KATA PENGANTAR

Segala puji bagi Allah, Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan buku ajar ini. Tak lupa juga mengucapkan shalawat serta salam semoga senantiasa tercurahkan kepada Nabi Besar Muhammad SAW, karena berkat beliau, kita mampu keluar dari kegelapan menuju jalan yang lebih terang.

Kami ucapkan juga rasa terima kasih kami kepada pihak-pihak yang mendukung lancarnya buku ajar ini, yaitu orang tua kami, rekan-rekan kami, pembimbing kami, dan masih banyak lagi yang tidak bisa kami sebutkan satu per satu.

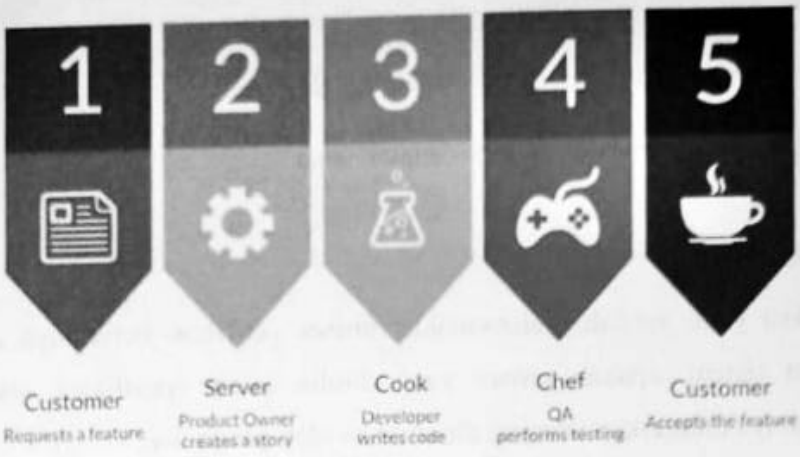
Adapun, buku ajar kami yang berjudul "Software Testing Simple" ini telah selesai kami buat secara semaksimal dan sebaik mungkin agar menjadi manfaat bagi pembaca yang membutuhkan informasi dan pengetahuan mengenai pengujian perangkat lunak untuk pemula.

Dalam buku ini, tertulis mengenai pengetahuan-pengetahuan secara umum terkait pengujian perangkat lunak yang relevan dengan mata kuliah kualitas perangkat lunak sebagai alternatif pegangan bagi mahasiswa dan dosen yang menempuh studi tersebut.

Kami sadar, masih banyak luput dan kekeliruan yang tentu saja jauh dari sempurna tentang buku ini. Oleh sebab itu, kami mohon agar pembaca memberi kritik dan juga saran terhadap karya buku ajar ini agar kami dapat terus meningkatkan kualitas buku.

Demikian buku ajar ini kami buat, dengan harapan agar pembaca dapat memahami informasi dan juga mendapatkan wawasan mengenai bidang sistem informasi manajemen serta dapat bermanfaat bagi masyarakat dalam arti luas. Terima kasih.

15 November 2022



Gambar 1.1 Gambaran sederhana tentang SDLC

B. Testing Definition

Untuk memastikan kualitas dari suatu aplikasi, testing harus dilakukan yaitu dengan cara menguji apakah sistem data yang dihasilkan sesuai dengan testing yang telah dilakukan. Tujuan dari testing ini sendiri antara lain, yaitu:

1. Memastikan aplikasi berjalan sebagaimana mestinya
2. Mendeteksi terjadinya error serta memvalidasi apakah sudah memenuhi keinginan user
3. Melakukan pengecekan/pengetesan entitas-entitas, termasuk software,untuk pemenuhan dan konsistensi dengan melakukan evaluasi hasil terhadap kebutuhan yang telah ditetapkan
4. Melakukan validasi untuk dapat melihat kebenaran sistem, apakah proses yang telah ditulis dalam spesifikasi adalah apa yang sebenarnya.
5. Mendeteksi adanya error, testing untuk mendeteksi kesalahan secara insentif, yaitu menentukan apakah suatu hal tersebut terjadi bilamana tidak seharusnya terjadi.

