Pengembangan Website License Management dengan Metode Test Driven Development

Indra Taftazani Wisnuaji
Program Studi Informatika
Fakultas Teknologi Industri
Universitas Islam Indonesia
Jl. Kaliurang, Sleman, Yogyakarta
18523286@students.uii.ac.id

Irving Vitra Paputungan
Program Studi Informatika
Fakultas Teknologi Industri
Universitas Islam Indonesia
Jl. Kaliurang, Sleman, Yogyakarta
irving@uii.ac.id

***Abstract*—Ozeva mengembangkan sebuah aplikasi manajemen yang diberi nama Whizeva *Application Client* yang di dalamnya terdapat beberapa aplikasi berbeda yang dapat digunakan setelah membeli lisensi per aplikasi tersebut. Dikarenakan jangka waktu berlangganan yang dapat berbeda diperlukan pengelolaan lisensi yang dapat mempermudah proses pembelian serta monitoring status lisensinya. Guna mendukung proses tersebut, dikembangkanlah solusi cepat berupa aplikasi berbasis *website* yang dapat digunakan segera setelah proses pengembangan selesai dilakukan. Karena merupakan solusi cepat, maka waktu pengembangan tidak terlalu lama, sehingga faktor yang harus diperhatikan dari pengembangan ini adalah waktu, struktur sistem, bebas dari *error*, *maintainable* dan mudah digunakan. Pengembangan *website* menerapkan *Test Driven Development* sebagai metode pengembangan dan *MVC* sebagai pola arsitekturnya yang dinilai cocok dengan penggunaan *website* dimana terdapat 3 jenis pengguna berbeda yang mendukung pengembangannya menjadi modular. Dengan menjadi modular membuat pengembangan dan *maintainability* dari *website* mengalami peningkatan. Proses pengembangan *website* ini terdiri dari beberapa tahapan yaitu pembuatan *test case*, mengerjakan *working code*, *refactor code*, dan diakhiri dengan pengujian akhir berupa *black box testing* yang terdiri dari *functionality* dan *non functionality testing*. Hasil akhir dari pengembangan ini adalah sebuah aplikasi berbasis *website* yang digunakan untuk mengelola dan memantau lisensi aplikasi yang dimiliki penggunanya.**

***Keywords*—*Test Driven Development*, *Black Box Testing*, *Testing*, *MVC*, Modular, *TDD*.**

I. PENDAHULUAN

Ozeva Technology merupakan perusahaan penyedia solusi IT yang berspesialisasi dalam pengembangan perangkat lunak, konsultasi dan pelatihan IT. Sebagai perusahaan teknologi, Ozeva mengembangkan perangkat lunak yang dibutuhkan bisnis dan industri untuk memaksimalkan keuntungan dan meningkatkan efisiensi mereka dengan memberikan solusi yang tepat [1]. Ozeva sendiri telah mengeluarkan beberapa aplikasi yang dapat digunakan setelah dilakukannya pembelian lisensi aplikasinya. Seiring berjalannya waktu pengguna aplikasi Ozeva mengalami peningkatan, dan tidak sedikit pengguna yang membeli lisensi aplikasi lebih dari satu dengan durasi lisensi yang berbeda, serta keadaan pandemi yang menyulitkan penggunanya datang ke lokasi untuk melakukan pembelian lisensi. Hal ini mendorong Ozeva untuk mengembangkan sistem yang dapat membantu penggunanya dalam mengelola lisensi mereka. Selain itu, dengan keadaan pandemi Ozeva melihat beberapa peluang bisnis dimana banyak orang membutuhkan pekerjaan sampingan yang dapat dilakukan secara online. Berangkat dari hal itu, sistem menambahkan 1 jenis pengguna yaitu *reseller*, dimana *reseller* dapat menawarkan dan menjadi mitra dalam melakukan penjualan lisensi aplikasi. Dengan adanya *reseller*, diharapkan dapat meningkatkan penjualan dengan terjadinya promosi dari mulut ke mulut.

Oleh karena itu, dibutuhkan *website* sederhana yang dapat dirilis segera setelah pengembangan selesai dengan persentase *error* dan *bug* se-sedikit mungkin sebagai wadah kegiatan tersebut berjalan. Maka dari itu, *website* dikembangkan dengan menerapkan metode *Test Driven Development* atau yang biasa disingkat sebagai *TDD* dan konsep *MVC* atau *model – view – controller*, serta menggunakan *black box testing* untuk melakukan pengujian akhir atau evaluasi. Terdapat beberapa pertimbangan mengapa *TDD*, *MVC*, dan *black box testing* digunakan.

Dengan menerapkan konsep *MVC*, pengembang dapat mengembangkan sistemnya secara modular, dengan masing-masing *role* yang ditetapkan dianggap sebagai satu modul. Penggunaan konsep ini juga digunakan untuk mengurangi perubahan *code* lain saat salah satu *code* ditambahkan atau dilakukan perubahan.

