PENGUJIAN PERANGKAT LUNAK

Untuk memenuhi tugas mata kuliah Manajemen Proyek dan Perangkat Lunak

Oleh :

Ade Lisna

164060063

SEKOLAT TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER JABAR

BANDUNG

2017

KATA PENGANTAR

Puji dan syukur kami panjatkan kehadirat Allah سبحانه وتعالى yang telah memberikan rahmat dan hidayah-Nya, sehingga dapat menyelesaikan pembuatan tugas ini guna memenuhi tugas mata kuliah Rekayasa Perangkat Lunak.

Dalam pembuatan tugas ini mengalami beberapa kesulitan, namun berkat bimbingan dan bantuan dari semua pihak akhirnya tugas ini dapat terselesaikan tepat pada waktunya. Oleh karena itu, Ucapan terima kasih juga tak lupa ucapkan kepada Bapak Agung Wahana ST,.M.Kom, selaku dosen mata kuliah Manajeme Proyek Rekayasa Perangkat Lunak yang telah memberikan tugas ini. Semoga tugas ini dapat bermanfaat bagi kita semua.

Tak ada gading yang tak retak. Begitu pula dengan tugas yang di buat ini yang masih jauh dari kesempurnaan. Oleh karena itu, mohon maaf apabila ada kekurangan ataupun kesalahan. Kritik dan saran sangat diharapkan agar tugas ini menjadi lebih baik serta berdaya guna dimasa yang akan datang.

Bandung, 21 Januari 2017

Ade Lisna

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar belakang

Pengujian perangkat lunak memegang peranan penting dalam menjaga kualitas perangkat lunak. Menurut Galin (2004) pengujian perangkat lunak atau software testing diartikan sebagai proses formal dimana suatu perangkat lunak diuji dengan cara menjalankan perangkat lunak tersebut dalam komputer dan menjalankan prosedur serta kasus tertentu.

Dalam pengembangan perangkat lunak, tekanan untuk menyelesaikan perangkat lunak dengan cepat sering ditemui. Selain itu, perangkat lunak yang dikembangkan di era modern memiliki kompleksitas yang tinggi, sehingga meningkatkan tingkat kesulitan dalam melakukan pengujian. Hal-hal tersebut seringkali menyebabkan manajer proyek memutuskan untuk mengurangi aktivitas ataupun sumber daya yang diperlukan untuk melakukan pengujian perangkat lunak (Konka, 2011).

Pengujian perangkat lunak yang dilaksanakan dengan tidak sempurna tentu akan membawa pengaruh yang kurang baik terhadap kualitas perangkat lunak yang dihasilkan. Pengujian perangkat lunak yang tidak efektif dan tidak lengkap dapat mengakibatkan berbagai masalah ketika perangkat lunak tersebut digunakan oleh *end-user* (Catelani dkk., 2011).

Dalam teori pengujian perangkat lunak terdapat berbagai metode yang bisa digunakan untuk melakukan pengujian, misalnya metode *white-box testing* dan metode *black-box testing*. Sistem penguji perangkat lunak tentunya harus mampu menguji berbagai aspek dalam perangkat lunak sehingga penggunaan lebih dari satu metode pengujian sangat diharapkan (Galin, 2004).

Di dalam merancang dan membangun sebuah perangkat lunak berbasis proyek, semua kebutuhan pengguna harus bisa diakomodir oleh perangkat lunak yang dibuat. Untuk itu, salah satu cara memastikan kesesuaian antara kebutuhan dan *output* yang dihasilkan, adalah dengan membuat *use case*. *Use case* menjadi elemen dasar dan terpenting dalam tahap desain perangkat lunak, pembuatan diagram *robustness, sequence* bahkan hingga *class diagram* didasarkan pada *skenario* yang dijabarkan pada *usecase*.

Sebuah perangkat lunak yang baik, idealnya adalah yang telah memenuhi semua kebutuhan penggunanya. Cara yang paling lazim digunakan untuk mengetahui apakah perangkat lunak yang dibuat telah sesuai dengan *use case*-nya, adalah cara *test case*. Berdasarkan skenario *basic* dan *alternate path* pada *use case*, dikembangkan seperangkat

skenario *testing*. Selain itu, setiap skenario *testing* akan diberikan serangkaian *data dummy* yang akan dilakukan sebagai perangkat *testing*. Hasil dari *testing* ini akan menunjukkan sejauh mana kesesuaian antara *use case* dengan perangkat lunak.