Terdapat dua tahapan proses perangkat lunak bebas bug dan memenuhi semua persyaratan, yaitu:

1. Verifikasi

Verifikasi adalah suatu pengecekan ataupun pengetesan entitas-entitas, termasuk aplikasi, untuk pemenuhan serta konsistensi dengan melakukan penilaian hasil terhadap kebutuhan yang telah ditetapkan.

"are we building the product right?" (menurut Boehm)

"apakah kita telah membuat produk dengan benar?"

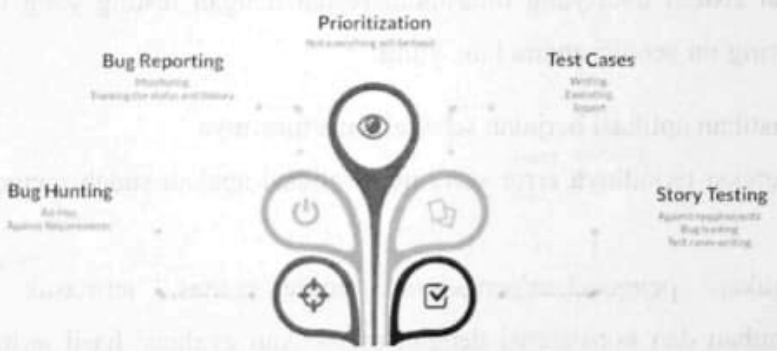
2. Validasi

Validasi yaitu setelah melaksanakan proses verifikasi berikutnya adalah melihat kebenaran sistem, apakah proses yang ditulis dalam spesifikasi adalah apa yang sesungguhnya diinginkan ataupun dibutuhkan oleh pengguna.

"are we building the right product?" (menurut Boehm)

"apakah kita telah membuat produk yang benar?"

Berikut 5 kegiatan utama yang akan dibahas pada *testing definition*, yaitu:



Gambar 1.2 Kegiatan-kegiatan utama pada pengujian

1. *Bug Hunting*, yaitu aktivitas untuk mencari bug dan ketika bug sudah ditemukan akan dilaporkan.
2. *Bug reporting*, yaitu pemantauan dalam melacak status dan riwayat, yang mana memerlukan tempat pusat dalam proses pemantauan.
3. *Prioritization*, yaitu memprioritaskan perbaikan pada kekurangan yang masuk akal saja karena tidak semuanya akan diperbaiki.
4. *Test cases*, yaitu menulis, mengeksekusi, dan melaporkan kasus yang akan diuji.

5. *Story testing*, yaitu aktivitas untuk melakukan beberapa pengujian terhadap persyaratan, sehingga dapat menemukan bug, melaporkan dan menulis kasus yang diuji.

C. Software Defect (Bug)

Bug adalah kesalahan dalam program komputer yang menyebabkan hasilnya tidak benar atau tidak terduga, atau berperilaku dengan cara yang tidak diinginkan. Berikut ini adalah beberapa jenis bug yang paling sering terjadi selama proses pengembangan *software*.

1. *Functional error*

Functional error merupakan sebuah kategori luas yang mencakup masalah yang terkait fungsionalitas sebuah program. *Bug* jenis ini bervariasi, mulai dari tombol yang tidak dapat di-klik hingga masalah pada kegunaan aplikasi itu sendiri.

2. *Performance defects*

Performance defects adalah kategori bug yang terkait dengan kecepatan, stabilitas, *response time*, dan penggunaan sumber daya dari sebuah *software*.

3. *Usability defects*

Usability defects adalah jenis *bug* yang menyebabkan pengguna tidak dapat memanfaatkan sebuah *software* secara maksimal. Jenis *bug* ini biasanya menyebabkan *software* sulit atau tidak nyaman untuk digunakan. Selain masalah pada kode *software*, *usability defects* juga dapat disebabkan oleh desain *user interface* (UI) yang terlalu rumit sehingga pengguna kesulitan menemukan fungsi yang mereka cari.

