Jurnal Ilmu Komputer dan Informatika (JIKI)
Vol. 3, No. 2, Desember 2023, Hal. 125-141
https://jiki.jurnal-id.com

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

Pengembangan Arsitektur *Microservice* pada *Learning Management System E-learning* Menggunakan Metode *Web Service*

Calvin Alvito Dinova[*1], Ihsan Cahyo Utomo[2]

[1,2]Teknik Informatika, Fakultas Komunikasi dan Informatika, Universitas Muhammadiyah Surakarta, Indonesia
Email: [1]l200190213@student.ums.ac.id, [2]icu866@ums.ac.id

Abstrak

Learning Management System merupakan perangkat lunak dengan skala besar maka akan melibatkan berbagai media dan jaringan yang rumit, dalam pengembangan LMS, masih umum digunakan arsitektur monolitik yang menjalankan semua logika aplikasi dalam satu server. Namun, arsitektur monolitik memiliki kelemahan dalam mengadaptasi perubahan kebutuhan sistem, kompleksitas kode, dan pemeliharaan. Sebagai solusi alternatif, muncul arsitektur microservice yang membangun aplikasi sebagai kumpulan layanan kecil yang bekerja secara terpisah dan berkomunikasi melalui berbagai mekanisme. Pendekatan ini membantu menjaga keorganisan dan independensi kode program, sehingga memungkinkan pengujian aplikasi secara terpisah jika terjadi kesalahan.Dalam pertukaran data antar layanan, metode web service dengan format JSON (JavaScript Object Notation) sering digunakan. Penggunaan JSON memberikan kemudahan dalam implementasi dan tidak memberikan beban berlebih pada server. Penulis ingin membangun sebuah learning managaement system untuk *e-learning* yang menggunakan metode web *service* yang menggunakan arsitektur *microservice* yang mana akan membantu merancang bagian *service* agar lebih terorganisir dan mudah untuk di *maintenace,* yang dimana telah dilakukan pengujian menggunakan black box membuktikan bahwa sistem *back-end* yang menggunakan arsitektur *microservice* berfungsi sesuai dengan yang diharapkan. Selain itu, pengujian integrasi antara API *Gateway* dan layanan lainnya juga berhasil, di mana empat skenario koneksi antara layanan online menghasilkan kode respon 200 dan *response body* data yang sesuai, sedangkan empat skenario ketika salah satu layanan *offline* menghasilkan *response code* 500 dan pesan *error "Service Unavailable"*.

Kata Kunci: *E-Learning, Learning Management System, Microservice, Web Service.*

Abstract

The Learning Management System (LMS) is a large-scale software that involves various complex media and networks. In the development of LMS, the commonly used architecture is still monolithic, where all application logic runs on a single server. However, the monolithic architecture has drawbacks in adapting to system requirement changes, code complexity, and maintenance. As an alternative solution, the microservice architecture has emerged, building applications as a collection of small, independent services that communicate through various mechanisms. This approach helps maintain code organization and independence, enabling separate testing of applications in case of errors. In data exchange between services, web service methods using JSON (JavaScript Object Notation) format are frequently used. The use of JSON facilitates implementation and reduces server overhead.The author intends to develop a learning management system for e-learning that utilizes web service methods and adopts a microservice architecture. This approach will help design the service components to be more organized and easily maintainable. Testing using black box methodology has demonstrated that the backend system using microservice architecture functions as expected. Additionally, integration testing between the API Gateway and other services has been successful, where four scenarios of online service connections resulted in a response code of 200 and the appropriate response body data, while four scenarios involving offline services produced a response code of 500 and an error message "Service Unavailable."

Keywords: *E-Learning, Learning Management System, Microservice, Web Service*

1. PENDAHULUAN

Perkembangan teknologi saat ini sangat bermanfaat dalam berbagai bidang, termasuk pendidikan. Salah satu manfaatnya adalah pengembangan *Learning Management System* (LMS), sebuah aplikasi

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

web untuk mengelola pembelajaran, interaksi siswa-guru, penilaian, dan pelaporan kemajuan siswa [1]. Dengan menggunakan metode *E-learning*, siswa dapat mengakses materi di mana pun dan kapan pun [2]. *E-learning* ini juga mengakomodasi berbagai latar belakang siswa, termasuk penyandang disabilitas. Dalam pembelajaran jarak jauh, kualitas *e-learning* dianggap baik ketika informasinya berkualitas tinggi dan digunakan secara signifikan.[3].

LMS ini merupakan perangkat lunak dengan skala besar yang maka akan melibatkan berbagai media dan jaringan yang rumit, Namun, secara umum, pengembangan tersebut masih menggunakan arsitektur monolitik. Arsitektur monolitik menjalankan semua logika aplikasi dalam satu server, namun memiliki kelemahan dalam mengadaptasi perubahan kebutuhan sistem, kompleksitas kode, dan pemeliharaan. ketika ada perubahan atau penambahan teknologi, diperlukan perubahan menyeluruh pada seluruh aplikasi [4]. Arsitektur *microservice* hadir sebagai aplikasi yang dibangun sebagai kumpulan dari layanan-layanan kecil yang bekerja secara terpisah dalam proses masing-masing dan berkomunikasi melalui berbagai mekanisme [5]. Arsitektur *microservice* membuat kode program lebih ringkas dan independen sehingga memungkinkan pengujian aplikasi secara terpisah jika terjadi kesalahan [6]. Untuk pertukaran data menggunakan metode web *service* dengan format JSON (*JavaScript Object Nation*). Dengan menggunakan JSON memberikan kemudahan dalam implementasi dan juga tidak terlalu membebani server. [7].

Untuk mendukung permasalahan yang dibahas, penulis mengkaji penelitian terdahulu yang relevan dan valid terkait objek penelitian. Berikut adalah ringkasan penelitian terdahulu yang relevan dengan permasalahan tersebut.

Dalam penelitian yang dilakukan oleh D. J. Riyanto, P. Pizaini, N. S. H., and M. Affandes mereka menyajikan hasil penelitian berjudul "Impementasi *Service Choreography* Arsitektur *Microservice Classroom* Akademik Menggunakan *Docker*". Penelitian ini menghasilkan implementasi arsitektur *microservice* dengan menggunakan teknik kontainerisasi untuk mengisolasi setiap layanan dan lingkungannya. Tujuan dari penelitian ini adalah mengimplementasikan pola *service choreography* [4].

Penelitian yang dilakukan D. A. Bauer Hochschule Emden, M. Assaad, D. Alessandro Bauer, B. Penz, *and* J. Mäkiö, membahas peningkatan teknis dari prototipe platform e-learning STIMEY dengan menggunakan pola arsitektur mikro. Pendekatan pertama menggunakan teknologi fragmen halaman yang mengalami kesulitan dalam pemeliharaannya. Pendekatan kedua lebih efisien dengan menggunakan satu layanan mikro untuk menyimpan semua fragmen halaman, dan data spesifik diberikan secara terpisah oleh layanan mikro yang sesuai dengan domainnya [8].

Penelitian A. Milovanović membahas solusi menggunakan arsitektur *microservice* dengan mengintegrasikan beberapa LMS. Pendekatan yang digunakan adalah domain *driven design* dan penggunaan *integration cloud data platform* juga dipertimbangkan dalam arsitektur tersebut. [9].

