SOFTWARE TESTING

ERIC AGUSTIAN S.KOM

V 1.5

PENDAHULUAN

- Ketika kita membeli buah pir, kita bisa melihat kualitasnya dari bentuk, kematangan, dan tidak ada memar yang terlihat.
- Tetapi kita dapat mengetahui bahwa kualitas pir benar-benar bagus setelah kita memakan pir tersebut.
- Walaupun bentuknya terlihat bagus bisa jadi rasanya asam atau ada cacing didalamnya.

- Hal ini juga berlaku untuk produk software.
- Mungkin Ketika kita membuka halaman website awalnya baik-baik saja, tapi ketika kita scroll down, atau berpindah ke halaman lain, bisa jadi terdapat desain yang cacat atau error,

AKIBAT KUALITAS SOFTWARE YANG BURUK

- Kualitas Software self-driving autopilot yang buruk mungkin dapat menyebabkan **kecelakaan.**
- Satu Error di system EHR (Electronic Health Record) mungkin dapat menyebabkan **nyawa pasien dalam bahaya**
- Performance Issue di website E-Commerce mungkin dapat menyebabkan **kerugian besar.**

- Contoh Kasus Nyata
- Starbucks terpaksa memberikan minuman gratis karena kegagalan di software POS.
- Pesawat Militer F-35 yang gagal mendetek target karena kegagalan di software radar.

CARA MEMASTIKAN SOFTWARE BERJALAN SESUAI

- **Kualitas Software** : Tingkat kesesuaian dengan **requirement** (persyaratan) dan harapan yang secara eksplisit dan implisit.
- Harapan Eksplisit dan Implisit sesuai dengan dua **Level Pengujian** Kualitas Software:
  - **Fungsional** : Kesesuain produk sesuai dengan persyaratan fungsional (eksplisit) dan spesifikasi desain. Aspek ini berfokus pada penggunaan dari perspektif sudut pandang pengguna. Misalnya : fitur-fiturnya, performancenya, kemudahan pengguna, dan tidak ada yang cacat.
  - **Non-Fungsional** : Karakteristik & Arsitektur internal system, yaitu persyaratan structural (implisit). Hal ini meliputi kemudahan pemeliharaan kode, pemahaman, efisiensi, dan security.

QA, QC, SOFTWARE TESTING

- **QA (Quality Assurance)** adalah **serangkaian proses** sistematis untuk menentukan apakah suatu produk atau jasa memenuhi **syarat** yang ditentukan.
- **QC (Quality Control**) adalah proses untuk memastikan bahwa produk yang dihasilkan sesuai dengan **persyaratan** dan bebas dari **defect.**
- **Software Testing** adalah adalah aktivitas dasar yang bertujuan untuk mendeteksi dan memecahkan masalah teknis dalam source code software, serta menilai penggunaan produk secara keseluruahan**.**

