Pengembangan Aplikasi Mentor untuk Studi Kasus Pembelajaran Keterampilan Digital dengan Metode *Test-Driven Development*

1st Aditya Rohman
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
enteryornics@student.telkomuniversity.ac.id

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

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

Abstrak—Perkembangan teknologi yang kian pesat berperan penting untuk kemajuan dalam berbagai aspek kehidupan masyarakat terutama aspek ekonomi di Indonesia. Hal tersebut dapat dilihat dari mulai banyaknya perusahaan rintisan teknologi yang berlomba-lomba menemukan dan mengembangkan solusi untuk berbagai kebutuhan masyarakat. Sejalan dengan kemajuan tersebut, kebutuhan akan orang-orang yang ahli di bidang keterampilan digital juga semakin meningkat. Kebutuhan akan talenta digital pada keterampilan digital seperti pemrograman dan desain mencapai angka 600 ribu per tahun. Pada Tugas Akhir ini membahas tentang pengembangan aplikasi sebagai media pembelajaran keterampilan digital dengan dukungan mentor dan komunitas di dalamnya. Secara garis besar aplikasi yang dikembangkan terdiri dari layanan kursus daring yang didukung oleh adanya forum diskusi untuk bertanya kepada mentor dan sesi live mentorship berkala dengan mentor. Pengembangannya menerapkan prinsip *Test-Driven Development* (TDD) dimana pengujian unit lebih didahulukan sebelum membuat sebuah fitur. Penerapan metode *Test-Driven Development* dalam pengembangan berdampak pada tingkat usabilitas dari aplikasi yang sangat baik. Hal tersebut dapat dilihat dari hasil pengujian usabilitas pada pengguna dengan menggunakan skala *Likert*. Seluruh fitur dalam aplikasi mendapatkan nilai *index* di atas 90% dimana nilai *index* terendah adalah 92,90% dan nilai *index* tertinggi mencapai 96,12%.

Kata kunci—aplikasi, keterampilan digital, mentor, perangkat seluler, test-driven development, usabilitas.

I. PENDAHULUAN

Perkembangan teknologi yang kian pesat berperan penting untuk kemajuan dalam berbagai aspek kehidupan masyarakat terutama aspek ekonomi di Indonesia. Hal tersebut dapat dilihat dari mulai banyaknya perusahaan rintisan teknologi yang berlomba-lomba menemukan dan mengembangkan solusi untuk berbagai kebutuhan masyarakat. Sejalan dengan kemajuan tersebut, kebutuhan akan orang-orang yang ahli di bidang keterampilan digital juga semakin meningkat. Menurut Dirjen Pendidikan Tinggi, Riset dan Teknologi (2021) bahwa pemintaan akan talenta digital di Indonesia mencapai angka 600 ribu setiap tahun. Hal serupa juga dituturkan oleh Presiden Republik Indonesia, Ir. H. Joko Widodo pada acara *Google for Indonesia* (2020) bahwa Indonesia masih membutuhkan total 9 juta talenta digital untuk menunjang prekonomian hingga tahun 2035.

Untuk memenuhi kebutuhan tersebut, disamping memperoleh pendidikan formal pada berbagai bidang keterampilan digital, dibutuhkan pula sebuah media untuk dapat belajar lebih dalam tentang suatu bidang keterampilan digital sesuai dengan kebutuhan di industri. Mempelajari keterampilan digital secara mandiri mungkin mudah bagi sebagian orang, namun masih banyak yang kesulitan ketika belajar sendiri. Mereka membutuhkan seorang mentor atau pembimbing yang dapat mengarahkan dan memberikan saran serta bantuan dalam proses belajar. Oleh karena itu, penulis berinisiatif untuk mengembangkan aplikasi sebagai media untuk belajar berbagai bidang keterampilan digital dengan dukungan mentor dan komunitas sebagai salah satu dari sekian banyak solusi untuk permasalahan di atas.

Dalam pengembangannya dilakukan dengan metode *Test-Driven Development* (TDD) dimana pengujian *unit* aplikasi lebih didahulukan sebelum membuat implementasi fitur yang sesungguhnya. Pada tahap akhir, aplikasi diuji kembali dengan menggunakan *Usability Testing* kepada pengguna potensial untuk mengukur bagaimana pengaruh pengembangan aplikasi dengan metode *Test-Driven Development* terhadap kualitas dari aplikasi yang dihasilkan.

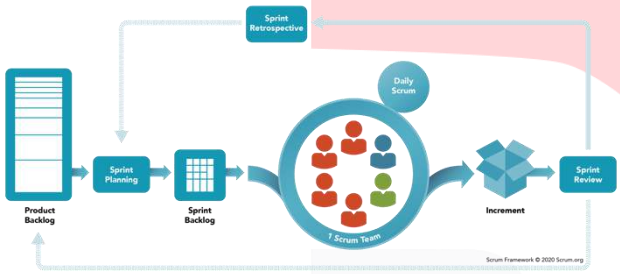
II. KAJIAN TEORI

A. Mentor

Mentor dideskripsikan sebagai seseorang yang sudah berpengalaman di bidang tertentu dan memiliki minat serta aktif dalam pengembangan karir profesional yang lebih muda. Ada beberapa prinsip yang ada dalam diri seorang mentor yaitu: (a) mereka mendorong impian dan mendukung aspirasi karir anak didiknya, (b) mereka sering memberikan kesempatan dan melibatkan anak didiknya dalam sebuah pekerjaan atau proyek [1]. Kegiatan *mentorship* memberikan keuntungan bagi kedua pihak. Mentor mendapatkan jejaring yang lebih luas, potensial *partner*, dan retensi ilmu dan keterampilan yang meningkat. Adapun anak didik (*mentee*) mendapatkan praktik terbaik untuk keterampilan tertentu dan kemajuan karir mereka [2].

B. Pengembangan Aplikasi Mobile

Perangkat seluler pintar (*smartphone*) merupakan salah satu perangkat komputasi yang berkembang sangat pesat selama beberapa tahun ini sejak dirilisnya dua buah sistem operasi terbesar untuk perangkat seluler yaitu *Android* dan *iOS*. Dalam pengembangan aplikasi secara umum termasuk pengembangan aplikasi mobile, terdapat berbagai model daur ulang pengembangan perangkat lunak atau Software *Development Life Cycle* (SDLC) yang umum digunakan. Contohnya seperti *Waterfall Model*, *B-Model*, *Incremental Model*, *V-Model*, *Spiral Model*, *Wheel-and-spoke Model*, *Unified Process Model*, dan *Rapid Application Development* (RAD) [3]. Pada Tugas Akhir ini, karena pengembangannya dalam lingkungan *WRAP Entrepreneurship Bandung Techno Park* (WRAP BTP), sistem yang dikembangkan memiliki target dan spesifikasi yang mungkin untuk berubah seiring dengan proses validasi ide dan pengguna. Maka, penulis menggunakan metode *Agile Scrum* yang berupakan bagian dari *Rapid Application Development* (RAD).