Dengan menerapkan metode *Test Driven Development*, pengembang memfokuskan pengembangan kepada 1 fitur, tidak berpindah fokus hingga semua *unit* yang berhubungan dengan fitur tersebut selesai / lulus dalam *unit test*. Dengan perulangan seperti ini, kualitas *code* akan terus mengalami peningkatan, membuat *error* dan *bug* mudah ditemukan, serta menjadikan *code* yang ditulis lebih *reusable*. Karena berfokus kepada satu fitur per siklus, pengembangan akan secara tidak langsung menjadi modular, sehingga penerapan *TDD* dan *MVC* memiliki sinergi yang baik dikarenakan keduanya membuat pengembangan sistem berbentuk modular.

Dengan menjadikan pengembangan menjadi modular, membuat *code* menjadi lebih mudah dibaca, lebih mudah dilakukan pengujian dan lebih mudah dilakukan *refactoring*. Tiap modul pada modular memiliki fungsi yang spesifik yang memperkecil tempat munculnya *bug* maupun *error*, serta meningkatkan *reuseability* karena kita dapat memanggil fungsi dari satu sumber yang ada di modul apapun.

Dengan mempertimbangkan hal – hal diatas, tim pengembang memutuskan menggunakan metode *Test Driven Development* sebagai metode pengembangan yang didukung dengan konsep *Model – View – Controller* sebagai pola arsitekturnya, karena *TDD* dan *MVC* memiliki sinergi yang baik dan memungkinkan pengembang mengidentifikasi dan memperbaiki *error* dan *bug* sejak dini.

Pada praktiknya pengembangan *website* metode *TDD*, mengurangi kemungkinan *bug* pada *output* yang dihasilkan akan tetapi ada kemungkinan ditemukannya kasus *error* dan *bug*, oleh karena penting dilakukannya pengujian akhir atau evaluasi. Evaluasi dilakukan menggunakan metode *black box testing*.

Black box testing perlu dilakukan untuk memastikan apakah setiap fungsionalitas dari fitur bekerja sesuai spesifikasi yang diinginkan [2]. Selain itu karena pengembangan berdasarkan *TDD* melakukan pengujian dari sudut pandang pengembang, maka diperlukan pendekatan pengujian dari perspektif pengguna *website*, *black box testing* melakukan pengujian kepada keseluruhan sistem berbeda dengan *TDD* yang fokus pengujiannya dilakukan per *unit* dan modul. Sehingga *black box testing* sangat cocok dilakukan sebagai evaluasi akhir dimana pengujiannya mengisi hal – hal yang tidak dicakup pada saat pengujian di *TDD*.

Fokus pada makalah ini adalah penerapan metode *Test Driven Development* dalam melakukan pengembangan *website* dan pengaruh metode tersebut untuk memperkecil kemungkinan terjadi *error* atau *bug* yang ditemukan pada saat evaluasi akhir, yang juga menjadi patokan apakah *website* dapat langsung dirilis atau tidak, serta pengaruh eksternal seperti pandemi terhadap pengembangan *website*.

II. KAJIAN PUSTAKA

Ditengah pandemi seperti yang kita alami sekarang banyak individu yang tidak dapat atau lebih memilih melakukan sesuatu secara daring daripada bertemu langsung, sehingga terdapat masalah yang mengharuskan dikembangkannya *website license management* yang dapat digunakan dalam jangka panjang / *maintainability* yang tinggi. Karena masalah ini menjadi salah satu masalah yang harus segera terselesaikan, biaya yang tidak besar serta waktu pengembangan juga tidak terlalu panjang sehingga penerapan prinsip *agile* sangatlah membantu dalam pengembangannya, waktu pengembangan yang tidak terlalu lama dapat menyebabkan banyaknya terjadi *error* dan *bug* serta, adanya *code* yang tidak terstruktur dengan baik, sehingga pengembangan menitik beratkan kepada waktu pengerjaan dan pengujian *error*.

Terdapat beberapa metode yang menjadi pertimbangan untuk menjadi metode dasar pengembangan *website* ini diantaranya : *continuous integration*, *scrum*, *extreme programming*, *acceptance test driven development*, dan *test driven development* dimana *test driven development* yang terpilih menjadi metode pengembangan yang disebabkan oleh beberapa perbandingan antara metode yang ada.

Metode *continuous integration* merupakan metode dimana pengembang menambahkan perubahan-perubahan kecil pada *code* dan diikuti dengan *automated testing* secara terus menerus yang memungkinkan *bug* diperbaiki dengan sangat cepat, yang secara tidak langsung membantu dalam *continuous deployment*. Akan tetapi *continuous integration* pada jangka panjang memerlukan biaya yang sangat besar sehingga *maintainability* dari metode ini cukup rendah, dikarenakan penggunaan sejumlah *pipeline* terus menerus [3]. Hal ini menjadi alasan utama tidak digunakannya *CI* karena *maintainability* dari metode ini cukup sulit untuk dipertahankan dalam jangka panjang.