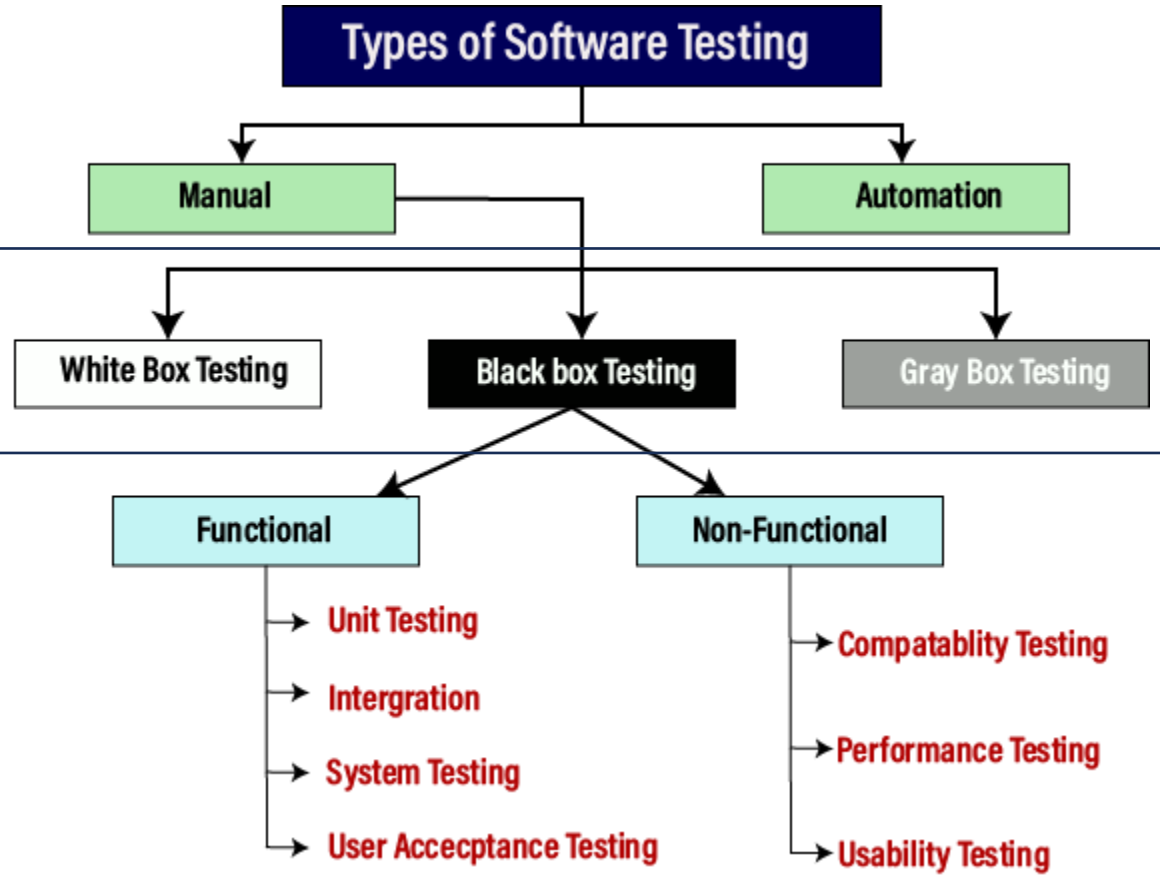
HUBUNGAN QA, QC & SOFTWARE TESTING

QA, QC and Testing in software development process

PERBEDAAN QA, QC & SOFTWARE TESTING

	QA	QC	Testing
Purpose	Setting up adequate processes, introducing the standards of quality to prevent the errors and flaws in the product	Making sure that the product corresponds to the requirements and specs before it is released	Detecting and solving software errors and flaws
Focus	Processes	Product as a whole	Source code and design
What	Prevention	Verification	Detection
Who	The team including the stakeholders	The team	Test Engineers, Developers
When	Throughout the process	Before the release	At the testing stage or along with the development process

