MAKALAH
METODOLOGI DESAIN PERANGKAT LUNAK
"TEST DRIVEN DEVELOPMENT"

KELOMPOK SQUAD

1	NASRULLAH KHOMAENI	5200411208
2	CLAUDIO ORLANDO DE ARAUJO	5200411214
3	RIDHO ICHVAN	5200411256

UNIVERSITAS TEKNOLOGI YOGYAKARTA
PROGRAM STUDI TEKNIK INFORMATIKA

TAHUN 2021/2022

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa. Atas rahmat dan hidayah-Nya, penulis bisa menyelesaikan makalah yang berjudul "Metode Agile."

Makalah ini berisi tentang pengetahuan akan Metode agile . Penulis menyadari ada kekurangan pada Makalah ini. Oleh sebab itu, saran dan kritik senantiasa diharapkan demi perbaikan karya penulis. Penulis juga berharap semoga karya ilmiah ini mampu memberikan pengetahuan tentang Test Driven Development Dalam Pengembangan Aplikasi Berbasis Web

.

DAFTAR ISI

BAB I
PENDAHULUAN

A. Latar Belakang

Pada saat ini perkembangan teknologi perangkat lunak telah berkembang pesat dan menjadi pendukung utama bagi sebuah perusahaan. Suatu perusahaan atau lembaga yang menempatkan teknologi perangkat lunak menjadi salah satu pendukung dalam kemajuan perusahaan dapat mencapai rencana strategis organisasi. Perangkat lunak (*software*) sendiri merupakan program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model desain, dan cara penggunaannya (*user manual*). Perangkat lunak pada saat ini sudah menjadi kebutuhan umum di setiap usaha karena dengan memanfaatkan teknologi perangkat lunak akan membantu suatu perusahaan untuk memecahkan sebuah permasalahan yang terjadi.

Test Driven Development merupakan proses pengembangan perangkat lunak yang bergantung pada pengulangan siklus pengembangan yang sangat singkat. Persyaratan untuk melakukan sebuah perubahan menjadi kasus uji yang sangat spesifik, maka software yang sedang dikembangkan diharuskan memenuhi test baru. Hal ini bertentangan dengan pengembangan perangkat lunak yang memungkinkan perangkat lunak untuk ditambahkan belum tentu memenuhi persyaratan. Pada proses *software development* ada beberapa proses yang pasti dilakukan, sehingga terbentuklah iterasi suatu langkah — langkah yang mungkin sudah lazim dilakukan.

Proses pengembangan perangkat lunak akan menghasilkan suatu produk yang berkualitas apabila terjadi sinkronisasi antar analisis dan pemograman. Sebuah perangkat lunak dikatakan berkualitas apabila perangkat lunak dapat bekerja sesuai apa yang diinginkan pengguna *(user requirement and bussines process).* Selain itu, dapat memudahkn pengguna dalam modifikasi perbuahan, serta memudahkan dalam menemukan bug. Semua spesifikasi dapat diakomodasi oleh metode TDD. Sebab pada metode TDD terdapat unit test yang berperan menjaga agar pengembang tetap stay on track Ketika membuat sebuah perangkat lunak. Sehingga memungkinkan hasil perangkat lunak yang dibuat akan sesuai dengan kebutuhan dan proses bisnis pengguna.

Dalam artikel ini, akan dijelaskan mengenai apa itu Test Driven Development, tujuan, tahapan, manfaat, kelebihan dan kekurangannya.

B. Rumusan Masalah

1. Apa itu Test Driven Development?
2. Apa tujuan Test Driven Development?
3. Apa tahapan-tahapan Test Driven Development?
4. Apa Manfaat Test Driven Development?
5. Apa saja Kelebihan dan kekurangan Test Driven Development?

C. Tujuan Penelitian

1. Mengetahuhi apa itu Test Driven Development
2. Mengetahuhi tujuan Test Driven Development
3. Mengetahuhi tahapan-tahapan Test Driven Development
4. Mengetahuhi manfaat Test Driven Development
5. Mengetahuhi apa saja Kelebihan dan kekurangan Test Driven Development

BAB II

PEMBAHASAN

A. Pengertian Test Driven Development

Pertama, kita harus mengenal lebih dahulu apa itu Test Driven Development. Dalam pembangunan software diawali dengan membuat test casenya terlebih dahulu, baru implementasi (coding). Bedakan dengan tahapan pembangunan software, pada umumnya di coding dulu baru test.