Penelitian P. Niemelä *and* H. Hyyrö sistem arsitektur *microservice* menyediakan berbagai layanan yang modular, independen, dan tahan terhadap kesalahan. Namun, saat ini masih kurangnya sistem *open-source* berbasis *microservice* untuk keperluan pendidikan. Dalam penelitian ini, dilakukan studi terhadap sistem manajemen pembelajaran berbasis *microservice* seperti WETO dan Plussa, dengan tujuan mengembangkan proposal untuk menciptakan sistem manajemen pembelajaran yang ideal dan terlepas [10].

Penelitian oleh U. Syarif *and* P. Pizaini menerapkan *Event-Driven Microservice* pada Aplikasi Layanan Penerimaan Peserta Didik Baru. Menggunakan metode SDLC *waterfall* dan pengujian dengan *Software Architecture Analysis Method* (SAAM). Hasil evaluasi menunjukkan arsitektur *microservice* lebih unggul dibandingkan dengan arsitektur *monolith*. [11].

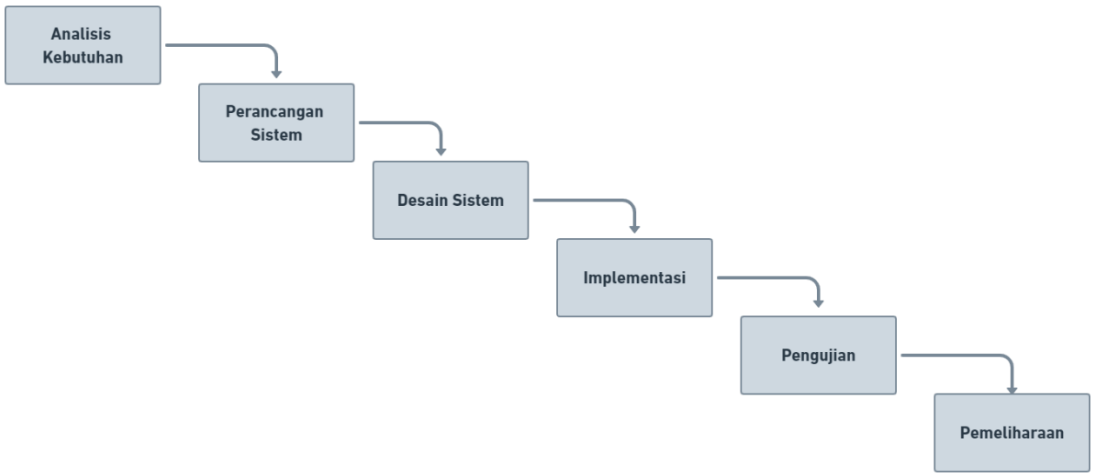
Penulis bertujuan untuk membuat LMS untuk *e-learning* dengan menggunakan arsitektur *microservice*. Dalam pendekatan ini, metode *web service* digunakan untuk memungkinkan organisasi dan pemeliharaan yang lebih baik dalam pengembangan bagian layanan sistem.

2. METODE

Pada perancangan, digunakan metode penelitian *Software Development Life Cycle* (SDLC) *waterfall* yang terkenal dan umum digunakan dalam pembuatan sistem. Metode ini melibatkan tahapan berurutan dalam pembuatan sistem, dimana setiap tahapan harus diselesaikan sebelum melanjutkan ke

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

tahapan berikutnya [12]. Tahapan penelitian ini terdiri dari beberapa langkah, yang dimulai dari analisis, perancangan, desain, implementasi, pengujian, hingga pemeliharaan. Gambar 1 menunjukkan metode SDLC *waterfall*.



Gambar 1. Metode SDLC Waterfall

2.1. Analysis

Tahap Analisis sering juga disebut *Software Requirement Specification* (SRS) / Spesifikasi Kebutuhan Perangkat Lunak, dimana ini adalah deskripsi terperinci dan lengkap tentang perilaku perangkat lunak yang akan dibangun. [13]. Tahap ini menentukan layanan-layanan yang diperlukan dalam arsitektur *microservice* LMS. Setiap layanan beroperasi secara terpisah dengan tugas yang spesifik dan saling berinteraksi menggunakan API (*Application Programming Interface*) yang digunakan untuk menghubungkan dan berkomunikasi antara berbagai service. Berikut ini adalah ringkasan dan penjelasan fungsi dari setiap layanan yang akan dibuat.

a. API *Gateway*
   API *Gateway* merupakan komponen penting dari manajemen API. Layanan ini mengirimkan permintaan layanan *backend* dan bertanggung jawab atas fungsionalitas [14]. API berfungsi sebagai *single entry point* yang menghubungkan antara *client* dan semua service yang ada diaplikasi. API *gateway* juga bertindak sebagai tempat berkomunikasi antar server atau bisa disebut dengan *middleware*.
b. *Service Media*
   Service Media merupakan *service* yang berfungsi untuk mengelola data media gambar.
c. *Service User*
   Service User merupakan *service* yang berfungsi untuk mengelola data user dan untuk aunthentikasi user.
d. *Service Course*
   Service Course merupakan *service* yang berfungsi untuk memberikan atau menyimpan data-data kursus.
e. *Service Order & Payment*
   Service Order & Payment merupakan *service* yang berfungsi untuk melakukan pembelian dari kursus yang disediakan dan sebagai perantara antara customer dan merchant yang memastikan transaksi berjalan dengan aman dan cepat.

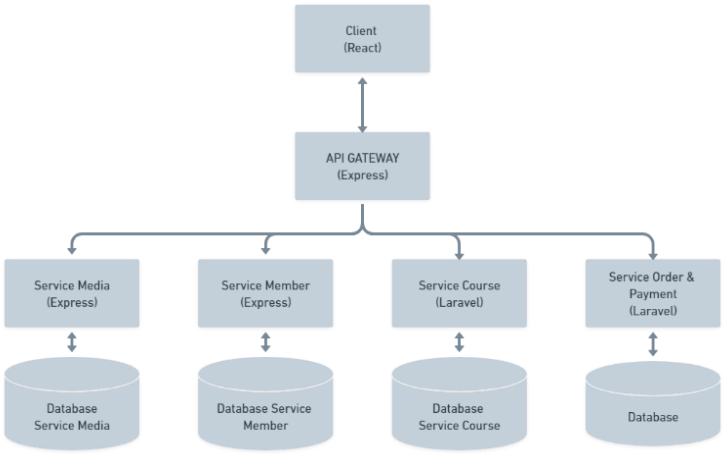
2.2. Perencanaan

Dalam penelitian ini perencanaan terbagi menjadi dua tahapan, yaitu :

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

2.2.1. Perencanaan Arsitektur

Tahap awal adalah membuat desain arsitektur LMS yang memenuhi kebutuhan dengan menggunakan arsitektur *microservice*. Dalam arsitektur *microservice*, digunakan pola *choreography pattern*, dimana setiap *microservice* beroperasi secara independen dengan logika layanan yang terdesentralisasi. Dengan begitu, tidak ada penumpukan logika proses bisnis pada satu layanan tertentu, sehingga dapat menghindari penundaan untuk klien [11].



Gambar 2. Arsitektur Microservice

Pada Gambar 2, komunikasi antara klien dan API *gateway*, serta antara API *gateway* dengan layanan-layanan lainnya menggunakan *REST API*. API *gateway* berperan sebagai titik masuk tunggal untuk *microservice* dan bertanggung jawab atas autentikasi menggunakan *JSON Web Token (JWT)*. Setelah autentikasi berhasil, permintaan akan diteruskan ke layanan-layanan seperti service media, service *user*, service *course*, dan service *order & payment*.