TIPE-TIPE SOFTWARE TESTING



Jenis Pengujian

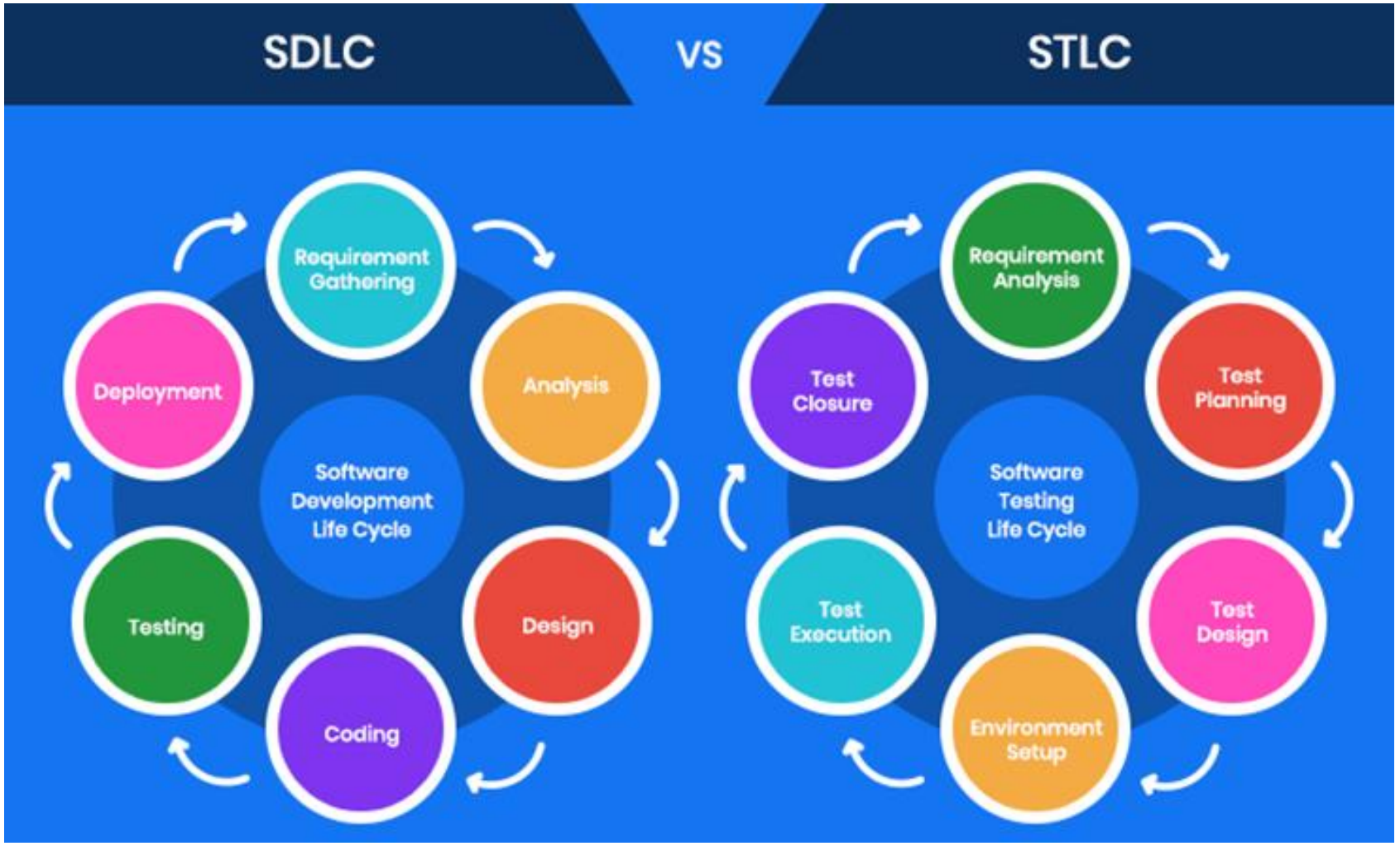
Metode Pengujian

Level Pengujian

SOFTWARE DEVELOPMENT

- SDLC
- STLC
- Project Management
- Desain UI/UX
- Standar Kualitas Internasional

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)
SOFTWARE TESTING LIFE CYCLE (STLC)



SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC) SOFTWARE TESTING LIFE CYCLE (STLC)

- Software Development Life Cycle (SDLC) adalah serangkaian proses sistematis pembuatan software untuk mengembangkan aplikasi dan website dengan kualitas terbaik, biaya yang efisien, dan waktu yang efektif.
- Software Testing Life Cycle (STLC) merupakan sebuah siklus pengujian yang dirancang dengan baik secara sistematis yang dilakukan untuk memastikan kualitas produk.

PERAN UMUM YANG TERLIBAT DENGAN SDLC (1)

- **Stakeholders (Pemangku Kepentingan)**
  - Pihak-pihak yang memiliki kepentingan dalam proyek, seperti manajemen eksekutif, pemilik perusahaan, dan investor.
- **Business Analyst (Analisa Bisnis)**
  - Bertugas untuk menganalisis kebutuhan bisnis dan merumuskan persyaratan perangkat lunak.
  - Berinteraksi dengan pengguna dan pemangku kepentingan untuk memahami kebutuhan mereka dan mengubahnya menjadi dokumen kebutuhan yang jelas
- **Product Owner (Pemilik Produk)**
  - Bertanggung jawab untuk mewakili kebutuhan pengguna dan pemangku kepentingan.
  - Berperan dalam merumuskan visi produk, mengatur prioritas fitur, dan memberikan masukan berdasarkan kebutuhan pasar.