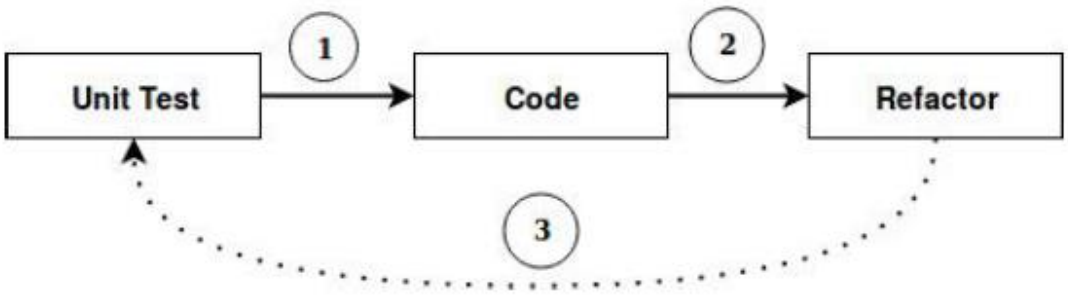
Test Driven Development adalah sebuah strategi pembangunan software yang menggunakan prinsip extreme programming oleh Kent Beck yang menggunakan agile method. Agile berarti membangun aplikasi dengan cara yang lebih efektif dan cerdas dan sebagai metode pengembangan di mana pengembang hanya menulis kode baru berdasarkan pengujian otomatis yang gagal. Test Driven Development dimulai dengan merancang dan mengembangkan pengujian otomatis yang dapat diulang untuk setiap fungsionalitas kecil sebuah aplikasi. Dalam metode Test Driven Development, pengujian pertama kali dikembangkan yang menentukan dan memvalidasi apa yang harus dilakukan kode.

Dalam proses pengembangan normal, pertama kita memprogram kode program dan kemudian mengujinya. Di Test Driven Development, kami mulai dengan pengujian yang kemudian salah karena sudah dilakukan sebelum pengembangan. Untuk lulus ujian, tim pengembangan harus mengembangkan kode dan terus mengembangkannya. Pengembangan yang digerakkan oleh tes adalah proses mengembangkan dan melakukan pengujian otomatis untuk pengembangan aplikasi yang sebenarnya.

B. Tujuan Test Driven Development

Tujuan Test-Driven Development adalah menghasilkan kode program yang lebih baik karena dirancang dari kode test yang telah ditulis sebelumnya. *Test-Driven Development* digunakan oleh *developer* untuk mengetahui kualitas kode *software* itu sendiri.

C. Tahapan-tahapan Test Driven Development



1. Pengembang menuliskan *unit test* terlebih dahulu, sesuai spesifikasi dan bisnis proses dari system yang diinginkn oleh pengguna.

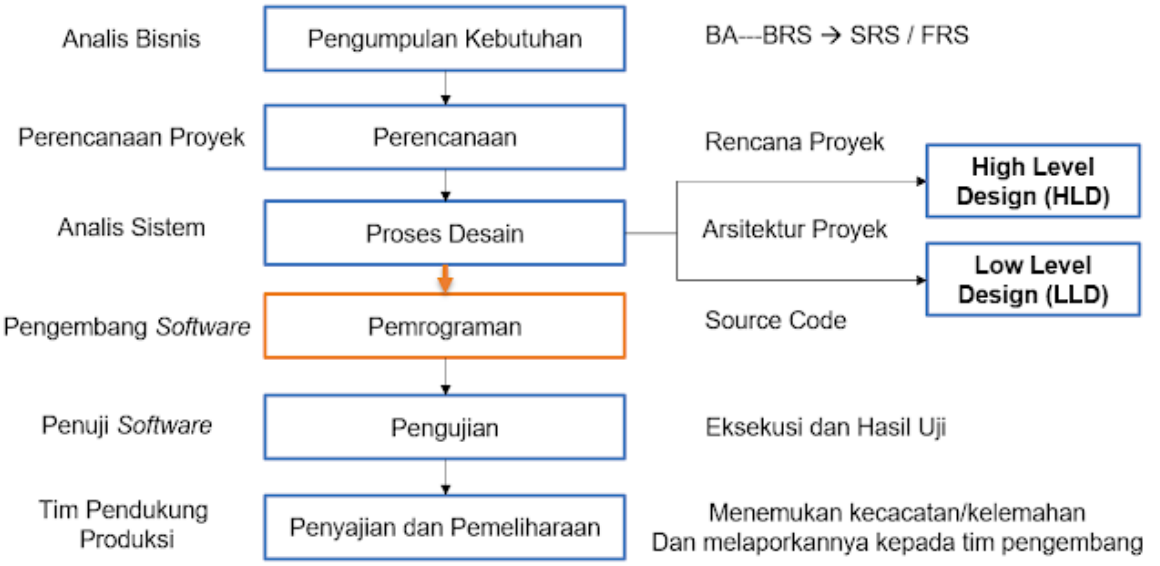
2. Setelah membuat *unit test*, pengembang menuliskan kode program untuk menyelesaikan *unit test* sampai *pass* (tidak terjadi error).
3. Pengembang melakukan refactoring kode program yaitu mengubah struktur program agar lebih mudah dipahami dan dimodifikasi, tanpa harus mengubah dari program.

TDD merupakan gabungan antara *Test First Development* dan *Refactoring.* Tahapan unit test menggunakan metode *Test First Development*(TFD), dimana alur kerjanya di gambar sebagai berikut :

Berdasarkan gambar, hal pertama yang dilakukan ketika menerapkan metode TFD adalah membuat *unit test* hasil turunan dari *user requirement*. Saat pertama kali tes dijalankan hasilnya akan *error*, sebab belum terdapat kode apapun yang membuat tes menjadi terpenuhi. Oleh sebab itu, tugas pengembangan adalah membuat kode sampai tes yang membuat berhasil terpenuhi. Pengembang akan terus melakukan revisi kode program, dan apabila tes pertama sudah berhasil terpenuhi maka akan berganti ke tes-tes berikutnya sesuai fungsinya masing-masing. Apabila semua *unit test* sudah berhasil dijalankan, maka pengembangan sistem selesai dibangun.