2.2.2. Perancangan *Service*

Langkah ini merupakan kelanjutan dari tahap analisis untuk membangun *e-learning* dengan menggunakan arsitektur *microservice*. Pada tahap ini, akan dijelaskan teknologi apa yang digunakan pada masing-masing layanan, seperti bahasa pemrograman, *framework*, dan *runtime*. Berikut ini adalah rencana untuk setiap layanan yang akan dibangun :

a. API gateway
   API *gateway* dibuat dengan menggunakan bahasa pemrograman *javascript* dengan menggunakan *framework express* dan *runtime NodeJs*.
b. *Service Media*
   Service media dibuat dengan menggunakan bahasa pemograman *javascript* dengan *framework* express dan menggunakan *runtime NodeJs*.
c. *Service Member*
   Service member dibuat dengan menggunakan bahasa pemograman *javascript* dengan *framework* express dan menggunakan *runtime NodeJs*.
d. *Service Course*
   Service course dibuat dengan menggunakan bahasa pemograman PHP dengan framework laravel dan menggunakan runtime laravel artisan.
e. *Service Order & Payment*
   Service order & payment dibuat dengan menggunakan bahasa pemograman *PHP* dengan *framework* laravel dan menggunakan *runtime laravel artisan*.

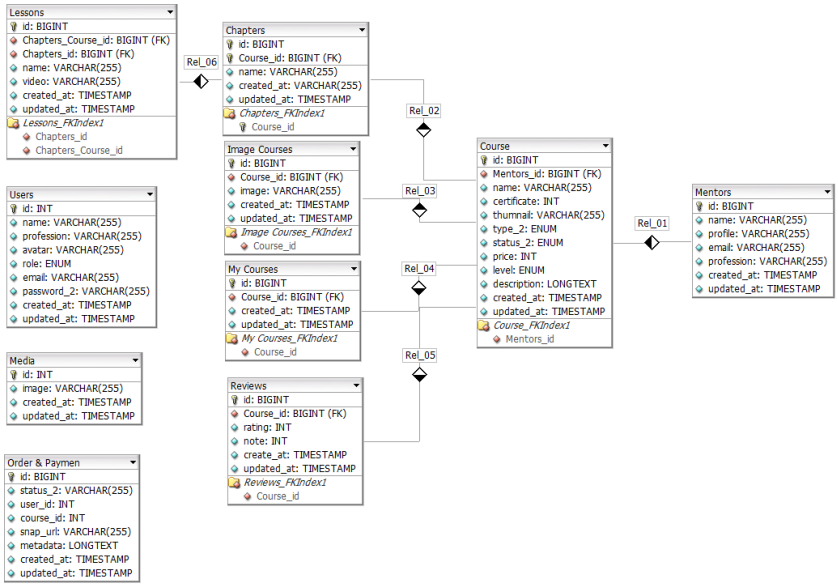
DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

2.3. *Design*

Setelah mendapatkan data analysis *service* apa saja yang dibutuhkan untuk pengembangan arsitektur *microservice*, maka akan dilanjutkan ke tahap selanjutnya yaitu tahap *design*. Pada tahapan design dilakukan dengan cara mendefinisikan *flow* setiap service dan modeling *Entity Relational Diagram* (ERD) yang nantinya akan membantu untuk mengembangkan arsitektur *microservice* pada LMS dan akan dijelaskan berikut ini :

2.3.1. *ERD (Entity Relational Diagram)*

Entity Relationship Diagram (ERD) merupakan perancangan basis data yang menggunakan suatu diagram, yang digunakan untuk membuat sebuah rancangan hubungan antar tabel pada database [15]. Dalam arsitektur microservice, setiap entitas berdiri sendiri dan terisolasi satu sama lain. Berikut adalah ERD yang telah dibuat, dan hasil ERD tersebut dapat dilihat pada Gambar 3.

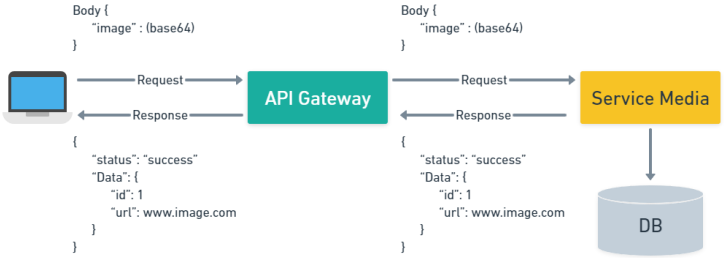


Gambar 3. ERD (Entiry Relational Diagram)

2.3.2. Flow *Service*

Dalam *flow service* mendefinisikan bagaimana alur dari sebuah *service* bekerja dimulai dari server melakukan *request* dan *service* melakukan *response*. Dalam arsitektur *microservice* yang digunakan dalam *e-learning* mengimplementasikan *web service* dimana setiap service menggunakan *REST API* untuk berkomunikasi.

2.3.2.1. Service Media

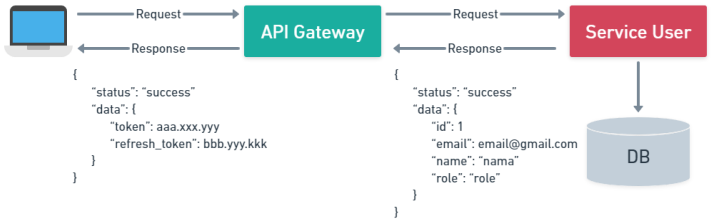


Gambar 4. Flow Service Media

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

Dalam *flow* media seperti Gambar 4. , client mengirimkan request image terenkripsi dengan *base64* dikirim ke API *Gateway*. API *Gateway* meneruskan *request* ke *service* media. *Service* media mengirimkan *response* berisi status, id, dan URL image yang diminta oleh *client*.

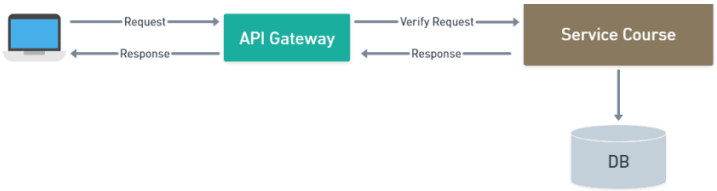
2.3.2.2. Service User



Gambar 5. Flow Service User

Dalam alur *service user seperti* Gambar 5, *request* dikirim ke API *Gateway* dan kemudian ke *service user*. *Service user* memeriksa data di database dan memberikan respons ke API *Gateway* dengan status dan data (id, email, nama, peran). API *Gateway* mengubah respon tersebut menjadi token JWT untuk otentikasi dan otorisasi pengguna. Token tersebut digunakan untuk mencegah penyalahgunaan data. Terdapat juga refresh_token untuk memperbarui token yang kedaluwarsa.

2.3.2.3. Service Course



Gambar 6. Flow Service Course

Pada *flow service course* seperti Gambar 6, *client* melakukan *request* dan *API Gateway* melakukan *verify request* tersebut dengan melakukan cek ke *header* apakah *client* memasukan *authorization* dan apakah *user* tersebut memiliki akses atau tidak. Jika telah *verify* maka *service course* akan melakukan *response* data yang akan dikirim ke *API Gateway* dan akan diteruskan ke *client*.

