IMPLEMENTASI METODE TEST DRIVEN DEVELOPMENT PADA PENGEMBANGAN WEB SERVICE DENGAN GAYA ARSITEKTUR REST (STUDI KASUS: APLIKASI WEB THE-SKILLS.ID)

Skripsi

Diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk menyelesaikan pendidikan Sarjana (S1) Program Studi Teknologi Informasi Fakultas Ilmu Komputer Universitas Jember dan mencapai gelar Sarjana Komputer

Oleh

Denta Firdaus Fatoni Maulana

NIM 182410102056

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI UNIVERSITAS JEMBER
FAKULTAS ILMU KOMPUTER
PROGRAM STUDI TEKNOLOGI INFORMASI
JEMBER
2023

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Allah SWT yang telah memberikan rizki dalam pengerjaan skripsi ini;
2. Almarhumah ibu, bapak, kakak dan semua orang yang telah memberikan dukungan;
3. Almamater Program Studi Teknologi Informasi Fakultas Ilmu Komputer Universitas Jember;
4. Tim startup theskills.

MOTTO

Fortis Fortuna Adiuvat [1]

[1] Alexandria Infante. 2018. *Fortis Fortuna Adiuvat: Fortune Favors the Bold.* English: Independently Published.

PERNYATAAN ORISINALITAS

Saya yang bertanda tangan di bawah ini:

Nama : Denta Firdaus Fatoni Maulana

NIM : 182410102056

Menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "*Implementasi Metode Test Driven Development pada Pengembangan Web Service dengan Gaya Arsitektur REST (Studi Kasus : Aplikasi Web the-skills.id)"* adalah benar-benar hasil karya saya sendiri, kecuali jika dalam pengutipan substansi disebutkan sumbernya, belum pernah diajukan pada instansi manapun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika dikemudian hari pernyataan ini tidak benar.

Jember, Juli 2023

Yang menyatakan,

Denta Firdaus Fatoni Maulana
NIM 182410102056

HALAMAN PERSETUJUAN

Skripsi berjudul *"Implementasi Metode Test Driven Development pada Pengembangan Web Service dengan Gaya Arsitektur REST (Studi Kasus : Aplikasi Web the-skills.id)"* telah diuji dan disahkan oleh Fakultas Ilmu Komputer Universitas Jember pada:

Hari : Selasa

Tanggal : 11 Juli 2023

Tempat : Fakultas Ilmu Komputer Universitas Jember

Pembimbing Tanda Tangan

1. Pembimbing Utama
   Nama : Anang Andrianto, ST., MT
   NIP : 196906151997021002 (.....)
2. Pembimbing Anggota
   Nama : Dr. Dwiretno Istiyadi S.ST., M.Kom
   NIP : 197803302003121003 (.....)

Penguji

1. Penguji Utama
   Nama : Windi Eka Yulia Retnani, S.Kom., MT
   NIP : 198403052010122002 (.....)
2. Penguji Anggota 1
   Nama : Tri Agustina Nugrahani, S.Kom., M.Kom
   NIP : 199208222022032014 (.....)

ABSTRAK

Penelitian ini bertujuan untuk menerapkan metode pengembangan aplikasi web service berbasis *Test Driven Development* (TDD) pada aplikasi web the-skills.id. Sebelumnya, aplikasi web the-skills.id telah dikembangkan menggunakan metode *Test Last Development* (TLD) Namun, seiring dengan perkembangan teknologi dan kebutuhan yang semakin kompleks, beberapa kelemahan atau keterbatasan dari metode *Test Last Development* mulai terlihat. Hal ini menimbulkan kebutuhan untuk mencari alternatif metode pengembangan yang dapat mengatasi kekurangan yang ada dan meningkatkan kualitas perangkat lunak yang dihasilkan. Metode *Test Driven Development* (TDD), sebagai metode pengembangan alternatif, telah menarik perhatian banyak peneliti dan praktisi dalam industri perangkat lunak. Metode *Test Driven Development* menawarkan pendekatan yang berbeda dalam pengembangan perangkat lunak dengan fokus pada kelebihan dan keunggulan tertentu.

Oleh karena itu, metode TDD dipilih sebagai pendekatan alternatif dalam pengembangan aplikasi web the-skills.id. TDD menekankan pembuatan skenario pengujian atau test case sebelum menulis kode fungsional program, yang kemudian diikuti dengan proses refactor. Tujuannya adalah untuk meningkatkan kualitas aplikasi, mendorong pengembang untuk menulis kode yang lebih baik, dan memberikan umpan balik sebelum implementasi kode.

Penelitian ini akan membandingkan hasil pengembangan aplikasi menggunakan metode TDD dengan metode TLD. Diharapkan penggunaan TDD dapat menghasilkan aplikasi yang bebas bug, meningkatkan cakupan kode (*code coverage*), dan mengurangi jumlah cacat (*number of defect*) dibandingkan dengan pengembangan aplikasi dengan metode TLD.

Hasil penelitian ini diharapkan akan memberikan manfaat bagi para pengembang aplikasi dalam industri. Metode *Test Driven Development* menunjukkan keunggulan untuk mengurangi *defect* sedangkan dari segi *code coverage* metode *Test Last Development* memiliki rerata kualitas *code coverage* yang lebih baik.

Kata kunci: *Test Driven Development*, *Code Coverage*, *Number of Defects*

ABSTRACT

This research aims to implement the Test Driven Development (TDD) method in the development of the web application for the-skills.id. Previously, the web application for the-skills.id was developed using the Test Last Development (TLD) method. However, as technology advances and the complexity of requirements increases, some limitations and weaknesses of the Test Last Development method have become apparent. This has created a need to find an alternative development method that can overcome these limitations and improve the quality of the software produced. The Test Driven Development (TDD) method, as an alternative development method, has attracted the attention of many researchers and practitioners in the software industry. The Test Driven Development method offers a different approach to software development with specific advantages and strengths.

Therefore, the TDD method was chosen as an alternative approach in the development of the web application for the-skills.id. TDD emphasizes the creation of test scenarios or test cases before writing functional code, followed by the refactoring process. The goal is to improve the quality of the application, encourage developers to write better code, and provide feedback before the implementation of the code.

This research will compare the results of developing the application using the TDD method with the TLD method. It is expected that the use of TDD can result in a bug-free application, increase code coverage, and reduce the number of defects compared to application development using the TLD method.

The results of this research are expected to provide benefits to application developers in the industry. The Test Driven Development method demonstrates advantages in reducing defects, while in terms of code coverage, the Test Last Development method has shown better average code coverage quality.

Keywords: *Test Driven Development*, *Code Coverage*, *Number of Defects*

RINGKASAN

Implementasi Metode Test Driven Development pada Pengembangan Web Service dengan Gaya Arsitektur REST (Studi Kasus : Aplikasi Web the-skills.id); Denta Firdaus Fatoni Maulana, 182410102056, 2023; 51 halaman, Program Studi Teknologi Informasi Fakultas Ilmu Komputer Universitas Jember.

Pengembangan menggunakan metode *Test Last Development* pada aplikasi web the-skills.id memiliki kualitas perangkat lunak yang kurang baik sebab menimbulkan banyak kegagalan aplikasi setelah di*deploy.* Banyak penelitian menyimpulkan metode *Test Driven Development* dapat meningkatkan kualitas perangkat lunak dan dapat mengurangi jumlah bug pada kemudian hari. Karena alasan tersebut metode *Test Driven Development* akan digunakan untuk pengembangan web service pada startup the-skills.id.

Hasil perbandingan metode *Test Driven Development* dan *Test Last Development* menggunakan dua *quality factor* yaitu *number of defects* dan *code coverage*. Pendekatan *Test Driven Development* menunjukkan keunggulan pada segi *external quality* yaitu memunculkan lebih sedikit *defect* dengan jumlah 80 *defects* pada saat masa pengembangan dibandingkan dengan *Test Last Development* yang menimbulkan lebih banyak defect dengan jumlah 90 *defects*. Adapun dari segi *code coverage* metode *Test Last Development* memiliki rerata kualitas *code coverage* yang lebih baik yakni 86,23% sedangkan *Test Driven Development* 71,7%.

PRAKATA

Puji syukur kehadirat Allah Yang Maha Esa atas segala rahmat dan nikmat-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul "*Implementasi Metode Test Driven Development pada Pengembangan Web Service dengan Gaya Arsitektur REST (Studi Kasus : Aplikasi Web the-skills.id)"*. Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan Strata Satu (S1) pada Program Studi Teknologi Informasi Fakultas Ilmu Komputer Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Drs. Antonius Cahya Prihandoko, M.App.Sc, Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Jember;
2. Gayatri Dwi Santika, S.SI., M.Kom selaku Dosen Pembimbing Akademik;
3. Anang Andrianto, ST., MT selaku Dosen Pembimbing Utama dan Dr. Dwiretno Istiyadi S.ST., M.Kom selaku Dosen Pembimbing Pendamping yang senantiasa sabar meluangkan waktu, nasehat dan tenaga serta petunjuk selama penyusunan skripsi ini;
4. Segenap civitas akademik Fakultas Ilmu Komputer Universitas Jember;
5. Teman-teman saya yang sering berbagi keluh kesah;
6. CEO startup theskills Cintania Syurga Alifa beserta tim;
7. Semua pihak yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih terdapat banyak kekurangan dan penulis mengharapkan adanya masukan yang bersifat membangun. Penulis berharap skripsi ini mampu memberikan manfaat bagi semua pihak.

Jember, 2023

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Seiring berkembangnya zaman, kebutuhan teknologi di setiap lini masyarakat semakin meningkat baik dari perusahaan, usaha mandiri, instansi pemerintah, dan bisnis digital. Web *Service* adalah salah satu jenis teknologi yang sering digunakan dalam pengembangan perangkat lunak dari segala jenis *platform* yang ada. Web *service* merupakan kumpulan dari berbagai layanan berbasis web dengan menggunakan protokol komunikasi HTTP, layanan tersebut dapat dimanfaatkan dan diolah oleh pengguna dengan jenis bahasa pemrograman, model arsitektur dan platform yang berbeda (*interoperability*). Web *service* dapat dikembangkan dengan model arsitektur *Simple Object Access Protocol* (SOAP) atau *Representational State Transfer* (REST), secara umum web *service* menggunakan JSON atau XML dalam proses transaksi dan penyimpanan data (Rizal & Rahmatulloh, 2019).

Salah satu arsitektur aplikasi web yang sedang banyak digunakan saat ini adalah REST atau *Representational State Transfer*. REST merupakan arsitektur web *service* yang bersifat *client server* dimana client melakukan *request* kepada server kemudian *server* memproses request dan mengembalikan *response* (Wardhana et al., 2020). Dalam proses *request, client* akan mengirimkan data yang berupa URI, *request header, method* (GET, POST, PUT, DELETE dll) dan *request body*. Lalu, *server* menerima *request* dari *client*. Berikutnya, *server* memberikan *response* kepada *client* berupa *status code response, response header* dan *response body*.

Salah satu *platform* yang memiliki layanan berupa web *service* adalah *startup* pendidikan digital yaitu the-skills.id. The-skills.id merupakan salah satu layanan berbasis web yang dikelola oleh startup TheSkills. TheSkills adalah *startup* yang menawarkan kursus-kursus pendidikan khusunya untuk anak- anak dan orangtua dengan metode *blended learning* untuk model pembelajarannya. *Blended learning* adalah sebuah tipe pembelajaran yang memadukan antara pembelajaran secara langsung atau tatap muka dengan pembelajaran berbasis teknologi informasi atau online. *Blended learning* adalah inovasi baru dalam proses belajar mengajar dimana

kegiatan antara murid dan guru dapat berlangsung di kelas atau *online* (Wardani et al., 2018).

Dalam pengembangan perangkat lunak sebelumnya, metode *Test Last Development* (TLD) telah digunakan sebagai pendekatan utama. Metode tersebut memiliki kelebihan tertentu dan telah terbukti berhasil digunakan dalam beberapa proyek perangkat lunak. Namun, seiring dengan perkembangan teknologi dan kebutuhan yang semakin kompleks, beberapa kelemahan atau keterbatasan dari metode *Test Last Development* mulai terlihat. Hal ini menimbulkan kebutuhan untuk mencari alternatif metode pengembangan yang dapat mengatasi kekurangan yang ada dan meningkatkan kualitas perangkat lunak yang dihasilkan. Metode *Test Driven Development* (TDD), sebagai metode pengembangan alternatif, telah menarik perhatian banyak peneliti dan praktisi dalam industri perangkat lunak. Metode *Test Driven Development* menawarkan pendekatan yang berbeda dalam pengembangan perangkat lunak dengan fokus pada kelebihan dan keunggulan tertentu. Kebutuhan user yang semakin tinggi dibutuhkan pengembangan ataupun perbaikan perangkat lunak dilakukan secara cepat dan efisien maka dibutuhkan skenario yang tepat dalam implementasi suatu metodologi pengembangan perangkat lunak.

Test Driven Development adalah metode yang akan digunakan untuk mengembangkan RESTful web *service*. *Test driven development* merupakan metode untuk mengembangkan aplikasi atau *software* yang langkah mengedepankan dalam pembuatan *test case* terlebih dahulu lalu menuliskan kode program dan *refactoring* (Thohari & Amalia, 2018). Menurut (Baldassarre et al., 2021), pengembangan perangkat lunak dengan menerapkan metode *Test Driven Development* dapat meningkatkan kualitas produk perangkat lunak baik dari segi fungsi atau dan kualitas internal serta meningkatkan *productivity* dari sisi pengembang. Selain itu, penerapan skenario *Test Driven Development* lebih memudahkan proses penggabungan unit karena setiap unit akan diuji dan dipastikan berjalan dengan baik sebelum digabungkan dengan unit yang lain. Manfaat yang lain dari menerapkan *Test Driven Development* adalah mudah dalam penelusuran penyebab bug pada sebuah unit atau sistem (Jonathan et al., 2021). Dengan

menggunakan metode *Test Driven Development* diharapkan menghasilkan sebuah web *service* yang bebas *bug* dan sesuai dengan bisnis proses yang diinginkan dan mempercepat masa pengembangan dari web *service* lebih dari pengembangan secara konvensional.

Berdasarkan permasalahan diatas, dalam penelitian ini peneliti mencoba menggunakan metode *Test Driven Development* untuk diimplementasikan dalam pengembangan RESTful web *service* pada aplikasi web the-skills.id dan mengukur perbedaan kualitas kode antara metode *Test Last Development* dengan *Test Driven Development*.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas, dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana hasil implementasi metode *Test Driven Development* pada RESTful web *service* untuk aplikasi web the-skills.id pada startup The skills?
2. Bagaimana perbedaan kualitas kode antara pengembangan *Test Last Development* dengan pengembangan dengan metode *Test Driven Development*?

1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini yaitu:

1. Mengimplementasikan strategi metode pengembangan perangkat lunak *Test Driven Development* web *service* pada startup The skills.
2. Membandingkan kualitas perangkat lunak antara pengembangan *Test Last Development* dengan pengembangan menggunakan metode *Test Driven Development*.

1.4 Manfaat

Manfaat dari tercapainya penelitian ini yaitu:

1. Bagi peneliti sebagai bahan penyusunan tugas akhir dan juga untuk menerapkan ilmu yang telah didapatkan selama masa kuliah.
2. Bagi objek penelitian untuk dapat membantu meningkatkan kualitas produk dalam hal ini aplikasi web the-skills.id.

1.5 Batasan Masalah

Adapun beberapa batasan masalah dalam penelitian ini antara lain:

1. Objek penelitian yang akan dikembangkan adalah aplikasi web dari *startup* Theskills.
2. Pengembangan aplikasi hanya pada sisi *server* dan tidak meliputi tampilan antarmuka pengguna.

BAB 2. TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian yang dilakukan oleh (Thohari & Amalia, 2018) dengan judul "*Implementasi Test Driven Development Dalam Pengembangan Aplikasi Berbasis Web*". Pada penelitian ini metode *Test Driven Development* diimplementasikan terhadap aplikasi web *rating* dan review film-film Indonesia yang dibangun menggunakan bahasa pemrograman ruby dengan *framework* rails. Siklus pengembangan aplikasi web dimulai dengan pembuatan dan menjalankan test *case*, melakukan implementasi program dan terakhir melakukan *refactoring*. Adapun *tools* pengujian unit yang digunakan adalah *tools* bawaan framework rails yaitu Rspec. Penerapan metode *Test Driven Development* membantu dalam penerapan proses bisnis melalui *test case* dan kebutuhan fitur dari pengguna aplikasi IMDB. Selain itu, pengembang mudah dalam menemukan dan memperbaiki kesalahan atau bug. Kekurangan dari *Test Driven Development* yaitu pengujian otomatis hanya dilakukan dari segi fungsionalitas sisi server dari sebuah web *service* sedangkan untuk sisi *client* pengetesan dengan cara manual.

Penelitian selanjutnya adalah penelitian yang dilakukan oleh (Rizal & Rahmatulloh, 2019) dengan judul "*Restful Web Service Untuk Integrasi Sistem Akademik Dan Perpustakaan Universitas Perjuangan*". Penelitian ini merupakan penerapan web *service* bergaya arsitektur REST dengan tujuan untuk mengintegrasikan sistem informasi akademik dan sistem informasi perpustakaan Universitas Perjuangan. Pada penelitian ini belum ada penjelasan yang lebih komprehensif terkait keamanan data dengan sistem otentikasi sehingga perlu ada pembahasan pada penelitian lain.

Berikutnya penelitian yang dilakukan oleh (Alhammad et al., 2016) dengan judul "*Comparison between Test-Driven Development and Conventional Development: A Case Study*". Pada penelitian ini dilakukan komparasi antara metode *Test Driven Development* dengan metode *Test Last Development* dengan menggunakan *code coverage* dan *number of defects* sebagai *quality factor* pengukuran kualitas perangkat lunak. Hasil dari penelitian ini menyimpulkan code

coverage dari kedua metode memiliki kemiripan pada presentasenya sedangkan untuk kriteria *number of defects* metode TDD memiliki lebih banyak defects daripada metode TLD. Namun, metode TDD memiliki *clean code* yang lebih baik daripada metode TLD.

2.2 The Skills

TheSkills merupakan platform pendidikan digital yang memiliki konten-konten berupa video, quiz dan modul di dalam aplikasi web theskills. Aplikasi web ini haruslah memiliki komponen-komponen baik dari segi interaksi dengan user yang baik hingga keamanan data pengguna maupun keamanan konten di dalamnya. Adapun major fitur atau *service* yang akan dikembangkan di dalam aplikasi web theskills antara lain:

2.2.1 Autentikasi

Sistem autentikasi yang akan diterapkan pada aplikasi web theskills adalah sistem berbasis level pengguna. Sebagai contoh user normal hanya akan dapat melihat konten sedangkan user level admin dapat mengelola konten lebih dari user biasa. Sehingga keamanan data di dalam aplikasi web lebih dapat diminimalisir dari penyalahgunaan akses. Fitur lain yang tergabung dalam autentikasi yaitu login, register, reset password dan logout.

2.2.2 Kursus

Aplikasi web theskills memiliki fitur utama dalam bentuk kursus atau kelas online. Kursus akan dimuat dalam bentuk media berupa video, dokumen atau audio.

2.2.3 Quiz

Pada umumnya setiap kursus di aplikasi web ini akan memiliki quiz yang dapat dikerjakan oleh user sebagai acuan hasil belajar tiap user. Quiz dapat berupa pilihan ganda ataupun isian. Output dari quiz ini adalah skor berupa angka dan akan menjadi jejak historis dari masing-masing user pada kursus yang diambil.

2.2.4 Webinar

Aplikasi web theskills memiliki fitur untuk melakukan registrasi webinar yang tergabung dengan theskills. User dapat melihat informasi webinar seperti waktu pelaksanaan, poster dan form untuk pendaftaran. Selain itu, user juga dapat bergabung dengan room meeting webinar langsung melalui aplikasi web.

2.2.5 Artikel

Aplikasi web theskills akan menyediakan artikel yang dibedakan berdasarkan tag-tag tertentu sehingga akan memudahkan user dalam mencari tag artikel yang diinginkan. Dan juga aplikasi web ini akan memiliki fitur search artikel dan juga filter artikel agar lebih user friendly dan mudah digunakan.

2.3 Web *Service*

Web *Service* adalah salah satu jenis teknologi yang sering digunakan dalam pengembangan perangkat lunak dari segala jenis platform yang ada. Web *service* merupakan kumpulan dari berbagai layanan berbasis web dengan menggunakan protokol komunikasi HTTP, layanan tersebut dapat dimanfaatkan dan diolah oleh pengguna dengan jenis bahasa pemrograman, model arsitektur dan platform yang berbeda (*interoperability*) (Rizal & Rahmatulloh, 2019). Web *service* mengadopsi komunikasi *client-server* dan dapat menukar data tanpa harus memiliki database, bahasa pemrograman dan platform apa yang digunakan. Pemanfaatan sumber daya dari Web *Service* dapat dilakukan dari komputer baik dalam sistem operasi windows, linux atau macOS, smartphone, tablet, dan laptop sehingga Web *Service* banyak dimanfaatkan oleh instansi atau perusahaan di dalam sistem mereka karena fleksibilitas penggunaanya.