1.2 Rumusan Masalah

Dari latar belakang di atas dapat diambil rumusan masalah sebagai berikut :

1. Apa itu strategi pengujian perangkat lunak ?
2. Bagaimana cara membangun *test case* ?

1.3 Tujuan

Tujuan membuat makalah ini adalah :

1. Mengetahui tentang strategi pengujian perangkat lunak.
2. Mengetahui cara membangun *test case*.

1.4 Manfaat

Manfaat yang didapatkan dari makalah ini adalah sebagai berikut :

1. Mampu menerapkan srategi – strategi yang digunakan dalam pengujian perangkat lunak.
2. Mampu menerapkan *test case* dalam pengujian perangkat lunak.

BAB II

PEMBAHASAN

2.1 Perancangan UML.

Pada bab ini akan membahas eksperimen berupa pengujian terhadap studi kasus, yaitu Sistem Informasi Booking Tiket Bioskop Online Ambil Tiket Dot Com.

Pengguna Sistem informasi ini meliputi beberapa pihak sebagai berikut :

- **Pemesan**

  Pemesan adalah pihak yang secara langsung berkepentingan dengan sistem ini dalam hal booking tiket bioskop online.

- **Pengunjung**

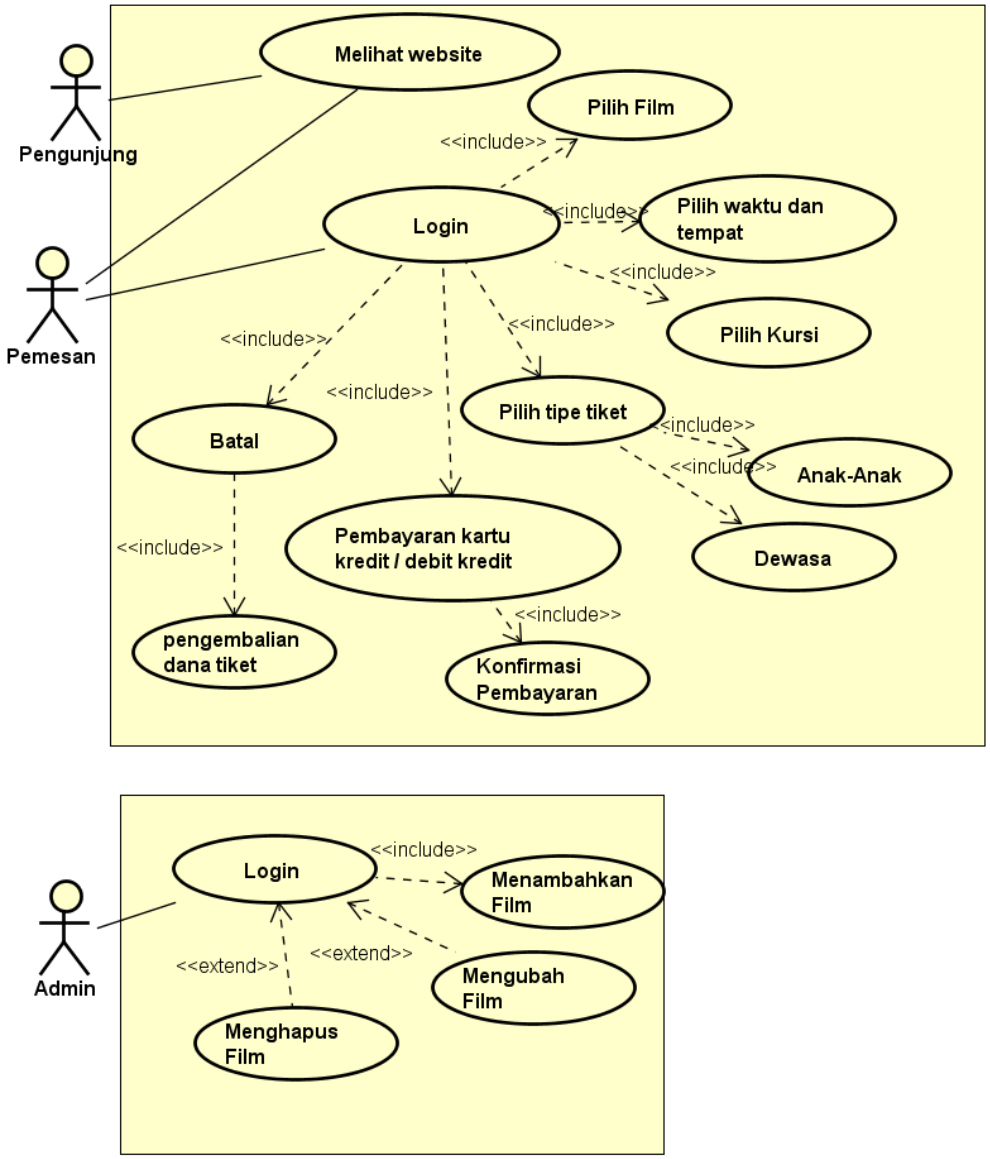
  Pengunjung merupakan pihak yang hanya melihat sistem tidak secara keseluruhan.