Tahapan berikutnya setelah melalui *unit test* adalah, melakukan proses *refactoring*. Pada metode TFD, kode yang dihasilkan biasanya kurang rapi atau acak-acakan. Oleh sebab itu dilakukan proses *refactoring* agar kode lebih mudah di *maintenance* dan dipahami oleh pengembang lain. Proses *refactoring* dilakukan dengan memperhitungkan hasil pengujian agar tetap *passed* artinya fitur tetap dapat bekerja sebagaimana mestinya (sesuai spesifikasi).

Test Drivven Development dalam Alur Pengembangan Software



D. Manfaat Penggunaan Test Driven Development

1. Desain yang lebih baik

TDD dianggap dalam banyak kasus sebagai proses desain daripada proses testing. Hal ini dikarenakan kode test yang ditulis diawal cenderung lebih kohesif serta mengurangi *coupling*.

2. **Efisiensi**

Penggunaan TDD akan mengidentifikasikan kesalahan atau *error* dengan cepat ketika kode baru ditambahkan ke sistem. Selain itu, dengan menerapkan TDD, kita tidak perlu lagi membuat *test* setelah implementasi kode selesai. Hal ini memberikan keuntungan sendiri, karena membuat *test* setelah implementasi kode sudah selesai akan jauh lebih susah dibandingkan menulis *test* di awal [3].

3. **Test assets**

Test case yang ditulis secara otomatis dalam TDD adalah aset yang berharga bagi proyek. Ketika ada sebuah peningkatan atau modifikasi pada kode, maka menjalankan unit test secara otomatis dapat mengidentifikasi terjadinya cacat/*defect* baru pada kode.

4**. Mengurangi *defect injection***

Kesalahan lebih rentan terjadi pada saat pemeliharaan kode atau perubahan kode daripada implementasi kode baru. Seringkali, *defect injection* terjadi pada saat dilakukan pemeliharaan kode. Dengan penggunaan TDD, *developer* dapat mengetahui apakah perubahan tersebut mengakibatkan kesalahan pada kode yang sudah jadi atau kode yang baru.

E. Kelebihan dan kekurangan Test Driven Development

Kelebihan :

Beberapa keuntungan yang kita dapat ketika menggunakan *Test Driven Development* adalah :

1. Saat terjadi *error* kita akan tahu persis letak *error* tersebut berada.
2. Dibandingkan meng-*test* sistem secara keseluruhan, melakukan TDD memberikan waktu yang cepat dalam penggabungan suatu sistem.

Kekurangan :

1. Sulit menentukan *unit test* yang benar dalam sebuah studi kasus karena tidak ada ukuran yang menunjukkan kebenaran dari suatu *unit test* yang dibangun.
2. Tim pembangun sulit membuat abstraksi dari sebuah *interface* karena tidak ada *physical design* yang dilakukan di awal proses pembangunan aplikasi. Tim pembangun hanya membangun aplikasi dengan *design thinking*.
3. Dalam pembangunan aplikasi dengan metodologi ini dibutuhkan tim pembangun yang sudah berpengalaman dan memahami bagaimana menulis *test* yang baik dan juga mengerti sedikit tentang arsitektur yang baik sehingga tidak cocok bagi tim pembangun pemula

BAB III
CONTOH METODE JURNAL

Mengambil jurnal dengan judul IMPLEMENTASI TEST DRIVEN DEVELOPMENT DALAM PENGEMBANGAN APLIKASI BERBASIS WEB

1. METODOLOGI PENELITIAN

Metodologi yang digunakan dalam penelitian ini adalah metodologi kualitatif. Dimana menurut [6] metodologi penelitian kualitatif adalah suatu metode yang digunakan untuk memahami fenomena tentang yang dialami oleh objek penelitian. Pada metodologi kualitatif tidak terdapat perhitungan matematika dan statistik.

1.1 Pengumpulan Data

Metode pengumpulan data pada penelitian ini dilakukan dengan melakukan pengamatan langsung terhadap website-website resmi seperti broadway.com dan imdb.com. Tujuan pengamatan tersebut adalah untuk melakukan kajian tentang kebutuhan pengguna dan proses bisnis yang berjalan pada sebuah sistem rating dan review film.

1.2 Analisis Data

Berdasarkan hasil pengamatan pada tahap pengumpulan data, dilakukan analisa untuk menentukan kebutuhan pengguna dan proses bisnis dari aplikasi IMDB yang dibuat. Agar lebih memudahkan pengembang dalam bekerja, analisis kebutuhan sistem tersebut diimplementasikan dalam bentuk *user stories.* Secara mendasar *user stories* tidak memiliki rumus baku dalam pembuatannya. Namun demikian, terdapat rumus yang banyak diadopsi oleh tim Agile, yaitu dengan menggunakan rumus **As-I-So** sebagai berikut [7].