2.3.2.4. Service Order & Payment

Dalam alur *service order & payment* seperti Gambar 7, *client* mengirim permintaan ke API *Gateway* untuk bergabung dengan kelas yang tersedia. API *Gateway* meneruskan permintaan tersebut ke *service course* untuk verifikasi kelas premium. Jika kelas premium, *service course* menghubungi service *order & payment* untuk mendapatkan *snap* URL dari midtrans. *Snap* URL tersebut dikirim kembali ke *service order & payment* yang memberikan respon ke *service course*. API *Gateway* menerima respon tersebut dan mengirimkannya kembali ke *client*.

Gambar 7. Flow Service Order & Payment

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

2.4. Implementasi

Langkah berikutnya adalah pelaksanaan atau implementasi, yang melibatkan pengembangan aplikasi web dari hasil desain sebelumnya. Pengembangan *e-learning* ini menggunakan *framework Laravel dan Express JS* untuk *API* dan untuk *front-end* menggunakan *React Js* kode editor yang digunakan adalah *VCS (Visual Studi Code). Web browser Google Chrome* dan *web server Apache* beserta *database MySQL* diperlukan dalam proses pembuatan.

2.5. *Testing*

Pengujian *black box* adalah sebuah metode untuk menguji fungsionalitas suatu sistem atau komponen. Saat melakukan pengujian *black box*, dilakukan pembuatan *test case* yang berisi formulir input dan output yang diharapkan dari fungsi aplikasi yang diuji. [16]. Pengujian dapat dilakukan pada aplikasi dengan granularitas rendah, sehingga waktu yang dibutuhkan relatif singkat [17].

3. HASIL DAN PEMBAHASAN

3.1. Hasil

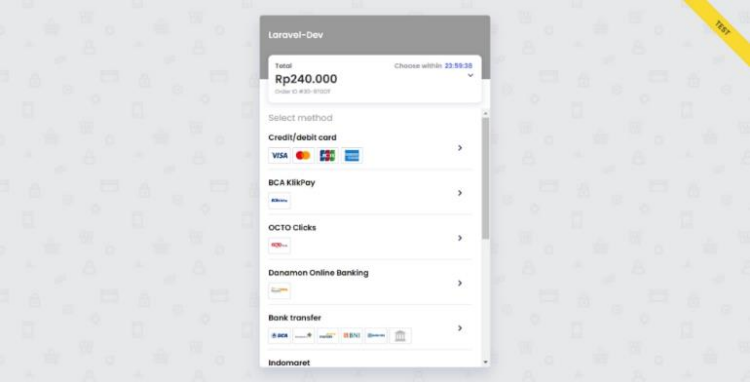
Hasil yang dicapai dari sistem ini menghasilkan sebuah sistem *learning management system e-learning* yang menerapkan arsitektur *microservice* dalam server-servernya. Dihalaman awal *user* akan diarahkan ke halaman utama yang mana berisi *list course* yang dapat dipilih oleh *user*, fitur daftar akun dan terdapat pengelompokan *course* berdasarkan jenisnya. Halaman utama bisa dilihat pada Gambar 8.

Gambar 8. Halaman Home Page

Jika *course* merupakan *premium* seperti Gambar 9. *user* akan diarahkan ke *link midtrans* seperti Gambar 10. dan diarahkan untuk melakukan pembayaran terlebih dahulu agar dapat mengakses kelas.

Gambar 9. Halaman Course (Premium)

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

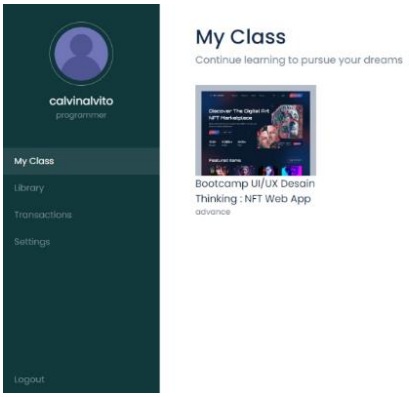


Gambar 10. Halaman Pembayaran Midtrans

Halaman *class room* merupakan halaman yang dimana *user* mendapatkan materi berupa *video learning* dan informasi *course* berupa judul, *lesson dan chapter*. Halaman *class room* seperti pada Gambar 11.

Gambar 11. Halaman Class room

Halaman *dashboard user* merupakan halaman yang berisi informasi course user yang sedang diperlajari seperti pada Gambar 12. riwayat transaksi *user* yang berisi *link midtrans* seperti dan *setting akun user* untuk mengatur data *user* juga tersedia didalam halaman dashboard user.



Gambar 12. Halaman My Class

3.2. *Testing*/Pengujian

Dalam bagian ini akan dibahas mengenai dua jenis pengujian, yaitu pengujian *black box* dan pengujian integrasi, yang berfokus pada pengguna aplikasi khususnya pada bagian *backend*.

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

3.2.1. Pengujian *Black Box*

Pengujian *black box* dipakai untuk memverifikasi fungsionalitas sistem secara menyeluruh serta menjamin kesesuaian antara data input dan output yang dihasilkan oleh sistem yang telah dirancang. Disini peneliti membedakan tabel pengujian black box menjadi dua bagian, yaitu bagian *student* dan bagian admin.

Berdasarkan hasil pengujian black box pada halaman bagian *student* ditunjukan oleh Tabel 1 maka sistem telah berfungsi sebagaimana yang diharapkan sehingga pengguna dapat menggunakan sistem ini tanpa adanya masalah.

Kemudian, berdasarkan hasil pengujian black box pada halaman bagian admin oleh Tabel 2 maka sistem telah berfungsi sebagaimana yang diharapkan sehingga admin dapat menggunakan sistem ini tanpa adanya masalah.

Tabel 1.Pengujian Black Box Testing Student

No	Langkah Pengujian	Kondisi Pengujian	Harapan	Hasil
1	*Student* melakukan *register* ke sistem.	*Student* melakukan *register* dengan mengisi data dan *email* yang belum terdaftar	Sistem akan menyimpan data ke database dan kemudian menuju halaman *login*.	Valid
2	*Student* gagal melakukan *register* ke sistem.	*Student* melakukan *register* dengan mengisi data dan *email* yang sudah terdaftar.	Sistem akan gagal *register* dan terdapat *error message "email already exist".*	Valid
3	*Student* melakukan *login* ke sistem	*Email* dan *password* benar	Sistem akan menerima akses *login* dan berhasil masuk ke halaman utama	Valid
4	*Student* gagal *login* ke sistem	*Email* dan *password* salah	Sistem akan gagal *login* dan terdapat *error message "user not found".*	Valid
5	*Student* melakukan transaksi kelas premium.	*Student* melakukan transaksi pembelian kelas premium dan pembayaran lewat midtrans.	*Student* berhasil melakukan transaksi pembelian kelas premium lewat midtrans dan mendapatkan akses kelas premium.	Valid
6	*Student* gagal melakukan transaksi kelas premium.	*Student* tidak menyelesaikan pembayaran lewat midtrans.	Sistem akan melabeli kelas tersebut dengan label pending dan *user* tidak akan mendapatkan akses kelas premium.	Valid
7	*Student* melakukan transaksi kelas gratis.	*Student* melakukan transaksi pembelian kelas gratis.	*Student* berhasil melakukan transaksi pembelian kelas gratis.	Valid
8	*Student* melakukan pembelajaran video kelas.	*Student* melakukan pembelajaran video kelas yang dapat diakses di halaman *member page.*	*Student* berhasil melakukan pembelajaran vidio kelas yang dapat diakses.	Valid