Metode *scrum* merupakan metode yang sangat populer digunakan karena dapat melakukan pengerjaan dengan tempo yang cepat dan progres proyek dan kinerja masing – masing individu yang terlihat jelas dengan adanya *daily meeting* [4], akan tetapi belajar dari proyek – proyek sebelumnya, terdapat kemungkinan adanya kesulitan melakukan *daily meeting*, dikarenakan terdapat beberapa masalah terkait koneksi masing-masing anggota tim, serta pada pengembangan kali ini, anggotanya berasal dari 2 departemen berbeda sehingga terdapat beberapa anggota yang mengalami kesulitan dalam melaksanakan *daily meeting* secara daring karena terdapat beberapa jadwal tim satu dan lainnya.

Metode *extreme programming* merupakan metode pengembangan yang memungkinkan tim kecil dan menengah sistem yang berkualitas tinggi dan memiliki *adaptability* yang tinggi terhadap perubahan kebutuhan perangkat [5]. *Adaptability* ini diperoleh dari *feedback* yang diberikan terus – menerus dan diberikannya kemampuan kepada pengembang untuk menambahkan atau mengurangi spesifikasi jika terdapat perubahan kebutuhan. Tetapi pada penerapannya *extreme programming* memerlukan peran pengguna atau calon pengguna sejak awal pengembangan sistem, sehingga diperlukannya perwakilan calon pengguna yang harus ada pada proses pengembangan sistem, hal ini menjadi alasan utama mengapa *extreme programming* tidak cocok digunakan pada pengembangan kali ini, dikarenakan pandemi yang terjadi.

Metode *acceptance test driven development* atau *ATDD* merupakan metode yang mirip dengan penerapan *test driven development*, yang membedakan *ATDD* dari metode *TDD* adalah, pada *ATDD* pengujian berfokus kepada *acceptance test* bukan berfokus kepada *unit testing*, sehingga pada praktiknya pengujiannya lebih menitik beratkan kepada *behavior interface* atau dengan kata lain, *ATDD* melakukan pengujian dengan perspektif *pengguna* bukan dari perspektif pengembang sedangkan pada *TDD* fokus pengujian menitik beratkan kepada *unit testing* [6], akan tetapi dengan anggota tim yang berasal dari departemen *web developer* dan *QA*, pengujian dapat dilakukan dari 2 perspektif dengan menggabungkan *TDD* dan *black box testing*, sehingga penggunaan *TDD* dinilai lebih baik dengan pertimbangan anggota tim yang ikut serta dalam pengembangan proyeknya, serta dengan sifat *TDD* yang modular membuat *maintainability* dari sistem lebih baik. *Maintainability* menjadi alasan utama terpilihnya metode pengembangan dengan mengetahui bahwa semua metode yang memegang prinsip *agile* memiliki waktu pengembangan yang relatif cepat dan menggunakan banyak pengujian, hal ini dimiliki oleh sebagian besar metode *agile*, akan tetapi terdapat faktor lain yaitu komposisi anggota tim dan keadaan pandemi yang dialami menjadi salah satu faktor yang harus dipertimbangkan dalam pemilihan metode yang diterapkan, faktor – faktor tersebut menjadi alasan utama mengapa penggunaan *TDD* terpilih sebagai metode pengembangan *website*.

Untuk mendukung perkembangan sistem dengan metode *TDD*, diperlukan *code* yang terstruktur dengan baik, struktur modular sangat berperan besar dalam kesuksesan pengembangan dengan metode *TDD*, maka

penerapan *MVC* sangat membantu, dimana *MVC* merupakan sebuah pola arsitektur yang membagi aplikasi ke dalam 3 komponen yaitu *model*, *view* dan *controller* [7], yang setiap bagiannya memiliki peran yang berbeda – beda. *Model* merupakan bagian dari sistem yang bertanggung jawab atas semua tugas yang berhubungan dengan data. *View* bertanggung jawab atas semua elemen yang berhubungan dengan *user interface* baik itu tombol, *form* dan lainnya. *Controller* bertanggung jawab untuk menangani semua *event* yang terjadi saat *website* berjalan, baik *event* yang terjadi karena interaksi dengan pengguna aplikasi maupun yang terjadi karena proses *website* itu sendiri.

Pada *TDD* pengembang melakukan *unit testing* untuk tiap fungsionalitas yang ada. *Unit testing* merupakan salah satu jenis pengujian *software* dengan cara menguji tiap *unit* atau modul [8]. *Unit testing* buat diawal sebagai acuan penulisan *code*, serta sebagai alat uji *code* yang baru dituliskan ataupun *code* yang telah dilakukan *refactoring* sebelumnya.