As [an actor] I want to [action] so that [benefit]

Selanjutnya berdasarkan *user stories* tersebut, maka dirancanglah sebuah *rule* untuk membuat *unit test.* Setiap *unit test* akan diuji validitasnya dengan menggunakan *tools* yang berjalan pada *framework* rails yaitu Rspec. Adapun *user stories* dari aplikasi IMDB, ditunjukan pada Tabel 1.

Tabel 1. *User stories* dari aplikasi IMDB

No	*As {Actor)*	*I Want to {Action}*	*So that {Object)*
MV-001	Admin, member	Menambah daftar film (*create post*)	Admin dan member dapat menambah daftar film.
MV-002	Admin	Mengubah dan menghapus daftar film yang dibuat oleh member	Admin dapat mengedit dan menghapus postingan film yang dibuat oleh semua member
MV-003	Admin, member	Memberikan komentar dan rating dari sebuah film	Admin dan member dapat memberikan komentar dan rating dari sebuah film yang telah di posting member lain
MV-004	Admin	Mengubah dan menghapus komentar dari member	Admin dapat menertipkan komentar dari member yang mengganggu
MV-005	Admin, member	Melihat daftar film sesuai kategori	Admin dan member dapat memilih kategori film yang ingin di review
MV-006	Member	Mengubah dan menghapus daftar film yang dibuat	Member dapat mengubah dan menghapus daftar film yang telah member buat sendiri

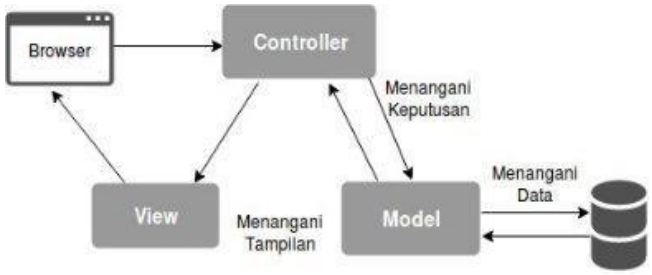
MV-007	Member	Mengubah dan menghapus komentar yang dibuat	Member dapat mengubah dan menghapus komentar yang telah dibuat
MV-008	Pengunjung	Mendaftar aplikasi IMDB	Pengunjung dapat mendaftar menjadi member dari IMDB

1.1 Desain Sistem

Pada tahap ini akan dilakukan penentuan arsitektur sistem, desain model proses dan model data. Arsitektur sistem yang dikembangkan pada aplikasi web ini adalah MVC. Desain model proses dari aplikasi web digambarkan menggunakan *United Modelling Language* (UML). Sedangkan untuk model data digambarkan dengan relasi antar tabel dalam basis data. Masing-masing tahap pada desain sistem, dijelaskan sebagai berikut

1.1.1 Perancangan Arsitektur Sistem

Seperti pada umumnya framework pengembangan web, Rails mengabdopsi model arsitektur MVC. Cara kerja dari model MVC ini yaitu memisahkan data (*Model*) dari tampilan (*View*) dan cara mengaksesnya (*Controller*). Arsitektur dari MVC pada sebuah pengembangan web, ditunjukan pada Gambar 1.

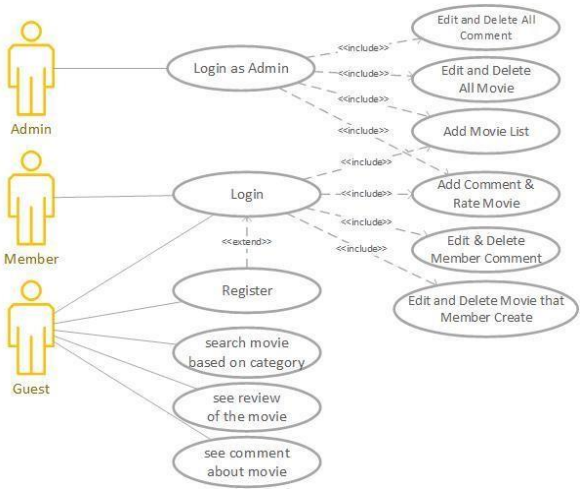


Penjelasan mengenai arsitektur MVC pada Gambar 1 adalah sebagai berikut

a. Pengguna melakukan permintaan untuk mengakses aplikasi web ke browser.
b. Setiap permintaan yang diberikan oleh pengguna akan ditangani oleh Controller.
c. Kemudian dari Controller akan dilanjutkan ke Model.
d. Saat mengakses Model, dilakukan pengecekan mengenai kebutuhan basis data.
e. Jika permintaan membutuhkan basis data, maka Model akan mengambil data dari basis data. Namun jika tidak maka dari Controller akan langsung ke proses View.
f. Informasi dari Model dikembalikan ke Controller, kemudian Controller melanjutkan ke *state* View untuk ditampilkan di browser.
g. View menampilkan antarmuka pengguna dan juga informasi yang telah diperoleh dari basis data.