Web *Service* mampu mengatasi masalah interoperabilitas yang umum menjadi kelemahan dari aplikasi non *webservice* yang hanya dapat berkomunikasi dengan teknologi yang sama. Aplikasi dengan teknologi berbeda dapat berkomunikasi dengan web *service* dari perangkat apapun. Manfaat lain dari penerapan web *service* yaitu kemudahan dalam *maintenance* modul aplikasi karena client maupun server merupakan komponen yang independen.

2.4 *HTTP (Hypertext Transfer Protocol)*

HTTP merupakan salah satu protokol komunikasi pada lapisan aplikasi TCP/IP yang menjadi dasar pada World Wide Web (WWW). HTTP memiliki konsep komunikasi *client-server*, dimana *client* akan mengirim sebuah *request* kepada *server* dan *server* akan memberikan timbal balik berupa sebuah *response*. Menurut (Firdaus et al., 2019), HTTP *request* memiliki komponen yaitu *verb*, URI,

request header dan *request body*. Sedangkan, HTTP *response* memiliki komponen yaitu status atau *response code*, HTTP *version*, *response header* dan *response body*.

Setiap *request* dan *response* dari *client* ke atau dari *server* yang menggunakan protokol HTTP maka akan mengembalikan kode respon sebagai penanda agar dapat diterima oleh *client*. Kode ini menunjukkan status *server* terhadap *resource* yang hendak diakses. Pengelompokkan dari kode status protokol HTTP dapat dilihat pada table berikut:

Tabel 2. 1 HTTP Code Group

Kelompok Kode	**Keterangan**
1xx	Informational response
2xx	Success
3xx	Redirection
4xx	Client Errors
5xx	Server Errors

Kode status yang paling sering muncul dan digunakan pada protocol HTTP adalah sebagai berikut:

Tabel 2. 2 HTTP Status Code

Kode Status	**Nama Status**	**Keterangan**
200	OK	*Request* berjalan sukses.
401	*Unauthorized*	Tidak diperbolehkan mengakses *resource*. Umumnya karena *client* belum memiliki *credential*.
422	*Not validated*	*Form data Client* tidak tervalidasi dengan sukses atas *resource* yang ingin diakses.
404	*Not Found*	*Resource* yang hendak diakses tidak ditemukan.
405	*Method Not Allowed*	*Method* salah, atau *method* tidak diizinkan.
500	*Internal Server Error*	Terjadi kesalahan atau error pada *server*.

2.5 *REST (Representational State Transfer)*

Sebuah Web *Service* dapat diaplikasikan dengan gaya arsitektur SOAP (*Simple Object Access Protocol*) atau REST (*Representational State Transfer*). Saat ini arsitektur yang paling banyak diimplementasikan pada web *service* adalah REST karena memiliki lebih banyak kelebihan daripada arsitektur SOAP.

Model arsitektur REST memiliki kelebihan dalam hal kecepatan transfer data dan performa daripada metode SOAP, karena model REST lebih fleksibel dalam hal format pengiriman dan penyimpanan data yang dapat kompatibel dengan JSON atau XML sedangkan model SOAP hanya mendukung bentuk XML (Safitri & Putro, 2021).

Menurut (Choirudin & Adil, 2019), prinsip utama yang mendasari konsep REST yaitu:

1. *Resource identifier* melalui *Uniform Resource Identifier* (URI), yaitu sekumpulan sumber daya yang interaksi antar *client* diindetifikasi oleh REST Web *service*.
2. *Uniform interface*, menggunakan operasi PUT, GET, POST dan DELETE untuk memanipulasi manipulasi CRUD (*Create, Read, Update, Delete*).
3. *Self-descriptive messages*, disamping Metadata dapat digunakan, sumber daya informasi tidak terikat, juga berbagai format konten (HTML, XML, PDF, JPEG, Plain Text dan lainnya) dapat diakses.
4. *Stateful interactions*, dapat berinteraksi dengan suatu sumber daya bersifat *stateless*, dengan *request messages* tergantung jenis kontennya melalui *hyperlink*.

2.6 *Test Driven Development (TDD)*

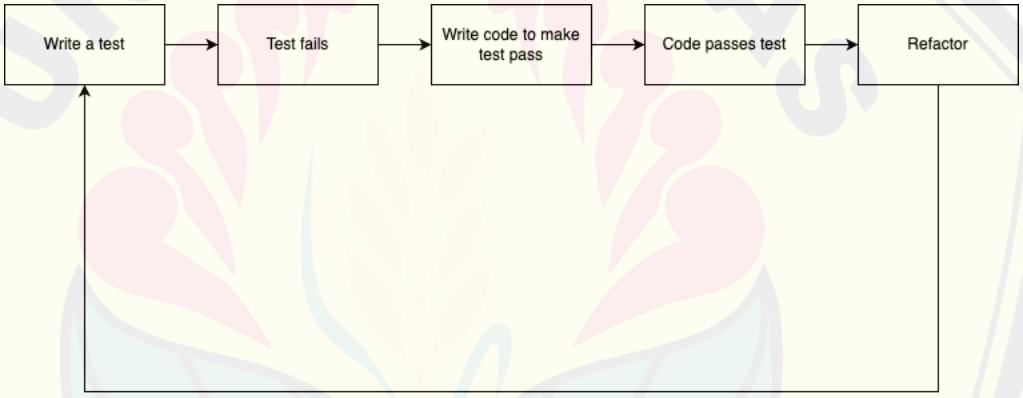
Test driven development (TDD) merupakan metode pengembangan aplikasi atau software yang langkah mengedepankan dalam pembuatan *test case* terlebih dahulu lalu menuliskan kode program dan *refactoring* (Thohari & Amalia, 2018).

Dalam pengembangan perangkat lunak dengan menggunakan metode TDD, pengembang harus menentukan *test case* dengan kriteria – kriteria tertentu dari sebuah *resource* yang akan dapat diakses melalui sebuah URI terlebih dahulu. Lalu,

pengembang akan menuliskan kode untuk memenuhi kriteria – kriteria dari *resource* tersebut agar lolos dari *test case* yang sudah dibuat.

Terakhir, pengembang akan melakukan *refactor* untuk meningkatkan kualitas kode dengan menerapkan *clean code. Clean code* merupakan konsep dalam cara penulisan kode pada pengembangan sebuah aplikasi dengan aturan tertentu. Aturan *clean code* antara lain penamaan file, *function*, variabel, *class* dan folder, menghilangkan kode tak terpakai, gaya penulisan kode yang rapi, menyantumkan komentar dan mengurangi ambiguitas pada sebuah blok kode, sehingga kode lebih mudah dipahami baik oleh diri sendiri atau orang lain yang membacanya (Inayah, n.d.).

Adapun menurut (Subhiyakto & Astuti, 2020), alur pengembangan dalam metode *Test Driven Development* adalah sebagai berikut:



Gambar 2. 1 Metode Test Driven Development

1. Menulis kode test / *Write a test*

   Langkah awal adalah penulisan test untuk setiap unit dari fitur yang akan dibuat. Penulisan test berdasarkan scenario dan deskripsi yang jelas dari tiap unit yaitu spesifikasi lengkap mulai dari input hingga output dari unit.

2. Menjalankan test / *Tests fails*

   Test yang telah dibuat akan dijalankan dan memastikan test tersebut gagal atau tidak memenuhi kebutuhan dari *scenario*. Kegagalan test adalah normal sebab kode fungsional belum dibuat pada tahap ini.

3. Menulis kode fungsional / *Write code to make test pass*

Tahap ini bertujuan untuk memenuhi fungsional dari tiap unit hingga kode fungsional memenuhi kebutuhan dari test case.

4. Menjalankan ulang test / *Code passes test*

   Testing ulang pada tahap ini bertujuan untuk memastikan tiap unit dari fitur telah sesuai dengan kebutuhan test case dan juga developer dapat memastikan kode bebas dari error atau bugs.

5. *Refactor*

   Tahap ini bertujuan untuk merapikan kode yang berantakan, duplikasi kode dan tidak efisien sehingga kode lebih mudah untuk dibaca dan di *maintenance* oleh developer itu sendiri atau developer lain.

2.7 Laravel

Laravel adalah salah satu web *framework* berbasis bahasa pemrograman PHP. Laravel cukup popular di kalangan web *developer* khususnya *developer* PHP. Laravel mendukung pengembangan API yang lengkap dengan berbagai package yang dibutuhkan oleh *developer*. Adapun kelebihan dari *framework* laravel antara lain:

Salah satu kelebihan Laravel adalah megadopsi arsitektur MVC yang mana dapat mempermudah dan membuat kode menjadi lebih konsisten. MVC adalah arsitektur aplikasi yang terbagi dalam 3 lapisan yaitu model untuk berinteraksi dengan database, lapisan *view* untuk antarmuka dan controller yang menjadi jembatan antara model dan *view*. Keuntungan dalam menerapkan konsep MVC yaitu aplikasi dipisah berdasarkan *class* sehingga kode menjadi lebih *reusable* (Aulia Wicaksono et al., 2020).

2.8 *Unit Testing*

Menurut (Hasibuan, 2021), Unit testing merupakan tahap dasar dalam pengujian. Unit testing adalah salah satu jenis pengujian yang dilakukan untuk menguji blok kode atau *function* yang lebih kecil dari program atau sistem. Dengan kata lain, unit testing berfokus pada salah satu bagian dari keseluruhan bagian sistem seperti *function, method, procedure* atau objek. Pengujian unit sendiri menjalankan setiap unit atau blok kode untuk memastikan masing-masing fitur tersebut berfungsi sesuai dengan yang diharapkan. Unit testing akan dilakukan

setelah pengembangan satu unit selesai, sehingga dilakukan ketika pengembangan sedang berlangsung. Implementasi unit testing dilakukan dengan menuliskan sebuah algoritma testing untuk menguji satu blok kode apakah berjalan dengan baik atau tidak. Unit testing dapat dijalankan dengan metode manual atau otomatis. Pada pengujian otomatis akan menggunakan sebuah tool unit testing sesuai dengan bahasa pemrograman yang dipakai.

2.9 PEST

PEST merupakan *framework* atau *tool* khusus untuk unit testing pada bahasa pemrograman PHP. *Tool* PEST menyediakan beragam kebutuhan yang dibutuhkan developer untuk melakukan *unit testing* secara mudah seperti melakukan request ke suatu resource tertentu.

BAB 3. METODOLOGI PENELITIAN

3.1 Jenis Penelitian

Penelitian ini merupakan penelitian *research and development* (R&D). Pada dasarnya metode ini adalah metode penelitian yang digunakan untuk mengembangkan suatu produk dengan kualitas yang baik dan sesuai harapan. Penelitian R&D adalah jenis penelitian yang bertujuan untuk mengembangkan dan menciptakan suatu produk dengan prosedur tertentu yang diawali dengan analisis kebutuhan, proses pengembangan dan evaluasi (Purnama, 2016). Produk yang dikembangkan pada penelitian ini adalah web *service* dari aplikasi web pada startup Theskills. Pengembangan produk yang dilakukan adalah menerapkan skenario pengembangan software dengan metode TDD untuk menggantikan pengembangan software secara konvensional.

3.2 Objek Penelitian

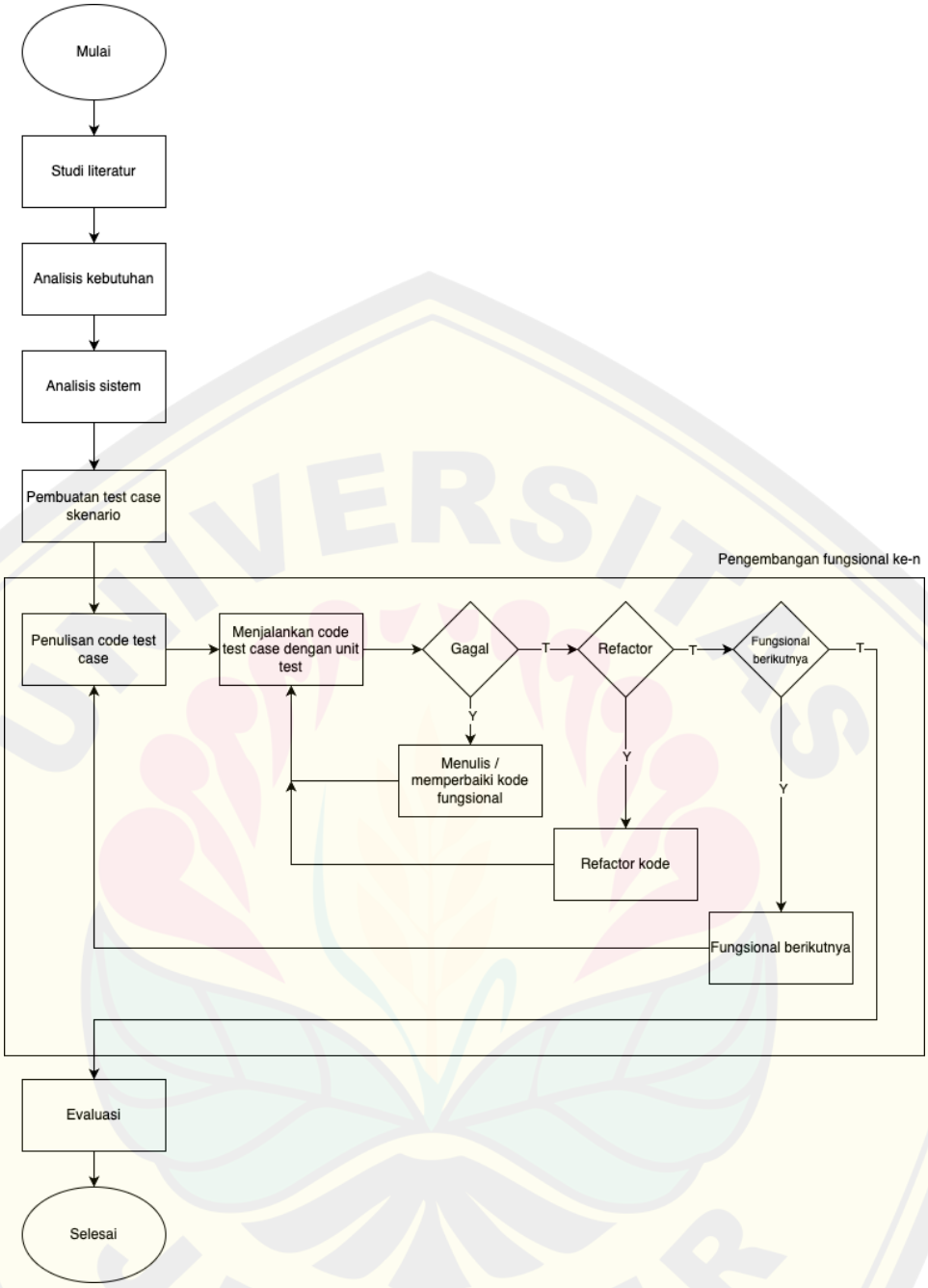
Objek penelitian adalah web service pada aplikasi web the- skills.id milik startup Theskills. Web *service* akan disediakan untuk kebutuhan *resource* untuk aplikasi web the-skills.id.

3.3 Tempat dan Waktu Penelitian

Tempat dilaksanakannya penilitian ini adalah startup theskills di Kabupaten Jember. Penilitian dilakukan pada bulan Desember 2022 sampai Maret 2023.

3.4 Tahapan Penelitian

Gambar 3.1 menggambarkan tahapan yang digunakan dalam penelitian ini:



Gambar 3. 1 Tahapan Penelitian

3.4.1 Studi Literatur

Studi literatur dilakukan dengan mencari referensi terkait teori dan metode yang akan digunakan selama proses penelitian melalui sumber yang relevan. Sumber literasi diperoleh dari jurnal atau karya ilmiah dengan topik yang relevan pada penelitian lain sebelumnya.

3.4.2 Analisis Kebutuhan

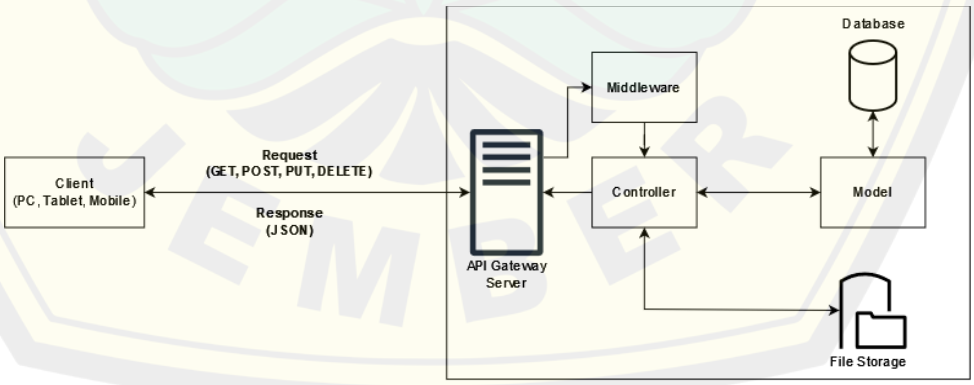
Tahap analisis kebutuhan merupakan tahap awal dalam melakukan penelitian. Pada tahap ini peneliti melakukan pembelajaran dan studi literatur untuk mendapatkan teori-teori ilmiah yang akan digunakan pada tahap-tahap selanjutnya. Pada penelitian ini studi literatur adalah aktivitas untuk mengumpulkan teori – teori pada pengembangan web *service* dan implementasi *test driven development* di dalamnya. Studi literatur dilakukan dengan mencari referensi pada jurnal atau buku yang berkaitan dengan topik penelitian. Selanjutnya dilakukan analisis user untuk memetakan kebutuhan fitur-fitur dari web *service* yang sesuai dengan kebutuhan proses bisnis pada objek penelitian.

3.4.3 Analisis Sistem

Analisis sistem adalah proses untuk menentukan tools apa saja yang akan digunakan untuk pengembangan sistem pada objek penelitian. Pada penelitian ini web *service* menggunakan framework pemrograman Laravel. Laravel merupakan *framework* pengembangan web yang ditulis menggunakan bahasa pemrograman PHP. Kebutuhan sistem lain yang digunakan pada pengembangan antara lain:

1. Apache web server
2. MySQL 5.7
3. PHP 8.1

Sistem design dari web *service* yang akan dikembangkan adalah diilustrasikan sebagai berikut:



Gambar 3. 2 Design System

Client mengakses endpoint dengan request tertentu melalui API gateway server. HTTP *request* akan diteruskan ke modul *middleware* untuk difilter dan

ditentukan hak otoritas untuk dilanjutkan ke modul berikutnya yaitu modul *controller*. Modul *controller* bertugas sebagai otak dimana logika dari sebuah aplikasi diproses. Selain itu, *controller* juga berhak untuk menentukan data yang akan dikirimkan ke *client* melalui *response*.

Modul *controller* berisi unit-unit fungsi logika dari *endpoint* yang disediakan oleh web *service*. Pada modul *controller* juga ditentukan apakah *resource* yang diakses oleh client membutuhkan data dari modul model untuk melakukan transaksi data terhadap database atau transaksi data file dari modul file storage yang memiliki otoritas sebagai penyimpanan file.

3.4.4 Pembuatan skenario test case

Pada tahap ini *test case* dibuat berdasarkan fitur fitur yang akan dikembangkan kedepannya. Adapun isi dari *test case* berupa kebutuhan teknis pada tiap fungsi atau unit. Kebutuhan tersebut antara lain seperti *body params, header, authorization, HTTP method* dan sebagainya. Tujuan dari pembuatan *test case* dimaksudkan agar sistem dapat berfungsi normal dan berjalan sesuai kebutuhan awal yang sudah ditentukan.

3.4.5 Implementasi

Berikut ini adalah tahapan-tahapan dalam implementasi:

1. Pembuatan *test case*

   Pembuatan *test case* merupakan tahap dimana fitur-fitur yang akan diimplementasikan pada web *service* dibuatkan kriteria- kriteria tertentu.

2. Menjalankan *test case* dan identifikasi error

   Test case yang telah ditentukan akan dijalankan untuk mengetahui error atau kriteria apa saja yang harus dipenuhi.

3. Penulisan Kode

   Setelah *test case* dijalankan dan kriteria dari sebuah *resource* telah diketahui maka kode akan dituliskan untuk memenuhi kriteria dari sebuah *resource* tersebut.