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

Tabel 2. Pengujian Black Box Admin

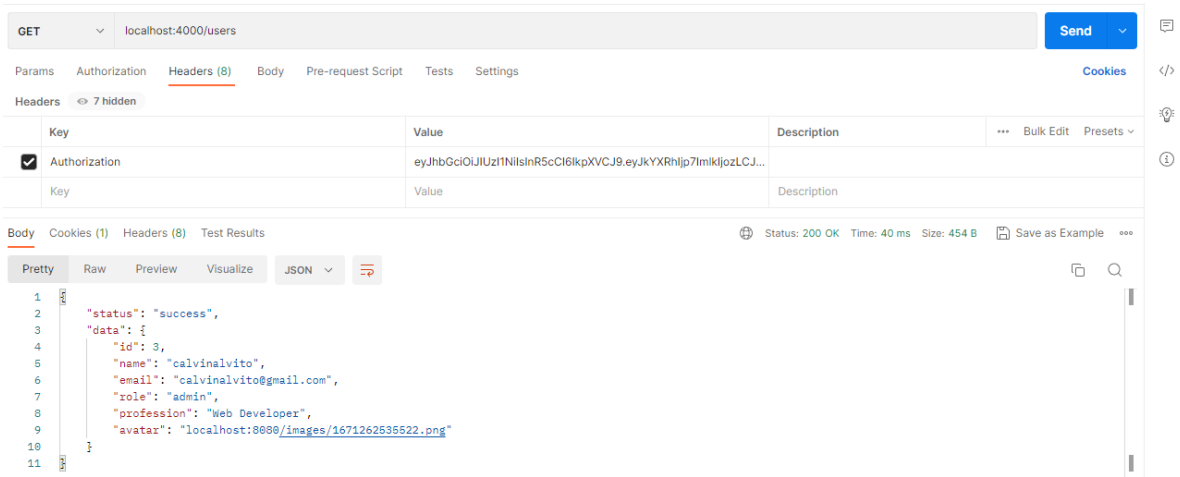
No	Langkah Pengujian	Kondisi Pengujian	Harapan	Hasil
1	Admin masuk ke sistem	*Email* dan *password* benar	Sistem akan menerima akses *login* dan berhasil masuk ke halaman admin.	Valid
2	Admin gagal masuk ke sistem	*Email* dan *password* salah	Sistem akan gagal login dan terdapat *error message.*	Valid
3	Admin melakukan tambah, ubah dan hapus data *service* media.	Admin menambah, mengubah dan menghapus data *service* media.	Admin berhasil menambah, mengubah dan menghapus data *service* media.	Valid
4	Admin melakukan tambah, ubah dan hapus data *service course.*	Admin menambah, mengubah dan menghapus data *service course.*	Admin berhasil menambah, mengubah dan menghapus data *service course.*	Valid
5	Admin mengelola data *service order & payment.*	Admin mengelola data service data service *order & payment.*	Admin berhasil mengelola data *service order & payment.*	Valid

3.2.2. Pengujian Integrasi

Pengujian integrasi memeriksa *log* status melalui layanan API, menampilkan respon dan *error message* jika terjadi kesalahan. Melibatkan lima layanan: API *Gateway*, *Service User*, *Service* Media, *Service Course*, dan *Service Order & Payment*. Ada dua jenis pengujian: dengan kedua layanan aktif atau salah satu layanan *offline*. Pengujian ini bertujuan untuk memastikan bahwa semua layanan dapat bekerja sama secara efektif dan dapat mengidentifikasi masalah integrasi jika terdapat kesalahan lewat error message.

3.2.2.1. Integrasi API Gateway dengan Service User

Pada Gambar 13, dilakukan pengujian integrasi API *Gateway* dengan *service user* dalam kondisi kedua *service* tersebut *online*.



Gambar 13. Integrasi API Gateway dengan Service User

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

Gambar 13 menunjukkan bahwa *Service User* digunakan oleh API *Gateway* untuk mengambil detail data pengguna yang melakukan login. Detail spesifik dari *output* REST API yang terlihat pada Gambar 13 adalah sebagai berikut:

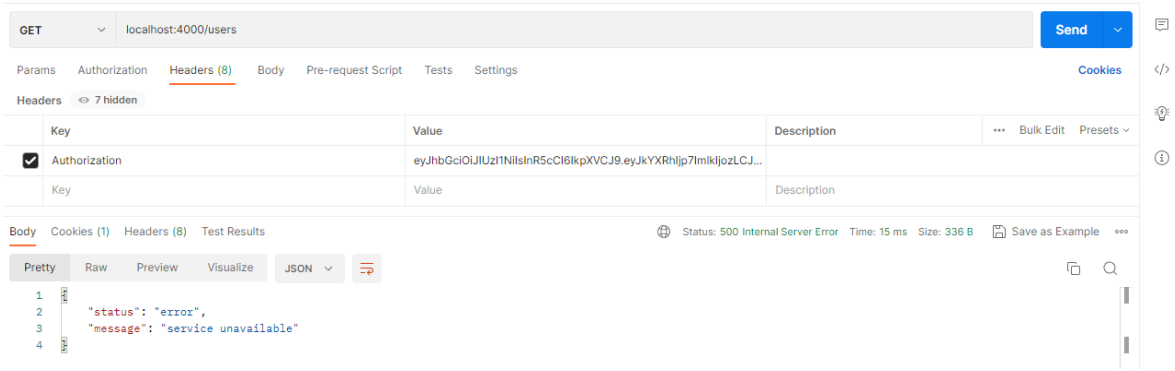
Permintaan HTTP :

a. Metode *: GET*
b. URI : http://localhost:4000/users
c. *Header : Authorization*

Respon *:*

a. Kode Respon : 200
b. Waktu : 40 *ms*
c. *Body Response* : data.id, data.*name*, data.email, data.*role*, data.*profession*, data.avatar.

Berikut adalah pengujian yang dilakukan dengan kondisi offline pada service user, seperti yang ditunjukkan pada Gambar 14 di bawah ini:



Gambar 14. Integrasi API Gateway dengan Service User (Offline)

Gambar 14 menampilkan API *Gateway* melakukan permintaan (*request*) ke *service user* dalam kondisi *offline*. *Output* REST API pada Gambar 14 dijelaskan secara detail sebagai berikut:

Permintaan HTTP :

a. Methode *: GET*
b. *URI : http://localhost:4000/users*
c. *Header : Authorization*

Respon *:*

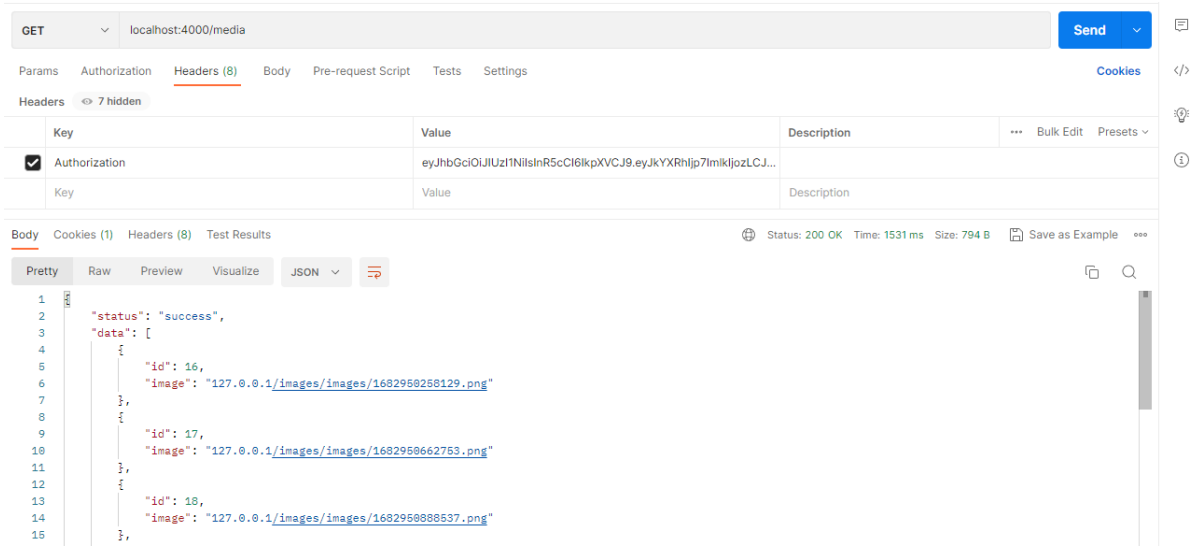
a. Kode respon : 500
b. Waktu : *15 ms*
c. *Body Response* : Mendapatkan status *error* dengan e*rror message "Service Unavailable"*.