GAMBAR 1
DIAGRAM PROSES SCRUM

Agile secara istilah adalah kemampuan untuk membuat dan merespon perubahan dalam sebuah lingkungan yang tidak pasti [4]. Dalam pengembangan aplikasi, *Agile* diartikan sebagai proses pengembangan aplikasi dengan iterasi atau tahapan yang dinamis. Adapun *Scrum* yang merupakan bagian dari *Agile* adalah sebuah kerangka kerja (*framework*) dimana sekelompok orang dapat bekerja untuk menyelesaikan masalah adaptif yang kompleks, sembari secara produktif dan kreatif mengembangkan dan memberikan sebuah produk dengan nilai tinggi untuk permasalahan tersebut [4]. Terdapat dua komponen utama dalam kerangka kerja *Scrum* seperti pada Gambar 1, yaitu sebagai berikut:

1. *Scrum Events*

   Scrum Events merepresentasikan serangkaian kegiatan dalam proses pengembangan sebuah produk atau aplikasi (selanjutnya disebut dengan *Sprint*). *Sprint* memiliki waktu yang terbatas selama satu bulan atau kurang dan tidak ada jeda antara satu kegiatan dengan kegiatan lain. *Scrum Events* terdiri dari:

   a. *Sprint Planning*

      Sprint Planning adalah kegiatan yang dibatasi selama 8 jam atau kurang untuk memulai *Sprint*. Tujuan utama dari kegiatan ini adalah untuk menentukan prioritas pekerjaan dari *Product Backlog* yang harus diselesaikan terlebih dahulu dalam satu *Sprint*.

   b. *Daily Scrum*

      Daily Scrum adalah kegiatan yang dilaksanakan terjadwal setiap hari selama 15 menit. Dalam kegiatan ini pengembang akan memaparkan pekerjaan yang akan dilakukan pada hari tersebut dan melaporkan kemanjuan dari pekerjaan pada hari sebelumnya.

   c. *Sprint Review*

      Sprint Review adalah kegiatan yang dibatasi selama 4 jam atau kurang untuk meninjau hasil pekerjaan (*Increment*).

   d. *Sprint Retrospective*

      Sprint Retrospective adalah kegiatan yang dibatasi selama 3 jam atau kurang untuk mengakhiri proses *Sprint*. Dalam kagiatan ini, seluruh pihak yang terlibat akan meninjau *Sprint* sebelumnya dan memberikan saran serta masukan untuk *Sprint* berikutnya.

2. *Scrum Artifacts*

   Scrum Artifacts merepresentasikan pekerjaan yang dikerjakan dalam proses pengembangan sebuah produk atau aplikasi. *Scrum Artifacts* terdiri dari:

   a. *Product Backlog*

      Product Backlog adalah daftar terurut dari pekerjaan yang akan diselesaikan, dipelihara, atau ditingkatkan.

   b. *Sprint Backlog*

      Sprint Backlog adalah daftar pekerjaan yang telah dipilih dalam *Sprint Planning* untuk dikerjakan selama satu *Sprint*.

   c. *Increment*

      Increment adalah hasil akhir yang dicapai dalam satu Sprint. Dalam hal pengembangan aplikasi, *Increment* dapat berupa sebuah fitur.

C. *Flutter Software Development Kit*

Flutter adalah sebuah *Software Development Kit* (SDK) atau *UI toolkit* yang dirancang dan dikembangkan oleh *Google* untuk mengembangkan aplikasi *multi- platform* yang dapat berupa aplikasi *mobile*, *website*, dan *desktop*. Aplikasi yang dikembangkan dengan *Flutter* hanya membutuhkan satu basis kode yaitu *Dart* sebagai bahasa pemrogramannya [5]. Pengembangan aplikasi *mobile* secara umum termasuk dengan menggunakan *Flutter* terdiri dari dua proses yaitu pembuatan *User Interface* (UI) dan menghubungkan *User Interface* (UI) dengan data baik dari internet maupun basis data lokal.

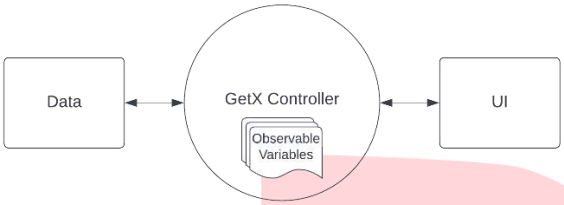
1. *State Management*

   Untuk mengelola perukaran data tersebut, *Flutter* menggunakan *State Management* yaitu sebuah metode untuk mengatur bagaimana *User Interface* (UI) aplikasi bereaksi terhadap perubahan data. Selain itu, *State Management* juga berfungsi untuk mengatur kode sumber agar terciptanya *Separation of Concern* (SoC) antara logika bisnis aplikasi dan tampilan aplikasi [5]. Dalam

Tugas Akhir ini, penulis menggunakan jenis *State Management* pihak ketiga yaitu *GetX*.

2. *GetX Reactive State Management*

GetX merupakan pendekatan *State Management* sederhana yang ringan, memudahkan pengelolaan *dependency injection*, dan perpindahan (*routing*) antar halaman pada aplikasi baik dengan membawa data maupun tidak.



GAMBAR 2
CARA KERJA *GETX*

Cara kerja dari *GetX* sebagai sebuah *State Management* sangat sederhana yaitu dengan memanfaatkan *GetX Controller* sebagai pusat dari logika bisnis aplikasi untuk mengolah data sebelum diteruskan ke halam *User Interface* (UI) aplikasi. *GetX* adalah jenis *State Management* yang reaktif. Pada Gambar 2 di dalam *controller* terdapat variabel yang bersifat *observable* dimana nilai dari variabel akan berubah sejalan dengan perubahan pada data yang masuk ke *controller*. Dengan pendekatan *State Management* yang reaktif, aplikasi yang dihasilkan dapat interaktif secara langsung (*real-time*) dan tetap ringan [6].

D. Teknologi *WebRTC*

WebRTC merupakan sebuah proyek sumber terbuka (*Open-Source*) yang dikelola oleh perusahaan-perusahaan teknologi seperti *Apple*, *Google*, *Microsoft*, *Mozilla*, dan lainnya. *WebRTC* adalah layanan yang memungkinkan pengembang perangkat lunak untuk menyediakan fitur komunikasi langsung seperti komunikasi video, suara, dan data umum lainnya [7]. Layanan video conference seperti *Zoom*, *Google Meet*, dan lain sebagainya dibangun di atas teknologi *WebRTC*.

1. *VideoSDK*

VideoSDK adalah sebuah layanan pihak ketiga yang menyediakan *Software Development Kit* (SDK) yang berjalan dengan teknologi *WebRTC* untuk membuat fitur *video conference* dan *live streaming*. *VideoSDK* menyediakan template fitur yang dapat digunakan langsung (*Prebuilt Integration*) dan pilihan fitur yang dapat dikustomisasi mengikuti spesifikasi dan desain dari aplikasi (*Custom Integration*) [8]. Dalam Tugas Akhir ini, penulis menggunakan layanan pihak ketiga *VideoSDK* untuk membuat fitur *Live Mentorship*.

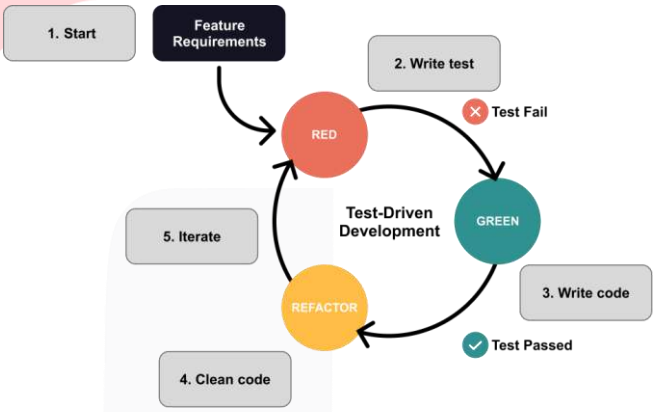
E. *Figma*

Figma adalah sebuah perangkat lunak berbasis *web* dan *desktop* yang digunakan untuk merancang desain antarmuka aplikasi mulai dari *wireframe*, *high- fidelity design*, sampai prototipe. Selain itu, *Figma* juga digunakan secara kolaboratif dengan pengembang untuk berbagi desain aplikasi yang sudah siap untuk dikembangkan ke tahap prouksi [9].