- **Admin**

  Admin merupakan pihak yang mengelola keseluruhan dari sistem ini.

Untuk menjelaskan kebutuhan fungsional, lalu digunakan *use-case diagram*. *Use-case diagram* adalah sebuah diagram yang menggambarkan interaksi dan keterhubungan antara sistem yang akan dibuat dengan sistem-sistem eksternal lainnya, termasuk *user* yang akan menggunakan sistem tersebut.

1. Use Case Diagram



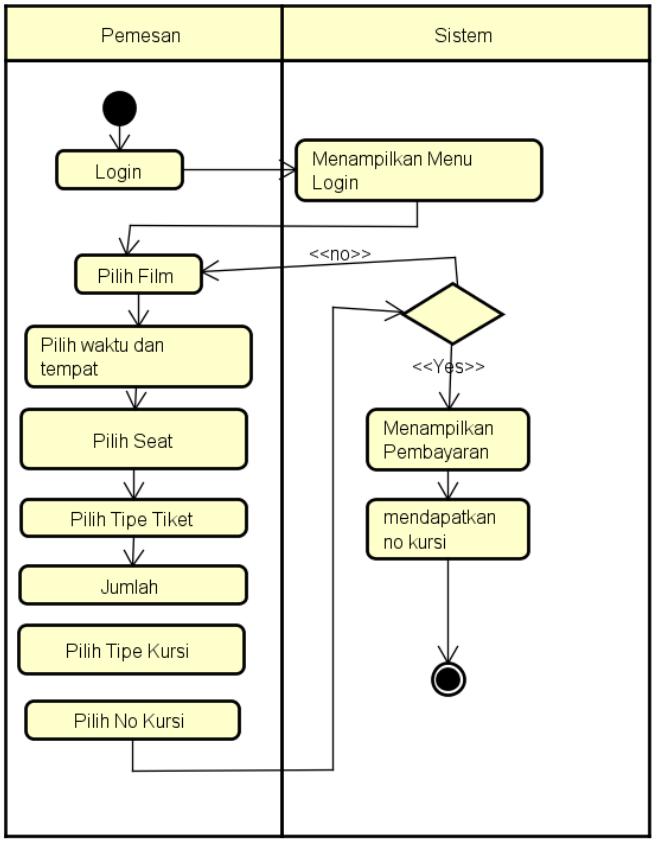
Gambar 1 Use case Sistem Booking Tiket Bioskop Online

2. Rancangan Skenario

IDENTIFIKASI	
Nomor	UCS-01
Nama	Use case Pemesanan Tiket
Deskripsi	Untuk memesan Tiket Bioskop online
Aktor	Pemesan
Kondisi Awal	Aktor telah login
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
1. Aktor memilih menu film	2. Sistem menampilkan halaman data film
3. Aktor menginput data booking	4. Sistem memproses pemesanan tiket

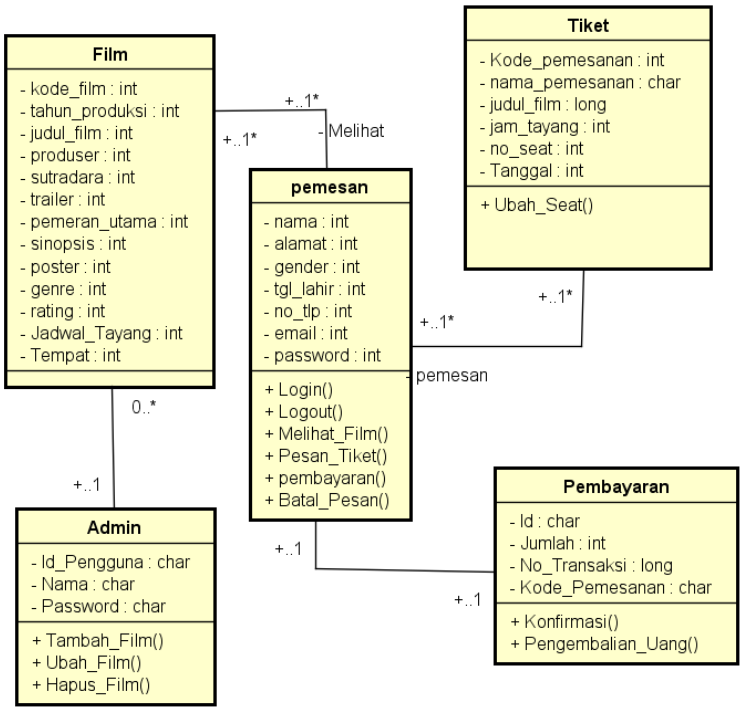
	5. Sistem menampilkan Detail pemesanan Tiket
EXCEPTIONAL CASE	