4. Pengujian Kode

   Kode yang telah dibuat akan memasuki tahap pengujian untuk memastikan kualitas *resource* yang disediakan oleh sebuah unit kode dan diuji

menggunakan kriteria sesuai dari *test case*. Pengujian dilakukan dengan metode unit testing.

5. *Refactor*

   Pada tahap terakhir siklus TDD dilakukan refactor untuk meningkatkan kualitas kode dengan memecah blok kode yang reusable menjadi modul-modul sesuai dengan fungsinya.

3.4.6 Evaluasi

Evaluasi kualitas produk merupakan tahap akhir dari pengembangan web *service* mengenai hasil dari pengembangan dengan menggunakan metode *Test driven development*. Selain itu akan dilakukan perbandingan kualitas dari hasil pengembangan antara pengembangan menggunakan metode *Test driven development* dengan menggunakan metode *Test last development* pada web *service* yang sudah ada pada pengembangan aplikasi the-skills.id sebelumnya. Pada penelitian ini akan dilakukan perbandingan keefektifan dari metode yang telah disebutkan dengan *quality factor* sebagai berikut (Alhammad et al., 2016):

1. *Number of defect*

   Menunjukkan jumlah *defect* yang ditemukan pada tiap fungsi atau unit. Adapun cara mendapatkannya adalah dengan menghitung manual setiap *defect* yg muncul selama masa pengembangan pada tiap unit.

2. *Code Coverage*

   Code coverage adalah suatu ukuran untuk mengukur sejauh mana kode telah dijalankan selama pengujian. Hal ini membantu pengembang dalam mengevaluasi sejauh mana pengujian telah dilakukan secara menyeluruh dan melihat bagian kode mana yang belum teruji dengan baik. Dengan menganalisis code coverage, pengembang dapat mengidentifikasi bagian kode yang belum atau kurang diuji dan mengambil tindakan yang tepat untuk meningkatkan kualitas dan keandalan perangkat lunak. Adapun rumus perhitungan code coverage adalah:

$$Coverage = \frac{CI}{(CI + MI)} * 100$$

Keterangan:

CI = Covered instructions

MI = Missed instructions

BAB 4. HASIL DAN PEMBAHASAN

4.1 Analisis Kebutuhan

4.1.1 Kebutuhan Fungsional

Kebutuhan fungsional menjabarkan proses – proses yang akan dan harus disediakan oleh sistem. Adapun fungsional yang ada pada sistem antara lain:

1. Role admin:
    a. Sistem dapat mengolah data *course* dan quiz yang meliputi *create, read, update dan delete*.
    b. Sistem dapat mengolah data webinar yang meliputi *create, read, update dan delete*.
    c. Sistem dapat mengolah data artikel yang meliputi *create, read, update dan delete*.
2. Role Customer:
    a. Sistem dapat menyajikan data artikel, course dan quiz, webinar dalam bentuk listing.
    b. Sistem dapat menyajikan data artikel, course dan quiz, webinar dalam bentuk detail.
    c. Sistem dapat menyimpan course pribadi untuk tiap customer.
    d. Sistem dapat menyimpan progress belajar tiap customer.
    e. Sistem dapat menyimpan webinar pribadi untuk tiap customer.

4.1.2 Kebutuhan Non Fungsional

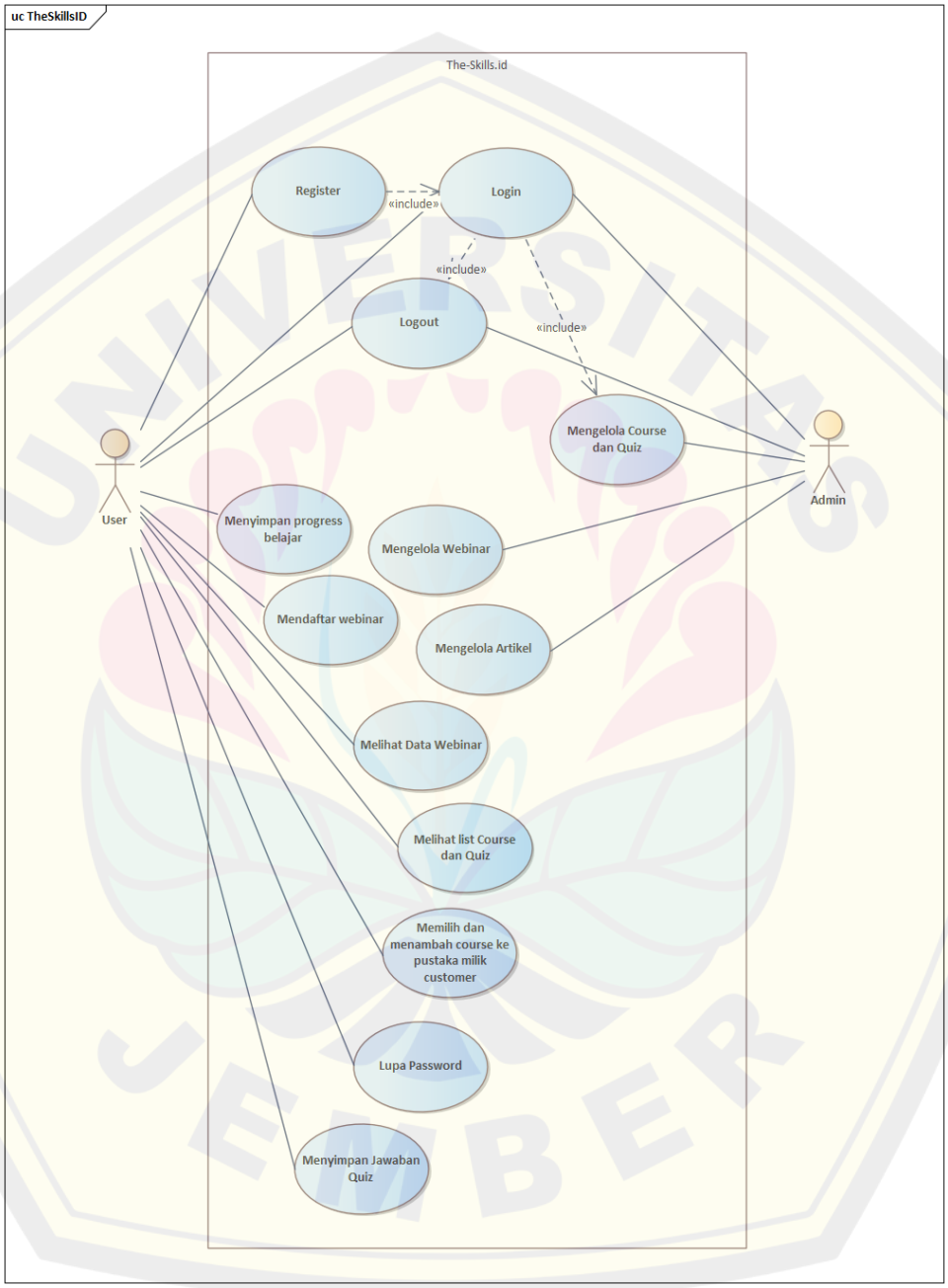
Adapun kebutuhan non fungsional yang ada pada sistem antara lain:

1. Sistem berbasis web dan menggunakan *server shared hosting*.
2. Sistem harus terhubung dengan internet.

4.2 Analisis Sistem

4.2.1 Use case diagram

Use case diagram merupakan gambaran atau rangkaian kebutuhan fungsional dari sebuah sistem dan gambaran behavioral sistem yang akan dibuat.



Gambar 4. 1 Use Case Diagram

Berdasarkan use case diagram pada Gambar 4.1 sistem yang akan dibuat memiliki roles admin dan customer. Deskripsi dari masing – masing role dijabarkan pada Tabel 4.1:

Tabel 4. 1 Role pada sistem

Role	Deskripsi
Admin	Admin memiliki hak akses penuh dalam ke semua fitur dan pengelolaan data course, artikel, webinar, quiz dan user management.
Customer	Customer memiliki peranan dalam akses fitur seperti register akun, melihat course, melihat artikel, melihat webinar, mendaftarkan course pribadi, mengikuti webinar, menyimpan jawaban quiz dan mengedit profile.

Adapun penjelasan dari masing-masing fungsionalitas dari use case diagram pada sistem dapat dilihat di tabel 4.2:

Tabel 4. 2 Deskripsi Use Case

Nama Fungsional	Deskripsi
Login	Proses login terhadap sistem dengan menggunakan akun yang sudah terdaftar dan mendapatkan akses token.
Register	Proses pendaftaran akun terhadap sistem.
Mengelola course dan quiz	Role yang berkaitan dapat melakukan aksi akses penuh yaitu create, read, update dan delete pada fitur course dan quiz
Mengelola Webinar	Role yang berkaitan dapat melakukan aksi akses penuh yaitu create, read, update dan delete pada fitur webinar
Mengelola artikel	Role yang berkaitan dapat melakukan aksi akses penuh yaitu create, read, update dan delete pada fitur artikel

Nama Fungsional	Deskripsi
Melihat webinar	Role yang berkaitan dapat melakukan aksi akses terbatas yaitu read pada fitur webinar
Melihat course dan quiz	Role yang berkaitan dapat melakukan aksi akses terbatas yaitu read pada fitur course dan quiz
Menambah course ke daftar milik pribadi	Role yang berkaitan dapat menambahkan course ke daftar pribadi milik customer dan dapat mengikuti atau mempelajari course
Mengikuti webinar	Role yang berkaitan dapat mendaftarkan akun untuk mengikuti kegiatan webinar
Menyimpan jawaban quiz	Role yang berkaitan dapat menambahkan jawaban quiz dan menyimpannya
Lupa password	Role yang berkaitan dapat merubah password melalui email verifikasi
Logout	Proses logout atau keluar dari sistem dan menghapus token akses

4.2.2 *Scenario* diagram

1. *Scenario* diagram tambah course

   Berikut adalah penjelasan alur dari aksi role admin dan respon sistem dalam aktifitas menambah course pada Tabel 4.3.

Tabel 4. 3 Scenario tambah course

Nama Use case	Menambah Course
Aktor	Admin
Deskripsi	Aktor terkait menambahkan course
Prakondisi	Halaman listing data course
Pascakondisi	Penambahan data course berhasil disimpan
Event Flow	
Normal Flow : Menambah Course	
Aksi Aktor	Reaksi Sistem
Mengirim request tambah course	
	Validasi form data
	Validasi Berhasil
	Menyimpan data course ke database

	Mengirim response dengan pesan sukses
Menerima response sukses	
Alternatif Flow : Isi form data tidak lengkap	
	Mengirim response gagal tambah course dengan pesan “Data tidak lengkap”
Menerima response gagal	
Alternatif Flow : Gagal simpan server error	
	Mengirim response gagal internal server error
Menerima response gagal	

2. Fitur aplikasi menambah webinar

Berikut adalah penjelasan alur dari aksi role admin dan respon sistem dalam aktifitas menambah webinar pada Tabel 4.4.

Tabel 4. 4 Scenario tambah webinar

Nama Use case	Menambah Webinar
Aktor	Admin
Deskripsi	Aktor terkait menambahkan webinar
Prakondisi	Halaman listing data course
Pascakondisi	Penambahan data webinar berhasil disimpan
Event Flow	
Normal Flow : Menambah Course	
Aksi Aktor	Reaksi Sistem
Klik Tambah Webinar	
	Menampilkan form tambah course
Aktor mengisi form tambah webinar	
Klik Simpan	
	Menampilkan listing course
Alternatif Flow : Isi form data tidak lengkap	
	Menampilkan form tambah webinar dengan pesan “Data tidak lengkap”
Alternatif Flow : Kembali	
Klik kembali	
	Menampilkan halaman listing webinar

3. Fitur aplikasi melihat webinar

Berikut adalah penjelasan alur dari aksi role admin dan respon sistem dalam aktifitas menambah webinar pada Tabel 4.5.

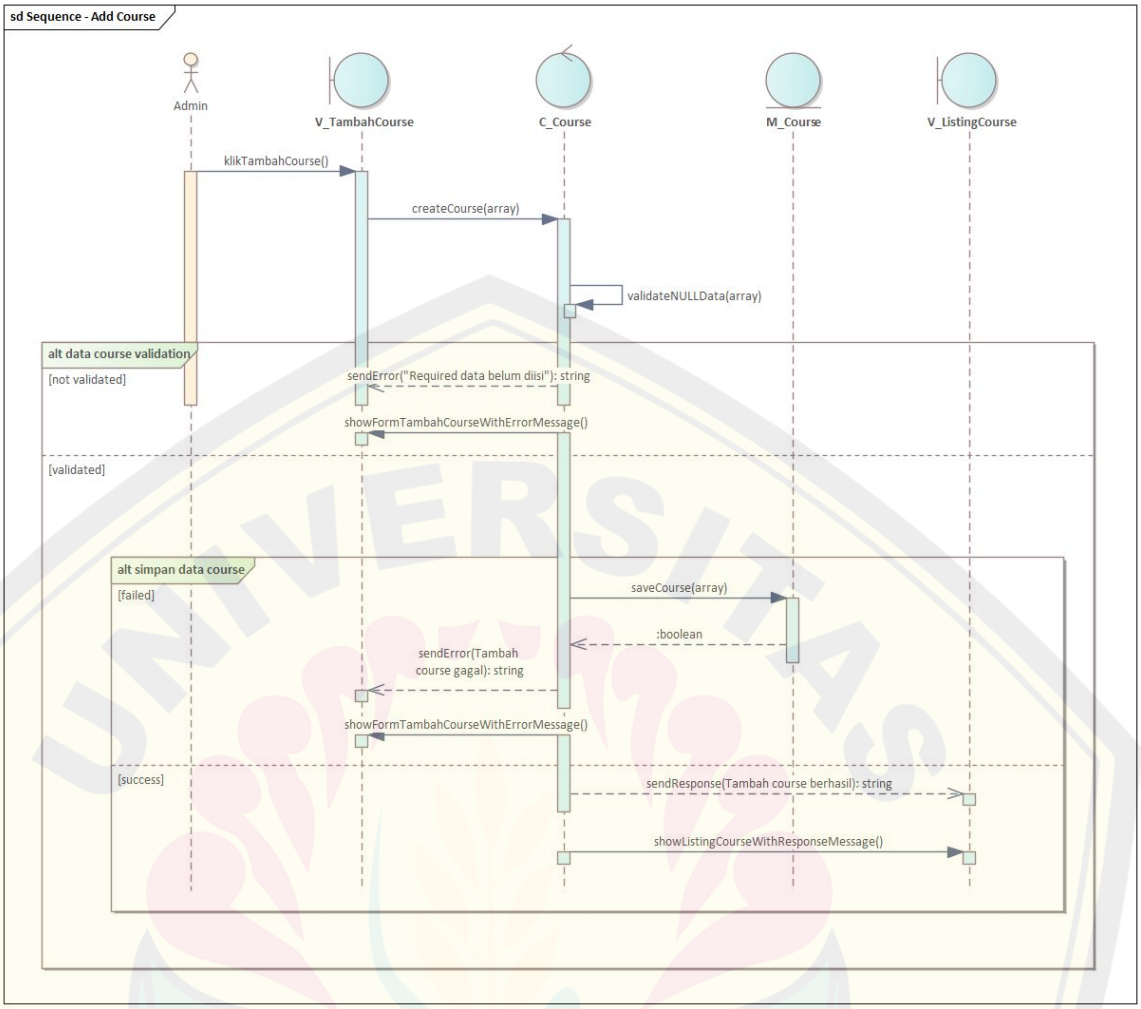
Tabel 4. 5 Scenario melihat webinar

Nama Use case	Melihat Webinar
Aktor	Admin
Deskripsi	Aktor terkait melihat list webinar
Prakondisi	Halaman listing data webinar
Pascakondisi	Menampilkan tabel listing data webinar
Event Flow	
Normal Flow : Melihat Webinar	
Aksi Aktor	Reaksi Sistem
Klik Menu Webinar	
	Menampilkan halaman tabel listing webinar
Alternatif Flow : Data webinar kosong	
	Menampilkan pesan pemberitahuan bahwa data webinar masih kosong

4.2.3 *Sequence* Diagram

1. *Sequence* diagram tambah course

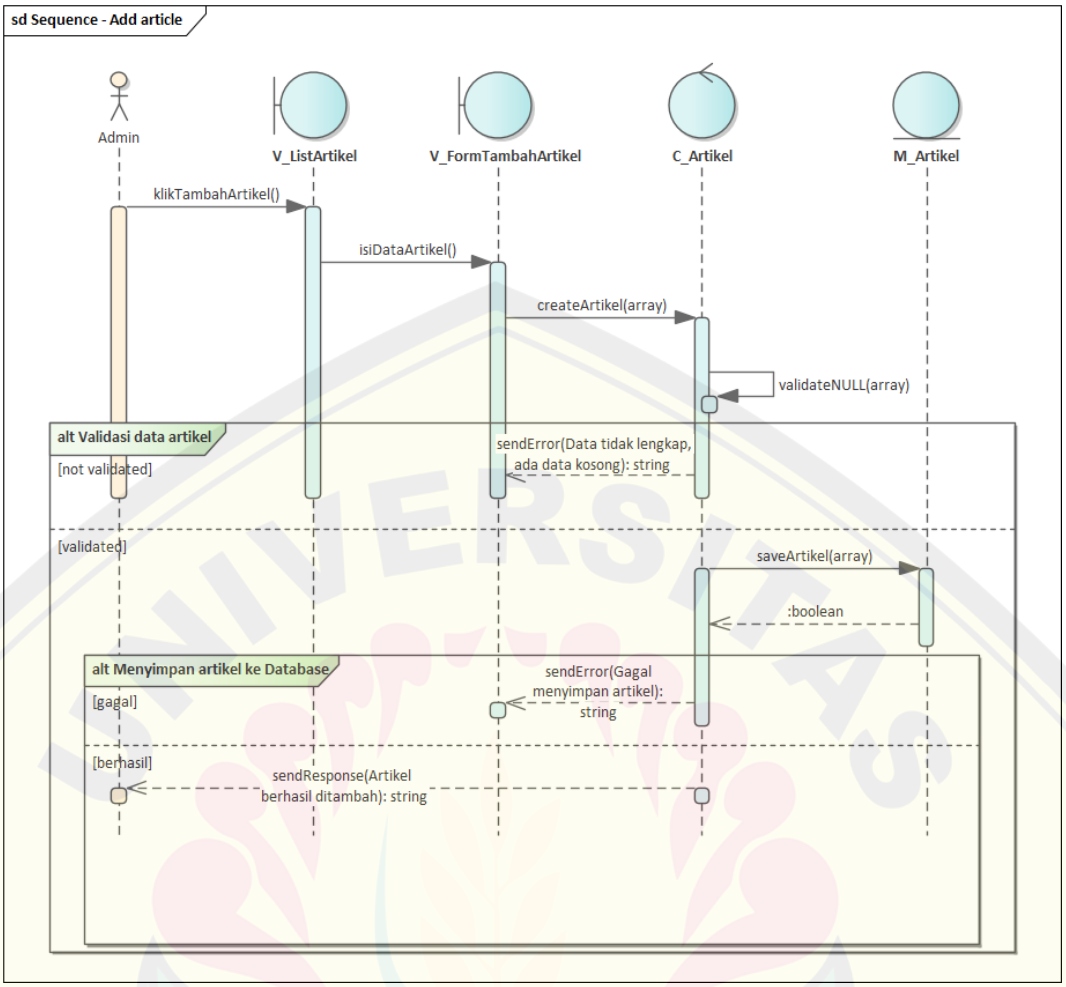
   Rangkaian proses - proses pada fitur tambah course digambarkan dalam *sequence* diagram pada gambar berikut.