F. *Test-Driven Development*

Test-Driven Development (TDD) merupakan sebuah pendekatan yang digunakan dalam proses pengembangan sebuah perangkat lunak atau aplikasi dimana pengembang diharuskan untuk melakukan pengujian terlebih dahulu sebelum mengembangkan fungsionalitas dari aplikasi tersebut. *Test-Driven Development* merupakan bagian dari *Extreme Programming*. Dimana *Extreme Programming* adalah sebuah metode *agile* untuk pengembangan perngkat lunak berorientasi objek dengan iterasi yang cepat [10].

Dalam *Test-Driven Development*, pengembang akan terlebih dahulu menulis skenario pengujian *unit* (*unit testing*). *Unit* adalah bagian terkecil dari perangkat lunak yang dapat diuji (*testable*). Pada pemrograman berorientasi objek, *unit* umumnya dapat berupa *class*, *method*, atau *procedure*.



GAMBAR 3
DIAGRAM *TEST-DRIVEN DEVELOPMENT*

Alur kerja dari *Test-Driven Development* bersifat iteratif dimulai dengan penulisan skenario pengujian dan menjalankan pengujian. Pada tahap ini, pengujian akan gagal karena kode *unit* belum ditulis. Kemudian proses dilanjutkan dengan menulis kode *unit* seminimal mungkin dengan tujuan untuk membuat kode pengujian sebelumnya berhasil. Setelah itu, kode *unit* dan kode pengujian akan dioptimasi (*refactor*) untuk meningkatkan kualitas dan keterbacaan kode tanpa mengubah fungsionalitas atau perilakunya [10]. Proses *Test-Driven Development* diakhiri dengan melakukan pengujian kembali untuk memastikan kode *unit* sudah sesuai dengan apa yang diharapkan dalam skenario pengujian. Proses tersebut dikenal dengan istilah *Red-Green-Refactor*.

Tujuan mendahului pengujian (*testing*) sebelum menulis kode *unit* adalah untuk mendapatkan alur pengembangan yang rapi dan terukur serta menghasilkan aplikasi yang berjalan dengan baik dan mudah dipelihara.

G. *Behaviour-Driven Development*

Metode *Behavior-Driven Development* (BDD) pertama kali diperkenalkan oleh *Dan North* yang terinspirasi oleh *Kent Beck's*, yaitu orang yang pertama kali memperkenalkan metode *Test-Driven Development* (TDD). Metode *Behavior-*

Driven Development secara eksplisit dituliskan dalam tiga sintaks sederhana yaitu: (a) *Given some initial context*, (b) *When an event occurs*, dan (c) *Then ensure some outcomes* [9]. Dalam prakteknya, metode ini berfokus kepada tingkah laku sebuah sistem. Pihak-pihak yang terlibat dalam proses pengembangan dengan metode ini adalah pengembang (*developers*), QA (*Quality Assurance*), dan pengguna (*customers*). Pelanggan juga terlibat karena metode ini fokus kepada memahami persyaratan sistem (*requirements*) dan proses pengujian dilakukan dengan Bahasa Inggris sederhana sehingga mudah dipahami oleh orang-orang non-teknis [11]. Hal ini menjadikan metode *Behavior-Driven Development* cocok diimplementasikan pada sebuah perusahaan.

H. Kunci Pembeda Antara TDD dan BDD

Baik *Test-Driven Development* (TDD) maupun *Behavioral-Driven Development* (BDD) merupakan bagian dari kerangka kerja *Agile Scrum*. Berdasarkan kunci pembeda pada tabel di atas, kedua metode memiliki kelebihan dan kekurangan masing-masing ketika diimplementasikan pada sebuah proyek pengembangan perangkat lunak atau aplikasi [12]. Namun, *Test-Driven Development* dalam implementasinya sangat sesuai ketika hanya ada pengembang (*developers*) yang terlibat.

TABEL 1
PERBEDAAN TDD DAN BDD

Parameter	TDD	BDD
Definisi	TDD adalah sebuah metode pengembangan yang lebih fokus kepada implementasi sebuah fitur	BDD adalah sebuah metode pengembangan yang lebih fokus kepada tingkah laku sebuah sistem
Partisipan	*Developer*	*Developer, Customer, QAs*
Bahasa	Menggunakan bahasa yang sama dengan bahasa yang digunakan pada pengembangan dan implementasi fitur (contohnya: *Dart*, *Java*, dan lain sebagainya)	Menggunakan Bahasa Inggris sederhana
Fokus Utama	Pengujian *unit* (*unit test*)	Memahami *requirements*

I. Pengujian *Alpha*

Pengujian *Alpha* merupakan jenis pengujian yang dilakukan pada lingkungan pengembangan. Dalam pengujian ini, hanya tim internal yang terlibat. Pengujian *Alpha* dilakukan dengan tujuan untuk mengetahui kesalahan pada sebuah aplikasi lebih dini sehingga dapat dengan cepat diperbaiki sebelum aplikasi berada di tangan pengguna [13].

J. Pengujian *Beta*

Pengujian *Beta* merupakan jenis pengujian yang dilakukan secara terbuka pada lingkungan yang sebenarnya dengan keterlibatan orang lain yang berpotensi menggunakan aplikasi tersebut [13].

1. Pengujian *Usability*

Pengujian *Usability* pada sebuah perangkat lunak atau aplikasi dilakukan untuk mengukur dan mengetahui apakah aplikasi yang dikembangkan dapat digunakan oleh pengguna dengan mudah dan sesuai dengan target serta kebutuhan pengguna akan aplikasi tersebut. Menurut Neilsen [14], terdapat lima aspek yang harus dimiliki oleh sebuah perangkat lunak atau aplikasi agar dikatakan *usable*, antara lain sebagai berikut:

a. Efisiensi (*Efficiency*)
   Perangkat lunak atau aplikasi dikatakan efisien ketika pengguna dapat dengan cepat menyelesaikan tujuan mereka dalam aplikasi.
b. Kepuasan (*Satisfaction*)
   Perangkat lunak atau aplikasi dikatakan memberikan kepuasan bagi pengguna ketika pengguna dapat menggunakan aplikasi dengan nyaman dan memilih aplikasi tersebut tetap ada diperangkat mereka untuk digunakan lagi.
c. Kemampuan untuk dipelajari (*Learnability*)
   Perangkat lunak atau aplikasi dapat dipelajari dengan mudah pada saat pengguna pertama kali menggunakan aplikasi tersebut.
d. Kemampuan untuk diingat (*Memorability*)
   Pengguna dapat mengingat fungsi-fungsi dari aplikasi ketika membuka dan menggunakan aplikasi kembali.
e. Kesalahan (*Error*)
   Aplikasi memiliki tingkat kesalahan yang kecil dan penanganan kesalahan yang baik. Jadi ketika pengguna mendapati kesalahan, aplikasi dapat mengarahkan pengguna untuk menanggulangi kesalahan tersebut.