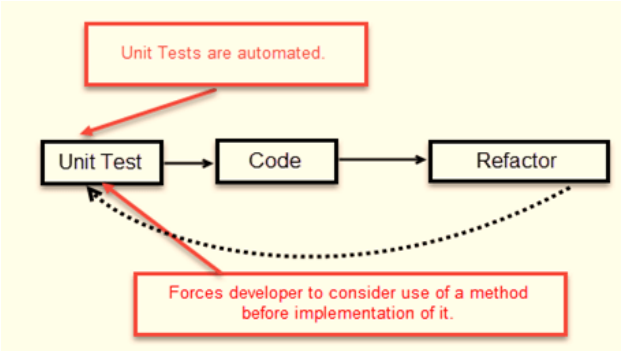
Selain waktu, struktur, dan bebas dari *error* atau *bug*, tentunya salah satu hal terpenting yaitu apakah *website* mudah untuk digunakan atau tidak, akan tetapi pada pengembangan *TDD* semua pengujian dilakukan berdasarkan perspektif pengembang, sehingga sangatlah penting untuk dilakukan *black box testing* karena *black box testing* lebih berpusat kepada perspektif pengguna. Pengujian ini dilakukan diluar siklus *TDD* dengan penerapan metode *boundary value analysis* atau *boundary testing* dimana penguji memasukkan data pada nilai batas data yang ada pada spesifikasi. Pengujian ini dilakukan untuk memastikan *error* atau *bug* yang mungkin terlewat pada saat proses pengembangan, terutama pada bagian interaksi di *user interface* seperti *error messages* dan lain sebagainya.

Penelitian lain telah mengkaji dampak dari penggunaan *TDD* sebagai metode dalam mengembangkan aplikasi, hasil dari penelitian tersebut mengindikasikan terjadinya pengurangan *bug* dan *error pre-release* sebanyak 40% [9].

Dengan penggunaan metode – metode tersebut sudah mencakup hal – hal penting yang menjadi fokus dalam pengembangan *website* ini yaitu pengembangan *website* yang singkat, terstruktur, *maintainable*, bebas dari *bug* sehingga dapat cepat dirilis setelah pengembangan selesai, dan yang paling penting adalah mudah digunakan oleh pengguna aplikasi.

III. METODOLOGI

Metodologi yang diterapkan dalam proses pengembangan sistem mengikuti siklus atau alur *Test Driven Development* setelah siklus selesai, pengerjaan dilanjutkan oleh tim penguji untuk dilakukan *black box testing* yang terdiri dari serangkaian skenario *functionality* dan *non-functionality testing*. Siklus *Test Driven Development* ditunjukkan pada gambar 1.



gambar 1 *Siklus* Test Driven Development [9]

A. *Pembuatan Unit Test*

Pembuatan *unit* test diawali dengan diskusi dengan keseluruhan anggota tim pengembang, dimana hasil dari diskusi tersebut berupa gambaran umum apa yang dapat dilakukan tiap komponen atau *unit* tersebut, yang biasa disebut sebagai *envisioning*. Berdasarkan hasil dari *envisioning*, pembagian tugas terkait pembuatan *unit test* serta spesifikasi tiap *unit* ditentukan. Kemudian *test case* dibuat dan diujikan yang akan menghasilkan *fail test* dikarenakan *code* yang belum dituliskan. Contoh *test case* tertera pada gambar 2.

```
/**
* @test
*/
public function customer_can_store()
{
    $current = Carbon::now();
    $expired = $current->addMonths(6);
    $price = DB::table('features')->where('feature_id',17)->value('feature_price');


    $user = User::where('role','=','customer')->first();
    $subs_id = subscription::max('subscription_id');
    $response = $this->actingAs($user)->post(route('subscriptions.store'),
[
            'user_id' => 15,
            'feature_id' => 10,
            'chosen_feature' => 10,
            'total_price' => $price,
            'expire_date' => $expired,
            'extra_device' => 1,
]);
    $response->assertResponseStatus(302);
}
```

gambar 2 *Unit Test Case Store Function*

B. *Menuliskan code*

Pada tahap ini, pengembang mulai menuliskan *code* untuk menjalankan *unit test* hingga pengujian dinyatakan berhasil, setelah itu pengembang melakukan diskusi kembali untuk memastikan apakah diperlukan *refactor* pada *code* yang telah dituliskan, contoh penulisan *code* yang dilakukan ditunjukkan pada gambar 3. Terdapat beberapa alasan *refactor* perlu dilakukan di antaranya :

a. Penulisan *code* yang terlalu rumit.

b. *Code* yang berulang atau terdapat *code* yang tidak diperlukan.