Berdasarkan dua pengujian API *Gateway* dengan *service user* dengan kedaan *online* pada Gambar 13. dan yang kedua pengujian API *Gateway online* dan *service user offline* pada Gambar 14. telah sesuai dengan yang diharapkan ketika keduanya *online* maka dapat menerima *body response* data yang sesuai yang direquest dan jika salah satu *offline* maka akan mendapatkan *error message*.

3.2.2.2. Integrasi API Gateway dengan Service Media

Pada Gambar 15, terdapat pengujian integrasi antara API *Gateway* dan *Service* Media dengan syarat bahwa kedua *service* harus dalam kondisi online.

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591



Gambar 15. Integrasi API Gateway dengan Service Media

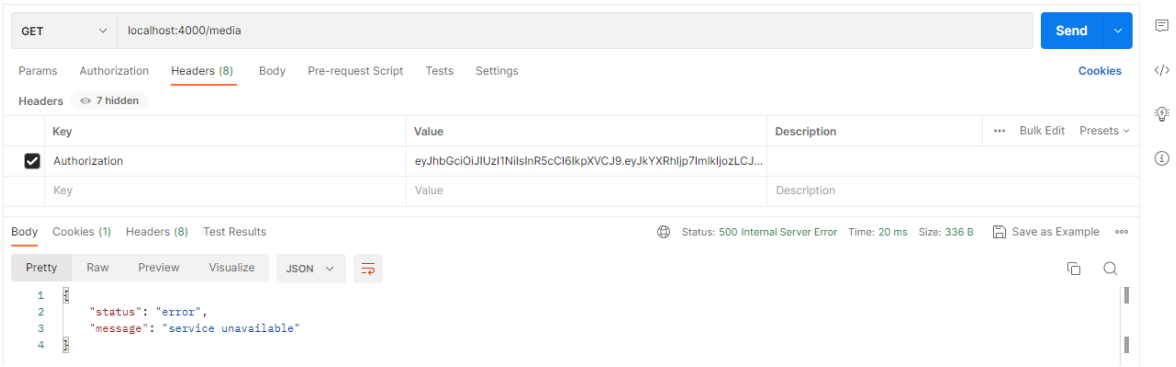
Gambar 15 menunjukkan bahwa Service *Media* di-*request* oleh API *Gateway* untuk mendapatkan semua data media. *Output* REST API dari Gambar 15 dijelaskan secara detail sebagai berikut::
Permintaan HTTP :

a. Metode : *GET*
b. *URI : http://localhost:4000/media*
c. *Header : Authorization*

Respon :

a. Kode Respon : 200
b. Waktu : *1531 ms*
c. *Body Response* : data.id, data.*image*.

Berikut adalah pengujian yang dilakukan saat kondisi layanan media offline seperti yang ditunjukkan pada Gambar 16.



Gambar 16. Integrasi API Gateway dengan Service Media (Offline)

Gambar 16 menunjukkan bahwa dalam situasi ketika *service* media dalam kondisi *offline*, API *Gateway* melakukan permintaan kepada *service* tersebut. Rincian hasil dari REST API pada Gambar 16 sebagai berikut :
Permintaan HTTP :

a. Metode *: GET*
b. *URI : http://localhost:4000/media*
c. *Header : Authorization*

Respon :

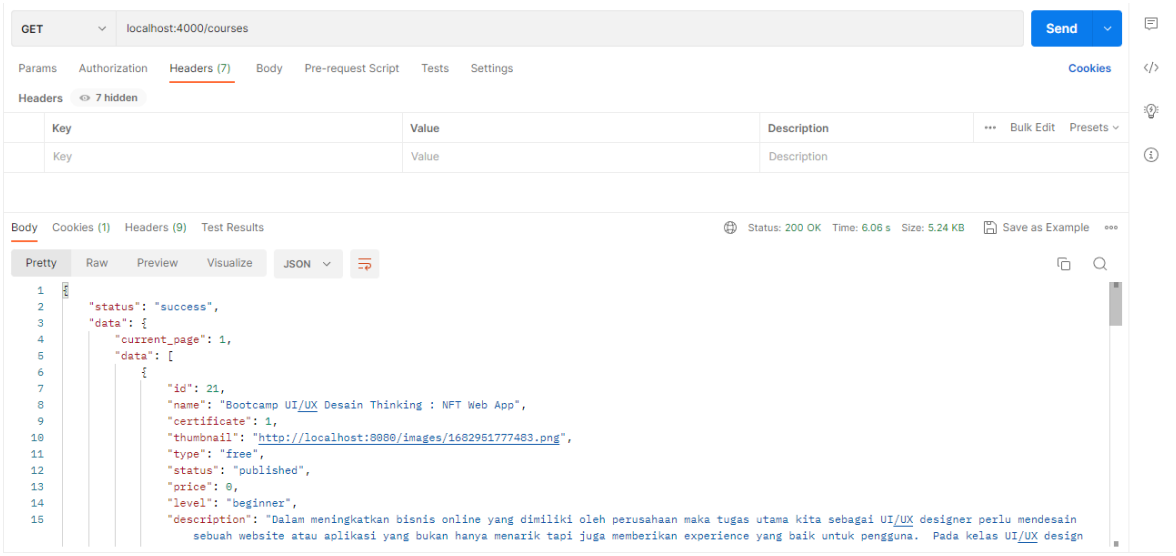
DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

a. Code Respon : *500*
b. Waktu : *20 ms*
c. *Body Response* : Mendapatkan status *error* dengan *error message "Service Unavailable"*.

Berdasarkan dua pengujian API *Gateway* dengan *service* media dengan kedaan *online* pada Gambar 15. dan yang kedua pengujian API *Gateway online* dan *service* media *offline* pada Gambar 16. telah sesuai dengan yang diharapkan ketika keduanya *online* maka dapat menerima *body response* data yang sesuai yang di *request* dan jika salah satu *offline* maka akan mendapatkan *error message*.

3.2.2.3. Integrasi API Gateway dengan Service Course

Pada Gambar 17, ditampilkan pengujian integrasi API *Gateway* ke *Service Course* dimana kedua layanan tersebut berada dalam kondisi *online*.