Pengukuran tingkat *usability* sebuah perangkat lunak atau aplikasi dapat dilakukan dengan menggunakan Skala *Likert* [14]. Dengan Skala *Likert*, pengguna dapat dengan mudah menentukan nilai dari *usability* aplikasi saat pengujian.

$$Skor = T\ x\ Pn \quad (1)$$

$$Interval = \frac{100}{Jumlah\ Skor\ Likert} \quad (2)$$

$$Index = \frac{\sum Skor}{Y}\ x\ 100 \quad (3)$$

Keterangan:
T = Jumlah responden yang memilih
Pn = Pilihan angka skor *Likert*
Y = $Skor\ tertinggi\ Likert\ x\ Jumlah\ responden$
Nilai dari interval digunakan untuk mendapatkan interpretasi persen seperti pada Tabel 2. Tabel ini akan digunakan untuk menentukan hasil akhir (*Index*).

TABEL 2
INTERPRETASI SKOR BERDASARKAN INTERVAL

Jawaban	Keterangan
Angka 0% - 19,99%	Tidak Mudah Sekali
Angka 20% - 39,99%	Tidak Mudah
Angka 40% - 59,99%	Cukup Mudah
Angka 60% - 79,99%	Mudah
Angka 80% - 100%	Mudah Sekali

2. Limitasi *Usability Testing* pada Aplikasi *Mobile*
   Melakukan pengujian *usability* pada perangkat lunak berupa aplikasi *mobile* memiliki tantangan dan limitasi tersendiri. Menurut Dongsong Zhang dan Boonlit Adipat [15], ada beberapa hal yang menjadi limitasi antara lain:
   a. Konteks *Mobile*
      Ketika menggunakan aplikasi *mobile*, pengguna tidak diam di satu tempat saja. Fokus pengguna dapat terdistraksi dengan lingkungan sekitar.
   b. Konektivitas
      Ketika koneksi perangkat *mobile* melambat maka akan berdampak pada penurunan pengalaman penggunaan aplikasi yang membutuhkan untuk terkoneksi ke internet.
   c. Ukuran Layar Kecil
      Karena perangkat *mobile* digunakan dengan alasan praktis, tentu informasi yang dapat ditampilkan juga terbatas.
   d. Perbedaan Resolusi Layar
      Resolusi layar yang berbeda antara satu perangkat dengan perangkat yang lain dapat mempengaruhi tingkat *usability* contohnya pada saat menampilkan sebuah gambar.
   e. Kemampuan Pemrosesan dan Daya yang Terbatas
      Perangkat *mobile* memiliki kemampuan pemrosesan yang berbeda-beda dan terbatas. Begitu pula dengan daya yang dimiliki. Hal ini akan membuat adanya limitasi aplikasi yang dapat dijalankan pada perangkat tersebut.
   f. Metode Entri Data
      Cara untuk melakukan entri data pada perangkat *mobile* sangat berbeda dengan perangkat komputer *desktop* dan memerlukan kemahiran tertentu.

K. Uji Validitas

Uji validitas bertujuan untuk mengetahui sejauh mana pengukuran yang dilakukan sudah tepat untuk mengukur apa yang hendak diukur. Nilai koefisien validitas suatu pengukuran berada berkisar antara -1,00 sampai +1,00. Semakin tinggi nilai koefisien yang dihasilkan, dalam hal ini mendekati +1,00 maka instrumen pengukuran semakin valid [16]. Rumus yang digunakan dalam uji validitas adalah teknik korelasi *Product Moment* [16] seperti berikut:

$$r_{xy} = \frac{n(\sum x_i y_i) - (\sum x_i)(\sum y_i)}{\sqrt{(n(\sum x_i^2) - (x_i)^2)(n(\sum y_i^2) - (y_i)^2)}} \quad (4)$$

Keterangan:
r_{xy} = Koefisien korelasi
n = Jumlah responden
x_i = Skor setiap item per responden
y_i = Skor total per responden

L. Uji Reliabilitas

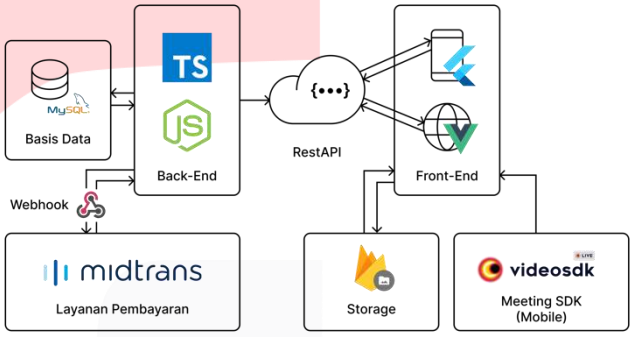
Uji reliabilitas dilakukan untuk mengetahui sejauh mana suatu pengukuran dapat dipercaya karena keajegannya. Terdapat beberapa metode yang dapat dilakukan untuk menguji reliabilitas antar lain *test-retest*, ekuivalen, dan *internal consistency*. Dalam metode *internal consistency* terdapat beberapa teknik diantaranya uji *split half*, KR 20, KR21, dan *Alpha Cronbach* [16]. Pada Tugas Akhir ini penulis menggunakan metode uji reliabilitas yaitu internal consistency dengan teknik *Alpha Cronbach* yang dapat digunakan untuk mengukur instrumen berbentuk kuesioner. Adapun rumus dari *Alpha Cronbach* adalah sebagai berikut:

$$r_i = \frac{k}{(k-1)}\left\{1 - \frac{\sum S_i^2}{S_t^2}\right\} \quad (5)$$

Keterangan:
r_i = Koefisien reliabilitas *Alpha Cronbach*
k = Jumlah item soal
$\sum S_i^2$ = Jumlah varian skor tiap item
S_t^2 = Varians total

III. METODE

A. Arsitektur Sistem



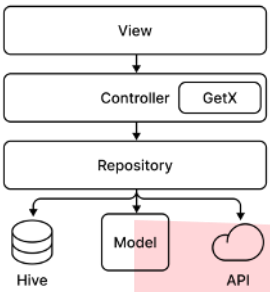
GAMBAR 4
ARSITEKTUR SISTEM

Arsitektur dari sistem yang dikembangkan secara keseluruhan mulai dari basis data sampai ke aplikasi yang berinteraksi dengan pengguna adalah sebagai berikut:

1. Basis Data (*Database*)
   Basis data yang digunakan dalam sistem adalah *Structured Query Language* (SQL) dengan bantuan *MySQL* sebagai *database management system* (DBMS) untuk mengelola seluruh basis data.
2. *Back-End*
   Seluruh aliran data yang keluar dan masuk ke basis data diatur pada sisi *back-end* sebelum data tersebut dikonsumsi oleh *front-end* baik itu aplikasi *mobile* mapupun *website*. Data-data tersebut akan didistribusikan dalam bentuk *JavaScript Object Notation* (JSON) melalui *Application Programming Interface* (API) sehingga dapat digunakan dengan mudah oleh aplikasi.
3. *Payment Gateway*
   Payment gateway merupakan layanan untuk melakukan transaksi atau pembayaran sebuah produk. Pada sistem ini, *payment gateway* yang digunakan adalah *Midtrans*.
4. *Front-End* Aplikasi dan *Website*
   Pada sisi *front-end*, terdapat aplikasi mobile yang dikembangkan dengan *Flutter SDK*, dan *website* yang dikembangkan dengan *framework Vue Js*.
5. *Firebase Storage*
   Layanan *Firebase* khususnya *Storage* digunakan untuk penyimpanan berkas berbentuk gambar.
6. *VideoSDK*
   Layanan *VideoSDK* digunakan untuk memenuhi kebutuhan *video conference* pada fitur *Live Mentorship*.