4. *Compatibility error*

Compatibility error merujuk pada masalah *software* yang tidak dapat berjalan sebagaimana mestinya dalam situasi tertentu. Pada umumnya, *compatibility error* akan muncul ketika dijalankan aplikasi lama pada sistem operasi yang lebih baru.

5. *Security error*

Security error merupakan jenis *bug* yang cukup berbahaya karena terkait langsung dengan sistem keamanan sebuah *software*. Sebuah bug dalam sistem keamanan *software*,

BAB II
TYPES OF TESTING

A. User Interface Testing

tulisan italic

Kerusakan/ketidaksempurnaan pada UI/design dapat dikategorikan dalam 5 kategori, yaitu:

1. *Layout*

   Kerusakan pada *layout* dapat terlihat pada ketidaksejajaran margin, ketidaksempurnaan model kotak IE6 (Internet Explorer 6), *overlapping*, kerusakan gambar, jarak lebar dan tinggi garis, ruang yang tidak diingankan diantara daftar item, jarak vertikal yang memiliki kecacatan.

2. *Font*

   Kerusakan pada *font* dapat terlihat dari tipe *font* yang berbeda, baik dari *family font* yang digunakan maupun ukuran *font* yang digunakan.

3. *Color*

   Kerusakan pada warna dapat terlihat dari tidak adanya keseragaman warna yang digunakan dan warna tombol yang tidak berfungsi sesuai dengan kebutuhan.

4. *Content*

   Kerusakan pada *content* dapat terlihat dari perbedaan besar kecilnya suatu ukuran *font*, tidak adanya keseragaman kata, kerusakan gambar serta kesalahan dalam ejaan kata.

B. Functional Testing

Functional testing adalah sistem yang diuji terhadap fungsional persyaratan atau spesifikasinya. Jenis pengujian ini juga memverifikasi bahwa setiap fungsi aplikasi telah beroperasi sesuai dengan kebutuhan, yang berfokus pada pengujian secara manual dan otomatis. *Functional testing* ini merupakan salah satu jenis pengujian metode *black box testing* di mana setiap fungsionalitas aplikasi diuji dengan memberikan suatu *input* untuk mengetahui reaksi aplikasi yang sebenarnya dan kemudian membandingkannya dengan hasil yang diharapkan sesuai dengan spesifikasi yang diberikan. Gambar 1.4 merupakan gambaran dari *black box testing*:

memori, efisiensi beban kerja, dan waktu respon perintah. Terdapat tiga faktor utama dalam memenuhi harapan/hasil dari *performance testing*, yaitu:

1. *Speed*: Melihat kecepatan sistem dalam memberikan respon dari setiap *request*.
2. *Scalability*: Menentukan berapa jumlah *load* maksimum yang dapat ditangani sistem.
3. *Stability*: menganalisis kondisi sistem saat diberikan beban yang bervariasi, apakah stabil atau tidak.

F. Usability Testing

Usability testing adalah metode yang digunakan untuk mengevaluasi seberapa mudah situs web menggunakan *User Experience* (UX). *Usability testing* dilakukan pada produk yang fokus pada identifikasi masalah kegunaan, mengumpulkan data kualitatif dan kuantitatif, dan menentukan kepuasan dari pengguna produk. Umumnya *usability test* mengevaluasi persyaratan fungsional dan kualitas dari *user interface*. *Usability test* ini juga memiliki hubungan antara pengguna dengan sistem yang dilakukan untuk menentukan apakah fungsi telah sama seperti yang diharapkan dan apakah *user interface* membuat sistem dapat mudah digunakan.