1.2.1 Perancangan Diagram Use Case

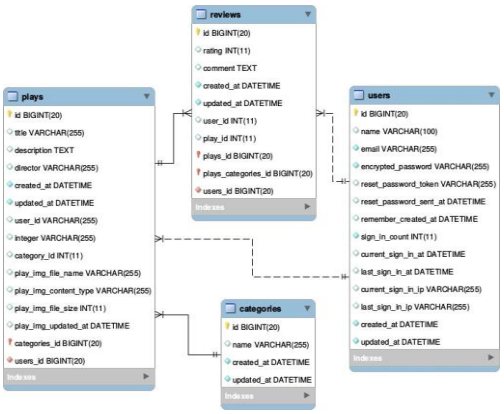
Pada tahap ini, ditentukan sebuah alur proses dari masing-masing pengguna. Setiap pengguna memiliki fitur-fitur tertentu dalam mengakses aplikasi IMDB yang dibuat. Terdapat tiga pengguna yang berperan dalam menggunakan aplikasi IMDB ini. Adapun diagram *use case* dari aplikasi IMDB, ditunjukan pada gambar 2.



Gambar 2. Diagram *Use Case* Aplikasi IMDB

Gambar 2 menunjukan bahwa terdapat tiga pengguna yang dapat mengakses aplikasi IMBD. Setiap pengguna memiliki hak akses yang berbeda-beda, tergantung dari levelnya. Pada level pengunjung, pengguna hanya memiliki akses untuk melihat daftar film, melihat review film dan membaca komentar dari member. Ketika pengunjung mendaftar sistem, maka status nya naik menjadi member. Pada level member, pengguna dapat menambahkan daftar film, menambahkan komentar dan memberikan *rating* film. Sedangkan pada level yang paling tinggi yaitu admin, pengguna dapat mengedit dan menghapus data film serta komentar-komentar yang dibuat oleh member.

1.2.2 Perancangan Model Data



Gambar 3. Relasi Antar Tabel Pada Aplikasi IMDB

Relasi antar tabel pada aplikasi IMDB ditunjukan pada gambar 3. Aplikasi IMDB yang dibangun memiliki 4 buah tabel yaitu tabel play yang berisi deskripsi dari film, tabel users yang berisi deskripsi dari pengguna, tabel categories berisi kategori (*genre*) dari film, dan tabel reviews berisi keterangan review yang diberikan oleh pengguna. Masing-masing tabel tersebut saling terintegrasi satu sama lain. Relasi antar tabel ini juga diimplementasikan ke dalam kode program, sebagai contoh pada berkas play.rb yang isinya sebagai berikut.

```
class Play <
      ApplicationRecord
      belongs_to
```

```
        :category
        belongs_to :user
        has_many :review
end
```

Maksud pada kode diatas adalah bahwa kelas plays berelasi dengan kelas category, user, dan review.

Kelas-kelas tersebut berada di direktori model yang digunakan untuk mengatur hubungan relasi antar tabel dalam basis data.

1.1 *Implementasi*

Implementasi program dilakukan dengan menerapkan metode *Test First Development* (TFD). Adapun langkah-langkah pada metode TFD ditunjukan seperti gambar 4.

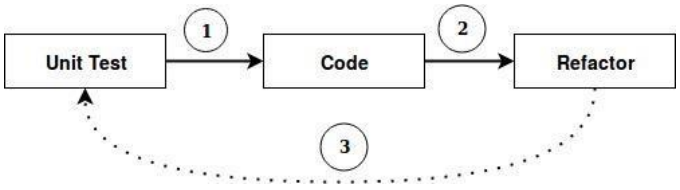
Gambar 4. Alur Kerja TFD

Berdasarkan gambar 4, hal pertama yang dilakukan ketika menerapkan metode TFD adalah membuat *unit test* hasil turunan dari *user requirement*. Saat pertama kali tes dijalankan hasilnya akan *error*, sebab belum terdapat kode apapun yang membuat tes menjadi terpenuhi. Oleh sebab itu, tugas pengembang adalah membuat kode sampai tes yang dibuat berhasil terpenuhi. Pengembang akan terus melakukan revisi kode program, dan apabila tes pertama sudah berhasil terpenuhi maka akan berganti ke tes-tes berikutnya sesuai fungsinya masing-masing. Apabila sumua unit test sudah berhasil dijalankan, maka pengembangan sistem selesai dibangun.

1.3 *Refactoring*

Pada metode TFD, kode yang dihasilkan biasanya acak-acakan atau kurang rapi. Oleh sebab itu dilakukan proses *refactoring* agar kode lebih mudah di *maintenace* dan dipahami oleh pengembang lain. Proses *refactoring* dilakukan dengan memperhitungkan hasil pengujian agar tetap *passed* artinya fitur tetap dapat bekerja sebagaimana mestinya (sesuai spesifikasi). Sehingga dapat dikatakan bahwa TTD adalah gabungan antara TFD dengan *refactoring*, sebagaimana

ditunjukan pada gambar 5.