Gambar 4. 2 Sequence tambah course

2. Fitur aplikasi menambah artikel

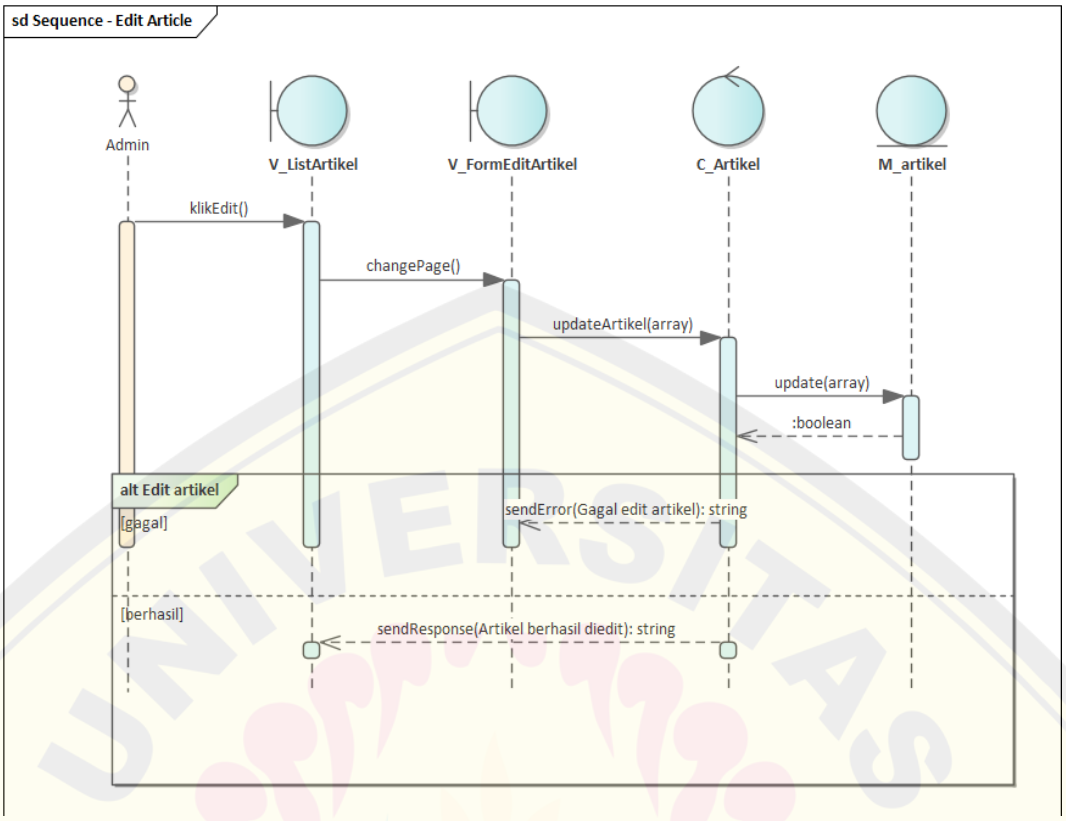
Rangkaian proses - proses pada fitur tambah artikel digambarkan dalam *sequence* diagram pada gambar berikut.



Gambar 4. 3 Sequence tambah artikel

3. Fitur aplikasi mengedit artikel

Rangkaian proses - proses pada fitur mengedit artikel digambarkan dalam *sequence* diagram pada gambar berikut.



Gambar 4. 4 Sequence mengedit artikel

4.3 Pembuatan skenario test case

Pada tahap pembuatan skenario test mengacu pada konsep REST. Adapun konsep atau aturan REST yang diterapkan pada skenario *test case* adalah sebagai berikut:

1. Penggunaan Metode HTTP: *RESTful* menggunakan metode HTTP seperti GET, POST, PUT, dan DELETE untuk berinteraksi dengan sumber daya (*resource*). Adapun pada skenario tes metode HTTP direpresentasikan sebagai *method*.
2. Sumber Daya (Resource): *RESTful* menganggap semua data sebagai sumber daya yang dapat diakses melalui URL (*Uniform Resource Locator*) atau biasa disebut *endpoint*.
3. Representasi Data: *RESTful* menggunakan format data JSON (*JavaScript Object Notation*) untuk merepresentasikan data yang dikirim antara klien

dan server. Pada skenario testing tiap *request* atau *response* nantinya akan selalu berformat JSON dan membawa data di dalamnya.

4. Penggunaan Respons HTTP: *RESTful* menggunakan kode status HTTP seperti 200 OK, 404 *Not Found*, atau 500 *Internal Server Error* dan sebagainya untuk memberikan respons yang sesuai kepada klien. Respon HTTP pada skenario testing akan selalu membawa kode status sebagai indikator sebuah respon.

Adapun skenario pada masing masing fitur aplikasi terdapat pada Lampiran 4 dan Lampiran 5.

Selain skenario untuk fitur utama aplikasi, skenario juga diterapkan untuk menangani *error* pada sisi server dan *middleware* atau fungsi global yang menyaring tiap request yang masuk. Adapun skenario penanganan error dijabarkan pada Tabel berikut:

Tabel 4. 6 Non fungsional skenario

Nama fungsi	**Deskripsi**	**Skenario Response**
Server error	Fungsi untuk menangani *error* yang bukan dari sisi fungsional aplikasi	Response harus memiliki status code 500, Header response Content-Type memiliki nilai application/json, Body response harus berformat objek
Middleware: Method not found	Fungsi untuk mengecek method yang dituju tersedia pada aplikasi	Response harus memiliki status code 405, Header response Content-Type memiliki nilai application/json, Body response harus berformat objek

4.4 Pengembangan fungsional *Web Service*

1. Fitur aplikasi menambah course dan quiz

Penulisan kode penerapan test case unit test pada fitur ini terletak pada folder "tests\unit\AddCourseTest.php". Test case pada fitur tambah course memiliki 2 skenario berdasarkan tabel 4.7. Kode test case dengan valid data dapat dilihat pada Gambar 4.5.

```php
<?php
use App\Models\Course;
use App\Models\CourseCategory;
use App\Models\CourseSection;
use App\Models\File;
use App\Models\User;

use function Pest\Laravel\actingAs;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Http\UploadedFile;

//uses(RefreshDatabase::class);

test("admin dapat menambah course baru", function () {
    $user = User::factory()->create([
        "role" => "1",
    ]);

    $courseCategory = CourseCategory::factory()->create();

    $course = [
        "course_name" => "Tes",
        "description" => "description",
        "price" => 11,
        "discount" => 10,
        "course_category" => $courseCategory->id,
    ];

    $course["thumbnail"] = UploadedFile::fake()->image("thumbnail.jpg");
    $course["course_sections"] = [
        [
            "section_name" => "Test",
        ],
    ];

    $course["course_sections"][0]["sub_sections"] = [
        [
            "title" => "Tes",
            "type" => "quiz",
            "questions" => [
                [
                    "question" => "lorem ipsum",
                    "answers" => [
                        [
                            "answer" => "A",
                            "is_correct" => true,
                        ],
                        [
                            "answer" => "B",
                            "is_correct" => false,
                        ],
```

```
                    ],
                ],
            ],
        ],
        [
            "title" => "Tes",
            "type" => "video",
            "video_url" => "google.com",
        ],
    ];

    actingAs($user)
        ->postJson(route("admin.course.store"), $course)
        ->assertStatus(200);
});
```

Gambar 4. 5 Kode test case fitur tambah course dengan valid data

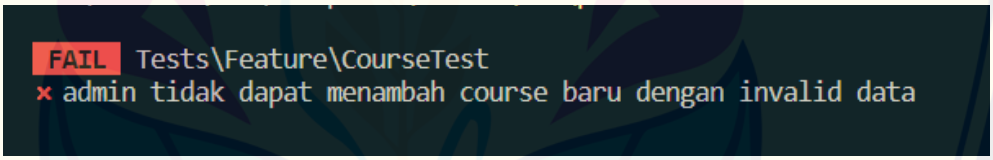
Fitur tambah course dan quiz juga dapat menangani apabila dalam proses penambahan course memiliki invalid data. Kode test case untuk tambah course dan quiz dengan invalid data terdapat pada Gambar 4.6.

```
<?php

use App\Models\Course;
use App\Models\CourseCategory;
use App\Models\CourseSection;
use App\Models\File;
use App\Models\User;

use function Pest\Laravel\actingAs;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Http\UploadedFile;

//uses(RefreshDatabase::class);

test("admin tidak dapat menambah course baru dengan invalid data", function () {
    $user = User::factory()->create([
        "role" => "1",
    ]);

    $courseCategory = CourseCategory::factory()->create();

    $course = Course::factory()->make([
        //'course_category' => $courseCategory->id,
    ]);

    $course["thumbnail"] = UploadedFile::fake()->image("thumbnail.jpg");
    $course["course_sections"] = CourseSection::factory()
        ->count(3)
        ->make([
            "course_id" => $course->id,
        ])
        ->toArray();

    actingAs($user)
        ->postJson(route("admin.course.store"), $course->toArray())
        ->assertStatus(422);
});
```

Gambar 4. 6 Kode test case fitur tambah course dengan invalid data

Unit test yang telah dibuat akan diuji dengan menjalankan perintah .\vendor\bin\pest --filter "admin dapat menambah course baru" untuk pengujian dengan valid data dan perintah .\vendor\bin\pest --filter "admin tidak dapat menambah course baru dengan invalid data" untuk pengujian dengan invalid data. Pengujian yang gagal dapat dilihat pada Gambar 4.7 dan Gambar 4.8.

Gambar 4. 7 Test case tambah course dengan valid data gagal



Gambar 4. 8 Test case tambah course dengan invalid data gagal

Selanjutnya, dilakukan penulisan kode fungsional untuk memperbaiki kesalahan yang muncul pada tahap *unit test* sampai *unit test* dinyatakan berhasil. Berikut adalah kode fungsional dapat dilihat pada Gambar 4.9.

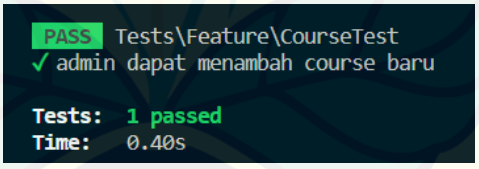
```
/**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $validated = Validator::make($request->all(), [
            'course_name' => 'required',
            'description' => 'required',
            'price' => 'required',
            'discount' => 'required',
            'thumbnail' => 'required',
            'course_category' => 'required',
```

```
            'course_sections' => 'array'
        ]);

        if($validated->fails()) {
            return response()->json([
                'success' => false,
                'message' => $validated->errors()
            ], 422);
        }

        DB::beginTransaction();
        try {
            $course = Course::create($request->except(['course_sections',
'thumbnail']));

            if($request->has('course_sections')) {
                foreach($request->course_sections as $sectione) {

                    $section = $course->sections()->create([
                        'section_name' => $sectione['section_name'],
                        'course_id' => $course->id
                    ]);

                    if($sectione['sub_sections']) {
                        $subsections = $sectione['sub_sections'];
                        foreach($subsections as $subsection) {
                            if($subsection['type'] === 'quiz') {
                                $ss = CourseSubSection::create([
                                    'title' => $subsection['title'],
                                    'type' => 'quiz',
                                    'section_id' => $section->id
                                ]);

                                foreach($subsection['questions'] as $question) {
                                    $quiz = Quiz::create([
                                        'question' => $question['question'],
                                        'sub_section_id' => $ss->id
                                    ]);

                                    foreach($question['answers'] as $answer) {
                                        QuizAnswer::create([
                                            'answer' => $answer['answer'],
                                            'is_correct' => $answer['is_correct'],
                                            'quiz_id' => $quiz->id
                                        ]);
                                    }
                                }

                            } else if($subsection['type'] === 'video') {
                                CourseSubSection::create([
                                    'title' => $subsection['title'],
                                    'video_url' => $subsection['video_url'],
                                    'type' => 'video',
                                    'section_id' => $section->id
                                ]);
                            } else {
                                continue;
                            }
                        }

                    }
                }

            }
```

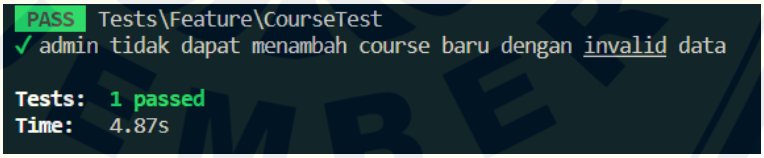
```
            if($request->has('thumbnail')) {
                $thumbnail = $request
                ->file('thumbnail')
                ->storeAs('public/course', time().$request->thumbnail-
>getClientOriginalName());

                $image = $course->image()->create([
                    'url' => $thumbnail,
                    'type' => 'image'
                ]);

                $course->update([
                    'thumbnail' => $image->id
                ]);
            }

            DB::commit();

            return response()->json([
                'success' => true,
                'message' => 'Course successfully created!'
            ], 200);
        } catch (\Throwable $th) {
            DB::rollBack();
            throw $th;
        }

    }
```

Gambar 4. 9 Kode fungsional tambah course

Sehingga, apabila perintah *unit test* dijalankan kembali akan menampilkan pesan *unit test* berhasil sesuai dengan skenario. Tampilan ketika unit test berhasil dapat dilihat pada Gambar 4.10.



Gambar 4. 10 Test case tambah course dengan valid data berhasil



Gambar 4. 11 Test case tambah course dengan invalid data berhasil

2. Fitur aplikasi melihat course

Penulisan kode penerapan test case unit test pada fitur ini terletak pada folder “tests\unit\ReadCourseTest.php”. Kode test case dapat dilihat pada Gambar 4.12.

```
test('admin dapat melihat daftar course', function() {
    $user = User::factory()->create([
        'role' => '1'
    ]);

    $courseCategory = CourseCategory::factory()->create();
    $thumbnail = UploadedFile::fake()->image('thumbnail.jpg')-
>storeAs('public/course', 'thumbnail.jpg');
    $file = File::factory()->create([
        'url' => $thumbnail,
        'type' => 'image'
    ]);

    $course = Course::factory()->create([
        'course_category' => $courseCategory->id,
        'thumbnail' => $file->id
    ]);

    CourseSection::factory()->count(3)->create([
        'course_id' => $course->id
    ]);

    actingAs($user)
    ->getJson(route('admin.course.index'))
    ->assertStatus(200);
});
```

Gambar 4. 12 Kode test case lihat course

Unit test yang telah dibuat akan diuji dengan menjalankan perintah .\vendor\bin\pest --filter "admin dapat melihat daftar course". Pengujian yang gagal dapat dilihat pada Gambar 4.13.

Gambar 4. 13 Test case lihat course gagal

Selanjutnya, dilakukan penulisan kode fungsional untuk memperbaiki kesalahan yang muncul pada tahap *unit test* sampai *unit test* dinyatakan berhasil. Berikut adalah kode fungsional dapat dilihat pada Gambar 4.14.

```
    public function index()
    {
        $data = Course::orderBy('created_at', 'desc')
        ->with(['category', 'image'])
        ->get();

        return response()->json([
            'success' => true,
            'data' => $data
        ], 200);
    }
```

Gambar 4. 14 Kode fungsional lihat course

Sehingga, apabila perintah *unit test* dijalankan kembali akan menampilkan pesan *unit test* berhasil sesuai dengan skenario.

Gambar 4. 15 Test case lihat course berhasil

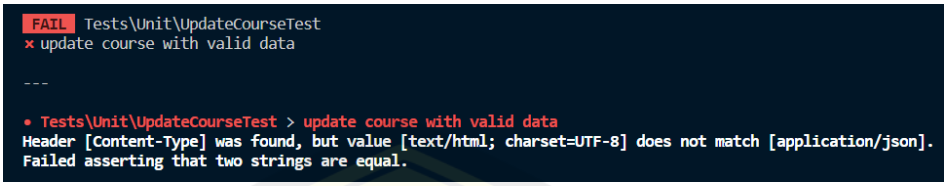
3. Fitur aplikasi mengedit course

Penulisan penerapan test case unit test pada fitur ini terletak pada folder "tests\unit\UpdateCourseTest.php". Kode test case dapat dilihat pada Gambar 4.16.

```
 test('admin dapat mengubah course dengan valid id', function() {
    $this->withoutExceptionHandling();
    $user = User::factory()->create([
        'role' => '1'
    ]);

    $courseCategory = CourseCategory::factory()->create();
    $thumbnail = UploadedFile::fake()->image('thumbnail.jpg')-
>storeAs('public/course', 'thumbnail.jpg');
    $file = File::factory()->create([
        'url' => $thumbnail,
        'type' => 'image'
    ]);

    $course = Course::factory()->create([
        'course_category' => $courseCategory->id,
        'thumbnail' => $file->id
    ]);

    $course['course_sections'] = CourseSection::factory()->count(3)->create([
        'course_id' => $course->id
    ]);
    $course['thumbnail'] = UploadedFile::fake()->image('thumbnailUpdated.jpg');
    $course['course_name'] = 'Course Name';

    actingAs($user)
    ->putJson(route('admin.course.destroy', $course->id), $course->toArray())
    ->assertStatus(200);
});
```

Gambar 4. 16 Kode test case update course

Unit test yang telah dibuat akan diuji dengan menjalankan perintah .\vendor\bin\pest –filter UpdateCourse. Pengujian yang gagal dapat dilihat pada Gambar 4.17.



Gambar 4. 17 Unit test update course gagal

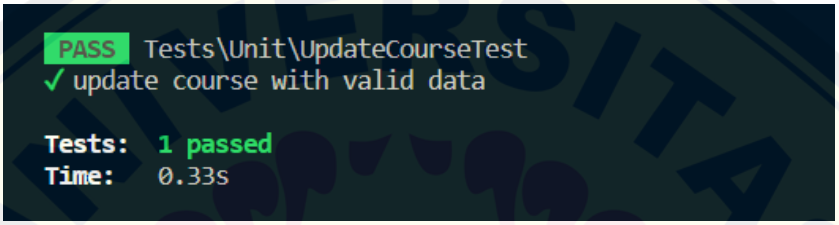
Selanjutnya, dilakukan penulisan kode fungsional untuk memperbaiki kesalahan yang muncul pada tahap *unit test* sampai *unit test* dinyatakan berhasil. Berikut adalah kode fungsional dapat dilihat pada Gambar 4.18.

```
/**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $course
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $course)
    {
        $validated = Validator::make($request->all(), [
            'course_name' => 'required',
            'description' => 'required',
            'price' => 'required',
            'discount' => 'required',
            'thumbnail' => 'required',
            'course_category' => 'required',
            'course_sections' => 'array'
        ]);

        if($validated->fails()) {
            return response()->json([
                'success' => false,
                'message' => $validated->errors()
            ], 422);
        }

        $courseChosen = Course::findOrFail($course);

        $courseChosen->update($request->except(['course_sections', 'thumbnail']));

        if($request->has('course_sections')) {
            foreach($request->course_sections as $section){
                $courseChosen->sections()->where('id', $section['id'])->first()-
>update($section);
            }
        }

        if($request->has('thumbnail')) {
            $thumbnail = $request
            ->file('thumbnail')
            ->storeAs('public/course', time().$request->thumbnail-
>getClientOriginalName());
```

```
            $courseChosen->image()->update([
                'url' => $thumbnail,
                'type' => 'image'
            ]);
        }

        return response()->json([
            'success' => true,
            'message' => 'Course Successfully Updated!'
        ], 200);
    }
```

Gambar 4. 18 Kode fungsional update course

Sehingga, apabila perintah *unit test* dijalankan kembali akan menampilkan pesan *unit test* berhasil sesuai dengan skenario.



Gambar 4. 19 Unit test update course berhasil

4.5 *Refactoring*

Refactoring adalah perubahan yang dilakukan pada struktur internal komponen perangkat lunak untuk membuatnya lebih mudah dipahami dan lebih murah untuk dimodifikasi, tanpa mengubah perilaku yang dapat diamati dari komponen perangkat lunak tersebut. Pada tahap refactoring terdapat istilah yang disebut *bad smell. Bad smell* dapat diartikan sebagai cacat desain pada sebuah kode dan menunjukkan bahwa ada sesuatu yang salah dalam kode yang harus kita refaktor (Kaur & Kaur, 2015). Berikut adalah contoh penanganan *bad smell* pada *class WebinarController* berdasarkan jenis-jenis *bad smell*:

1. *Long parameter list*

   Adalah daftar parameter di mana kita mendeklarasikan begitu banyak parameter yang sebenarnya tidak diperlukan. Daftar parameter panjang sulit dipahami karena mereka tidak konsisten dan sulit digunakan. Parameter seharusnya memiliki nilai yang sederhana. Sebagai contoh kode pada class *WebinarController* yang sudah tidak memiliki bad smell ini. Adapun contoh kodenya adalah sebagai berikut.

```
1. class WebinarController extends Controller
 2. {
 3. /**
 4. *
 5. */
 6. * Display a listing of the resource.
 7. * @return \Illuminate\Http\Response
 8. public function index()
 9. {}

10. /**
11. * Store a newly created resource in storage.
12. *
13. * @param \Illuminate\Http|Request $request
14. * @return \Illuminate|Http|Response
15. */
16. public function store (Request $request)
17. {}

18. /**
19. * Update the specified resource in storage.
20. *
21. * @param Illuminate Http| Request $request
22. * @param int $webinar
23. * @return \Illuminate|Http|Response
24. */
25. public function update (Request $request, $webinar)
26. {}

27. /**
28. * Remove the specified resource from storage.
29. *
30. * @param int $webinar
31. * @return \Illuminate|Http|Response
32. */
33. public function destroy ($webinar)
34. {}

35. public function joinWebinar (Request $request)
36. {}
38.
```

Gambar 4. 20 Kode Refactoring tanpa Long Parameter

2. *Dead code*

   Kode yang tidak pernah dijalankan atau tidak melakukan fungsi apa pun dalam kode sumber seharusnya dihilangkan dari kode sehingga kode menjadi lebih bersih dan rapi.