3. Activity Diagram



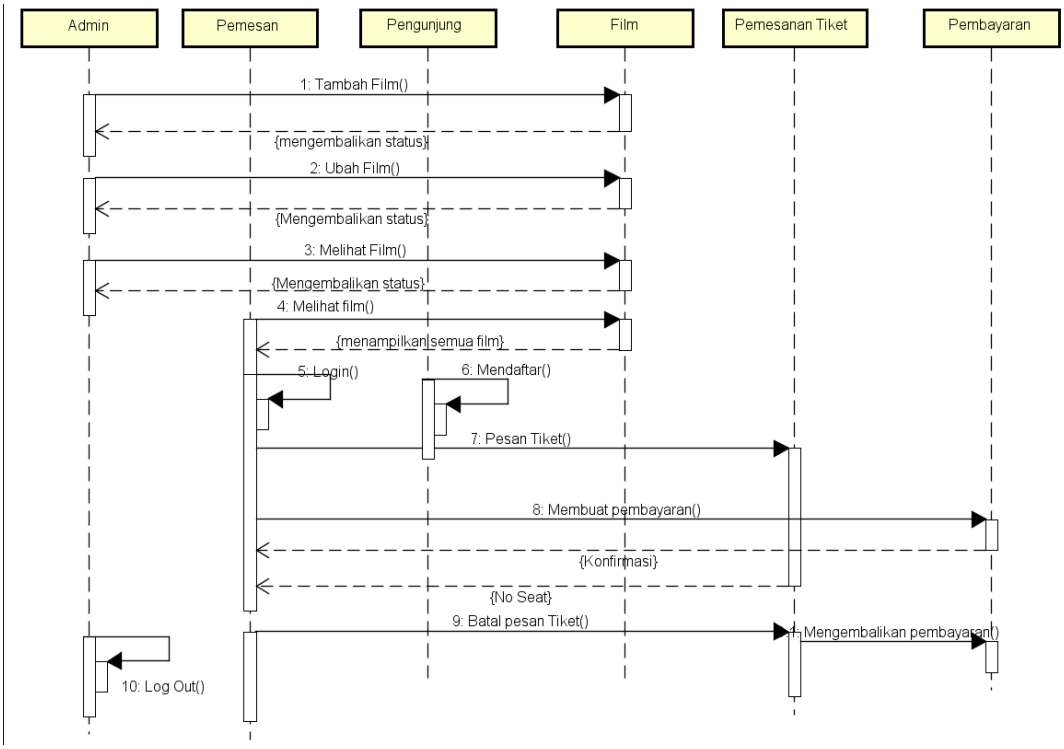
Gambar 3 Activity Diagram pemesanan Tiket Online