Gambar 5. Alur Kerja TDD

Penjelasan gambar 5 adalah sebagai berikut

a. Pengembang menuliskan *unit test* terlebih dahulu, sesuai spesifikasi dan bisnis proses dari sistem yang diingikan oleh pengguna.
b. Setelah membuat *unit test*, pengembang menuliskan kode program untuk menyelesaikan *unit test* sampai *pass* (tidak terjadi *error*).

Pengembang melakukan *refactoring* kode program yaitu mengubah struktur program agar lebih mudah dipahami dan dimodifikasi, tanpa harus mengubah dari *behaviour* program

1. HASIL DAN PEMBAHASAN

Luaran dari penelitian ini menghasilkan sebuah aplikasi web yang digunakan untuk memberikan rating dan review terhadap film-film Indonesia. Model pengembangan yang digunakan pada penelitian ini menggunakan metode *Test Driven Development.* Siklus pengembangan aplikasi yang dibangun dimulai dengan melakukan pengujian terlebih dahulu, kemudian melakukan implementasi program, dan terakhir melakukan *refactoring*.

1.1 Implementasi Pengujian Unit

Berdasarkan *user stories* yang telah dibuat, maka dirancang beberapa *unit test* dari aplikasi IMDB. Validasi dari *unit test* dilakukan dengan menggunakan sebuah *tools* bernama Rspec. Pemasangan Rspec pada *framework* Rails dilakukan dengan menambahkan sebuah sebuah kode pada berkas Gemfile sebagai berikut.

```
group :development,
  :test do gem 'rspec-
  rails'
end
```

Kemudian pada teminal, lakukan pemasangan Rspec dengan mengetikan perintah `rails generate rspec:install`. Apabila pemasangan Rspec berhasil, maka akan muncul direktori bernama ‘spec’. Selanjutnya di dalam direktori spec, tambahkan direktori baru bernama ‘features’. Pembuatan *unit test* dilakukan di dalam direktori ‘features’ dengan cara menambahkan berkas berisi kode dalam bahasa ruby. Adapun contoh *unit test* yang ditulis menggunakan bahasa ruby, ditunjukan seperti kode berikut.

```
require ‘rails_helper.rb’
feature ‘Membuat Daftar Film’ do
    scenario ‘Dapat Menambahkan Daftar
       Film’ do visit ‘/’
       click_link ‘Tambahkan
       Film’ fill_in ‘Judul’,
       with: ‘judul’
       fill_in ‘Deskripsi’, with:
       ‘deskripsi’ click_button ‘Tambah
       Film’ expect(page).to
       have_content(‘judul’)
       expect(page).to
       have_content(‘deskripsi’)
    end
```

```
end
```

Kode diatas adalah salah satu penerapan *unit test* dengan Rspec. Penulisan *unit test* pada Rspec dilakukan dengan cara menentukan terlebih dahulu fitur apa yang akan dibuat. Setelah itu menentukan skenario (langkah kerja) dari sebuah fitur. Kode diatas digunakan untuk menambahkan judul dan deskripsi film, dimana skenarionya adalah sebagai berikut.

a. Pada halaman beranda akan muncul tautan 'Tambah Film'
b. Ketika tautan 'Tambahkan Film' ditekan, maka akan diarahkan ke halaman tertentu yang berisi form untuk menambahkan judul dan deskripsi film.
c. Isikan judul dan deskripsi film, kemudian tekan tombol 'Tambah Film'.

Setelah tombol 'Tambah Film' ditekan, maka judul dan deskripsi yang telah *submit* akan ditampilkan.

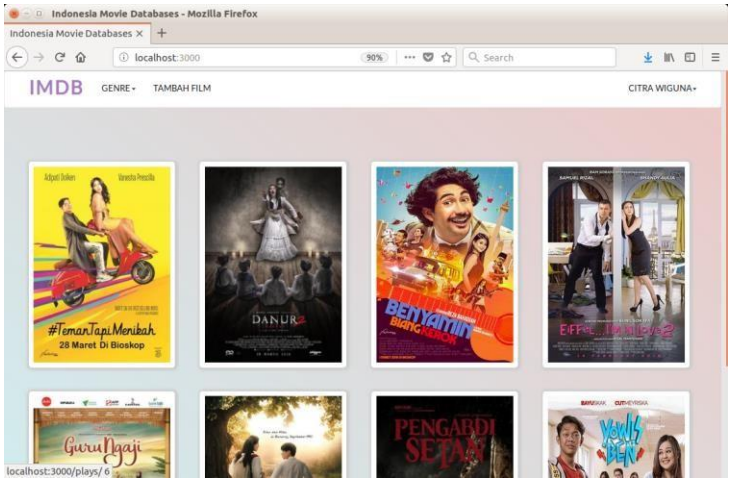
Berdasarkan *user stories* yang telah dibuat pada Tabel 1, dihasilkan 15 *unit test* pada aplikasi IMDB ini. Fitur-fitur tiap *unit test* ditunjukan pada Tabel 2. Setiap *unit test* telah berhasil diuji menggunakan Rpsec, tampilkan pengujian *unit test* yang berhasil di eksekusi ditunjukan pada Gambar 7.