c. *Code* yang menyebabkan *runtime* melewati standar yang telah ditentukan.

```php
public function store(Request $request)
{
    $request -> validate([
        'subscription_id' => 'required',
        'user_id' => 'required',
        'chosen_feature' => 'required',
        'durasi' => 'required',
    ]);

    $current = Carbon::now();
    $expired = $current->addMonths($request->durasi);
    $history_id = DB::table('histories')->max('subscription_id');
    $price = DB::table('features')->where('feature_id',$request->chosen_feature)->value('feature_price');
    $extradev = $request->extra_device * 100000;
    if($request->durasi < 12){
        $total_price = $price * $request->durasi + $extradev * $request->durasi;
    }elseif ($request->durasi = 12) {
        $total_price = $price * 10 + $extradev * $request->durasi;
    }
    subscription::create([
        'subscription_id' => $request->subscription_id,
        'user_id' => $request->user_id,
        'feature_id' => $request->chosen_feature,
        'chosen_feature' => $request->chosen_feature,
        'total_price' => $total_price,
        'expire_date' => $expired,
        'extra_device' => $request->extra_device,
        'status' => 0,
    ]);

    return redirect()->route('customer');
}
```

gambar 3 *store function*

C. *Refactoring code*

Pada tahap *refactoring*, pengembang melakukan *refactor* sesuai dengan standar yang dipaparkan pada saat diskusi. Kemudian diskusi dilanjutkan kembali untuk menyatakan apakah *code* sudah sesuai standard dan dianggap selesai.

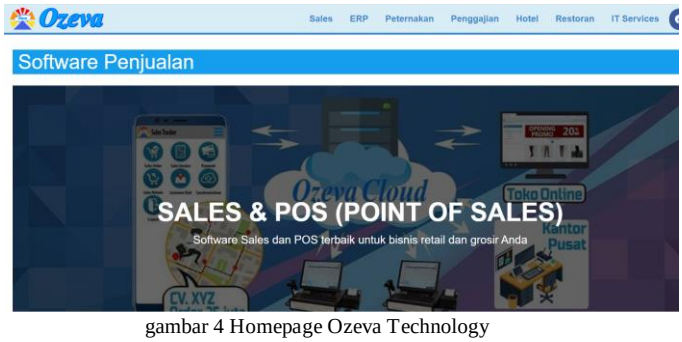
Melalui informasi diatas dapat disimpulkan bahwa alur atau siklus kerja dari pengembangan terbagi menjadi 2 yaitu siklus *TDD* yang kemudian dilanjutkan dengan evaluasi. Dimana *TDD* terdiri dari beberapa aktivitas yang dimulai dari pembuatan *test case* hingga *refactoring*, pada pengembangan kali ini satu siklus *TDD* dianggap selesai apabila telah menyelesaikan 1 modul.

IV. HASIL DAN PEMBAHASAN

A. Hasil

1. Homepage Ozeva technology

Sebelum melakukan login ke *license management*, pengguna berada di *website* milik ozeva dimana terdapat informasi lengkap terkait lisensi aplikasi serta fitur yang ada pada aplikasi yang akan digunakan yang ditunjukkan pada gambar 4 dan 5.

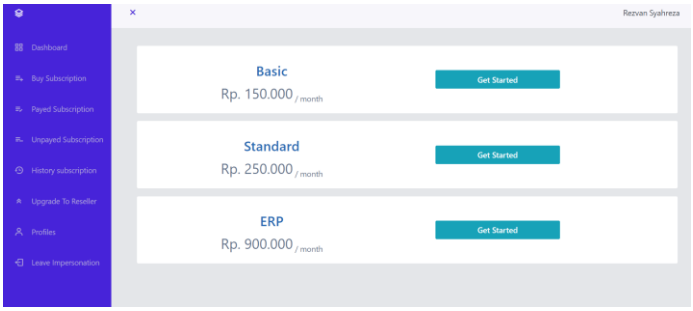


gambar 4 Homepage Ozeva Technology

gambar 5 Informasi Jenis Lisensi

2. Laman Pembelian lisensi

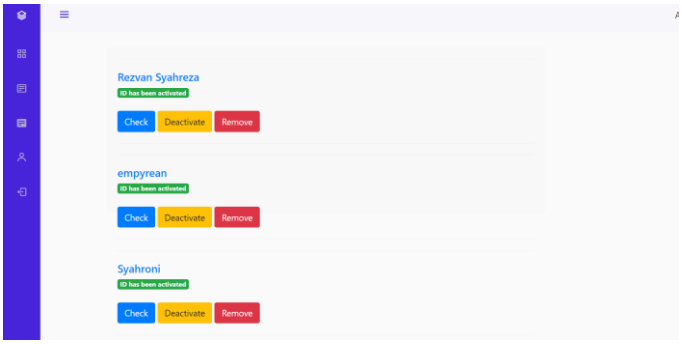
Pada saat pengguna melakukan pembelian lisensi, pengguna akan diarahkan untuk memilih license aplikasi yang akan dibeli serta jenis lisensinya yang terbagi menjadi beberapa kategori berdasarkan aplikasi yang dibeli, setelah selesai menentukan aplikasi dan jenis lisensinya, pengguna diarahkan ke laman pembelian dimana pengguna mengisi form untuk menentukan berapa lama pengguna ingin lisensi tersebut diaktifkan. Laman jenis lisensi ditunjukkan pada gambar 6.