4. Class Diagram



Gambar 4 Class Diagram Sistem Booking Tiket Bioskop Online

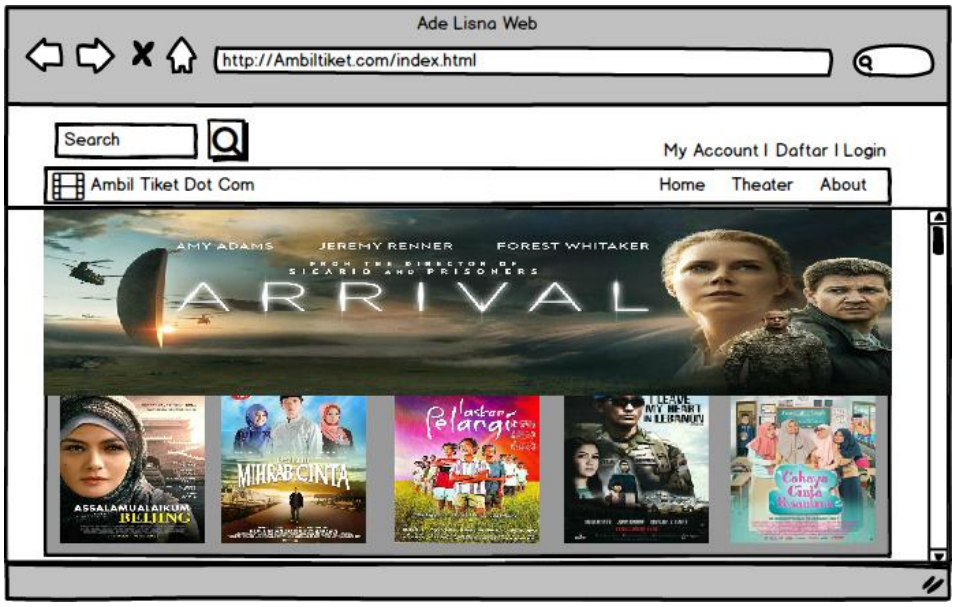
5. Sequence Diagram



Gambar 5 Sequence Diagram Sistem Booking Tiket Bioskop Online

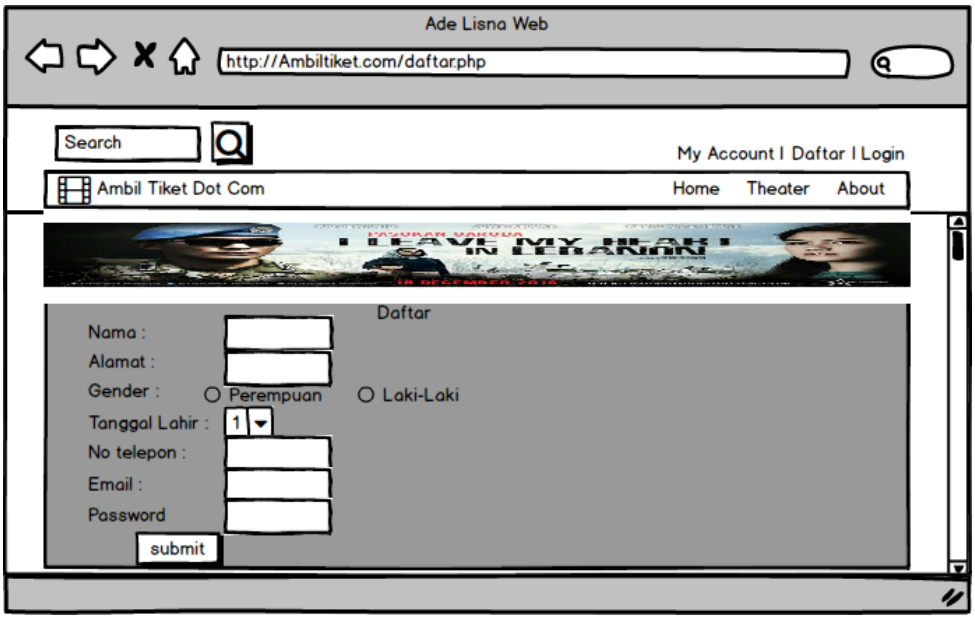
2.1 Perancangan Interface Sistem Informasi Booking Tiket Bioskop Online

1. Interface halaman utama Ambil tiket dot com.



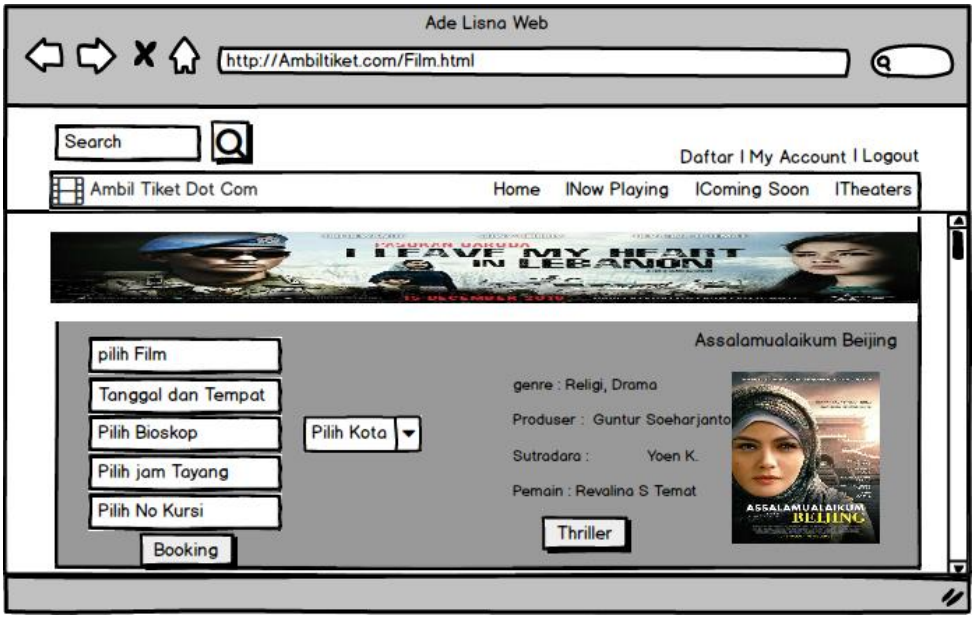
Gambar 6 Halaman Utama Sistem Booking Tiket Bioskop Online