3. *Long method*

   adalah metode yang terlalu panjang. Metode panjang sama dengan kelas besar. Metode panjang juga menyebabkan kebingungan bagi pengembang baru dan sulit dipahami. Contoh kode yang sudah bebas dari long method ini adalah sebagai berikut:

```
1. <?php
```

```
 2.
 3. namespace App\Http\Controllers;
 4.
 5. use App\Models\Webinar;
 6. use Illuminate\Http\Request;
 7. use Illuminate\Support\Facades\Auth;
 8. use Illuminate\Support\Facades\DB;
 9. use Validator;
10.
11. class WebinarController extends Controller
12. {
13.     /**
14.      * Display a listing of the resource.
15.      *
16.      * @return \Illuminate\Http\Response
17.      */
18.     public function index()
19.     {
20.         //code
25.     }
26.
27.     /**
28.      * Store a newly created resource in storage.
29.      *
30.      * @param  \Illuminate\Http\Request  $request
31.      * @return \Illuminate\Http\Response
32.      */
33.     public function store(Request $request)
34.     {
35.         //code
77.     }
78.
79.     /**
80.      * Update the specified resource in storage.
81.      *
82.      * @param  \Illuminate\Http\Request  $request
83.      * @param  int  $webinar
84.      * @return \Illuminate\Http\Response
85.      */
86.     public function update(Request $request, $webinar)
87.     {
88.         //code
116.    }
117.
118.    /**
119.     * Remove the specified resource from storage.
120.     *
121.     * @param  int  $webinar
122.     * @return \Illuminate\Http\Response
123.     */
124.    public function destroy($webinar)
125.    {
126.        //code
135.    }
136.
137.    public function joinWebinar(Request $request)
138.    {
139.        //code
156.    }
157. }
```

Gambar 4. 21 Kode Refactoring tanpa Long Method

Kode diatas dapat dinyatakan bebas dari long method Ketika function atau method di dalam kelas hanya spesifik kepada satu fungsi saja dan diberi nama sesuai dengan fungsinya dan juga nama method tidak terlalu panjang.

4. *Duplicate Code*

adalah kode yang diulang di banyak tempat dalam satu kode sumber yang sama. Masalahnya muncul ketika akan mencoba memperbarui kode ini. Developer harus memperbarui kode duplikat di semua tempat di mana kode ini ditempatkan. Jika developer lupa memperbarui kode di satu tempat, ini dapat menyebabkan masalah. Oleh karena itu, kode duplikat sulit dipelihara. Maka dari itu duplicate code dapat diatasi dengan *Method extraction* yang berarti mengekstrak sebuah metode yang muncul di banyak tempat dalam kode sumber dan menempatkannya dalam metode terpisah. Sebagai contoh pada class dibawah terdapat blok kode yang berulang akan tetapi blok kode tersebut memiliki fungsi yang sama.

```php
<?php

namespace App\Http\Controllers;

use App\Models\Webinar;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use Validator;

class WebinarController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $data = Webinar::with("thumbnail")
            ->orderBy("created_at", "desc")
            ->get();

        return response()->json(
            [
                "success" => true,
                "data" => $data,
            ],
            200,
        );
    }

    /**
     * Store a newly created resource in storage.
```

```
35.      *
36.      * @param  \Illuminate\Http\Request  $request
37.      * @return \Illuminate\Http\Response
38.      */
39.     public function store(Request $request)
40.     {
41.         $validated = Validator::make($request->all(), [
42.             "webinar_name" => "required",
43.             "description" => "required",
44.             "price" => "required",
45.             "link" => "required",
46.             "started_at" => "required",
47.             "register_end_at" => "required",
48.         ]);
49.
50.         if ($validated->fails()) {
51.             return response()->json(
52.                 [
53.                     "success" => false,
54.                     "message" => $validated->errors(),
55.                 ],
56.                 422,
57.             );
58.         }
59.
60.         DB::beginTransaction();
61.         try {
62.             $webinar = Webinar::create($request->except("thumbnail"));
63.
64.             if ($request->has("thumbnail")) {
65.                 $thumbnail = $request->file("thumbnail");
66.                 $thumbnailName = $thumbnail->storeAs(
67.                     "public/webinar",
68.                     time() . $thumbnail->getClientOriginalName(),
69.                 );
70.                 $wt = $webinar->thumbnail()->create([
71.                     "url" => $thumbnailName,
72.                     "type" => "image",
73.                 ]);
74.
75.                 $webinar->update([
76.                     "thumbnail" => $wt->id,
77.                 ]);
78.             }
79.
80.             DB::commit();
81.             return response()->json(
82.                 [
83.                     "success" => true,
84.                     "message" => "Webinar successfully created!",
85.                 ],
86.                 200,
87.             );
88.         } catch (\Throwable $th) {
89.             DB::rollBack();
90.             throw $th;
91.         }
92.     }
93.
94.     /**
95.      * Update the specified resource in storage.
96.      *
97.      * @param  \Illuminate\Http\Request  $request
98.      * @param  int  $webinar
```

```
 99.      * @return \Illuminate\Http\Response
100.      */
101.     public function update(Request $request, $webinar)
102.     {
103.         $webinar = Webinar::find($webinar);
104.
105.         if (!$webinar) {
106.             return response()->json(
107.                 [
108.                     "success" => false,
109.                     "message" => "Webinar Not Found!",
110.                 ],
111.                 404,
112.             );
113.         }
114.
115.         $webinar->update($request->all());
116.
117.         if ($request->has("thumbnail")) {
118.             $thumbnail = $request->file("thumbnail");
119.             $thumbnailName = $thumbnail->storeAs(
120.                 "public/webinar",
121.                 time() . $thumbnail->getClientOriginalName(),
122.             );
123.             $wt = $webinar->thumbnail()->create([
124.                 "url" => $thumbnailName,
125.                 "type" => "image",
126.             ]);
127.
128.             $webinar->update([
129.                 "thumbnail" => $wt->id,
130.             ]);
131.         }
132.
133.         return response()->json(
134.             [
135.                 "success" => true,
136.                 "message" => "Webinar successfully updated!",
137.             ],
138.             200,
139.         );
140.     }
141.
142.     /**
143.      * Remove the specified resource from storage.
144.      *
145.      * @param  int  $webinar
146.      * @return \Illuminate\Http\Response
147.      */
148.     public function destroy($webinar)
149.     {
150.         $webinarChosen = Webinar::find($webinar);
151.
152.         if (!$webinarChosen) {
153.             return response()->json(
154.                 [
155.                     "success" => false,
156.                     "message" => "Webinar Not Found!",
157.                 ],
158.                 404,
159.             );
160.         }
161.
162.         $webinarChosen->destroy($webinar);
```

```
163.          return response()->json(
164.              [
165.                  "success" => true,
166.                  "message" => "Webinar successfully deleted!",
167.              ],
168.              200,
169.          );
170.      }
171.
172.      public function joinWebinar(Request $request)
173.      {
174.          $user = Auth::user();
175.          $webinar = $request->webinar_id;
176.          $webinarChosen = Webinar::where("id", $webinar)->first();
177.
178.          if (!$webinarChosen) {
179.              return response()->json(
180.                  [
181.                      "success" => false,
182.                      "message" => "Webinar Not Found!",
183.                  ],
184.                  404,
185.              );
186.          }
187.
188.          if ($webinarChosen->price < 1) {
189.              $user->agendas()->attach($webinar);
190.
191.              return response()->json(
192.                  [
193.                      "success" => true,
194.                      "message" => "Webinar successfully joined!",
195.                  ],
196.                  200,
197.              );
198.          }
199.          //PAYMENT GATEWAY SERVICE
200.          $user->agendas()->attach($webinar);
201.
202.          return response()->json(
203.              [
204.                  "success" => true,
205.                  "message" => "Webinar successfully joined!",
206.              ],
207.              200,
208.          );
209.      }
210. }
211.
```

Gambar 4. 22 Kode Refactoring dengan Duplicated Code

Blok kode yang dimaksud yaitu terdapat pada gambar berikut.

```
return response()->json(
    [
        "success" => false,
        "message" => "Webinar Not Found!",
    ],
    404,
);
```

Gambar 4. 23 Kode Refactoring

```
return response()->json(
    [
        "success" => true,
        "message" => "Webinar successfully joined!",
    ],
    200,
);
```

Gambar 4. 24 Kode Refactoring Sebelum Method Extraction

Dua buah blok kode tersebut sebenarnya memiliki fungsi yang sama yaitu mengembalikan nilai response dengan format JSON. Sehingga apabila dilakukan *method extraction* akan menjadi seperti berikut:

```
1. public function sendResponse($message, $data = []) {
2.         return response()->json([
3.             'success' => true,
4.             'message' => $message,
5.             'data' => $data
6.         ], 200);
7.     }
8.
9.     public function sendError($message, $code = 500) {
10.        return response()->json([
11.            'success' => false,
12.            'message' => $message
13.        ], $code);
14.    }
15.
```

Gambar 4. 25 Kode Refactoring Method Extraction

Method sendResponse khusus untuk response sukses dan sendError khusus untuk menangani error. Lalu implementasi pada class controller adalah sebagai berikut.

```
1. <?php
2.
3. namespace App\Http\Controllers;
4.
5. use App\Models\Webinar;
6. use Illuminate\Http\Request;
7. use Illuminate\Support\Facades\Auth;
```

```
 8. use Illuminate\Support\Facades\DB;
 9. use Validator;
10.
11. class WebinarController extends Controller
12. {
13.     /**
14.      * Display a listing of the resource.
15.      *
16.      * @return \Illuminate\Http\Response
17.      */
18.     public function index()
19.     {
20.         ...
23.
24.         return $this->sendResponse("Fetched successfully!", $data);
25.     }
26.
27.     /**
28.      * Store a newly created resource in storage.
29.      *
30.      * @param  \Illuminate\Http\Request  $request
31.      * @return \Illuminate\Http\Response
32.      */
33.     public function store(Request $request)
34.     {
35.         $validated = Validator::make($request->all(), [
36.             "webinar_name" => "required",
37.             "description" => "required",
38.             "price" => "required",
39.             "link" => "required",
40.             "started_at" => "required",
41.             "register_end_at" => "required",
42.         ]);
43.
44.         if ($validated->fails()) {
45.             return $this->sendError($validated->errors(), 422);
46.         }
47.
48.         DB::beginTransaction();
49.         try {
50.             ...
69.             return $this->sendResponse(
70.                 "Webinar successfully created!",
71.                 $webinar,
72.             );
73.         } catch (\Throwable $th) {
74.             DB::rollBack();
75.             throw $th;
76.         }
77.     }
78.
79.     /**
80.      * Update the specified resource in storage.
81.      *
82.      * @param  \Illuminate\Http\Request  $request
83.      * @param  int  $webinar
84.      * @return \Illuminate\Http\Response
85.      */
86.     public function update(Request $request, $webinar)
87.     {
88.         ...
111.
112.         return $this->sendResponse(
113.             "Webinar successfully updated!",
```

```
114.             $webinar->fresh(),
115.         );
116.     }
117.
118.     /**
119.      * Remove the specified resource from storage.
120.      *
121.      * @param  int  $webinar
122.      * @return \Illuminate\Http\Response
123.      */
124.     public function destroy($webinar)
125.     {
126.         ...
133.
134.         return $this->sendResponse("Webinar successfully deleted!");
135.     }
136.
137.     public function joinWebinar(Request $request)
138.     {
139.         ...
154.
155.         return $this->sendResponse("Webinar successfully joined!");
156.     }
157. }
158.
```

Gambar 4. 26 Kode Refactoring Tanpa Duplicated Code

4.6 Evaluasi

Pada bagian ini membahas hasil analisis kualitas web *service* yang telah diperoleh. Analisis kualitas berisikan komparasi antara metode TDD dengan metode *Test Last Development* dan didapatkan hasil sebagai berikut.

Tabel 4. 7 Hasil Test Driven Development

Fitur	Test case ID	TDD		Test case description
		Number of Defect	Code Coverage	
Add Course	TC1	11	90,625	Menambah course dengan valid data
	TC2	1	9,375	Menambah course dengan invalid data
Edit Course	TC3	4	81,81818182	Merubah course dengan valid data
	TC4	2	27,27272727	Merubah course dengan invalid data
List Course	TC5	1	100	Melihat data course
Delete Course	TC6	2	93,75	Menghapus course dengan valid identity
	TC7	1	87,5	Menghapus course dengan invalid identity

Fitur	Test case ID	TDD		Test case description
		Number of Defect	Code Coverage	
Add Webinar	TC8	6	87,5	Menambah webinar dengan valid data
	TC9	1	27,27272727	Menambah webinar dengan invalid data
Edit Webinar	TC10	1	80	Merubah webinar dengan valid data
	TC11	1	30	Merubah webinar dengan invalid data
List Webinar	TC12	2	100	Melihat data webiniar
Delete Webinar	TC13	1	100	Menghapus webinar dengan valid identity
	TC14	1	33,3	Menghapus webinar dengan invalid identity
Add Article	TC15	4	80	Menambah artikel dengan valid data
	TC16	1	30	Menambah artikel dengan invalid data
Edit Article	TC17	3	80	Merubah artikel dengan valid data
	TC18	1	30	Merubah artikel dengan invalid data
List Article	TC19	2	100	Melihat data artikel
Delete Article	TC20	1	60	Menghapus artikel dengan valid identity
	TC21	2	60	Menghapus artikel dengan invalid identity
Join Course	TC22	7	80	Menambahkan course ke daftar course pribadi
List Personal Course	TC23	3	100	Melihat daftar course pribadi
Save Progress Course	TC24	3	100	Menyimpan progress belajar pada course
Save Quiz	TC25	4	100	Menyimpan jawaban quiz pada course
Detail Course	TC26	1	60	Melihat detail materi course
Join Webinar	TC27	6	80	Mendaftarkan akun pribadi ke webinar
List Agenda	TC28	7	100	Melihat list webinar yang diikuti oleh user

Fitur	Test case ID	TDD		Test case description
		Number of Defect	Code Coverage	
Rata rata		2,857142857	71,72905844	
Jumlah		80		

Tabel 4. 8 Hasil Test Last Development

Fitur	*Test Last Development*	
	Number of Defect	Code Coverage
Add Course	19	100
Edit Course	7	100
List Course	1	100
Delete Course	3	90,625
Add Webinar	13	57,35
Edit Webinar	7	60
List Webinar	0	100
Delete Webinar	0	65
Add Article	3	60
Edit Article	3	55
List Article	1	100
Delete Article	2	60
Join Course	7	80
List Personal Course	2	100
Save Progress Course	4	100
Save Quiz	4	100
Detail Course	1	60
Join Webinar	8	80
List Agenda	5	100
Total	90	-
Rata rata	-	86,23

Adapun hasil perbadingan pada tiap *quality factor number of defect* dan *code coverage* berdasarkan tabel hasil diatas adalah:

1. *Number of defects*

   Number of defect menunjukkan jumlah *defect* yang ditemukan selama masa pengembangan hingga tahap pengetesan dinyatakan sukses. Pada pengembangan aplikasi ini, jumlah *defect* pada metode *Test Last Development* menunjukkan angka 90 defect sedangkan untuk metode TDD 80 *defect*. Dari hasil tersebut metode TDD unggul dalam hal meminimalisir *defect* yang muncul sedangkan metode *Test Last Development* menghasilkan lebih banyak *defect* pada masa pengembangan.

2. *Code coverage*

   Code coverage menunjukkan presentase dari code sebuah unit atau *function* dieksekusi dan diuji. Perhitungan dilakukan dengan cara jumlah baris yang dieksekusi ditambah dengan jumlah baris yang tidak dieksekusi lalu dibagi 100. Aplikasi ini menunjukkan hasil dari *code coverage* yaitu 71% rata rata dari metode TDD dan 86,23% rata rata dengan metode *Test Last Development*. Dapat disimpulkan pada aplikasi yang dikembangkan dalam penelitian ini, metode *Test Last Development* memiliki kualitas kode yang lebih baik daripada metode TDD, jumlah rata rata *coverage* yang lebih tinggi menunjukkan lebih sedikit baris kode yang tidak tereksekusi yang dapat berpotensi menjadi bug.

BAB 5. PENUTUP

5.1 Kesimpulan

Berikut adalah kesimpulan yang dapat diambil dari penelitian ini:

Hasil yang diperoleh dari perbandingan kualitas menggunakan dua *quality factor* yaitu *number of defects* dan *code coverage* adalah pendekatan *Test Driven Development* menunjukkan keunggulan pada segi *external quality* yaitu memunculkan lebih sedikit *defect* dengan jumlah 80 *defects* pada saat masa pengembangan dibandingkan dengan *Test Last Development* yang menimbulkan lebih banyak defect dengan jumlah 90 *defects*. Adapun dari segi *code coverage* metode *Test Last Development* memiliki rerata kualitas *code coverage* yang lebih baik yakni 86,23% sedangkan *Test Driven Development* 71,7%.

5.2 Saran

Adapun saran untuk mengetahui keefektikan metode *Test Driven Development* lebih lanjut yaitu dengan membandingkan metode tersebut selain dengan metode *Test Last Development*, sehingga dapat mengetahui kualitas perangkat lunak dengan *Test Driven Development* yang lebih komprehensif.

DAFTAR PUSTAKA

Alhammad, N., Alkowiter, A., Althunayan, L., Alsahdi, N., & Alotaibi, T. (2016). Comparison between Test-Driven Development and Conventional Development: A Case Study. In *AlHammad. Int. Journal of Engineering Research and Applications www.ijera.com* (Vol. 6). www.ijera.com

Aulia Wicaksono, E., Ineke Pakereng, M. A., & Author, C. (2020). Implementation of Laravel Framework in the Development of Library Information System (Study Case: Smk Pgri 2 Salatiga). *Jurnal Pilar Nusa Mandiri*, *16*(2), 261–280. www.uksw.edu

Baldassarre, M. T., Caivano, D., Fucci, D., Juristo, N., Romano, S., Scanniello, G., & Turhan, B. (2021). Studying test-driven development and its retainment over a six-month time span. *Journal of Systems and Software*, *176*. https://doi.org/10.1016/j.jss.2021.110937

Choirudin, R., & Adil, A. (2019). Implementasi Rest Api Web Service dalam Membangun Aplikasi Multiplatform untuk Usaha Jasa. *MATRIK : Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer*, *18*(2), 284–293. https://doi.org/10.30812/matrik.v18i2.407

Firdaus, A., Widodo, S., Sutrisman, A., Nasution, S. G. F., & Mardiana, R. (2019). Rancang Bangun Sistem Informasi Perpustakaan Menggunakan Web Service Pada Jurusan Teknik Komputer Polsri. *Jurnal Informatika*, *5*(2407–1730), 83.

Hasibuan, A. N. (2021). *Pengujian dengan Unit Testing dan Test case pada Proyek Pengembangan Modul Manajemen Pengguna*.

Inayah, F. I. (2021). *Implementasi Clean Code pada Pengembangan Aplikasi Berbasis Web*. 2–6.

Jonathan, F., Pakereng, M. A. I., Informatika, T., Kristen, U., & Wacana, S. (2021). *Test-Driven Development pada Pengembangan Aplikasi Android untuk Memantau COVID-19*. *6*(1), 20–24.