B. *Design Pattern*

Pengembangan perangkat lunak atau aplikasi ini menggunakan pola desain (*design pattern*) *Model-View-Controller* (MVC) yang telah disesuaikan dengan kebutuhan pengembangan dan *Software Development Kit* (SDK) yang digunakan, dalam hal ini *Flutter*. Berikut adalah diagram dari *design pattern* tersebut.



GAMBAR 5
DIAGRAM *DESIGN PATTERN* APLIKASI

Tanda panah dalam diagram menunjukkan *dependency* (kebergantungan) dari sebuah komponen terhadap komponen lain. Penjelasan lebih lengkap mengenai masing-masing komponen pada *design pattern* di atas adalah sebagai berikut:

1. *Repository*

   Repository adalah komponen yang berada pada lapisan paling dalam dan bersifat abstrak terhadap lapisan di atasnya. *Repository* berfungsi sebagai tempat untuk berkomunikasi dengan *Application Programming Interface* (API) atau *server* baik itu menerima data maupun mengirim data. Dalam prosesnya, *repository* juga memiliki hubungan atau bergantung dengan beberapa komponen yaitu:

   a. *Model*

      Model merupakan sebuah objek yang merepresentasikan data. Dalam hal ini data yang dimaksud adalah data yang diperoleh dari *server*.

   b. *Hive*

      Hive merupakan sebuah plugin pihak ketiga untuk menangani kebutuhan basis data lokal yang digunakan untuk menyimpan informasi sederhana.

2. *Controller*

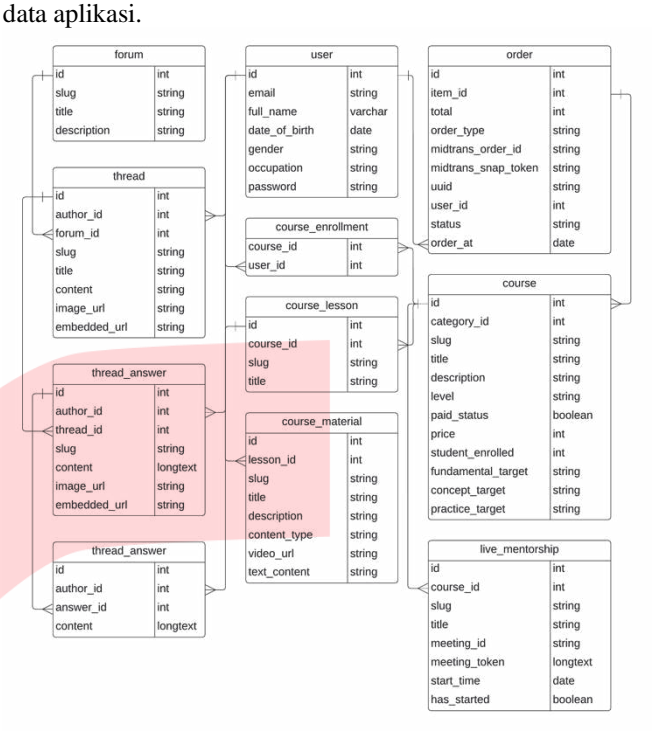
   Controller merupakan komponen yang berfungsi menghubungkan *repository* yang berisi data dengan *view* atau tampilan *User Interface* (UI) aplikasi yang berinteraksi dengan pengguna. Dalam arsitektur ini controller menerapkan *State Management* pihak ketiga untuk *Flutter* yaitu *GetX*.

3. *View*

   View merupakan komponen *User Interface* (UI) dari aplikasi yang terletak pada lapisan terluar. Pada prosesnya, *view* akan berkomunikasi dengan *controller* untuk mendapatkan maupun mengirimkan data.

C. Desain Basis Data

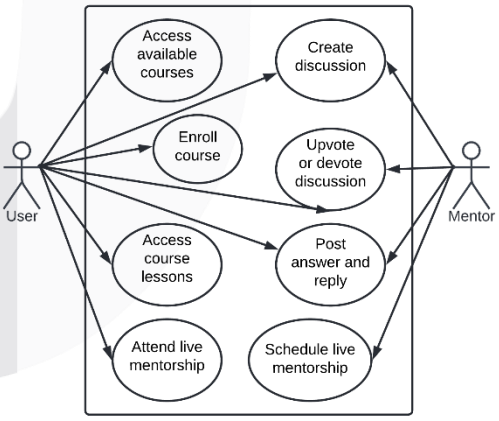
Berikut adalah desain basis data atau Entity-Relationship Diagram yang dimiliki aplikasi. Diagram pada Gambar 3.3 menunjukkan hubungan antar entitas atau tabel dalam basis data aplikasi.



GAMBAR 6
DIAGRAM *ENTITY-RELATIONSHIP*

D. Diagram *Use Case*

Berikut adalah *use case diagram* yang menggambarkan fitur yang dapat diakses oleh pengguna dengan *role user* dan *mentor*.



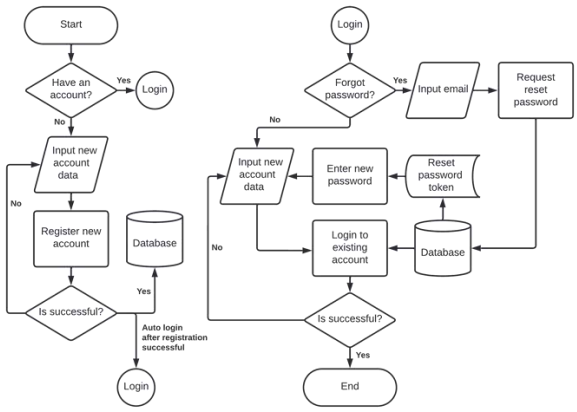
GAMBAR 7
DIAGRAM *USE CASE*

E. Diagram *Flowchart*

Berikut adalah diagram flowchart yang menggambarkan alur kerja dari masing-masing fitur yang ada pada aplikasi yang dikembangkan.

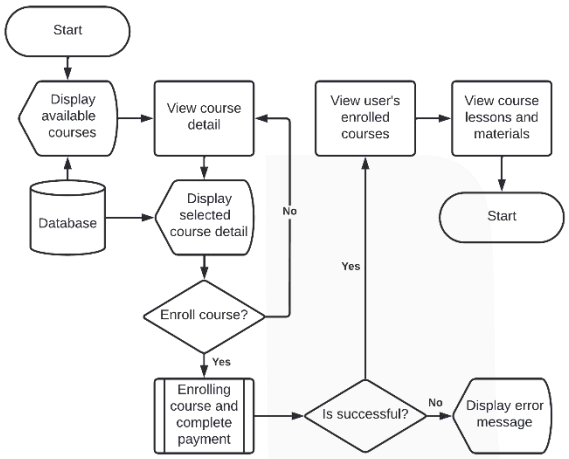
1. Autentikasi

   Fitur autentikasi pada aplikasi yang dikembangkan terdiri dari *login*, *register*, dan *forgot password* dengan alur seperti berikut.