2. Input Pendaftaran Pemesan.



Gambar 7 Pendaftaran Sistem Booking Tiket Bioskop Online

3. Input Pemesanan Tiket Bioskop Online



Gambar 8 Input Pemesanan Sistem Booking Tiket Bioskop Online

2.2 Pembuatan coding

Alogritma Pemesanan Tiket Bioskop Online

1. Pemesan Login
2. Pemesan memilih film terlebih dahulu
3. Pemesan Memilih waktu dan tempat
4. Pemesan memilih jumlah tiket, pemesan boleh memesan tiket lebih dari satu
5. Pemesan menentukan jumlah tiket yang akan di pesan dan mengelompokan kedalam kategori Dewasa atau Anak-anak.
6. Memlilih no kursi/seat.
7. Lakukan Perhitungan sub total (sub total)=(jumlah tiket Dewasa* harga tiket)+(jumlah tiket anak-anak*harga tiket)). Harga Tiket untuk anak-anak sebesar 5% dari harga tiket dewasa.
8. Konsumen memilih sistem pembayaran tiket.
9. User mempunyai pilihan untuk membatalkan atau menyetujui transaksi.
10. Jika user setuju maka transaksi selesai, bila tidak maka transaksi gagal.
11. Pemesan Logout

Pseudocode Pemesanan Tiket Bioskop Online

```
start
        Booking()
        Booking_done()
        output” Terima kasih Telah Booking Tiket”
stop

Booking()
        Pemesan()
        Film()
        Pembayaran()
return

 Booking_done()
     Output” Anda”,nama”telah booking”,kursi”Tiket”,film”di”,bioskop,kota”di”,tanggal.”
     Tiket Anda akan kami kirim lewat email”,email”Dengan Cepat.”
return

Pemesan()
        Deklarasi
                String nama
                Num MoNum
                string email
output”Masukan Nama”
input nama
Output “Masukan No Hp”
Input MoNum
Output”Masukan Email ID”
Input email
return

Film()
        Deklasrasi
                string kota
                string bioskop
                string film
                string kursi
                num date
        output”Masukkan Kota”
        input kota
        output”Masukan Tanggal”
        input date
        output”Pilh Bioskop “
        input bioskop
        output”Pilih Film”
        input film
        output” Pilih No Seat “
        input kursi
return
```

```
Pembayaran()
Deklarasi
        string kartu_kredit
        string kartu_debit
        string method
        num harga_tiket=25000
        num total
        string tiket
        string anak
total= kursi*harga_tiket
if kursi>10 then
        total=total-total(total*0.1)
else if tiket=anak then
        total=total-(total*0.5)
endif
        output" Total Yang Harus Di Bayar Adalah ",total
        output" Pilih Pembayaran
method(kartu_kredit/kartu_debit)"
        input method
        if method =kartu_kredit then
        Kartu_kredit()
else
        Kartu_debit()
endif
output" Pembayaran Anda Telah Di Konfirmasi"
return

Kartu_kredit()
        Deklarasi
                string nama_bank
                num no_rekening
                num validasi
                num kode
        output"Masukan Nama Bank"
        input nama_bank
        output" masukan No rekening"
        input no_rekening
        output" No Kartu Validasi (mm/yy)"
        input validasi
        output" Masukan No Pin "
        input kode
return
Debit_Card()
        Deklarasi
                string nama_bank
                num no_rekening
                num validasi
                num cvv_kode
                output"Masukan Nama Bank"
        input nama_bank
```

```
        output” masukan No rekening”
        input no_rekening
        output” No Kartu Validasi (mm/yy)”
        input validasi
        output” Masukan CVV “
        input cvv_kode
return
```

2.3 Pengujian perangkat Lunak

Pengujian dapat berarti proses untuk memeriksa apakah suatu perangkat lunak yang dihasilkan sudah dapat dijalankan sesuai dengan rancangan yang telah dibuat. Metode pengujian adalah cara atau teknik untuk menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap dan untuk menemukan kesalahan.

Dalam melakukan uji coba ada 2 masalah penting yang akan dibahas, yaitu :

A. Teknik uji coba PL

B. Strategi uji coba PL

Teknik Uji Coba Pl

Pada dasarnya, pengujian merupakan suatu proses rekayasa PL yg dapat dianggap (secara psikologis) sebagai hal yg destruktif daripada konstruktif.

Sasaran Pengujian (Glen Myers) :

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.
2. Test case yg baik adalah test case yg memiliki probabilitas tinggi untuk menemukan kesalahan yg belum pernah ditemukan sebalumnya.