PERAN UMUM YANG TERLIBAT DENGAN SDLC (II)

- **Project Manager (Manajer Proyek)**
  - Bertanggung jawab untuk mengelola proyek secara keseluruhan.
  - Merencanakan jadwal, mengalokasikan sumber daya, memantau kemajuan, dan mengelola risiko.
- **Developer (Pengembang)**
  - Membuat source code perangkat lunak berdasarkan rancangan dan spesifikasi yang telah ditetapkan.
  - Mengembangkan komponen-komponen perangkat lunak dan memastikan fungsionalitas yang tepat.
- **Quality Assurance Tester**
  - Melakukan pengujian untuk memastikan kualitas dan kinerja perangkat lunak.
  - Mengidentifikasi bug, masalah kinerja, dan menguji fungsionalitas sesuai spesifikasi.

TAHAP SDLC 1 : REQUIREMENT GATHERING & ANALYSIS

- Peran yang terlibat : **Stakeholder**, **Project Manager**, **Business Analyst, Product Owner**, **QA Lead, Developer Lead**
- Pada tahap ini tim developer akan mengumpulkan dan menganalisis kebutuhan user, untuk dibentuk menjadi requirement atau persyaratan dan standar yang diinginkan. Requirement tadi akan disusun dalam dokumentasi SRS atau BRD.
- SRS (Software Requirement Specificasions) adalah dokumen yang berisi tentang semua kebutuhan fungsional dan non fungsional serta teknik dari software yang akan dibangun atau dikembangkan.
- BRD (Business Requirement Document) adalah dokumen yang berisi tentang spesifikasi fungsional software dan semua kebutuhan dan syarat yang diajukan oleh user.

TAHAP SDLC 1 : REQUIREMENT GATHERING & ANALYSIS

- Peran QA setelah atau saat Tahap Requirement Gathering & Analysis
  - Mengumpulkan semua requirement
  - Review, Analisis, dan verifikasi dokumen SRS (Software Requirement Specification)
  - Jika diperlukan, QA juga bisa menambahkan saran untuk perbaikan.
  - Membuat Test Plan (QA Lead)

TAHAP SDLC 2 : DESIGN

- Peran yang terlibat : **Project Manager, Product Owner**, **Designer, QA Lead**
- Pada tahap ini designer mempersiapkan dokumen desain sistem dan software, sesuai dengan dokumen spesifikasi kebutuhan tahap sebelumnya.
- Adapun desain yang dibuat oleh tim UI/UX Designer adalah : Desain arsitektur, user interface (UI), diagram alur, diagram proses, dll.
- Tidak hanya bagi desainer, tahap ini juga digunakan oleh tim produk untuk : menentukan prioritas kerja, Menyusun produk roadmap, meminta persetujuan dari stakeholder.

TAHAP SDLC 2 : DESIGN

- Peran QA setelah atau saat tahap Design
  - Mengumpulkan flowchart dan mockup
  - Membuat Test Scenario dan Test Case
  - Review Design

TAHAP SDLC 3 : DEVELOPMENT / IMPLEMENTATION

- Peran yang terlibat : **Project Manager, Developer**
- Setelah tahap perancangan sistem selesai, kita akan masuk ke fase terpanjang dalam SDLC yaitu pengodingan.
- Pada fase ini, tim developer mulai mengimplementasikan script bahasa pemrograman atau ngoding, untuk membangun keseluruhan sistem.

TAHAP SDLC 3 : DEVELOPMENT / IMPLEMENTATION

- Peran QA setelah atau saat tahap development / implementation
  - Mengajukan test scenario dan test case ke tim Developer dan Project Manager.
  - Jika terdapat masalah yang muncul, QA juga bisa jump in untuk membantu sesuai kebutuhan tim Developer
  - Environment Setup, menyiapkan perangkat yang akan digunakan untuk testing (Misal HP Android, emulator, atau browser khusus)

TAHAP SDLC 4 : TESTING

- Peran yang terlibat : **Project Manager, QA Tester, Developer**
- Tahap pengujian aplikasi atau website untuk memastikan apakah produk kita sudah berjalan dengan baik sesuai kebutuhan user.
- Selama fase ini, QA mungkin akan menemukan beberapa bug / defects, eror, freeze yang nantinya mereka komunikasikan kepada developer.