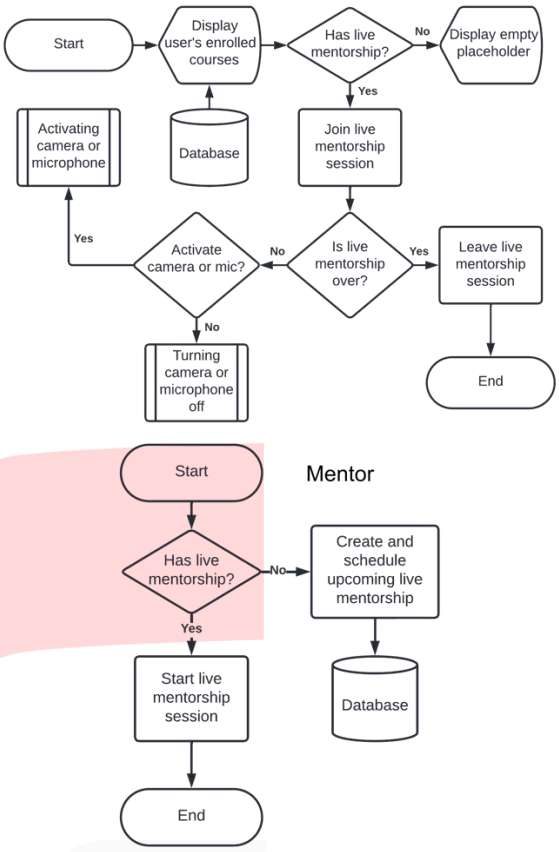
GAMBAR 8
FLOWCHART FITUR AUTENTIKASI

2. Kursus
Fitur kursus pada aplikasi yang dikembangkan terdiri dari daftar kursus, detail kursus, pembelian dan pembayaran kursus, dan *Learning Management System* (LMS) sederhana untuk belajar pada sebuah kursus dengan alur seperti berikut.



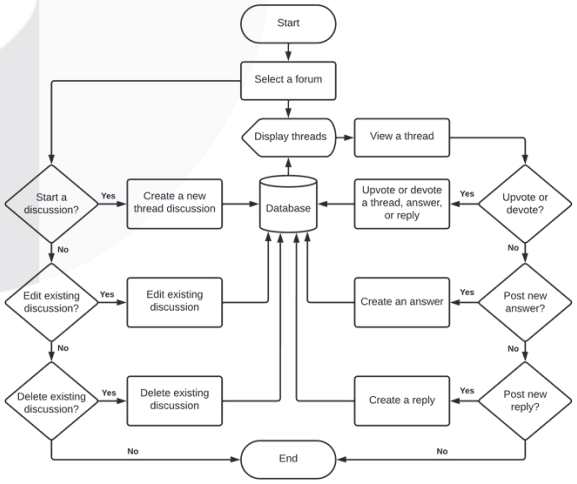
GAMBAR 9
FLOWCHART FITUR KURSUS

3. *Live Mentorship*
Fitur live mentorship yang dikembangkan terdiri dari menjadwalkan *live mentorship* pada sisi pengguna dengan *role* mentor, dan bergabung ke *live mentorship* pada sisi pengguna selain *role* mentor.



GAMBAR 10
FLOWCHART FITUR *LIVE MENTORSHIP*

4. Forum Diskusi
Fitur forum pada aplikasi yang dikembangkan terdiri dari daftar forum, *thread* diskusi dalam sebuah forum, memberikan jawaban pada *thread* diskusi, memberikan komentar pada jawaban dalam *thread* diskusi, membuat *thread* diskusi, mengubah *thread* diskusi, dan menghapus *thread* diskusi.



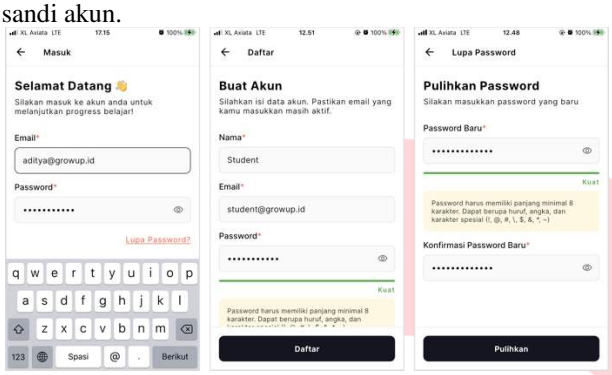
GAMBAR 11
FLOWCHART FITUR FORUM DISKUSI

F. Antarmuka Pengguna

Berikut adalah penjelasan dari masing-masing fitur yang ada pada aplikasi yang dikembangkan serta desain dari *User Interface* (UI) aplikasi.

1. Autentikasi

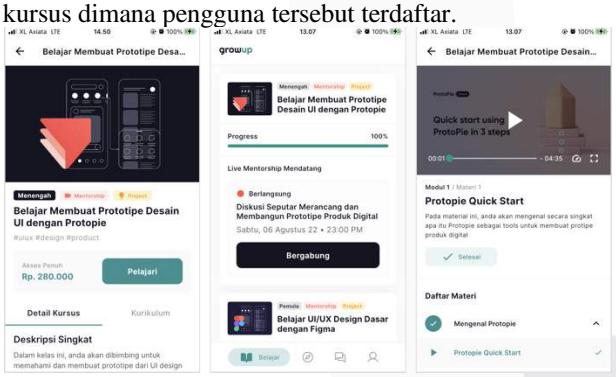
Merupakan fitur umum yang dimiliki aplikasi yaitu untuk masuk ke aplikasi dengan akun yang telah terdaftar, membuat dan mendaftarkan akun baru untuk mengakses fitur aplikasi, maupun memperoleh bantuan ketika mengalami kendala seperti pengguna lupa dengan kata sandi akun.



GAMBAR 12
DESAIN FITUR AUTENTIKASI

2. Kursus

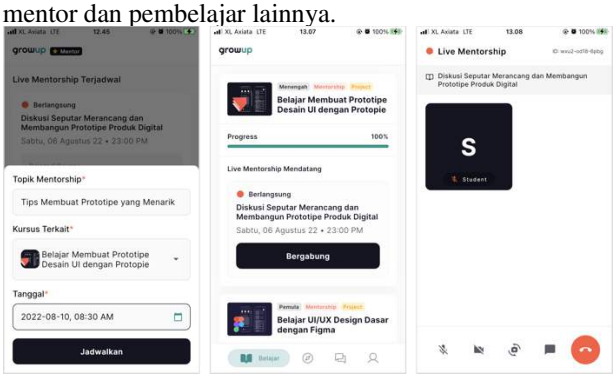
Merupakan fitur dimana pengguna dapat mengakses berbagai kursus online tentang keterampilan digital, melihat kursus yang tersedia, melihat detail kursus dan kurikulum, membeli kursus, serta mengakses materi pada kursus dimana pengguna tersebut terdaftar.



GAMBAR 13
DESAIN FITUR KURSUS

3. *Live Mentorship*

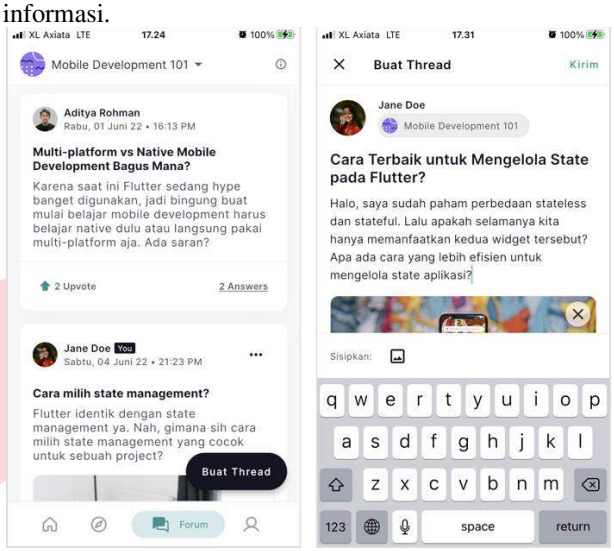
Merupakan fitur sebagai pendukung dari fitur kursus dimana mentor dapat mengadakan sesi live mentorship dan pengguna dapat berdiskusi secara langsung dengan mentor dan pembelajar lainnya.