3. Pengujian yg sukses adalah pengujian yg mengungkap semua kesalahan yg belum pernah ditemukan sebelumnya.

Prinsip Pengujian (diusulkan Davis) :

- Semua pengujian harus dapat ditelusuri sampai ke persyaratan pelanggan.
- Pengujian harus direncanakan lama sebelum pengujian itu dimulai.
- Prinsip Pareto berlaku untuk pengujian PL. Prinsip Pareto mengimplikasikan 80% dari semua kesalahan yg ditemukan selama pengujian sepertinya akan dapat ditelusuri sampai 20% dari semua modul program.
- Pengujian harus mulai "dari yg kecil" dan berkembang ke pengujian "yang besar".
- Pengujian yg mendalam tidak mungkin.
- Paling efektif, pengujian dilakukan oleh pihak ketiga yg independen

Testabilitas

Testabilitas PL adalah seberapa mudah sebuah program komputer dapat diuji. Karena pengujian sangat sulit, perlu diketahui apa yg dapat dilakukan untuk membuatnya menjadi mudah.

Karakteristik PL yg diuji :

- OPERABILITAS, semakin baik dia bekerja semakin efisien dia dapat diuji.
- OBSERVABILITAS, apa yg anda lihat adalah apa yg anda uji.
- KONTROLABILITAS, semakin baik kita dapat mengontrol PL semakin banyak pengujian yg adapat diotomatisasi dan dioptimalkan.
- DEKOMPOSABILITAS, dengan mengontrol ruang lingkup pengujian kita dapat lebih cepat mengisolasi masalah dan melakukan pengujian kembali.
- KESEDERHANAAN, semakin sedikit yg diuji semakin cepat pengujian.
- STABILITAS, semakin sedikit perubahan semakin sedikit gangguan pengujian.
- KEMAMPUAN DIPAHAMI, semakin banyak informasi yg dimiliki semakin detail pengujiannya.

Atribut Pengujian Yg Baik :

- Memiliki probabilitas yg tinggi menemukan kesalahan.
- Tidak redundan.
- Harusnya ‘jenis terbaik’.
- Tidak boleh terlalu sederhana atau terlalu kompleks.

Desain Test Case

Terdapat bermacam-macam rancangan metode test case :

1. Pengetahuan fungsi yg spesifik dari produk yg telah dirancang untuk diperlihatkan, test dapat dilakukan untuk menilai masing-masing fungsi apakah telah berjalan sebagaimana yg diharapkan.

2. Pengetahuan tentang cara kerja dari produk, test dapat dilakukan untuk memperlihatkan cara kerja dari produk secara rinci sesuai dengan spesifikasinya.

Dua macam pendekatan test yaitu :

1. Black Box Testing

Test case ini bertujuan untuk menunjukkan fungsi PL tentang cara beroperasinya, apakah pemasukan data keluaran telah berjalan sebagaimana yang diharapkan dan apakah informasi yang disimpan secara eksternal selalu dijaga kemutakhirannya.

2. White Box Testing

Adalah meramalkan cara kerja perangkat lunak secara rinci, karenanya logikal path (jalur logika) perangkat lunak akan ditest dengan menyediakan test case yang akan mengerjakan kumpulan kondisi dan atau pengulangan secara spesifik. Secara sekilas dapat diambil kesimpulan white box testing merupakan petunjuk untuk mendapatkan program yang benar secara 100%.

Uji Coba White Box

Uji coba white box adalah metode perancangan test case yang menggunakan struktur kontrol dari perancangan prosedural untuk mendapatkan test case. Dengan rnenggunakan metode white box, analis sistem akan dapat memperoleh test case yang:

- menjamin seluruh independent path di dalam modul yang dikerjakan sekurang-kurangnya sekali
- mengerjakan seluruh keputusan logikal
- mengerjakan seluruh loop yang sesuai dengan batasannya
- mengerjakan seluruh struktur data internal yang menjamin validitas

1. Uji Coba Basis Path

Uji coba basis path adalah teknik uji coba white box yg diusulkan

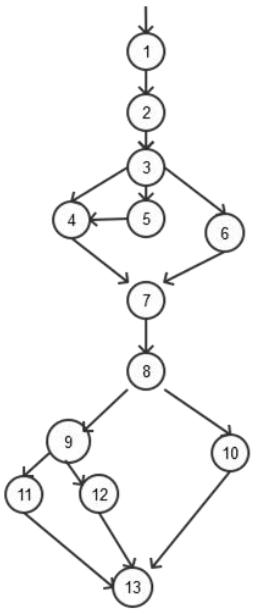
- Tom McCabe. Metode ini memungkinkan perancang test case mendapatkan
- ukuran kekompleksan logical dari perancangan prosedural dan menggunkan
- ukuran ini sbg petunjuk untuk mendefinisikan basis set dari jalur
- pengerjaan. Test case yg didapat digunakan untuk mengerjakan basis set
- yg menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.