TAHAP SDLC 4 : TESTING

- Peran QA setelah atau saat tahap testing
  - Menulis dan Mengetes sesuai Test Case dengan melakukan penelusuran secara detail di setiap bagian hingga bagian terkecil
  - Menulis Bug Report, termasuk langkah-langkah reproduksi dari kesalahan yang ditemukan, lampiran screenshot atau bukti, dan detail lainnya. Bukan hanya deskripsi singkat tentang masalah yang ditemukan.
  - Melacak update terbaru

TAHAP SDLC 5 : DEPLOYMENT

- Peran yang terlibat : **Project Manager, Deployment Team**
- Pada tahap ini setelah tim menyelesaikan issues atau memperbaiki bugs dan defects, aplikasi atau website kita siap dirilis ke pasar untuk digunakan oleh user.

TAHAP SDLC 5 : DEPLOYMENT

- Peran QA sebelum tahap deployment
  - Smoke Testing
  - Sanity Testing
  - Regression Testing
  - UAT Testing

TAHAP SDLC 6 : MAINTENANCE

- Peran yang terlibat : **Project Manager, Product Owner, QA Tester, Developer**
- Setelah deployment dan user mulai menggunakan produk yang telah dikembangkan, setidaknya akan ada 3 aktivitas berikut yang terjadi:
  - Perbaikan bug: bug dilaporkan karena beberapa skenario yang mungkin tidak diuji sama sekali
  - Upgrade: Upgrading aplikasi ke versi software yang terbaru
  - Enhancement: Menambahkan beberapa fitur baru ke dalam perangkat lunak yang ada

TAHAP SDLC 6 : MAINTENANCE

- Peran QA sebelum saat tahap maintenance
  - QA harus tetap stand by untuk melakukan pengujian lagi jika suatu saat produk yang sudah di luncurkan mengalami masalah.

PROJECT MANAGEMENT

- **Project Management** adalah Metode perencanaan dan pengelolaan sumber daya untuk menyelesaikan sebuah proyek
- Contoh Project Management adalah Waterfall, Critical Path Method, Critical Chain Project Management, Agile, Scrum, Kanban

PROJECT MANAGEMENT MODEL WATERFALL

- Waterfall adalah metode manajemen proyek tradisional yang mengikuti urutan linear dari tahap perencanaan hingga peluncuran produk.
- Alur Model Waterfall terdiri dari:
  - Requirement
  - Design
  - Implementation
  - Testing
  - Maintenance