GAMBAR 14
DESAIN FITUR *LIVE MENTORSHIP*

4. Forum Diskusi

Merupakan fitur sebagai pendukung dari fitur kursus dimana pengguna dalam hal ini pembelajar dapat dengan mudah berdiskusi dengan pembelajar lainnya serta mentor mengenai materi pada kursus tententu atau berbagi informasi.

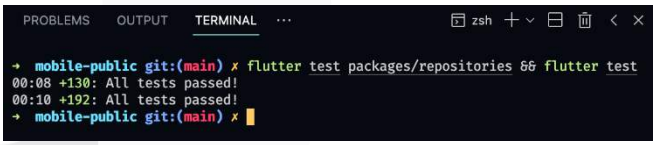


GAMBAR 15
DESAIN FITUR FORUM DISKUSI

IV. HASIL DAN PEMBAHASAN

A. Pengujian *Alpha*

Pengujian Alpha adalah pengujian yang dilakukan pada lingkungan pengembangan dengan hanya tim internal dan pengembang saja yang terlibat dalam pengujian. Pengujian *Alpha* pada Tugas Akhir ini dilakukan dengan pengujian *unit* (*unit testing*).



GAMBAR 16
HASIL *RUN UNIT TEST*

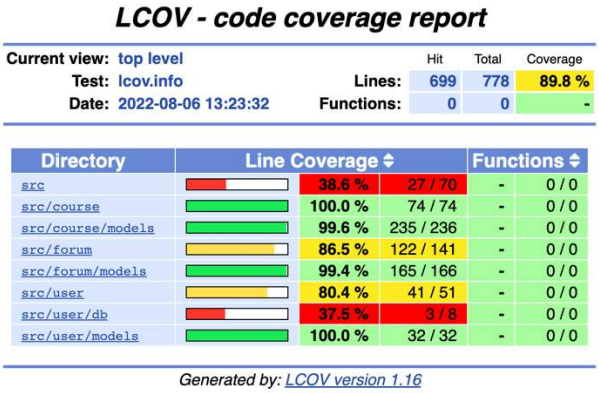
Berikut adalah hasil pengujian unit dari repository dan controller untuk seluruh fitur yang terdapat dalam aplikasi:

1. Modul *Repository*

TABEL 3
HASIL *UNIT TEST* PADA *REPOSITORY*

No.	Fitur	Jumlah *Test-Case*	Hasil
1	Autentikasi	26	100%
2	Kursus	54	100%
3	Forum Diskusi	50	100%

Dari total *test-case* sebanyak 130 pada modul *repository*, berikut adalah hasil analisa *test coverage* dengan menggunakan *LCOV*:



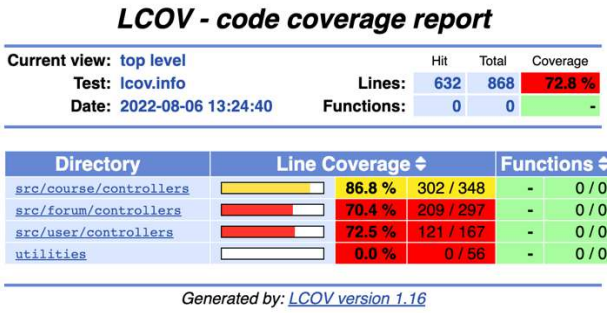
GAMBAR 17
HASIL *TEST COVERAGE* MODUL *REPOSITORY*

2. Modul *Controller*

TABEL 4
HASIL *UNIT TEST* PADA *CONTROLLER*

No.	Fitur	Jumlah *Test-Case*	Hasil
1	Autentikasi	26	100%
2	Kursus	54	100%
3	Forum Diskusi	50	100%

Dari total *test-case* sebanyak 192 pada modul *controller*, berikut adalah hasil analisa *test coverage* dengan menggunakan *LCOV*:



GAMBAR 18
HASIL *TEST COVERAGE* MODUL *CONTROLLER*

B. Pengujian *Beta*

Pengujian *Beta* merupakan pengujian yang dilakukan secara terbuka pada lingkungan sebenarnya dimana pengguna yang berpotensi menggunakan aplikasi dapat terlibat dalam proses pengujian. Pengujian *Beta* dilakukan dengan metode *Usability Testing*.

TABEL 5
KUESIONER PENGUJIAN *USABILITY*

No.	Pertanyaan Kuesioner
1	Bagaimana pengalaman menggunakan fitur register untuk membuat akun baru?
2	Bagaimana pengalaman menggunakan fitur login untuk masuk ke akun yang terdaftar?
3	Bagaimana pengalaman menelusuri dan mencari sebuah kursus pada aplikasi?
4	Bagaimana pengalaman melakukan pembelian sebuah kursus pada aplikasi?
5	Bagaimana pengalaman mencoba mengakses materi pada kursus yang telah dibeli?
6	Bagaimana pengalaman menggunakan fitur forum diskusi?
7	Bagaimana pengalaman menggunakan fitur live mentorship?

TABEL 6
SKOR *LIKERT*

Label	Nilai
Tidak Mudah Sekali	1
Tidak Mudah	2
Cukup Mudah	3
Mudah	4
Sangat Mudah	5

Setiap skenario pengujian mewakili aspek dari usability sebuah aplikasi. Perhitungan nilai tiap skenario menggunakan rumus (1), (2), dan (3). Tabel 7 di bawah ini menyajikan rekap hasil dari pengujian usability yang telah dilakukan.

TABEL 7
REKAP HASIL *USABILITY TESTING*

Pertanyaan	EF	SF	LR	MR	ER	Nilai
1	v	v	v	v	v	94,19%
2	v	v	v	v	v	94,83%
3	v	v	v	v	v	96,12%
4	v	v	v	v	v	94,19%
5	v	v	v	v	v	94,83%
6	v	v	v	v	v	92,90%
7	v	v	v	v	v	94,19%

Berdasarkan hasil pengujian *usability* pada tabel-tabel di atas. Diperoleh nilai *index* terendah yaitu 92,90% dan nilai *index* tertinggi yaitu 96,12%. Jika dilihat pada Tabel 7, nilai *index* dari masing-masing fitur yang diuji berada para interval 80% - 100%. Hal ini menunjukkan bahwa aplikasi yang dikembangkan sudah memenuhi kriteria usability yaitu *efficiency* (EF), *satisfaction* (SF), *learnability* (LR), *memorability* (MR), dan *error* (ER).

C. Uji Validitas

Uji validitas dilakukan untuk mengukur valid atau tidaknya pertanyaan-pertanyaan yang digunakan dalam kuesioner pada pengujian *beta*. Tingkat validitas dapat diketahui dengan menggunakan teknik korelasi *Product Moment* pada rumus (4) dengan membandingkan hasil r hitung dengan r tabel (taraf signifikan 5%). Pertanyaan kuesioner dikatakan valid apabila nilai r hitung lebih besar dari pada r tabel. Pada Tugas Akhir ini, jumlah responden pada pengujian beta sebanyak 31 responden.