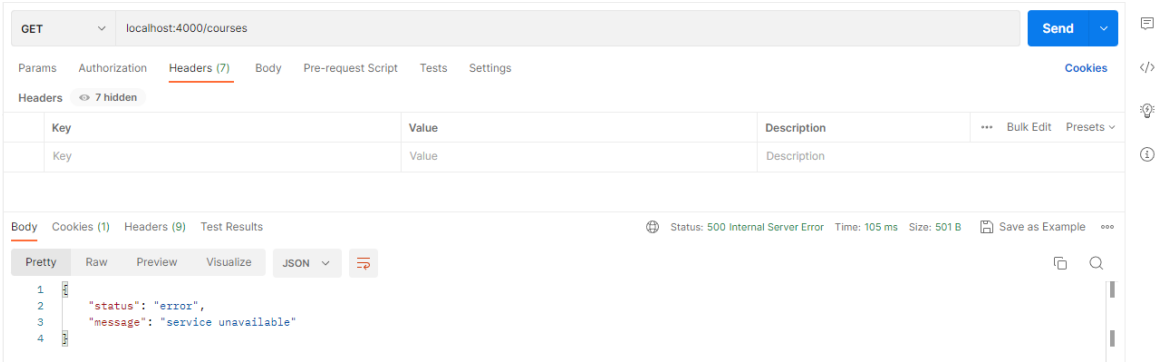


Gambar 17. Integrasi API Gateway dengan Service Course

Gambar 17 menunjukkan bahwa *Service Course* menerima permintaan dari API *Gateway* untuk mengambil seluruh data *course*. Berikut ini adalah detail *output* dari REST API yang ditunjukkan pada Gambar 17 di atas:
Permintaan HTTP :
a. Metode*: GET*
b. *URI : http://localhost:4000/couses*
c. *Header : Authorization*
Respon :
a. Code Respon : *200*
b. Waktu : *40 ms*
c. *Body Response* : data.id, data.*name*, data.*certificate*, data.*thumbnail*, data.*type*, data.status, data.*price*, data.level dan data.*description*.

Berikut adalah pengujian pada Gambar 18 yang dilakukan saat kondisi *service course* sedang *offline*.

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591



Gambar 18. Integrasi API Gateway dengan Service Course (Offline)

Gambar 18 menunjukkan bahwa pada saat *service course* sedang *offline*, API *Gateway* melakukan permintaan ke *service course*. Detail dari keluaran REST API pada Gambar 18 seperti berikut :
Permintaan HTTP :

a. Methode : *GET*
b. *URI : http://localhost:4000/courses*
c. *Header : Authorization*

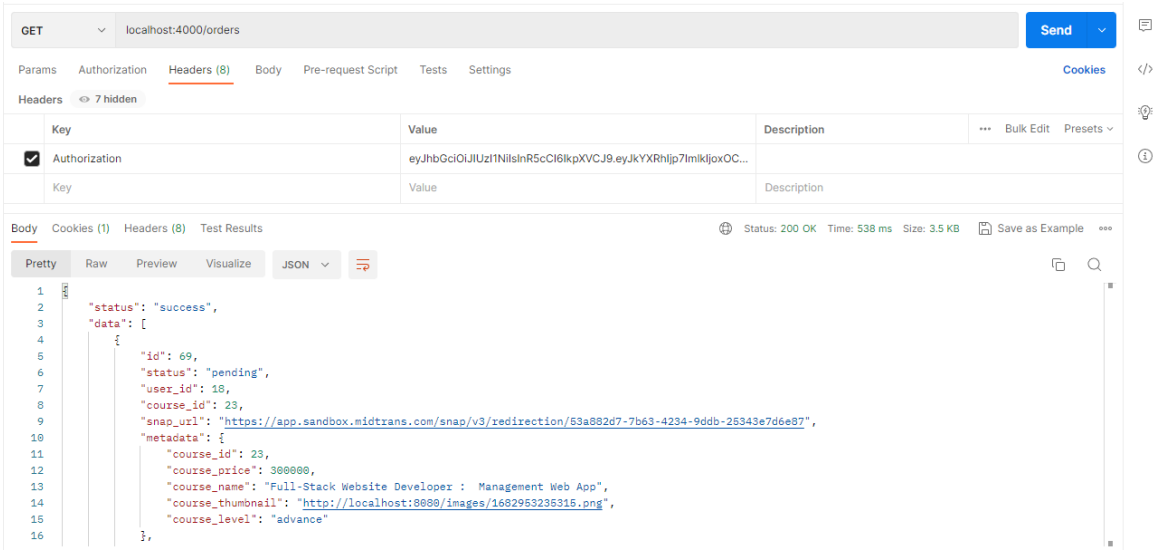
Respon :

a. Code Respon : 500
b. Waktu : *105 ms*
c. *Body Response* : Mendapatkan status *error* dengan *error message "Service Unavailable".*

Berdasarkan dua pengujian API *Gateway* dengan *service course* dengan kedaan *online* pada Gambar 17. dan yang kedua pengujian API *Gateway online* dan *service course offline* pada Gambar 18. telah sesuai dengan yang diharapkan ketika keduanya *online* maka dapat menerima *body response* data yang sesuai yang di *request* dan jika salah satu *offline* maka akan mendapatkan *error message.*

3.2.2.4. Integrasi API Gateway dengan Service Order & Payment

Gambar 19 menunjukkan pengujian integrasi antara API *Gateway* dan *Service Order & Payment* saat kedua layanan berada dalam kondisi *online*.



Gambar 19. Integrasi API Gateway dengan Service Order & Payment

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

Gambar 19 menunjukkan bahwa API *Gateway* melakukan permintaan ke *Service Order & Payment* untuk memperoleh semua data *order*. Detail *output* REST API dari gambar tersebut adalah sebagai berikut:

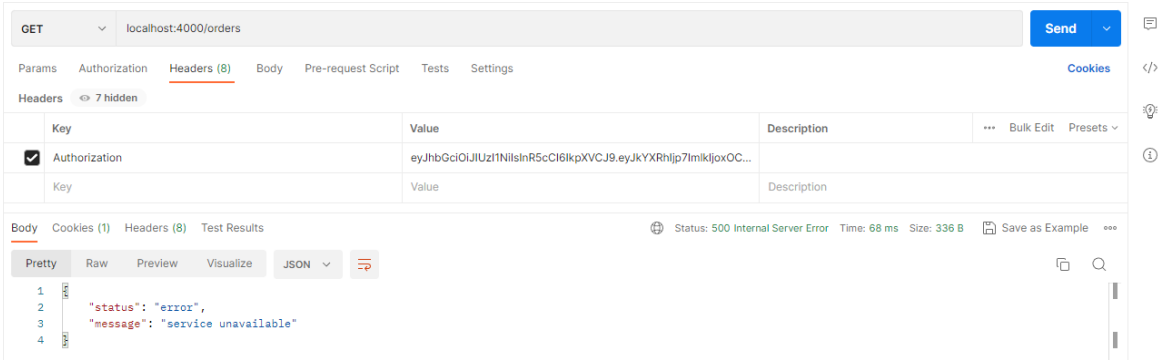
Permintaan HTTP :

a. Methode : *GET*
b. *URI : http://localhost:4000/orders*
c. *Header : Authorization*

Respon :

a. Kode Respon : *200*
b. Waktu : *538 ms*
c. *Body Response* : data.id, data.status, data.*user*_id, data.*course*_id, data.*snap_url*.

Berikut ini adalah pengujian yang dilakukan pada kondisi ketika *service order & payment* dalam keadaan *offline* seperti yang terlihat pada Gambar 20.