Kaur, S., & Kaur, E. H. (2015). REVIEW ON IDENTIFICATION AND REFACTORING OF BAD SMELLS USING ECLIPSE. In *International Journal For Technological Research In Engineering* (Vol. 2, Issue 7). www.ijtre.com

Purnama, S. (2016). Metode Penelitian Dan Pengembangan (Pengenalan Untuk Mengembangkan Produk Pembelajaran Bahasa Arab). *LITERASI (Jurnal Ilmu Pendidikan)*, *4*(1), 19. https://doi.org/10.21927/literasi.2013.4(1).19-32

Rizal, R., & Rahmatulloh, A. (2019). Restful Web Service Untuk Integrasi Sistem Akademik Dan Perpustakaan Universitas Perjuangan. *Jurnal Ilmiah Informatika*, *7*(01), 54. https://doi.org/10.33884/jif.v7i01.1004

Safitri, R. K., & Putro, H. P. (2021). Implementasi REST API untuk Komunikasi Antara ReactJS dan NodeJS (Studi Kasus : Modul Manajemen User Solusi247). *Automata*, *2*(1), 0–4. https://journal.uii.ac.id/AUTOMATA/article/view/17381

Subhiyakto, E. R., & Astuti, Y. P. (2020). Test-Driven Development (TDD) for Point of Sale System at Bicycle Shop. *Scientific Journal of Informatics*, *7*(2), 2407–7658. http://journal.unnes.ac.id/nju/index.php/sji

Thohari, A. N. A., & Amalia, A. E. (2018). Implementasi Test Driven Development Dalam Pengembangan Aplikasi Berbasis Web. *SITECH : Jurnal Sistem Informasi Dan Teknologi*, *1*(1), 1–10. https://doi.org/10.24176/sitech.v1i1.2255

Wardani, D. N., Toenlioe, A. J. E., & Wedi, A. (2018). Daya tarik pembelajaran di era 21 dengan blended learning. *Jurnal Kajian Teknologi Pendidikan (JKTP)*, *1*(1), 13–18. https://core.ac.uk/download/pdf/287323676.pdf

Wardhana, W. G., Arwani, I., & Rahayudi, B. (2020). Implementasi Teknologi Restful Web Service Dalam Pengembangan Sistem Informasi Perekaman Prestasi Mahasiswa Berbasis Website (Studi Kasus : Fakultas Teknologi Pertanian Universitas Brawijaya). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *4*(2), 680–689. http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/7024

LAMPIRAN

Lampiran 1 Scenario Diagram

1). *Fitur aplikasi mengedit webinar*

Nama Use case	Menambah Artikel
Aktor	Admin
Deskripsi	Aktor terkait mengirim request tambah artikel
Prakondisi	Halaman listing data artikel pada admin panel
Pascakondisi	Penambahan data artikel berhasil disimpan
Event Flow	
Normal Flow : Menambah Course	
Aksi Aktor	Reaksi Sistem
Mengirim request tambah artikel	
	Sistem melakukan validasi
	Sistem menyimpan data artikel
	Sistem mengirim response berhasil
Menerima response berhasil	
Alternatif Flow : Isi form data tidak lengkap	
	Megirimkan response tambah artikel dengan pesan “Data tidak lengkap”
Menerima response error validasi	
Alternatif Flow : Sistem gagal menyimpan	
	Mengirimkan response gagal dalam menyimpan
Menerima response error gagal menyimpan	

2). *Fitur aplikasi menambah artikel*

Berikut adalah penjelasan alur dari aksi role admin dan respon sistem dalam aktifitas menambah artikel pada

Nama Use case	Menambah Artikel
Aktor	Admin
Deskripsi	Aktor terkait mengirim request tambah artikel
Prakondisi	Halaman listing data artikel pada admin panel
Pascakondisi	Penambahan data artikel berhasil disimpan
Event Flow	
Normal Flow : Menambah Course	
Aksi Aktor	Reaksi Sistem
Mengirim request tambah artikel	
	Sistem melakukan validasi
	Sistem menyimpan data artikel

	Sistem mengirim response berhasil
Menerima response berhasil	
Alternatif Flow : Isi form data tidak lengkap	
	Megirimkan response tambah artikel dengan pesan “Data tidak lengkap”
Menerima response error validasi	
Alternatif Flow : Sistem gagal menyimpan	
	Mengirimkan response gagal dalam menyimpan
Menerima response error gagal menyimpan	

3). *Fitur aplikasi melihat artikel*

Nama Use case	Melihat Artikel
Aktor	Admin
Deskripsi	Aktor terkait melihat list artikel
Prakondisi	Halaman listing data artikel pada admin panel
Pascakondisi	Menampilkan tabel listing data artikel
Event Flow	
Normal Flow : Melihat Artikel	
Aksi Aktor	Reaksi Sistem
Mengirim request list artikel	
	Mengirimkan response data artikel
Alternatif Flow : Data artikel kosong	
	Mengirimkan response data artikel kosong

4). *Fitur aplikasi menambahkan course ke Pustaka pribadi*

Nama Use case	Menambah Course ke Pustaka pribadi
Aktor	Customer
Deskripsi	Aktor terkait menambahkan atau mengikuti course yang dipilih ke Pustaka pribadi
Prakondisi	Halaman listing data course
Pascakondisi	Penambahan data webinar berhasil disimpan
Event Flow	
Normal Flow : Menambah Course ke Pustaka pribadi	
Aksi Aktor	Reaksi Sistem
Mengirim request Join Course	
	Validasi data
	Menyimpan data join course
	Mengirim response join course berhasil
Menerima response sukses	
Alternatif Flow : Gagal menyimpan	
	Mengirimkan response gagal dalam menyimpan
Menerima response gagal menyimpan	
Alternatif Flow : Kembali	

Klik batal	
	Menampilkan halaman listing course

5). *Fitur aplikasi melihat detail course pada Pustaka pribadi*

Nama Use case	Melihat detail course pribadi
Aktor	Customer
Deskripsi	Aktor terkait menambahkan webinar
Prakondisi	Customer mengirim request untuk menampilkan detail course tertentu
Pascakondisi	Data detail course dikirim kepada customer
Event Flow	
Normal Flow : Request melihat detail Course pribadi	
Aksi Aktor	Reaksi Sistem
Mengirim URL Request detail course	
	Mengecek user terkait telah terdaftar di course
	Mengirim data detail course
Menerima data detail course	
Alternatif Flow : User belum terdaftar di course	
	Mengirim response pesan “Course belum diikuti”
Menerima response validasi	

6). *Fitur aplikasi menyimpan progress belajar*

Nama Use case	Menyimpan progress belajar course
Aktor	Customer
Deskripsi	Aktor terkait dapat menyimpan progress belajar pada course
Prakondisi	Customer mengirim request dan payload bab course yang sudah diselesaikan
Pascakondisi	Progress belajar course berubah
Event Flow	
Normal Flow : Menyimpan progress belajar	
Aksi Aktor	Reaksi Sistem
Mengirim URL Request dan data payload bab course yang sudah diselesaikan	
	Menyimpan progress sesuai dengan data payload
	Mengirim pesan response “Progress belajar telah disimpan”
Menerima response sukses menyimpan progress	

7). *Fitur aplikasi menyimpan progress quiz*

Nama Use case	Menyimpan progress quiz
Aktor	Customer
Deskripsi	Aktor terkait dapat menyimpan quiz

Prakondisi	Customer mengirim request dan data payload dari jawaban quiz yang sudah diselesaikan
Pascakondisi	Menambahkan data jawaban quiz oleh user
Event Flow	
Normal Flow : Menyimpan progress quiz	
Aksi Aktor	Reaksi Sistem
Mengirim URL Request dan data payload jawaban quiz yang sudah diselesaikan	
	Menyimpan jawaban quiz sesuai dengan data payload
	Mengirim pesan response "Jawaban Quiz Telah disimpan"
Menerima response sukses menyimpan	

8). *Fitur aplikasi menambahkan webinar ke agenda pribadi*

Nama Use case	Menambahkan webinar ke agenda pribadi
Aktor	Customer
Deskripsi	Aktor terkait dapat mendaftarkan akun ke webinar tertentu
Prakondisi	Customer mengirim request untuk mendaftar webinar
Pascakondisi	Customer terdaftar pada kegiatan webinar
Event Flow	
Normal Flow : Mendaftar webinar	
Aksi Aktor	Reaksi Sistem
Mengirim URL Request untuk mendaftarkan akun ke webinar	
	Validasi data
	Menyimpan data webinar
	Mengirim response sukses
Menerima response sukses	
Alternatif Flow : User belum terdaftar di webinar	
	Menambahkan webinar ke agenda pribadi
	Mengirim pesan response "berhasil mendaftar webinar"
Menerima pesan response validasi	
Alternatif flow: User sudah terdaftar	
	Mengirim pesan response "Sudah terdaftar di webinar"
Menerima pesan response validasi	

9). *Fitur aplikasi melihat detail webinar pada agenda pribadi*

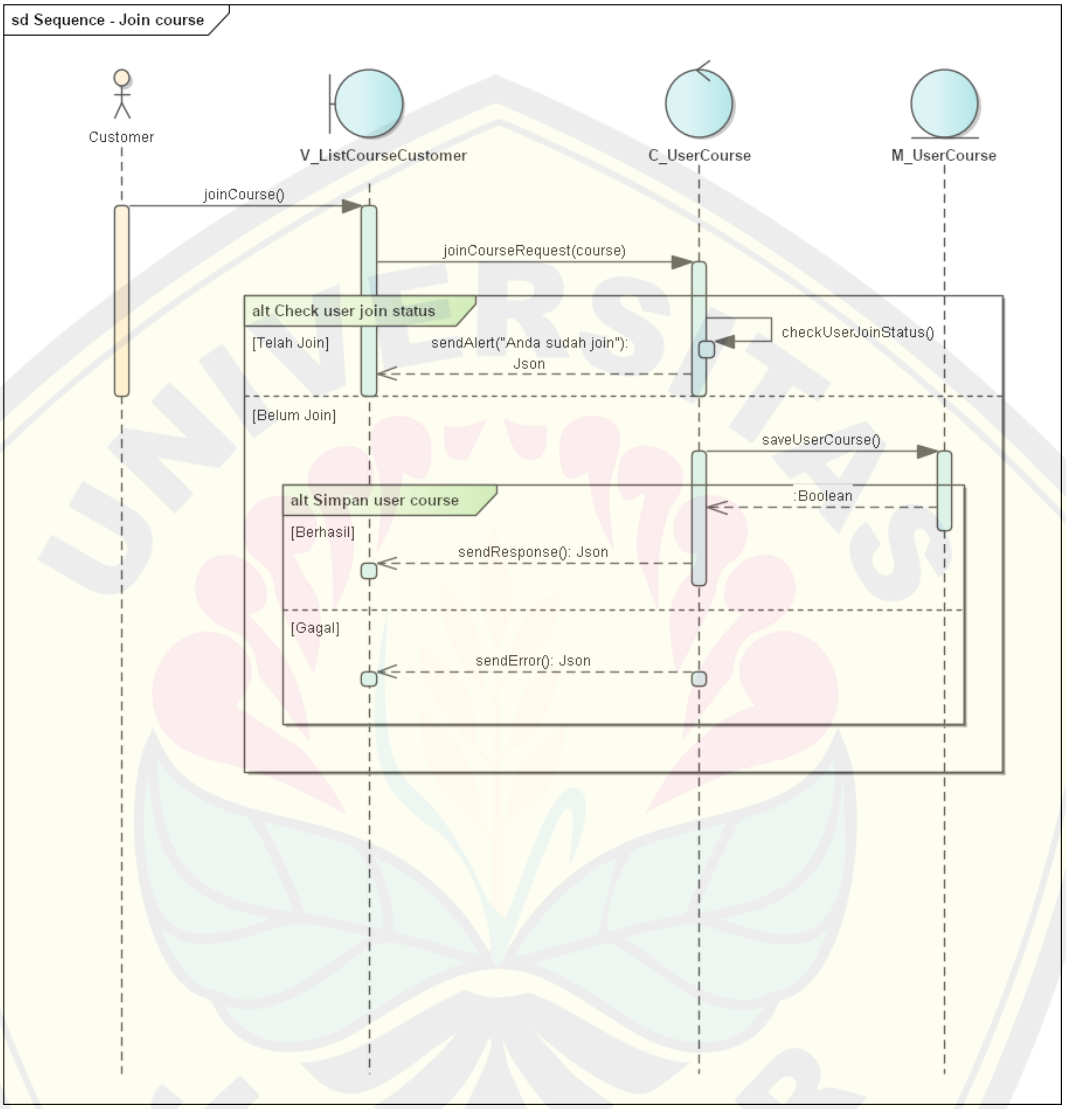
Nama Use case	Melihat detail webinar pada agenda pribadi
Aktor	Customer

Deskripsi	Aktor terkait dapat melihat detail webinar
Prakondisi	Customer mengirim request untuk melihat detail webinar
Pascakondisi	Data detail webinar dikirim terhadap customer
Event Flow	
Normal Flow : Melihat detail webinar	
Aksi Aktor	Reaksi Sistem
1. Mengirim URL Request untuk melihat detail webinar	
	Mengecek akun terkait telah mendaftar webinar
	Mengirim response data detail webinar
Menerima response data detail webinar	
Alternatif flow: Akun belum terdaftar	
	Mengirim pesan response “Belum mendaftar webinar”
Menerima pesan response	

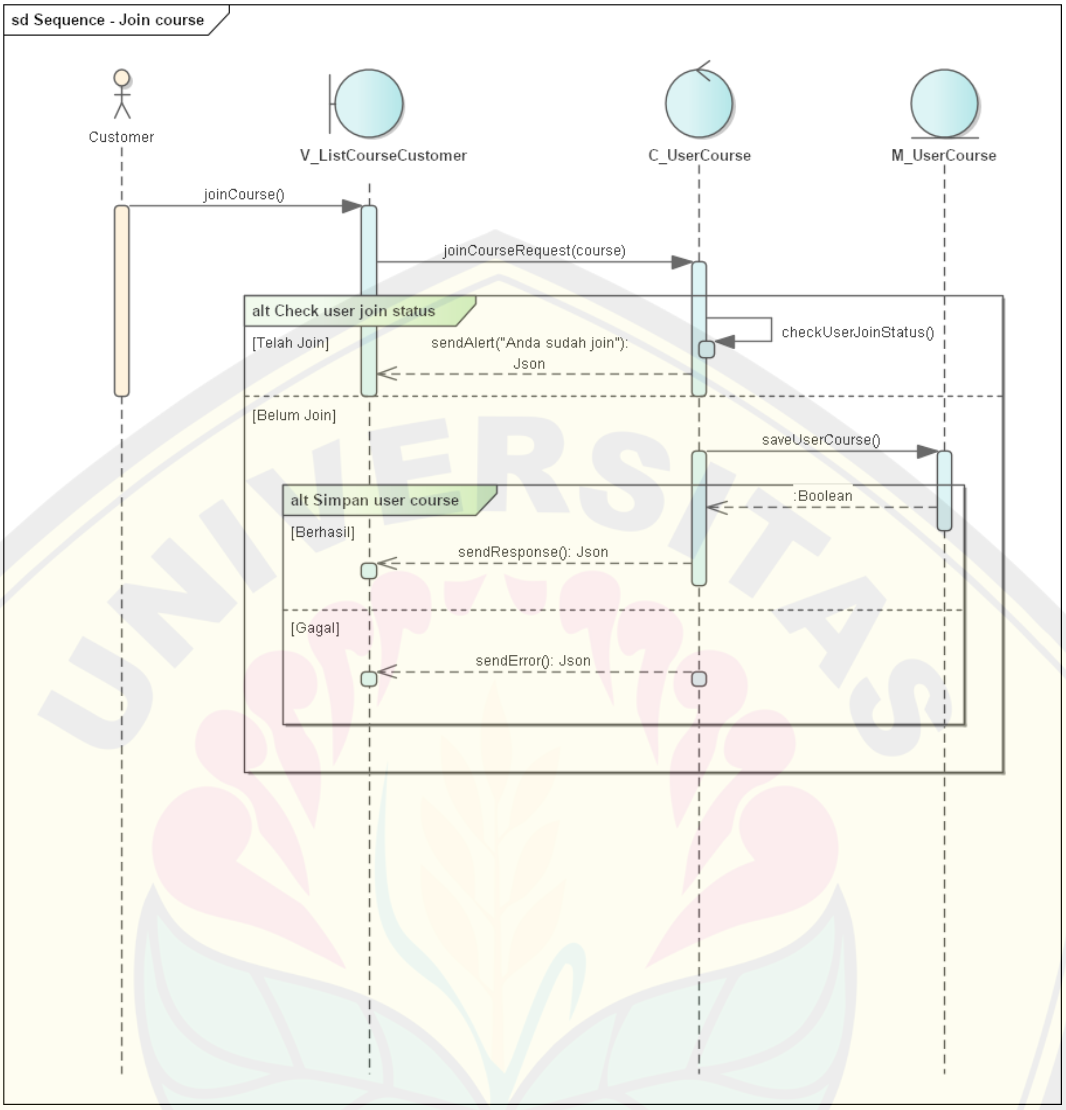
Lampiran 2 Sequence Diagram

1). *Fitur aplikasi menambah webinar*

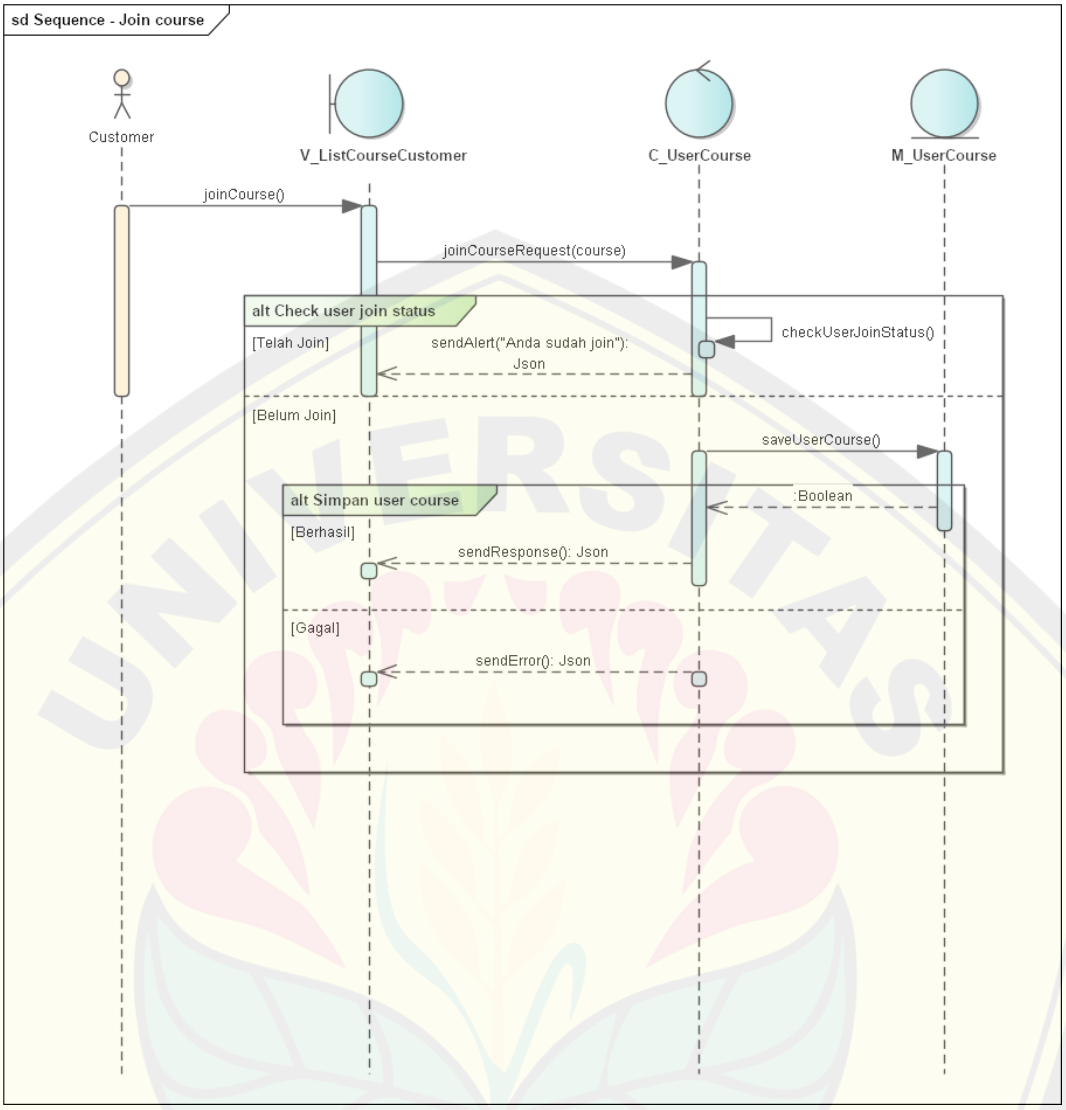
Berikut adalah penjelasan alur dari aksi role admin dan respon sistem dalam aktifitas menambah webinar pada website.