TABEL 8
HASIL UJI VALIDITAS

No.	Nilai r hitung	Nilai r tabel	Kesimpulan
1	0.5648	0.355	Valid
2	0.5471	0.355	Valid
3	0.5871	0.355	Valid
4	0.7087	0.355	Valid
5	0.5707	0.355	Valid
6	0.7454	0.355	Valid
7	0.6289	0.355	Valid

Berdasarkan Tabel 8, dapat disimpulkan bahwa kuesioner belum sepenuhnya valid karena ada dua buah pertanyaan yang memiliki nilai r hitung lebih kecil dari pada nilai r tabel.

D. Uji Reliabilitas

Uji reliabilitas dilakukan untuk mengukur konsistensi pertanyaan yang diajukan dalam kuesioner ketika

pengukurannya dilakukan secara berulang-ulang. Tingkat konsistensi dapat diketahui dengan menggunakan Alpha Cronbach pada rumus (5). Pada pengujian ini terdapat 7 butir pertanyaan (nilai k = 7).

TABEL 9
HASIL UJI RELIABILITAS

Jumlah varian item	Jumlah varian total	Nilai r	Nilai r tabel	Kesimpulan
1.7655	5.129	0.765	0.355	Reliabel

Berdasarkan Tabel 9, diperoleh nilai reliabilitas yaitu 0.765 dan nilai r tabel adalah 0.355. Karena nilai reliabilitas yang diperoleh lebih dari nilai r tabel, maka kuesioner yang digunakan masuk dalam kriteria reliabel.

E. Analisa

Pengujian *unit* yang termasuk ke dalam siklus pengembangan dengan metode *Test-Driven Development* dilakukan pada dua modul dalam aplikasi yang meliputi *repository* dan *controller*. Pengujian modul *repository* dilakukan untuk memastikan koneksi antara aplikasi dan *server* berjalan dengan baik, sedangkan pada modul *controller* dilakukan untuk memastikan *controller* bekerja dengan baik sebagai pusat logika bisnis aplikasi dan penghubung antara antarmuka pengguna dengan data pada *repository*. Hasil pengujian *unit* didapatkan 100% lolos.

Untuk memvalidasi pengaruh hasil pengujian *unit* terhadap tingkat kualitas aplikasi, maka dilakukan pengujian usabilitas kepada 31 pengguna. Didapatkan rata-rata nilai indeks pengujian usabilitas di atas 90%. Nilai indeks pengujian usabilitas terkecil terdapat pada fitur forum diskusi yaitu sebesar 92,90%. Hal ini dikarenakan pada proses pengujian beberapa pengguna, terdapat penundaan selama beberapa detik (*delay*) saat memberi *upvote*, maupun membuat komentar pada sebuah diskusi. Hal tersebut dapat ditingkatkan dengan mengaplikasikan penggunaan *webhook* pada server agar proses pertukaran data dapat berjalan secara langsung (*real-time*).

V. KESIMPULAN

Berdasarkan hasil dari pengujian dan analisa yang telah dilakukan pada penelitian Tugas Akhir ini, maka dapat diperoleh kesimpulan sebagai berikut:

1. Aplikasi yang dikembangkan sebagai media pembelajaran keterampilan digital dengan dukungan mentor dan komunitas memiliki tiga fitur yang saling berkaitan untuk menunjang proses pembelajaran yaitu kursus sebagai bahan utama pembelajaran, forum diskusi sebagai ruang bertanya dan berdiskusi baik dengan mentor maupun pembelajar lain, dan *live mentorship* sebagai tempat dilakukan kegiatan mentoring antara mentor dengan pembelajar. Berdasarkan hasil pengujian *alpha* diperoleh hasil 100%. Pada pengujian *beta* yaitu mengukur aspek *usability* dari aplikasi yang mencakup *efficiency*, *satisfaction*, *learnability*, *memorability*, dan *error*, diperoleh hasil rata-rata di atas 90%.
2. Berdasarkan hasil pengujian usability yang telah dilakukan diperoleh rata-rata hasil di atas 90%. Hal ini menunjukkan bahwa pengimplementasian metode pengembangan *Test-Driven Development* (TDD) pada pengembangan perangkat lunak baik untuk dilakukan dan berdampak besar pada hasil akhir aplikasi dan kelayakan aplikasi untuk digunakan.

REFERENSI

[1] C. A. Wright dan S. D. Wright, "The Role of Mentors in the Career Development of Young Professionals," *National Council on Family Relations,* vol. 36, no. 2, pp. 204-208, 1987.

[2] D. M. Hunt dan C. Michael, "Mentorship: A Career Training and Development Tool," *The Academy of Management Review,* vol. 8, no. 3, pp. 475-485, 1983.

[3] N. B. Ruparelia, "Software Development Lifecycle Models," *ACM SIGSOFT Software Engineering Notes,* vol. 35, no. 3, p. 8, 2010.

[4] K. Schwaber and J. Sutherland, The Scrum Guide, 2020.

[5] V. Guzzi, K. D. Moore, V. Ngo dan M. Katz, Flutter Apprentice (Second Edition): Learn to Build Cross-Platform Apps, McGaheysville: Razeware LLC, 2021.

[6] GetX, "About Get," GetX, [Online]. Available: https://pub.dev/packages/get#about-get. [Diakses 14 07 2022].

[7] Google, "Real-time communication for the web," Google, [Online]. Available: https://webrtc.org/. [Diakses 14 07 2022].

[8] "Live Audio & Video Calling SDK," Video SDK, [Online]. Available: https://www.videosdk.live/audio-video-calling-api-sdk. [Diakses 15 07 2022].

[9] Wikipedia, "Figma (software)," Wikipedia, 04 07 2022. [Online]. Available: https://en.wikipedia.org/wiki/Figma_(software). [Diakses 11 07 2022].

[10] D. Janzen dan H. Saiedian, "Test-Driven Development: Concepts, Taxonomy, and Future Direction," *IEEE Computer Society,* vol. 38, no. 9, pp. 43-50, 2005.

[11] M. Rahman dan J. Gao, "A Reusable Automated Acceptance Testing Architecture for Microservices in Behavior-Driven Development," dalam *IEEE Symposium on Service-Oriented System Engineering*, San Francisco, CA, USA, 2015.

[12] J. Unadkat, "BDD vs TDD vs ATDD : Key Differences," BrowserStack, 4 Mei 2021. [Online]. Available: https://www.browserstack.com/guide/tdd-vs-bdd-vs-atdd. [Diakses 15 Juni 2022].

[13] I. K. WAIROOY, "Alpha dan Beta Testing," School of Computer Science, BINUS University, [Online]. Available: https://socs.binus.ac.id/2020/06/30/alpha-dan-beta-testing/. [Diakses 15 07 2022].

[14] D. R. Rahadi, "Pengukuran Usability Sistem Menggunakan Use Questionnaire Pada Aplikasi Android," *Jurnal Sistem Informasi (JSI),* vol. 6, no. 1, pp. 661-671, 2014.

[15] R. Harrison, D. Flood dan D. Duce, "Usability of mobile applications: literature review and rationale for

a new usability model," *Harrison et al. Journal of Interaction Science,* vol. 1, no. 1, pp. 1-16, 2013.

[16] F. Yusup, "UJI VALIDITAS DAN RELIABILITAS INSTRUMEN PENELITIAN KUANTITATIF," *Jurnal Tarbiyah: Jurnal Ilmiah Kependidikan,* vol. 7, no. 1, pp. 17-23, 2018.

[17] T. Winters, T. Manshreck dan H. Wright, Software Engineering at Google: Lessons Learned from Programming Over Time, 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc., 2020.