUp.

2. Berdasarkan hasil pengujian yang dilakukan menggunakan Unit Testing menggunakan framework mocha dan chai assertion library, aplikasi backend Growup telah memenuhi rancangan pengujian dan dapat menyediakan serta memproses data untuk kebutuhan produk GrowUp.

REFERENSI

[1] Krystal, “Backend Development - The Beginner's Guide to Backend Technologies (2022),” Learn To Code With Me, 2 Februari 2022. [Online]. Available: https://learntocodewith.me/posts/backend-development/. [Diakses 26 Mei 2022].

[2] F. Doglio, REST API Development with Node.js, vol. 2, L. Corrigan, J. Markham and N. Chen, Eds., La Paz, Canelones: Apress Media LLC, 2018.

[3] L. Gupta, “What is REST - REST API Tutorial,” REST API Tutorial, 22 April 2022. [Online]. Available: https://restful.api.net. [Diakses 26 Mei 2022].

[4] K. Rungta, Learn NodeJS in 1 Day, Ahmedabad: Khrisna Rungta, 2016.

[5] S. Fenton, Pro TypeScript: Application-Scale JavaScript Development, Basingstoke: Apress Media LLC, 2018.

[6] V. Khorikov, Unit Testing, New York: Manning Publications Co., 2020.

[7] R. Julianto, “Apa itu UML? Beserta Pengertian dan Contohnya - Dicoding Blog,” Dicoding Indonesia, 12 Mei 2021. [Online]. Available: https://www.dicoding.com/blog/apa-itu-uml/. [Diakses 26 Mei 2022].

[8] A. D. Sole, Visual Studio Code Distilled : Evolved Code Editing for Windows, macOS, and Linux, Cremona: Apress Media LLC, 2019.