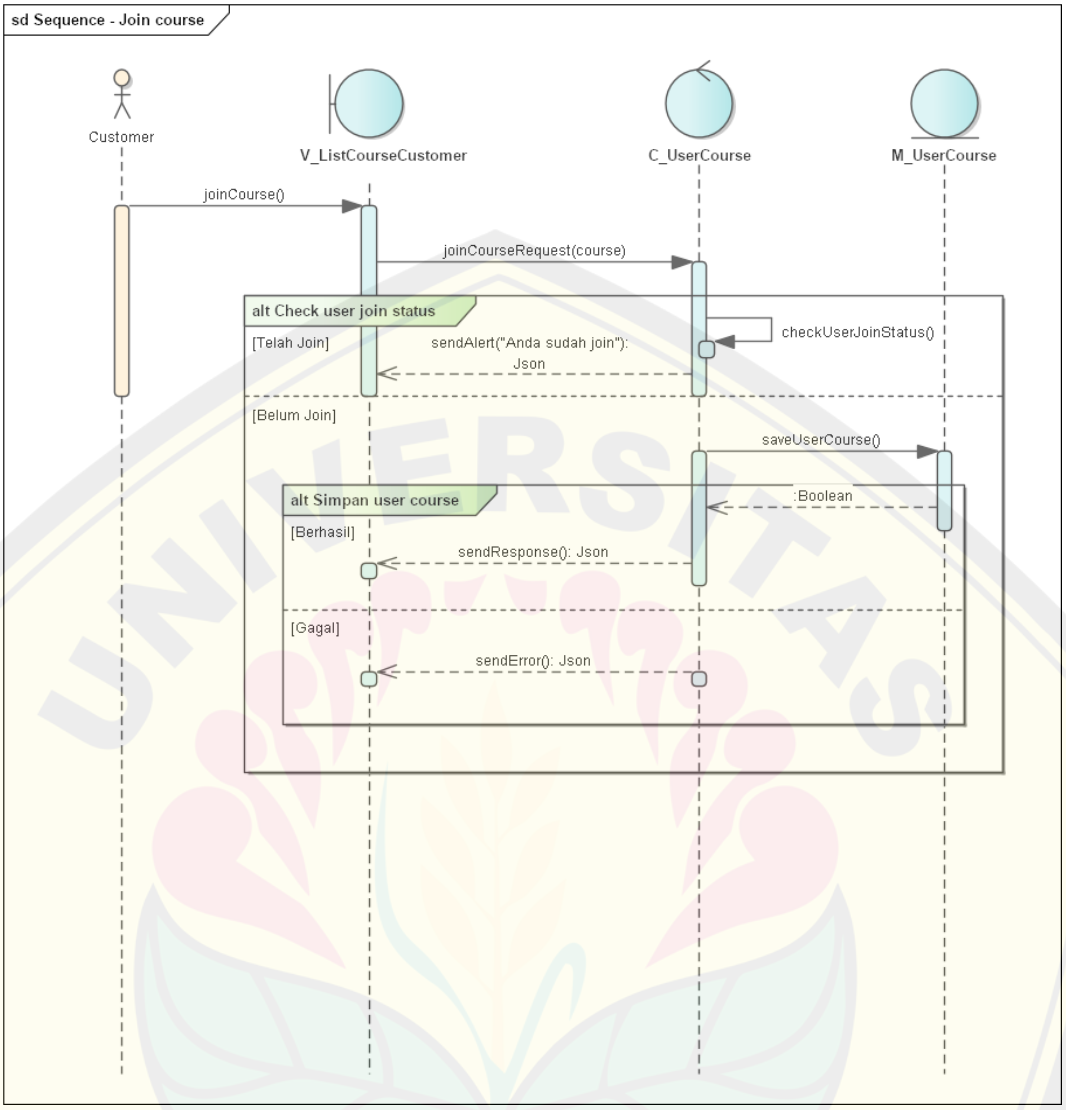
2). *Fitur aplikasi melihat webinar*



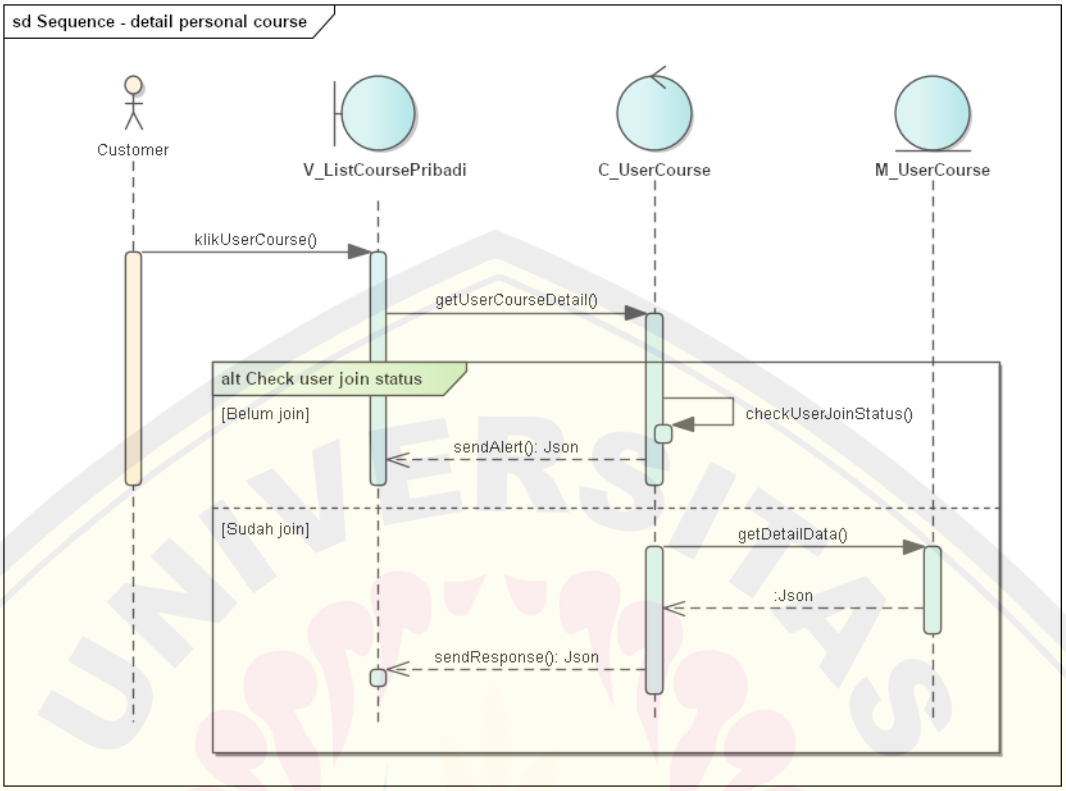
3). *Fitur aplikasi mengedit webinar*



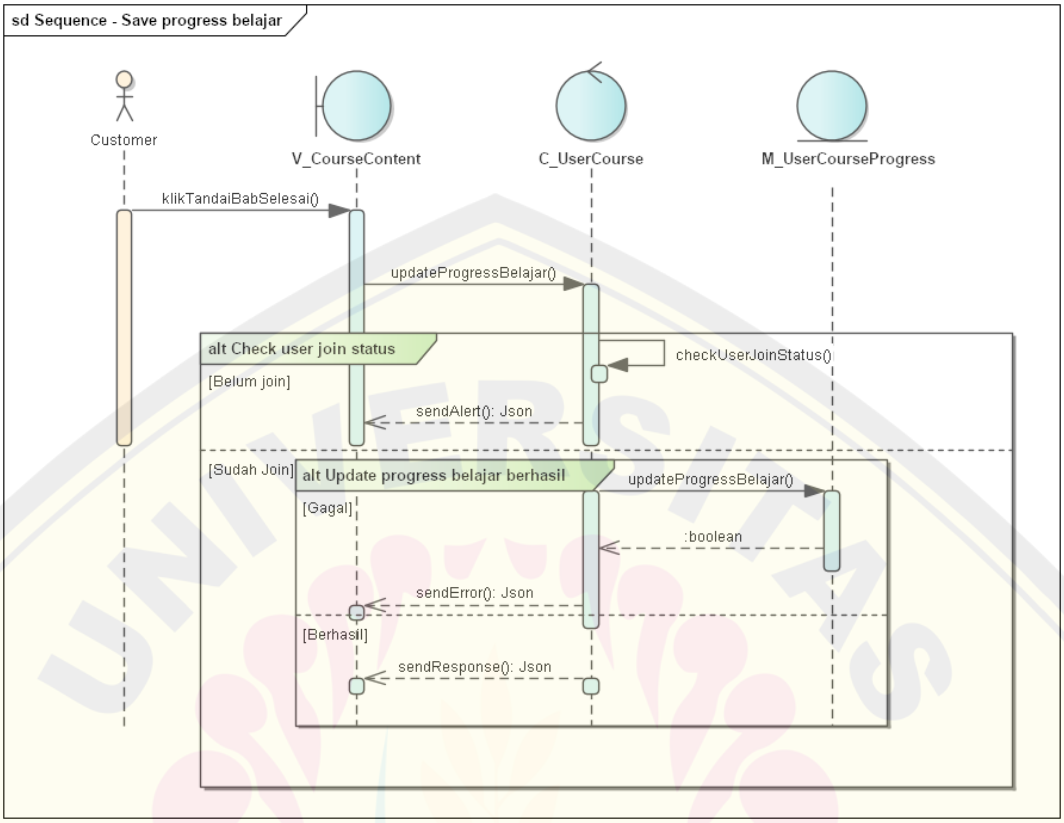
4). *Fitur aplikasi menambahkan course ke Pustaka pribadi*



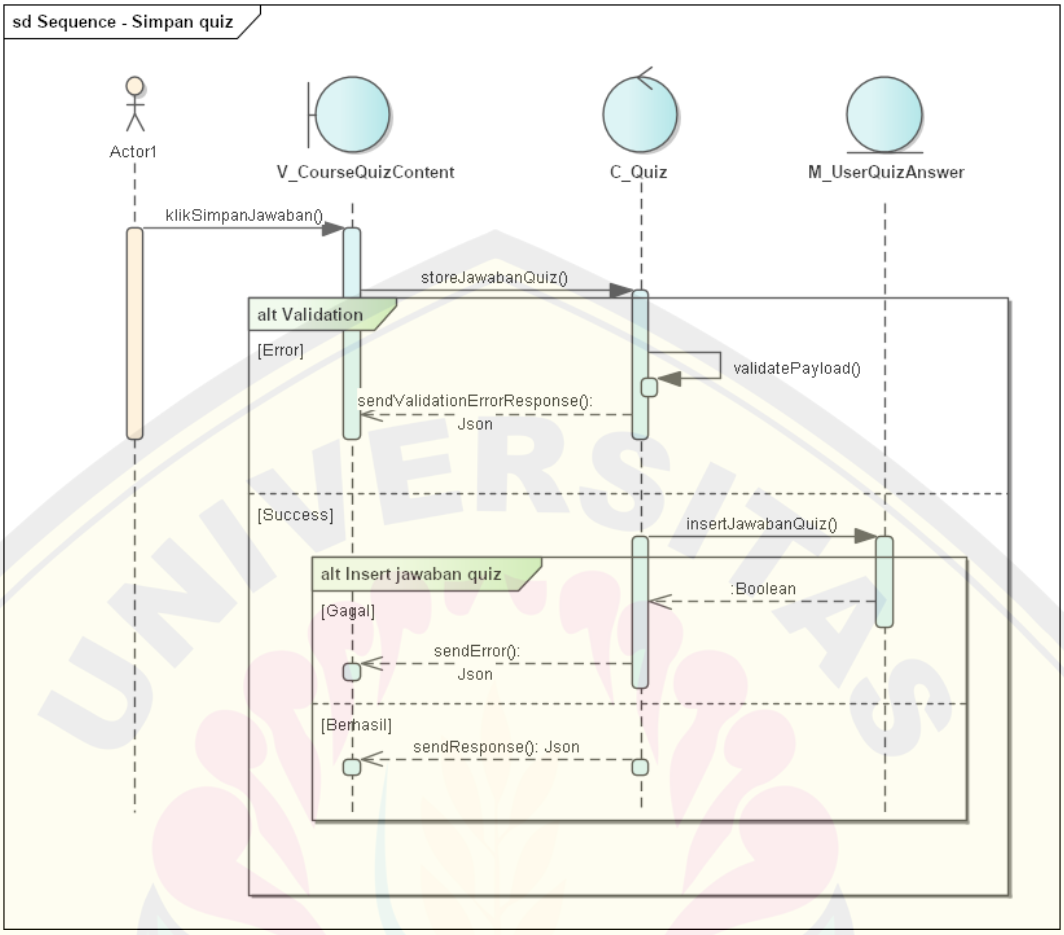
5). *Fitur aplikasi melihat detail course pada Pustaka pribadi*



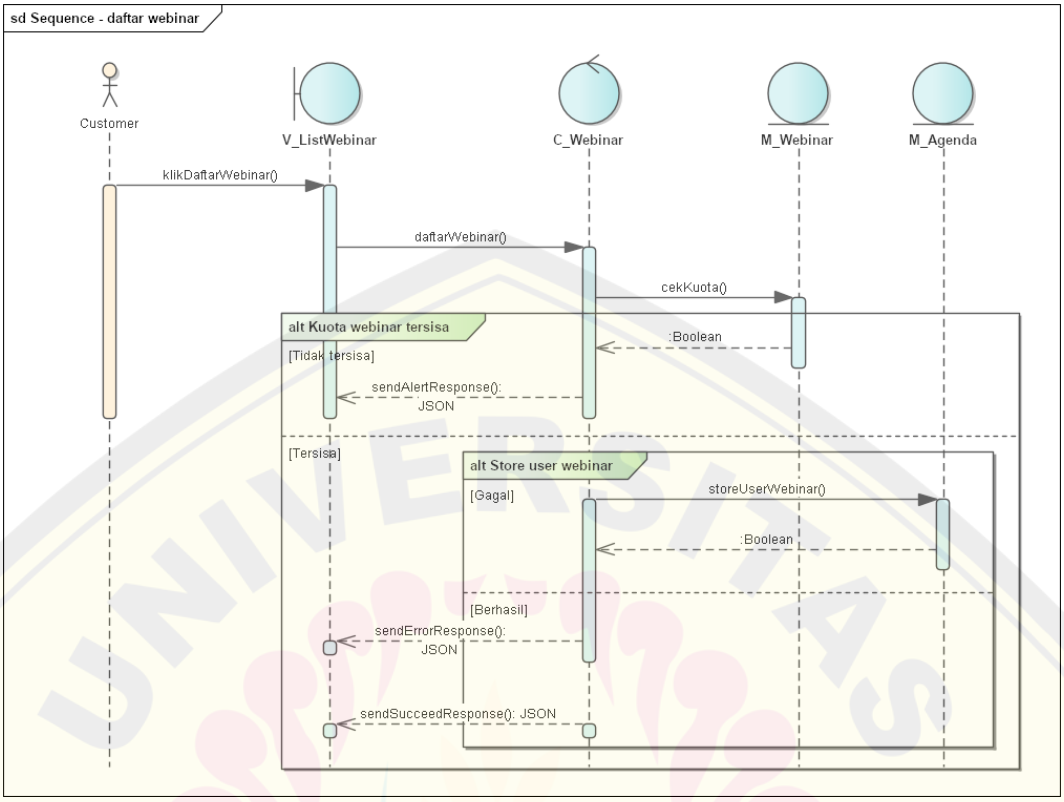
6). *Fitur aplikasi menyimpan progress belajar*



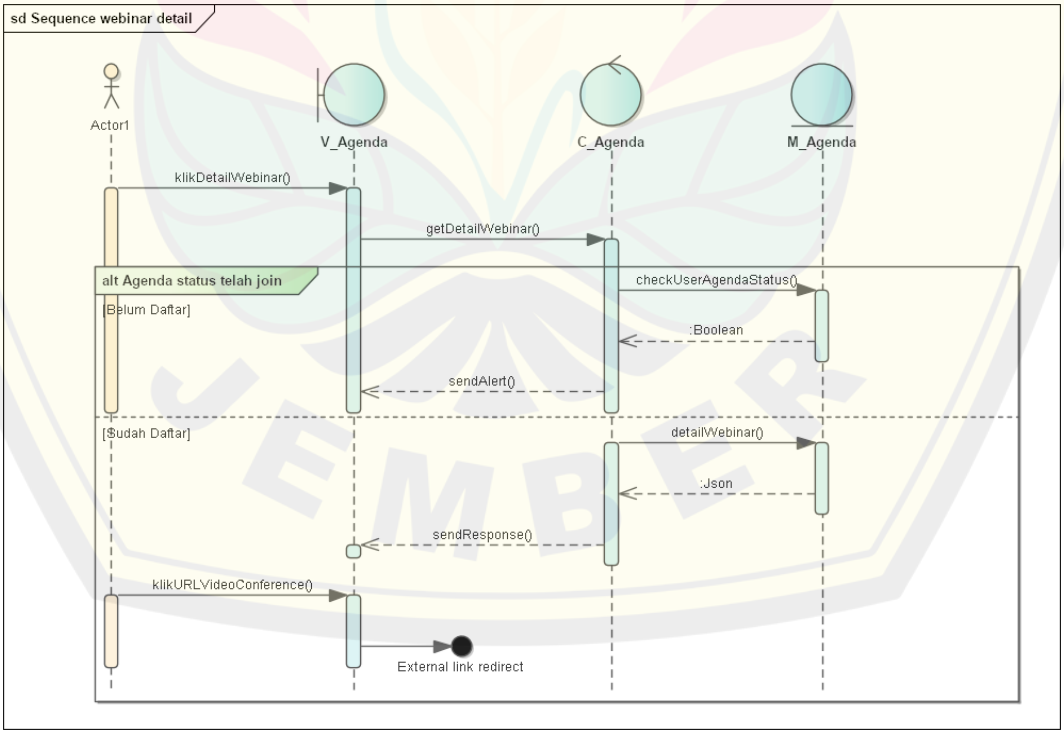
7). *Fitur aplikasi menyimpan progress quiz*



8). *Fitur aplikasi menambahkan webinar ke agenda pribadi*



9). *Fitur aplikasi melihat detail webinar pada agenda pribadi*



Lampiran 3 Pengembangan web service

1). *Fitur aplikasi menghapus course*

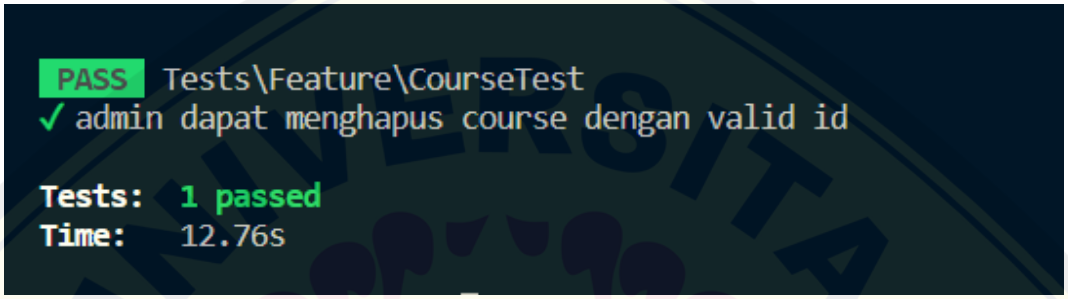
```
 test('admin dapat menghapus course dengan valid id', function() {
    $user = User::factory()->create([
        'role' => '1'
    ]);

    $courseCategory = CourseCategory::factory()->create();
    $thumbnail = UploadedFile::fake()->image('thumbnail.jpg')-
>storeAs('public/course', 'thumbnail.jpg');
    $file = File::factory()->create([
        'url' => $thumbnail,
        'type' => 'image'
    ]);

    $course = Course::factory()->create([
        'course_category' => $courseCategory->id,
        'thumbnail' => $file->id
    ]);

    CourseSection::factory()->count(3)->create([
        'course_id' => $course->id
    ]);

    actingAs($user)
    ->deleteJson(route('admin.course.destroy', $course->id))
    ->assertStatus(200);
});
```

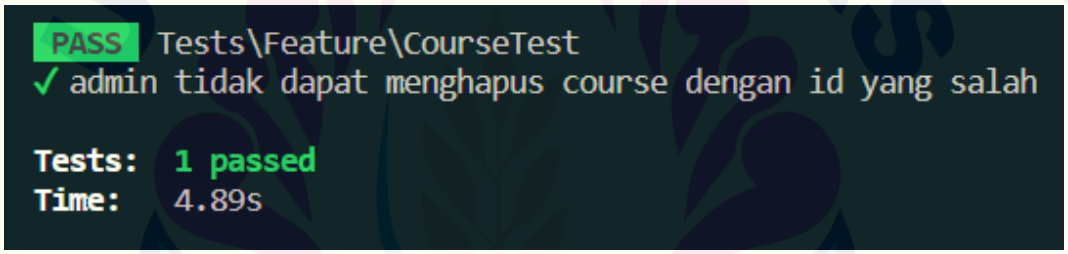
```
 test('admin tidak dapat menghapus course dengan id yang salah', function(){
    $user = User::factory()->create([
        'role' => '1'
    ]);

    actingAs($user)
    ->deleteJson(route('admin.course.destroy', 11111))
    ->assertStatus(404);
});
```

```
FAIL Tests\Unit\DeleteCoursesTest
× delete course with valid ID
× delete course with invalid ID

---
```

```
 /**
     * Remove the specified resource from storage.
     *
     * @param  int $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($course)
    {
        $courseChosen = Course::find($course);

        if(!$courseChosen) {
```

```
            return response()->json([
                'success' => false,
                'message' => 'Course Not Found!'
            ], 404);
        }

        $courseChosen->delete($course);

        return response()->json([
            'success' => true,
            'message' => 'Course Successfully Deleted!'
        ], 200);
    }
```