gambar 6 *License Page*

3. *Customer ID Management*

Terdapat beberapa kondisi unik dimana pengguna (*customer*) dapat meminta bantuan kepada pengguna khusus (pengguna dengan *role Admin* serta *Reseller*) untuk melakukan pembelian lisensi untuk dirinya. Maka dari itu *Admin* dan *Reseller* mempunyai kemampuan untuk masuk kedalam akun *customer*. Maka dari itu dibuatlah laman login khusus (disebut juga *check-in ID*) yang ditunjukkan pada gambar 7.



gambar 7 *Customer ID Management*

B. Pembahasan

1. Siklus Pengembangan *TDD*

Pengembangan *license management* terbagi menjadi beberapa aktivitas yang berlangsung selama kurang lebih 3 bulan lamanya yang dijabarkan dalam tabel 1.

Tabel 1 Timeline Pengembangan *Website*

NO	Aktivitas	Durasi
1	*TDD* 1 pengembangan modul *customer* sampai dengan *refactoring code*	1 Bulan
2	*TDD* 2 pengembangan modul *admin* sampai dengan *refactoring code*	2 Pekan
3	*TDD* 3 pengembangan modul *reseller* sampai dengan *refactoring code*	1 Bulan
4	Pengujian akhir / *black box testing*	1 Pekan
5	Perbaikan *error* dan *bug* yang ditemukan	1 pekan

Pada *TDD* 1 terdapat beberapa kendala yang dirasakan yang disebabkan oleh masalah koneksi, sehingga terdapat beberapa informasi yang tidak jelas yang menyebabkan pengerjaannya melebihi batas waktu awal yang sudah ditentukan, hal ini mulai dapat ditangani dengan beberapa adaptasi yang bisa dilakukan, dimana tim memutuskan untuk menambahkan pertemuan offline di kantor pada hari-hari yang telah ditentukan, hal ini dapat dilakukan karena lokasi sekitar kantor masih belum mencapai zona merah. Masalah pada *TDD* 1 kembali terjadi pada siklus ke 3, karena sudah menjadi zona merah pengembangan dilakukan secara *online*, sebagai antisipasi masalah yang terjadi pada *TDD* 1, durasi tim melaksanakan pertemuan diperpanjang sehingga pemaparan informasi serta diskusi yang dilakukan lebih jelas.

Seperti informasi yang terlihat pada table 1, pelaksanaan siklus *Test Driven Development* terbagi menjadi 3 sesuai dengan modul pengguna.

Pada siklus *TDD* 1, pengembangan berfokus kepada keseluruhan fitur yang berhubungan dengan modul *customer* baik fitur umum (fitur yang digunakan oleh ke 3 modul) ataupun fitur khusus.

Fitur yang dihasilkan pada siklus *TDD* 1 ditunjukkan pada tabel 2.

Tabel 2 Siklus T*DD* 1

NO	Fitur	Penjelasan
1	*Login & Sign In*	Fitur umum untuk melakukan login dan mendaftarkan akun
2	*Reset password*	Fitur umum untuk melakukan pengubahan password
2	*Customer Dashboard*	Fitur ini merupakan halaman awal yang muncul setelah login sebagai *customer* yang menampilkan lisensi yang dimiliki serta sisa waktu sebelum masa berlaku lisensi habis
3	Pembelian dan pembaruan lisensi aplikasi	Fitur untuk melakukan pembelian lisensi baru dan pembaruan lisensi yang sudah ada
4	*Profile*	Fitur untuk menambahkan profil pengguna
5	*Upgrade to reseller*	Fitur untuk mengajukan permintaan ke admin untuk mengganti *role* menjadi *reseller*
5	Riwayat pembelian lisensi	Fitur untuk melihat riwayat pembelian maupun perpanjangan lisensi yang dimiliki

Pada siklus *TDD* 2, pengembangan berfokus kepada keseluruhan fitur yang berhubungan dengan modul *admin.* Fitur yang dihasilkan pada siklus *TDD* 2 ditunjukkan pada tabel 3.

Tabel 3 Siklus *TDD 2*

NO	Fitur	Penjelasan
1	*Approved Reseller*	Fitur untuk mengubah role *customer* menjadi reseller
2	*Admin Dashboard*	Fitur ini merupakan halaman awal yang muncul setelah login sebagai *admin* yang menampilkan keaktifan *reseller* yang terdaftar
3	*Deactivate reseller*	Fitur untuk menonaktifkan akun *reseller* yang sudah tidak aktif atau tidak memiliki progress
4	*Check-in as customer*	Fitur ini dimiliki oleh modul *admin* dan *reseller*, untuk masuk sebagai *customer* dan memproses pembelian sesuai permintaan dari *customer* yang memiliki akun tersebut (*reseller* hanya dapat melakukan check-in untuk customer yang di daftarkan oleh *reseller* tersebut)

Pada siklus *TDD* 3, pengembangan berfokus kepada keseluruhan fitur yang berhubungan dengan modul *reseller*. Fitur yang dihasilkan pada siklus *TDD* 3 ditunjukkan pada tabel 4.