Tabel 2. Fitur dari masing-masing *unit test*

No.	***Fitur***	***Hasil***
1	*Create post*	Berhasil
2	*Display post*	Berhasil
3	*Edit and delete post*	Berhasil
4	*Add user to App*	Berhasil
5	*Navigation and model association*	Berhasil
6	*Categories for the movie*	Berhasil
7	*Edit page dropdown and navbar dropdown*	Berhasil
8	*Filtering with category*	Berhasil
9	*Image uploading*	Berhasil
10	*Display uploaded image*	Berhasil
11	*Add reviews*	Berhasil
12	*Display reviews*	Berhasil
13	*Edit and delete reviews*	Berhasil
14	*Add 5 star rating system*	Berhasil
15	*Average rating*	Berhasil

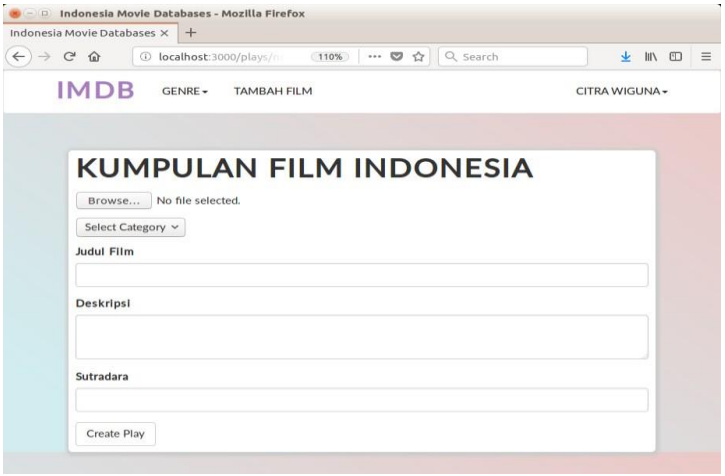
1.1 Tampilan Antarmuka Sistem

Tampilan antarmuka sistem setelah melalui serangkaian tes dan *production code*, ditunjukan pada Gambar 8 – 10. Ketika tes berhasil dijalankan (*pass*) maka dapat dipastikan bahwa fitur yang dikehendaki dapat bekerja dengan baik. Agar lebih memudahkan pengguna dalam menggunakan aplikasi (*user friendly*) digunakan bootstrap dan css untuk mempercantik tampilan.



Gambar 8. Tampilan Antarmuka Halaman *Home*

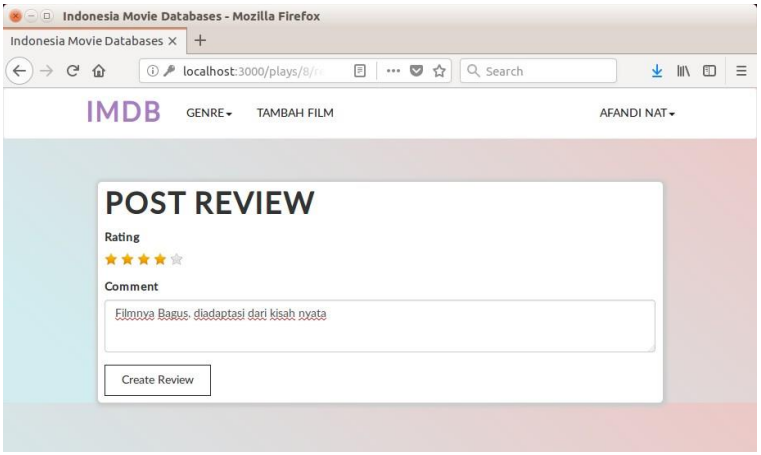
Gambar 8 menunjukan tampilan antarmuka pada halaman beranda aplikasi IMDB. Pada halaman beranda, pengguna disuguhkan daftar film-film terbaru yang



telah dimasukan admin atau member. Pengguna dapat mengelompokan film berdasarkan pada *genre* atau kategori.

Gambar 9. Tampilan Antarmuka Halaman *Add Movie*

Ketika pengunjung melakukan registrasi dan tercatat sebagai member IMDB, maka pengunjung memiliki hak akses untuk menambah daftar film. Gambar 9 merupakan tampilan antarmuka pada halaman tambah daftar film.



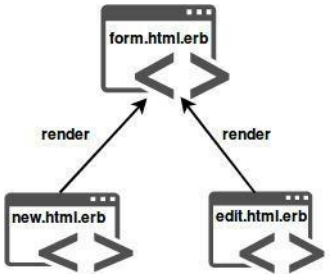
Gambar 10. Tampilan Antarmuka Halaman Tambah Komentar Dan Rating

Salah satu keuntungan menjadi member adalah dapat memberikan komentar dan rating terhadap sebuah film. gambar 10 menunjukan bahwa member dapat menuliskan komentar dan rating berupa pemberian bintang. Nilai maksimal rating memiliki bobot 5. Pada tampilan review film semua bobot rating yang diberikan oleh pengguna akan di rata-rata dan disajikan dalam tanda bintang.