Pada pengujian yang dilakukan, akan digunakan *use case* dari peran Pemesan. Menurut use case yang ada, fungsi-fungsi yang bisa dilakukan oleh pemesan adalah :

1. Login
2. Memilih film
3. Memilih waktu dan tempat
4. Memilih kursi
5. Memilih tipe tiket
6. Memilih tipe Pembayaran
7. Membatalkan pemesanan tiket
8. Logout

Penyusunan *test case*

Dari *use case* yang ada akan dibuat sebuah *graph* yang menggambarkan aktivitas pada aktor pemesan. *Graph* akan dibuat dengan asumsi seluruh proses hanya berjalan searah, tidak ada proses yang *looping*, dan minimal setiap *node* dilewati sekali. Tujuan pembuatan *graph* adalah untuk menentukan minimum *path coverage* untuk mencapai *full line coverage*. Dengan begitu *test case* yang akan dibuat adalah kumpulan *test case* minimum yang dapat memenuhi seluruh fungsi yang ada. Bentuk graph dari fungsi pemesan adalah :



Keterangan dari graph :

1. Login
2. Memilih Film
3. Memilih waktu dan tempat
4. Memilih no kursi
5. Loading
6. Memilih tiket
7. Memilih pembayaran
8. Submit
9. Menampilkan detail pemesanan
10. Batal
11. Submit
12. Kembali
13. Logout

Berdasarkan *flow graph* yang ada di atas, terdapat 68 kemungkinan *path* yang ada dengan jumlah *edge* sebanyak 29. Kemungkinan seluruh *path* yang ada adalah:

No.	*Path*
1.	1-2-3-4-7-8-9-11-13
2.	1-2-3-5-4-7-8-9-11-13
3.	1-2-3-6-7-8-9-11-13
4.	1-2-3-4-7-8-9-12-13
5.	1-2-3-5-4-7-8-9-12-13
6.	1-2-3-6-7-8-9-12-13
7.	1-2-3-4-7-8-10-13
8.	1-2-3-5-4-7-8-10-13
9.	1-2-3-6-7-8-10-13
10.	1-2-3-4-7-8-9-11-13
11.	1-2-3-5-4-7-8-9-11-13
12.	1-2-3-6-7-8-9-11-13
13.	1-2-3-4-7-8-9-12-13

Dengan menggunakan rumus *cyclomatic complexity metric*, dapat dihitung nilai kompleksitas dari program yang akan diuji **[GAL04]**. Perhitungannya adalah sebagai berikut:

E = 17

N = 13

V(G) = E – N + 2

V(G) = 17 – 13 + 2 = 6

Dari hasil pehitungan, nilai kompleksitas untuk *use case* mahasiswa adalah 6. Nilai tersebut termasuk kategori tidak terlalu sulit **.** Nilai kompleksitas yang ada adalah jumlah maksimum *path* yang dapat dibentuk. Jumlah set maksimum dari *path* yang ada:

No.	*Path*	*Edge* yang ditambahkan	Jumlah *Edge* yang ditambahkan
1.	1-2-3-4-7-8-9-11-13	1-2, 2-3, 3-4, 8-9, 9-11, 11-13,	9
2.	1-2-3-4-7-8-9-11-13	4-7, 7-8	4
3.	1-2-3-4-7-8-9-11-13	14-17, 17-18, 18-20	3
5.	1-2-3-5-4-7-8-	5-4	1

BAB III
PENUTUP

3.1 Kesimpulan

Strategi pengujian perangkat lunak dilakukan untuk memudahkan para perancang dalam menentukan keberhasilan *system* yang telah dikerjakan.

Test case merupakan suatu tes yang dilakukan berdasarkan pada suatu inisialisasi, masukan, kondisi ataupun hasil yang telah ditentukan sebelumnya. Segala produk perekayasaan, termasuk perangkat lunak, dapat diuji dengan dua cara, yaitu pengujian *white box* dan *black box*.

3.2 Saran

Saran yang dapat diberikan kepada pembaca adalah makalah ini agar menjadi informasi yang bermanfaat, serta dapat di kembangkan lagi agar menjadi lebih efektif dan lebih akurat dalam membangun sistem informasi booking tiket bioskop online.

DAFTAR PUSTAKA

NN. 2014. *Dokumen Test Case,* http://power.lecture.ub.ac.id/files/2014/11/Kelompok_3_ONick_Test_Case-dan-Screenshot-JUnit. diakses pada tanggal 24 Desember 2016.

NN. Tanpa Tahun. *Strategi Pengujian Perangkat* Lunak, http://wsilfi.staff.gunadarma.ac.id/Downloads/files/2204/Strategi+Pengujian+Perangkat+Luna k+mg+ke+8.pdf> diakses pada tanggal 24 Desember 2016.