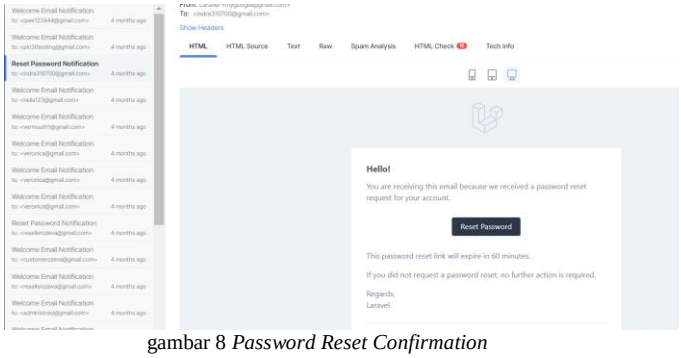
Tabel 4 Siklus *TDD* 3

NO	Fitur	Penjelasan
1	*Create customer account*	Fitur *reseller* untuk menambahkan *customer* baru
2	*Reseller Dashboard*	Fitur ini merupakan halaman awal yang muncul setelah login

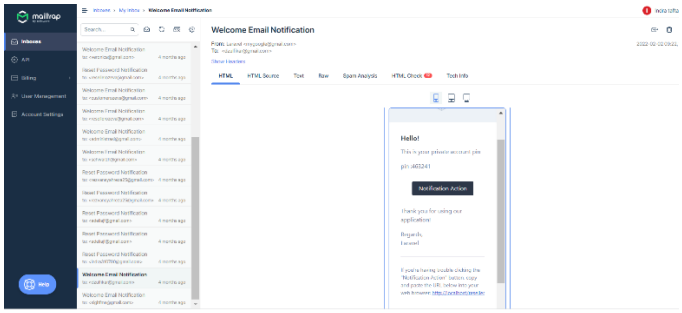
		sebagai *reseller* yang menampilkan *ranking reseller point*, bonus yang diterima *reseller*, dan *list customer* yang dimiliki oleh *reseller*

Setiap pengujian aplikasi dilakukan secara otomatis dengan menggunakan PHP *unit test* yang terintegrasi dengan *framework laravel*, tetapi terdapat beberapa fitur yang perlu dilakukan pengujian untuk kedua kalinya dikarenakan sangat berhubungan dengan *user interface* atau fitur yang berhubungan dengan pengiriman email. Diantaranya memastikan bahwa captcha muncul setelah 3 kali kesalahan login, peringatan yang muncul saat terjadi kesalahan, dan informasi yang disampaikan di dalam email yang dikirimkan.

Fitur yang berhubungan dengan email dilakukan pengujian dengan bantuan aplikasi *add-on* yang bernama mailtrap seperti yang ditunjukkan pada gambar 8 dan 9.



gambar 8 *Password Reset Confirmation*



gambar 9 *Account pin*

Terdapat beberapa adaptasi yang berbeda pada penerapan *Test Driven Development* dikarenakan anggota tim dan kondisi pandemi, seperti diskusi dan laporan yang biasanya dilakukan 1 minggu sebanyak 2 kali menjadi hanya 1 kali pertemuan dengan durasi lebih lama dikarenakan tim terdiri dari departemen yang berbeda, serta pertemuan sering terganggu karena beberapa kendala seperti koneksi membuat tim harus menentukan jadwal khusus untuk melakukan diskusi secara offline.

2. Pengujian Akhir / Evaluasi

Setelah dilakukannya evaluasi akhir dengan menerapkan *black box testing* untuk menguji fungsional dan non-fungsional sistem, semua scenario *test case* diuji oleh tester dan beberapa calon pengguna tanpa kesulitan yang cukup signifikan dikarenakan alur penggunaan aplikasi yang cukup sederhana membuat pengguna tidak terlalu sulit untuk menggunakannya, sistem dinyatakan layak untuk dilakukan *deployment* setelah dilakukan semua pengujian sesuai *test case* yang telah dibuat, dimana dari total 120 *test* ditemukan 6 *test case* dengan indikasi *error* atau sekitar 95%. Dengan tingkat fungsional yang lulus uji mencapai angka 95% menandakan bahwa *usability* sistem yang tinggi, serta alur penggunaan aplikasi yang simpel membuat pengguna tidak terlalu sulit untuk menggunakannya, walaupun terdapat beberapa *bug* yang terjadi diantaranya pada bagian *interface* terdapat tombol yang tertimpa atau hilang pada saat dibuka melalui perangkat mobile ataupun peringatan yang berbeda dari *error* yang terjadi. Berikut beberapa *test case* yang diujikan dalam *black box testing* dipaparkan pada tabel 5.