Pengujian ini sering dilakukan untuk mendapatkan hasil yang cepat dalam meningkatkan *interface* dan mengkoreksi kesalahan dalam komponen perangkat lunak. Salah satu contoh *usability box* adalah new away 2018, ketika semua orang dari negara bagian mendapatkan pesan peringatan hanya karena kesalahan seseorang meng-klik sesuatu yang salah dikomputer, karena masalah penggunannya yang mana memiliki dua tombol yang sama disamping satu sama lain. Oleh karena itu, lebih baik untuk menggunakan warna yang berbeda agar dapat membedakan antara satu tombol dengan tombol yang bersampingan.

G. Security Testing

Security testing yaitu jenis pengujian perangkat lunak yang dilakukan untuk mengidentifikasi kerentanan serta memastikan bahwa data dan sumber daya sistem di dalamnya sudah terlindungi dengan baik dari para penyusup. Pengujian ini dilakukan dengan tujuan untuk menemukan semua celah dan kelemahan sistem yang dapat mengakibatkan hilangnya informasi. Adapun tahapan-tahapan dari *security testing*, sebagai berikut:

BAB III
BUGS

A. How to Report Bugs?

Untuk melaporkan *bug* dapat menggunakan Jira, berikut 4 langkah utama *log bugs* dalam Jira:

Gambar 3.1 *Log bugs* pada Jira

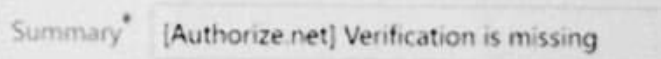
1. Type

Pilih jenis masalah *bug*, yang mana tergantung pada proyek yang telah disiapkan di JIRA. Jenis masalah dapat berisi nilai berikut: *bug, story, epic, task* dan sebagainya untuk melaporkan kerusakan.

Gambar 3.2 Tampilan *type* pada Jira

2. Summary

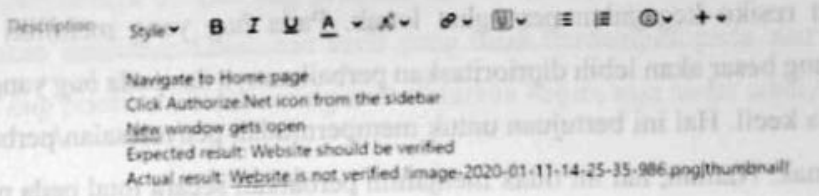
Menulis ringkasan singkat tentang apa dan dimana yang salah. Pesan yang dapat ditulis di *summary* yaitu [Authorize.net] Verification is missing.



Gambar 3.3 Tampilan *summary* pada Jira

3. *Description*

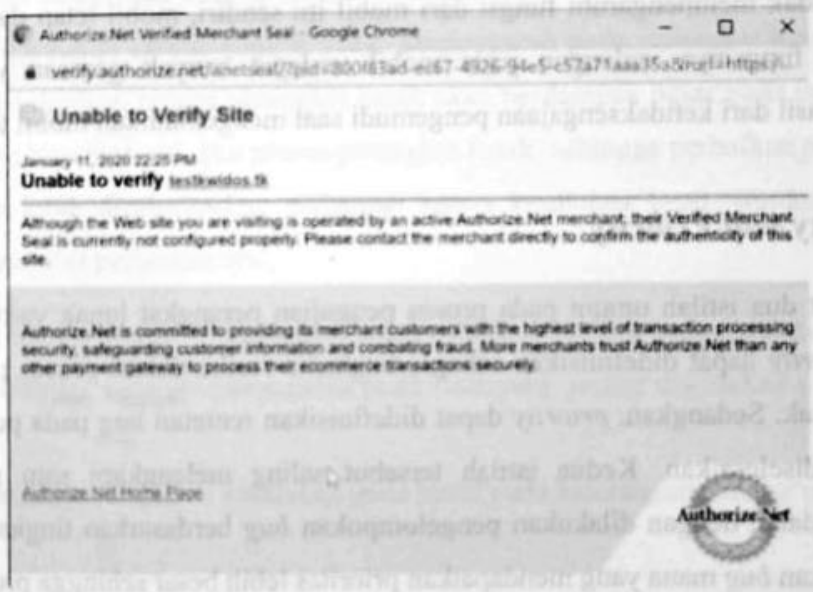
Deskripsi ini adalah bidang yang paling penting ketika membuat laporan *bug*, yang mana deskripsi ini sedikit lebih deskriptif dari dari ringkasan (*summary*).