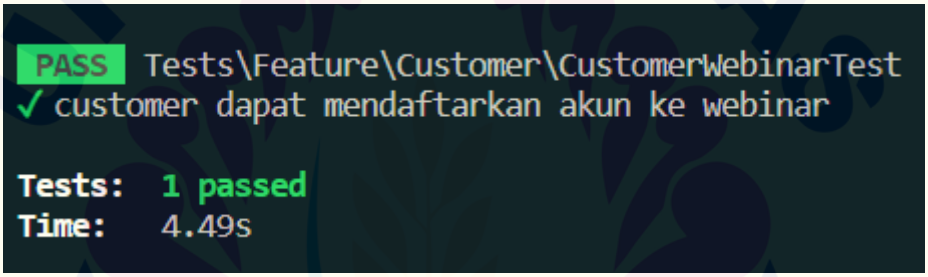


2). *Fitur aplikasi mendaftarkan akun ke webinar*

```
test('customer dapat mendaftarkan akun ke webinar', function(){
    $this->withoutExceptionHandling();

    $user = User::factory()->create([
        'role' => '0'
    ]);

    $webinar = Webinar::factory()->create([
        'price' => 0
    ]);

    actingAs($user)
    ->postJson(route('customer.join.webinar'), ['webinar_id' => $webinar->id])
    ->assertStatus(200);
});
```

```
public function joinWebinar(Request $request)
    {
        $user = Auth::user();
        $webinar = $request->webinar_id;
        $webinarChosen = Webinar::where('id', $webinar)->first();

        if(!$webinarChosen) {
```

```
            return response()->json([
                'success' => false,
                'message' => 'Webinar Not Found!'
            ], 404);
        }


        if($webinarChosen->price < 1) {
            $user->agendas()->attach($webinar);

            return response()->json([
                'success' => true,
                'message' => "Webinar successfully joined!"
            ], 200);
        }
        //PAYMENT GATEWAY SERVICE
        $user->agendas()->attach($webinar);

        return response()->json([
            'success' => true,
            'message' => "Webinar successfully joined!"
        ], 200);
    }
```

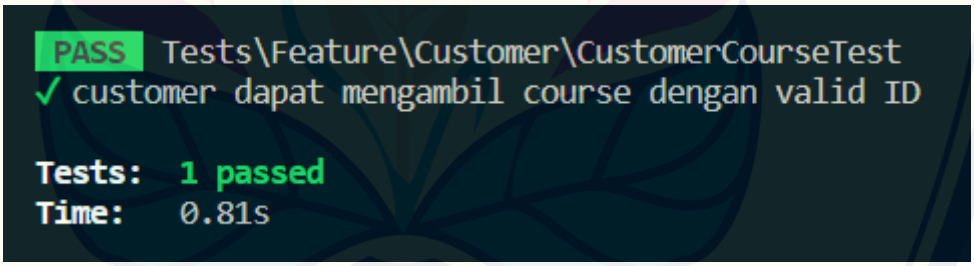


3). *Fitur aplikasi customer dapat mendaftarkan akun ke course tertentu*

```
test('customer dapat mengambil course dengan valid ID', function() {
    $this->withoutExceptionHandling();

    $user = User::factory()->create([
        'role' => '0'
    ]);
    $courseCategory = CourseCategory::factory()->create();
    $course = Course::factory()->create([
        'course_category' => $courseCategory->id
    ]);

    actingAs($user)
    ->postJson(route('customer.course.my-course'), [
        'course_id' => $course->id
    ])
    ->assertStatus(200);
});
```

```
/**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
```

```
    public function store(Request $request)
    {
        $input['user_id'] = Auth::user()->id;
        $input['course_id'] = $request->course_id;

        $course = Course::find($request->course_id);

        if(!$course) {
            return response()->json([
                'success' => false,
                'message' => 'Course Not Found!'
            ], 404);
        }

        if($course->price < 1) {
            UserCourse::create([...$input, 'status' => 'active']);

            return response()->json([
                'success' => true,
                'message' => 'You successfully join this course!'
            ], 200);
        }

        UserCourse::create([...$input, 'status' => 'active']);

        return response()->json([
            'success' => true,
            'message' => 'You successfully join this course!'
        ], 200);
        /* $checkoutService = new CheckoutService();
        dd($checkoutService->createInvoice()); */

    }
```



4). *Fitur aplikasi menambah webinar*

```
 <?php

use App\Models\File;
use App\Models\User;
use App\Models\Webinar;
use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Http\UploadedFile;

use function Pest\Laravel\actingAs;

uses(RefreshDatabase::class);

test("admin dapat menambah webinar baru dengan valid data", function () {
    $user = User::factory()->create([
```

```
        "role" => "1",
    ]);

    $webinar = Webinar::factory()->make();
    $webinar["thumbnail"] = UploadedFile::fake()->image("thumbnail.jpg");
    actingAs($user)
        ->postJson(route("admin.webinar.store"), $webinar->toArray())
        ->assertStatus(200);
});

test(
    "admin tidak dapat menambah webinar baru dengan invalid data",
    function () {
        $user = User::factory()->create([
            "role" => "1",
        ]);

        $webinar = Webinar::factory()->make();
        $webinar["webinar_name"] = null;
        $webinar["thumbnail"] = UploadedFile::fake()->image("thumbnail.jpg");
        actingAs($user)
            ->postJson(route("admin.webinar.store"), $webinar->toArray())
            ->assertStatus(422);
    },
);
```

```
FAIL  Tests\Unit\AddWebinarTest
× Add new webinar with valid data
× Add new webinar with invalid data

---

• Tests\Unit\AddWebinarTest > Add new webinar with valid data
Header [Content-Type] was found, but value [text/html; charset=UTF-8] does not match [application/json].
Failed asserting that two strings are equal.
```

```
/**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $validated = Validator::make($request->all(), [
            "webinar_name" => "required",
            "description" => "required",
            "price" => "required",
            "link" => "required",
            "started_at" => "required",
            "register_end_at" => "required",
        ]);

        if($validated->fails()) {
            return response()->json([
                'success' => false,
                'message' => $validated->errors()
            ], 422);
        }

        DB::beginTransaction();
        try {
            $webinar = Webinar::create($request->except('thumbnail'));
```

```
            if($request->has('thumbnail')) {
                $thumbnail = $request->file('thumbnail');
                $thumbnailName = $thumbnail->storeAs('public/webinar', time() .
$thumbnail->getClientOriginalName());
                $wt = $webinar->thumbnail()->create([
                    'url' => $thumbnailName,
                    'type' => 'image'
                ]);

                $webinar->update([
                    'thumbnail' => $wt->id
                ]);
            }

            DB::commit();
            return response()->json([
                'success' => true,
                'message' => "Webinar successfully created!"
            ], 200);
        } catch (\Throwable $th) {
            DB::rollBack();
            throw $th;
        }
    }
```

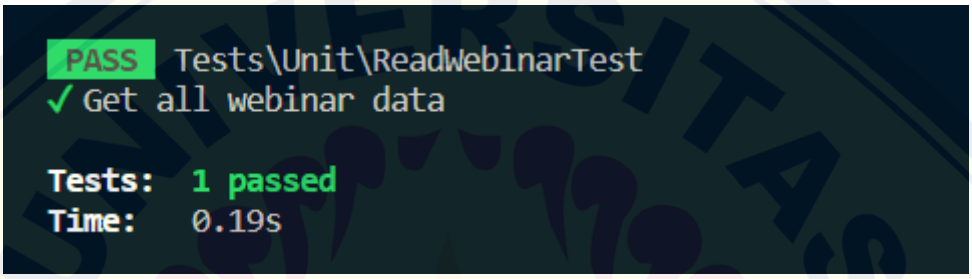
```
PASS  Tests\Unit\AddWebinarTest
✓ Add new webinar with valid data
✓ Add new webinar with invalid data

Tests:  2 passed
Time:   0.25s
```

5). *Fitur aplikasi melihat list webinar*

```
test('admin dapat melihat daftar webinar', function() {
    $this->withoutExceptionHandling();
    $user = User::factory()->create([
        'role' => '1'
    ]);

    $webinar = Webinar::factory()->create();

    actingAs($user)
    ->getJson(route('admin.webinar.index'))
    ->assertStatus(200);
});
```

```
FAIL  Tests\Unit\ReadWebinarTest
× Get all webinar data

---

• Tests\Unit\ReadWebinarTest > Get all webinar data
Header [Content-Type] was found, but value [text/html; charset=UTF-8] does not match [application/json].
Failed asserting that two strings are equal.
```

```php
/**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $data = Webinar::with('thumbnail')
        ->orderBy('created_at', 'desc')
        ->get();

        return response()->json([
            'success' => true,
            'data' => $data
        ], 200);
    }
```



6). *Fitur aplikasi menambah artikel*

```php
<?php

use App\Models\Article;
use App\Models\User;
use Illuminate\Http\UploadedFile;

use function Pest\Laravel\actingAs;

test("admin dapat membuat artikel dengan valid data", function () {
    $this->withoutExceptionHandling();

    $user = User::factory()->create([
        "role" => "1",
    ]);

    $article = Article::factory()->make();
    $article["thumbnail"] = UploadedFile::fake()->image("thumbnail.jpg");

    actingAs($user)
        ->postJson(route("admin.article.store"), $article->toArray())
        ->assertStatus(200);
});

test("admin tidak dapat membuat artikel dengan invalid data", function () {
    $this->withoutExceptionHandling();

    $user = User::factory()->create([
        "role" => "1",
    ]);

    $article = Article::factory()->make();
    $article["thumbnail"] = UploadedFile::fake()->image("thumbnail.jpg");
```

```
    $article["article_title"] = null;
    actingAs($user)
        ->postJson(route("admin.article.store"), $article->toArray())
        ->assertStatus(422);
});
```

```
FAIL  Tests\Unit\AddArticleTest
× Add new article with valid data
× Add new article with invalid data

---

• Tests\Unit\AddArticleTest > Add new article with valid data
Header [Content-Type] was found, but value [text/html; charset=UTF-8] does not match [application/json].
Failed asserting that two strings are equal.
```

```
 /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $validated = Validator::make($request->all(), [
            'article_title' => 'required',
            'article_content' => 'required',
        ]);

        if($validated->fails()) {
            return response()->json([
                'success' => false,
                'message' => $validated->errors()
            ], 422);
        }

        $request->merge(['created_by' => Auth::user()->id]);
        $article = Article::create($request->except('thumbnail'));

        if($request->has('thumbnail')) {
            $thumbnail = $request->file('thumbnail')->storeAs('public/article',
time().$request->file('thumbnail')->getClientOriginalName());
            $at = $article->image()->create([
                'url' => $thumbnail,
                'type' => 'image'
            ]);

            $article->update([
                'thumbnail' => $at->id
            ]);
        }

        return response()->json([
            'success' => true,
            'message' => 'Article successfully created!'
        ], 200);
    }
```

```
 PASS  Tests\Unit\AddArticleTest
✓ Add new article with valid data
✓ Add new article with invalid data

Tests:  2 passed
Time:   0.89s
```

7). *Fitur aplikasi melihat artikel*

```php
test('admin dapat melihat daftar artikel', function() {
    $this->withoutExceptionHandling();

    $user = User::factory()->create([
        'role' => '1'
    ]);

    $article = Article::factory()->create([
        'created_by' => $user->id
    ]);

    actingAs($user)
    ->getJson(route('admin.article.index'))
    ->assertStatus(200);
});
```

```
 FAIL  Tests\Unit\ReadArticlesTest
× Get all articles data

---

• Tests\Unit\ReadArticlesTest > Get all articles data
Header [Content-Type] was found, but value [text/html; charset=UTF-8] does not match [application/json].
Failed asserting that two strings are equal.
```

```php
/**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $data = Article::with(['image'])->orderBy('created_at', 'desc')->get();

        return response()->json([
            'success' => true,
            'data' => $data
        ], 200);
    }
```

```
 PASS  Tests\Unit\ReadArticlesTest
✓ Get all articles data

Tests:  1 passed
Time:   0.19s
```

8). *Fitur aplikasi login*

```php
<?php

use function Pest\Laravel\postJson;

uses(Tests\TestCase::class);

test("Login using valid data and valid credential", function () {
    postJson("api/login", [
        "email" => "test1@example.com",
        "password" => "123123123",
    ])
        ->assertHeader("Content-Type", "application/json")
        ->assertJsonStructure(["success", "message", "data"])
        ->assertStatus(200);
});

test("Login using invalid data", function () {
    postJson("api/login", [
        "email" => "test1@example.com",
        //"password" => "123123123"
    ])
        ->assertHeader("Content-Type", "application/json")
        ->assertJsonStructure(["success", "message"])
        ->assertStatus(422);
});

test("Login using invalid credential (wrong email)", function () {
    postJson("api/login", [
        "email" => "wrongemail@example.com",
        "password" => "123123123",
    ])
        ->assertHeader("Content-Type", "application/json")
        ->assertJsonStructure(["success", "message"])
        ->assertStatus(404);
});

test("Login using invalid credential (wrong password)", function () {
    postJson("api/login", [
        "email" => "test1@example.com",
        "password" => "wrongpassword",
    ])
        ->assertHeader("Content-Type", "application/json")
        ->assertJsonStructure(["success", "message"])
        ->assertStatus(403);
});
```

```
FAIL  Tests\Unit\LoginTest
× Login using valid data and valid credential
× Login using invalid data
× Login using invalid credential (wrong email)
× Login using invalid credential (wrong password)

---

• Tests\Unit\LoginTest > Login using valid data and valid credential
Header [Content-Type] was found, but value [text/html; charset=UTF-8] does not match [application/json].
Failed asserting that two strings are equal.
```

```php
public function login(Request $request)
    {
        $validated = Validator::make($request->all(), [
            'email' => 'required',
```

```
            'password' => 'required'
        ]);

        if($validated->fails()) {
            return response()->json([
                'success' => false,
                'message' => $validated->errors()
            ], 422);
        }

        $who = User::where('email', $request->email);

        if(! $who->exists()) {
            return response()->json([
                'success' => false,
                'message' => 'Credential not found!'
            ], 404);
        }

        $checkValidity = Hash::check($request->password, $who->first()->password);

        if($checkValidity) {
            $user = $who->first();
            $token = $user->createToken('token-ts');

            $data['token'] = $token->plainTextToken;
            $data['user'] = $user;

            return response()->json([
                'success' => true,
                'message' => 'Login successfull!',
                'data' => $data
            ], 200);
        }

        return response()->json([
            'success' => false,
            'message' => 'Wrong password!'
        ], 403);
    }
```

```
PASS  Tests\Unit\LoginTest
✓ Login using valid data and valid credential
✓ Login using invalid data
✓ Login using invalid credential (wrong email)
✓ Login using invalid credential (wrong password)

Tests:  4 passed
Time:   1.02s
```

Lampiran 4 Test case scenario customer

Fitur	Test Case ID	Test Case Description	Endpoint	Method	Request Body	Skenario Testing
Register User	TC01	Menambahkan user baru dengan data yang valid	/users/register	POST	Username: string Email: string Name: string Tanggal lahir: date	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek Body response memiliki property user id dan tidak kosong
	TC02	Menambahkan user baru dengan data yang duplikat	/users/register	POST	Username: string Email: string Name: string Tanggal lahir: date	Response harus memiliki status code 422 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
	TC03	Menambahkan user baru dengan data yang tidak valid	/users/register	POST	Data dinamis dengan invalid payload	Response harus memiliki status code 422 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
Login	TC04	Login dengan valid kredensial	/users/login	POST	Email: string Password: string	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek

Fitur	Test Case ID	Test Case Description	Endpoint	Method	Request Body	Skenario Testing
						Body response memiliki property token dan tidak kosong
	TC05	Login dengan invalid kredensial	/users/login	POST	Email: string Password: string	Response harus memiliki status code 401 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
	TC06	Login dengan invalid data	/users/login	POST	Data dinamis dengan invalid payload	Response harus memiliki status code 422 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
Course	TC07	Melihat daftar course	/courses	GET		Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat array objek
	TC08	Menambahkan course ke daftar course pribadi	/users/mycourses	POST		Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek

Fitur	Test Case ID	Test Case Description	Endpoint	Method	Request Body	Skenario Testing
		Melihat daftar course pribadi	/users/mycourses			Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat array objek
Artikel	TC10	Melihat artikel	/articles			Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat array objek
Webinar		Melihat daftar webinar				Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat array objek
		Mendaftarkan akun pribadi ke webinar	/webinars/join			Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek

Lampiran 5 Test case scenario admin

Fitur	Test Case ID	Test Case Description	Endpoint	Method	Request Body	Skenario Testing
Register User	TC01	Menambahkan user baru dengan data yang valid	/users/register	POST	Username: string Email: string Name: string Tanggal lahir: date	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek Body response memiliki property user id dan tidak kosong
	TC02	Menambahkan user baru dengan data yang duplikat	/users/register	POST	Username: string Email: string Name: string Tanggal lahir: date	Response harus memiliki status code 422 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
	TC03	Menambahkan user baru dengan data yang tidak valid	/users/register	POST	Data dinamis dengan invalid payload	Response harus memiliki status code 422 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
Login	TC04	Login dengan valid kredensial	/users/login	POST	Email: string Password: string	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek

Fitur	Test Case ID	Test Case Description	Endpoint	Method	Request Body	Skenario Testing
						Body response memiliki property token dan tidak kosong
	TC05	Login dengan invalid kredensial	/users/login	POST	Email: string Password: string	Response harus memiliki status code 401 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
	TC06	Login dengan invalid data	/users/login	POST	Data dinamis dengan invalid payload	Response harus memiliki status code 422 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
Create Course	TC07	Menambahkan cource baru dengan valid data	/courses	POST	Name : string Price : double Thumbnail : file Description : text Discount: double Course_sections: array	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
	TC08	Menambahkan cource baru dengan invalid data	/courses	POST	Data dinamis dengan invalid payload	Response harus memiliki status code 422 Header response Content-Type memiliki nilai application/json Body response harus berformat objek

Fitur	Test Case ID	Test Case Description	Endpoint	Method	Request Body	Skenario Testing
List Course	TC09	Melihat data course	/courses	GET	-	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
Edit Course	TC10	Mengubah data course dengan valid data	/courses/{id}	PUT	Name : string Price : integer Thumbnail : file Description : text	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
	TC11	Mengubah data course dengan invalid data	/courses/{id}	PUT	Data dinamis dengan invalid payload	Response harus memiliki status code 422 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
Delete Course	TC12	Menghapus data course dengan valid ID	/courses/{id}	DELETE	-	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
	TC13	Menghapus data course	/courses/{id}	DELETE	-	Response harus memiliki status code 404

Fitur	Test Case ID	Test Case Description	Endpoint	Method	Request Body	Skenario Testing
		dengan invalid ID				Header response Content-Type memiliki nilai application/json Body response harus berformat objek
Create Webinar	TC14	Menambahkan webinar baru dengan valid data	/webinars	POST	Name : string Date : date Status : enum[‘selesai’, ‘belum’] Thumbnail : file Meeting_link : string	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
	TC15	Menambahkan webinar baru dengan invalid data	/webinars	POST	Data dinamis dengan invalid payload	Response harus memiliki status code 422 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
List Webinar	TC16	Melihat data webinar	/webinars	GET	-	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
Edit Webinar	TC17	Mengubah data webinar dengan valid data	/webinars/{id}	PUT	Name : string Date : date Status : enum[‘sele	Response harus memiliki status code 200 Header response Content-Type

Fitur	Test Case ID	Test Case Description	Endpoint	Method	Request Body	Skenario Testing
					sai', 'belum'] Thumbnail : file Meeting_link : string	memiliki nilai application/json Body response harus berformat objek
	TC18	Mengubah data webinar dengan invalid data	/webinars{id}			Response harus memiliki status code 422 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
Delete Webinar	TC19	Menghapus data webinar dengan valid ID	/webinars/{id}	DELETE	-	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
	TC20	Menghapus data webinar dengan invalid ID	/webinars/{id}	DELETE	-	Response harus memiliki status code 404 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
Create Article	TC21	Menambahkan artikel baru dengan valid data	/articles	POST	Name : string Date : date Status : enum['selesai', 'belum']	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json

Fitur	Test Case ID	Test Case Description	Endpoint	Method	Request Body	Skenario Testing
					Thumbnail : file Meeting_link : string	Body response harus berformat objek
	TC22	Menambahkan artikel baru dengan invalid data	/articles	POST	Data dinamis dengan invalid payload	Response harus memiliki status code 422 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
List Artikel	TC23	Melihat data artikel	/articles	GET	-	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
Edit Artikel	TC24	Mengubah data artikel dengan valid data	/articles/{id}	PUT	Name : string Date : date Status : enum[‘selesai’, ‘belum’] Thumbnail : file Meeting_link : string	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json Body response harus berformat objek
Delete Artikel	TC25	Menghapus data artikel dengan valid ID	/articles/{id}	DELETE	-	Response harus memiliki status code 200 Header response Content-Type memiliki nilai application/json

Fitur	Test Case ID	Test Case Description	Endpoint	Method	Request Body	Skenario Testing
						Body response harus berformat objek
	TC26	Menghapus data artikel dengan invalid ID	/articles/{id}	DELETE	-	Response harus memiliki status code 404 Header response Content-Type memiliki nilai application/json Body response harus berformat objek