1.4 Refactoring Program

Refactoring bertujuan untuk menghasilkan *clean code* yang mudah dibaca, tidak mengandung duplikasi, dan mudah dilakukan *maintenance.* Menurut [9] dalam bukunya menerangkan bahwa cara untuk memenuhi *deadline* dan bekerja dengan cepat adalah sebisa mungkin menjaga agar tercipta *clean code.* Oleh karena

itu, pada aplikasi IMDB ini dilakukan *refactoring* kode program seperti ditunjukan pada gambar 11.



Gambar 11. Penerapan R*efactoring* Pada Modul

gambar 11 menunjukan bahwa berkas form.html.erb berisi *template* yang digunakan bersama oleh new.html.erb dan edit.html.erb. Sehingga isi dari berkas new.html.erb dan edit.html.erb hanya berisi satu baris kode yaitu render ‘form’. Penghematan listing penulisan kode ini dapat meningkatkan efisiensi dari pengembang dalam membangun aplikasi IMDB.

2. KESIMPULAN

Berdasarkan hasil perancangan dan implementasi terhadap metode TDD dapat disimpulkan bahwa fungsi-fungsi dari aplikasi web yang dibangun dapat berjalan dengan baik. Pengembang tidak perlu melakukan pengujian aplikasi dengan cara mencoba menu satu persatu. Penerapan metode TDD membantu pengembang dalam menerapkan proses bisnis dan kebutuhan dari pengguna aplikasi IMDB. Selain itu juga memudahkan pengembang menemukan kesalahan ketika mengimplementasikan kode program.Adanya *refactoring* pada metode TDD membuat kode program yang dihasilkan lebih rapi. Pengembang yang bekerja secara tim dapat dengan mudah memahami dan memodifikasi kode setelah melalui tahap *refactoring*. Pengujian *unit test* menggunakan Rspec membantu pengembang dalam menangani kesalahan dan memudahkan menambah fitur baru dari aplikasi web.Selain beberapa kemudahan yang telah diuraikan, penerapan aplikasi web dengan TDD ini juga memiliki beberapa kesulitan yang terjadi selama *development*. Salah satu kesulitan dalam penerapan metode TDD ini adalah ketika mengkonversikan kebutuhan pengguna ke dalam *unit test.* Terkadang *unit test* yang telah dibuat tidak sesuai dengan proses bisnis, sebab tidak ada ukuran yang menunjukan bahwa *unit test* yang dibuat telah benar. Kemudian dikarenakan TDD hanya menguji fungsionalitas (*back-end*) dari aplikasi web, maka masalah tampilan (*front-end*) dari aplikasi web dilakukan secara manual.

BAB IV
PERBANDINGAN WATERFAL PROTOTYPE DAN RAD

A. Perbandingan Test Driven Development dengan Waterfall

TDD

- Unit atau tes modul menunjukkan bahwa kode tersebut benar-benar berfungsi.
- Tes ini mengontrol desain program.
- Retesting memungkinkan untuk meningkatkan desain.
- Pengujian regresi tetap berukuran kecil.
- Biaya sejumlah besar bug dicegah.

Sedangkan Waterfall

- Waterfall adalah salah satu model termudah untuk dikelola. Karena sifatnya, setiap fase memiliki kiriman khusus dan proses penilaian.
- Waterfall bekerja dengan baik untuk proyek yang lebih kecil di mana persyaratannya mudah dipahami.
- Pengiriman proyek lebih cepat.
- Proses dan hasil pengembangan didokumentasikan dengan baik.
- Metode yang mudah diadaptasi untuk tim yang berbeda per fase.
- Spesialis di bidang tertentu lebih efektif.
- Metodologi manajemen proyek ini berguna untuk mengelola dependensi

B. Perbandingan Test Driven Development dengan Prototype

TDD

- Unit atau tes modul menunjukkan bahwa kode tersebut benar-benar berfungsi.
- Tes ini mengontrol desain program.
- Retesting memungkinkan untuk meningkatkan desain.
- Pengujian regresi tetap berukuran kecil.
- Biaya sejumlah besar bug dicegah

Sedangkan Prototype

- Mampu Mengetahui Kebutuhan Pengguna Lebih Awal
- Dapat Menghemat Biaya Pengembangan Produk
- Mendapatkan Gambaran secara Lebih Konkret

D. Perbandingan Test Driven Development dengan RAD

TDD

- Unit atau tes modul menunjukkan bahwa kode tersebut benar-benar berfungsi.
- Tes ini mengontrol desain program.
- Retesting memungkinkan untuk meningkatkan desain.
- Pengujian regresi tetap berukuran kecil.

- Biaya sejumlah besar bug dicegah

Sedangkan RAD

- Mudah mengakomodasi perubahan sistem
- Progress development bisa diukur
- Waktu iterasi bisa diperpendek
- Mengurangi waktu development
- Mempermudah untuk mendapatkan feedback user
- Cocok untuk sistem yang berbasis komponen dan terukur
- Mudah dalam menentukan dasar sistem
- Cocok untuk proyek pengembangan yang membutuhkan waktu singkat