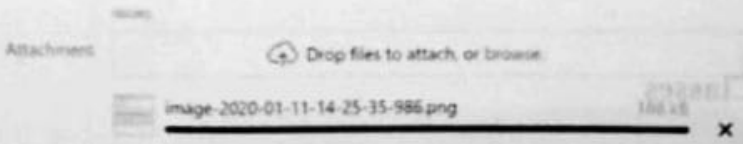
Gambar 3.4 Tampilan *description* pada Jira

4. *Screenshot*

Screenshot jendela Authorize.Net untuk lebih terpercaya dan aman. Setelah itu, masukkan file image di attachment.



Gambar 3.5 Tampilan jendela Authorize.Net pada Jira



Gambar 3.6 Tampilan *attachment* hasil screenshot pada Jira

B. Bug Triage

Pengelompokan *bug* pada proses pengujian kualitas perangkat lunak merupakan proses pencarian sejumlah *bug* pada perangkat lunak kemudian memilahnya berdasarkan tingkatan dampak atau resiko kegagalan perangkat lunak. Pada *bug* yang memiliki tingkat resiko kegagalan yang besar akan lebih diprioritaskan perbaikannya daripada *bug* yang tingkat resiko kegagalannya kecil. Hal ini bertujuan untuk mempermudah penyelesaian/perbaikan *bug* pada perangkat lunak. Namun, hal ini tidak menjamin perbaikan secara total pada perangkat lunak karena pada dasarnya *bug* pada perangkat lunak tidak mungkin dapat diselesaikan secara total.

Bug dengan resiko kegagalan kecil yang tidak mengganggu atau merusak alur proses perangkat lunak dapat dikesampingkan, karena kecacatan kecil pada perangkat lunak pasti akan selalu ada. Pernyataan tersebut dapat dianalogikan seperti goresan pada mobil. Goresan pada mobil tidak mempengaruhi fungsi dari mobil itu sendiri, mobil tetap dapat digunakan sebagaimana fungsinya walaupun pada mobil terdapat banyak goresan yang mungkin merupakan hasil dari ketidaksengajaan pengemudi saat mengemudikan mobil tersebut.

C. Severity and Priority

Terdapat dua istilah umum pada proses pengujian perangkat lunak yaitu *severity* dan *priority*. *Severity* dapat didefinisikan sebagai tingkatan resiko dalam proses pengembangan perangkat lunak. Sedangkan, *priority* dapat didefinisikan rentetan bug pada perangkat lunak yang harus diselesaikan. Kedua istilah tersebut saling melengkapi satu sama lainnya, maksudnya adalah dengan dilakukan pengelompokan *bug* berdasarkan tingkat resiko maka dapat ditentukan *bug* mana yang mendapatkan prioritas lebih besar sehingga proses perbaikan pada *bug* dapat lebih terorganisir dan tepat.

D. Priority Classes

Priority pada pengujian perangkat lunak terbagi menjadi tiga kelompok, yaitu *low, medium* dan *high priority*. *Bug* yang berdampak besar terhadap alur proses perangkat lunak bahkan bisa merusak alur proses perangkat lunak adalah *bug* yang sangat diprioritaskan perbaikannya atau bisa disebut sebagai *bug* dengan prioritas tinggi (*high priority*). Setiap *bug* dengan prioritas tinggi harus dilakukan perbaikan/penyelesaian secepat mungkin agar proses pengembangan dapat dilanjutkan dan tidak terhambat. Sedangkan *bug* dengan priotitas sedang