PROJECT MANAGEMENT MODEL AGILE

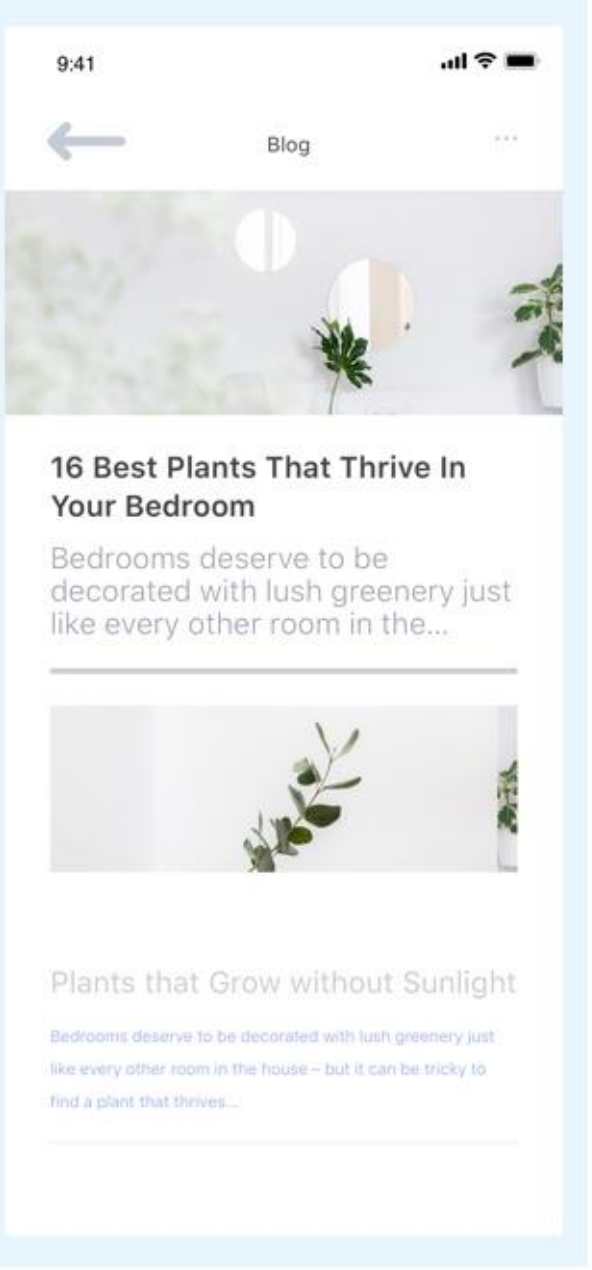
- Agile adalah metode manajemen proyek yang memprioritaskan adaptasi dan kolaborasi antara tim untuk mencapai tujuan bisnis.
- Alur Model Agile terdiri dari:
  - Product Planning : merumuskan visi produk, membuat daftar kebutuhan (product backlog), dan mengidentifikasi prioritas.
  - Sprint Planning : Pada awal setiap iterasi yang disebut "sprint", tim memilih sejumlah item dari backlog untuk dipecah menjadi tugas yang lebih kecil.
  - Sprint Execution : Tim melaksanakan tugas-tugas yang telah ditentukan dalam sprint planning.
  - Daily Standups : Tim mengadakan pertemuan pendek setiap hari untuk berbagi perkembangan, masalah, dan rencana ke depan
  - Sprint Review : Setelah selesai sebuah sprint, tim memperlihatkan hasil kerja kepada stake holder dan meminta feedback.
  - Sprint Retrospective : Tim merenungkan proses kerja selama sprint yang baru saja selesai.