Tabel 5 *Test Case* Pembelian lisensi

NO	*Test Case*	Hasil yang diharapkan	Hasil yang diperoleh
1	Langsung menuju form pembelian tanpa memilih aplikasi atau jenis lisensinya	Laman akan ke redirect ke halaman yang sebelumnya terbuka	Berhasil
2	Tidak mengisi durasi lisensi	Sistem akan menampilkan peringatan “this field is required ”	Berhasil
3	Mengisi durasi lisensi dengan angka 0 atau karakter	Sistem akan menampilkan peringatan “this field is must be fill with valid number”	Sistem tetap menyimpan data pembelian lisensi (Gagal)
4	Langsung menuju form pembelian tanpa melakukan login terlebih dahulu	Laman akan ke redirect ke halaman login, dan sistem akan menampilkan peringatan “Pengguna diharuskan melakukan login terlebih dahulu”	Berhasil
5	Melakukan Pembelian menggunakan sistem *check-in customer* oleh *reseller*	Pembelian berhasil dan email akan terkirim ke email customer yang digunakan dimana didalamnya terdapat informasi lisensi yang dibeli, serta jumlah dan nomor tagihan.	Berhasil
6	Menekan tombol navbar dalam kondisi portrait dan landscape di pada perangkat mobile	Navbar terbuka dan tombol terlihat dengan jelas	Pada saat membuka dalam kondisi portrait terdapat

			beberapa kalimat di navbar yang terpotong (gagal)
7	Admin atau Reseller keluar dari mode Check-IN customer	Kembali ke dashboard Admin atau Reseller sesuai dengan role masing-masing	Sistem menyatakan bahwa id telah terlogout dari sistem (Gagal)
8	Kesalahan login sebanyak 3 kali	Sistem akan mengeluarkan captcha pada usaha login selanjutnya	Berhasil
9	Melakukan reset pin dengan memasukkan angka pin sebanyak 5 digit	Sistem akan mengeluarkan peringatan “pin must be fill with 6 valid number”	Berhasil
10	Melakukan *password reset*	Sistem akan mengirimkan email konfirmasi terkait permintaan *reset password* serta link *reset password*	Berhasil

V. KESIMPULAN

Setelah dilakukan pengembangan *website* hingga mencapai pengujian akhir dapat disimpulkan bahwa penggunaan metode *Test Driven Development* dapat meminimalisir adanya *bug* dan *error* yang dibuktikan dengan data dari evaluasi akhir yang menyatakan bahwa sekitar 95% fungsionalitas dinyatakan lulus pengujian. Meskipun terdapat beberapa adaptasi yang harus dilakukan karena pandemi yang terjadi, efek dari pandemi tidak terlalu berpengaruh dikarenakan dalam penerapan kolaborasi antara anggota tim. Anggota tim lain dapat melakukan perubahan kepada *code* anggota tim lain dengan aman karena *unit testing* yang dilakukan akan memberi informasi jika perubahan tersebut memberikan efek yang tidak diperkirakan sebelumnya.

Penggunaan *black box testing* untuk menguji dari perspektif pengguna berjalan dengan baik yang dibuktikan dengan tidak adanya kesulitan yang cukup signifikan pada saat pengujian ke tim tester maupun beberapa calon pengguna.

REFERENCES

[1] "Ozeva," Ozeva Technology, [Online]. Available: https://www.ozeva.com/in/about.

[2] S. Nidhra and J. Dondeti, "BLACK BOX AND WHITE BOX TESTING TECHNIQUES –A LITERATURE REVIEW," *International Journal of Embedded Systems and Applications,* vol. 2, 2012 .

[3] M. Shahin, M. A. Babar and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," 2017.

[4] G. Y. Koi-Akrofi, J. Koi-Akrofi and H. A. Matey, "UNDERSTANDING THE CHARACTERISTICS, BENEFITS AND CHALLENGES OF AGILE IT PROJECT MANAGEMENT: A LITERATURE BASED PERSPECTIVE," *International Journal of Software Engineering & Applications,* vol. 10, 2019.

[5] A. FRUHLING and G.-J. D. VREEDE, "Field Experiences with eXtreme Programming:Developing an Emergency Response System," *Journal of Management Information Systems,* vol. 22, 2006.

[6] Z. Khanam and M. N. Ahsan, "Evaluating the Effectiveness of Test Driven Development: Advantages and Pitfalls," *International Journal of Applied Engineering Research,* 2017.

[7] D.-P. Pop and A. Altar, "Designing an MVC Model for Rapid Web Application Development," in *24th DAAAM International Symposium on Intelligent Manufacturing and Automation 2013*, Zadar, 2013.

[8] W. E. Lewis, D. Doobs and G. Veerapillai, Software Testing and Continuous Quality Improvement, New York: Auerbach Publications, 2009.

[9] N. Nagappan, E. M. Maximilien, T. Bhat and L. Williams, "Realizing quality improvement through test driven," Springer Science + Business Media, 2008.

[10] T. Hamilton, "What is Test Driven Development (TDD)? Tutorial with Example," Guru99, 2022.