Gambar 20. Integrasi API Gateway dengan Service Order & Payment (Offline)

Pada Gambar 20 terlihat bahwa API *Gateway* mengirimkan permintaan ke *service order & payment* ketika layanan pengguna sedang offline. Informasi rinci mengenai *output* REST API pada Gambar 20 seperti berikut :

Permintaan HTTP :

a. *Methode* : GET
b. *URI : http://localhost:4000/orders*
c. *Header : Authorization*

Respon :

a. Kode Respon : 500
b. Waktu : 68 *ms*
c. *Body Response* : Mendapatkan status *error* dengan *error message "Service Unavailable"*.

Berdasarkan dua pengujian API *Gateway* dengan *service order & payment* dengan kedaan *online* pada Gambar 19. dan yang kedua pengujian API *Gateway online* dan *service order & payment offline* pada Gambar 20. telah sesuai dengan yang diharapkan ketika keduanya *online* maka dapat menerima *body response* data yang sesuai yang di *request* dan jika salah satu *offline* maka akan mendapatkan *error message*.

4. KESIMPULAN

Hasil penelitian dan pengujian menunjukkan bahwa arsitektur *microservice* telah berhasil dikembangkan untuk *learning management system e-learning* yang terdiri dari beberapa layanan, termasuk service api-*gateway*, *service user*, *service* media, *service course*, dan *service order & payment*. Pengujian menggunakan *black box* membuktikan bahwa sistem *back-end* yang menggunakan arsitektur *microservice* berfungsi dengan baik. Selain itu, pengujian integrasi antara API *Gateway* dan layanan lainnya juga berhasil, di mana empat skenario koneksi antara layanan *online* menghasilkan kode respon 200 dan *response body* data yang sesuai, sedangkan empat skenario ketika salah satu layanan *offline*

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

menghasilkan *response code* 500 dan pesan *error "Service Unavailable"*. Sehingga dapat disimpulkan bahwa sistem layak untuk digunakan.

DAFTAR PUSTAKA

[1] S. Suwarno, "Application of the UTAUT Model for Acceptance Analysis of COBIT Implementation in E-Learning Management with Microsoft Teams on Distance Learning in Batam City," *Khazanah Inform. J. Ilmu Komput. dan Inform.*, vol. 8, no. 1, pp. 25–33, Mar. 2022, doi: 10.23917/KHIF.V8I1.15311.

[2] D. Priyawati, "Penerapan E-Learning Untuk Menunjang Kegiatan Belajar Mengajar Bagi Guru Di Lingkungan Pcm Kartasura," *Abdi Teknoyasa*, pp. 13–16, Jul. 2020, doi: 10.23917/abditeknoyasa.v1i1.48.

[3] F. A. Muqtadiroh, A. Herdiyanti, and N. Puspitasari, "The e-Learning Quality Model to Examine Students' Behavioral Intention to Use Online Learning Platform in a Higher Education Institution," *Khazanah Inform. J. Ilmu Komput. dan Inform.*, vol. 6, no. 2, Oct. 2020, doi: 10.23917/KHIF.V6I2.11344.

[4] D. J. Riyanto, P. Pizaini, N. S. H., and M. Affandes, "Implementasi Service Choreography Pattern Arsitektur Microservice Classroom Akademik Menggunakan Docker," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 7, no. 3, pp. 768–779, Aug. 2022, doi: 10.29100/jipi.v7i3.3126.

[5] M. Dudjak and G. Martinović, "An API-first methodology for designing a microservice-based Backend as a Service platform," *Inf. Technol. Control* , vol. 49, no. 2, pp. 206–223, Sep. 2020, doi: 10.5755/J01.ITC.49.2.23757.

[6] R. Mufrizal and D. Indarti, "Refactoring Arsitektur Microservice Pada Aplikasi Absensi PT. Graha Usaha Teknik," *J. Nas. Teknol. dan Sist. Inf.*, vol. 5, no. 1, pp. 57–68, Apr. 2019, doi: 10.25077/teknosi.v5i1.2019.57-68.

[7] H. Suryotrisongko, "Arsitektur Microservice untuk Resiliensi Sistem Informasi," *Sisfo*, vol. 06, no. 02, pp. 231–246, Jan. 2017, doi: 10.24089/j.sisfo.2017.01.006.

[8] D. A. Bauer Hochschule Emden, M. Assaad, D. Alessandro Bauer, B. Penz, and J. Mäkiö, "Improvement of an Existing Microservices Architecture for an E-learning Platform in STEM Education STIMEY View project Actor4j an actor-oriented Java framework View project Improvement of an Existing Microservices Architecture for an E-learning Platform i," 2018, Accessed: May 09, 2023. [Online]. Available: https://www.researchgate.net/publication/327018933

[9] A. Milovanović, "Microservice architecture in E-learning," *E-bus. Technol. Conf. Proc.*, vol. 1, no. 1, pp. 105–108, Sep. 2021, Accessed: Sep. 29, 2022. [Online]. Available: https://ebt.rs/journals/index.php/conf-proc/article/view/81

[10] P. Niemelä and H. Hyyrö, "Migrating learning management systems towards microservice architecture," *CEUR Workshop Proc.*, vol. 2520, pp. 10–20, 2019, Accessed: May 08, 2023. [Online]. Available: https://trepo.tuni.fi/handle/10024/129800

[11] U. Syarif and P. Pizaini, "Penerapan Event-Driven Microservices Pada Aplikasi Layanan Penerimaan Peserta Didik Baru," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 7, no. 3, pp. 745–756, Aug. 2022, doi: 10.29100/jipi.v7i3.3067.

[12] R. Setyawan and M. Maryam, "Sistem Informasi Penjualan Alat Elektronik Berbasis Web Pada Toko Mandiri Elektronik Purwantoro," *J. Digit. Teknol. Inf.*, vol. 4, no. 1, p. 8, Mar. 2021, doi: 10.32502/digital.v4i1.3071.

[13] M. M. Gultom and Maryam, "Sistem Informasi Penjualan Material Bangunan Pada Toko Bangunan Berkah," *J. Tek. Inform.*, vol. 1, no. 2, pp. 79–86, Dec. 2020, doi: 10.20884/1.jutif.2020.1.2.19.

[14] Y. A. Palamarchuk, "Methods of Building Microservice Architecture of E-Learning Systems,"

DOI: https://doi.org/10.54082/jiki.102
P-ISSN: 2807-6664
E-ISSN: 2807-6591

Inf. Technol. Comput. Eng., vol. 53, no. 1, pp. 43–54, Feb. 2022, doi: 10.31649/1999-9941-2022-53-1-43-54.

[15] A. A. Arif, D. Afriyanti, and P. Putri, "Perancangan Dan Implementasi Web Penjualan Pada Toko Juragan Laptop Second Pati," *Emit. J. Tek. Elektro*, vol. 1, no. 1, pp. 56–65, Mar. 2023, doi: 10.23917/EMITOR.V1I1.21300.

[16] N. B. Naseri and N. Nurgiyatna, "Sistem Informasi Pemesanan Makanan Berbasis Client Server di Kopi We Salatiga," *JITU J. Inform. Technol. Commun.*, vol. 5, no. 1, pp. 1–12, Jul. 2021, doi: 10.36596/JITU.V5I1.497.

[17] C. Seviro, B. Sakti, and I. Hermawan, "Implementasi Arsitektur Microservice pada Back End Sistem Informasi Atlantas berbasis Website," *J. Teknol. Terpadu*, vol. 6, no. 2, pp. 96–104, Dec. 2020, doi: 10.54914/JTT.V6I2.281.