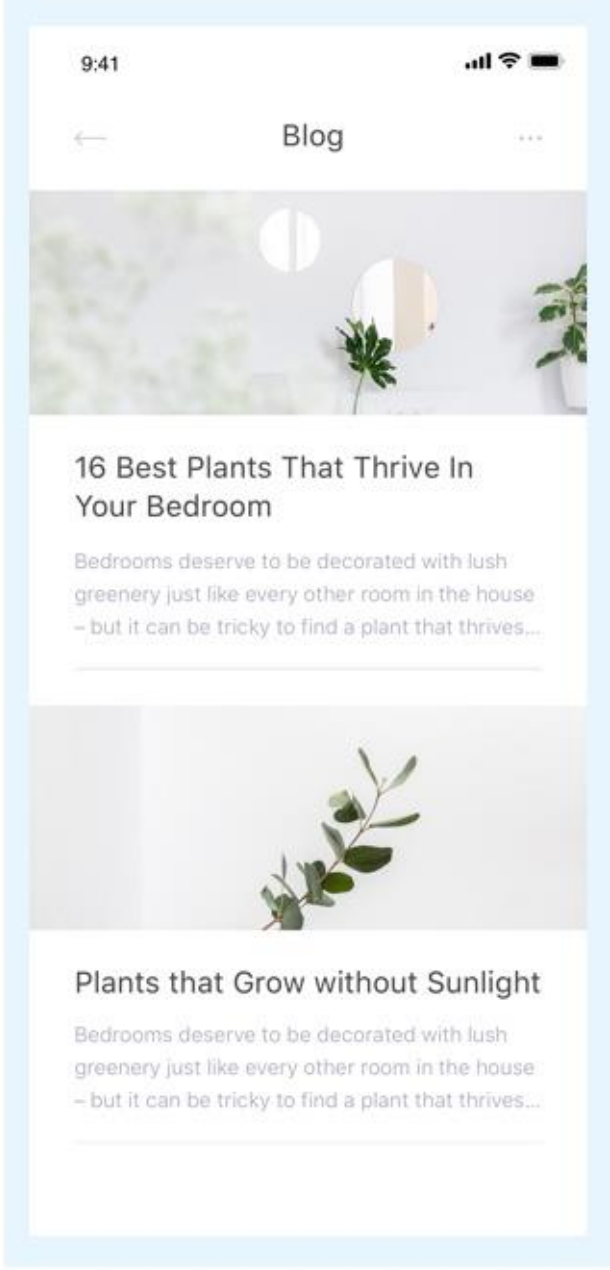
DESAIN UI & UX

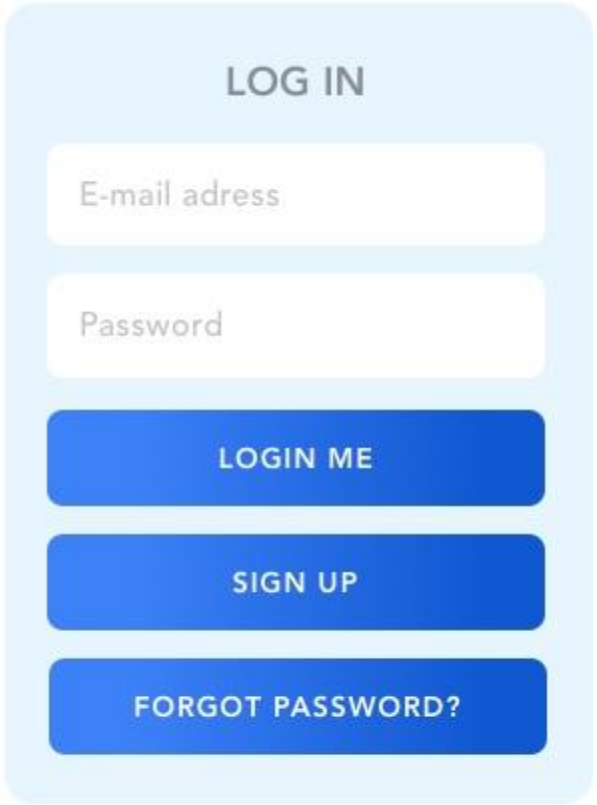
- UI (User Interface) adalah bagian visual dari website, aplikasi, software, atau hardware yang menentukan bagaimana seorang pengguna berinteraksi dengan produk tersebut.
- UX (User Experience) adalah bagaimana seorang pengguna berinteraksi dengan sebuah produk, khususnya produk digital.

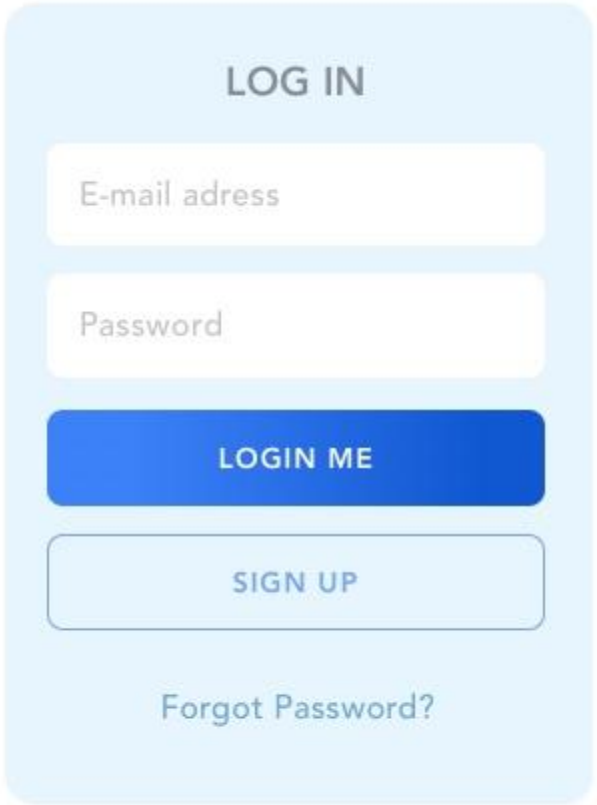
PERBANDINGAN UI & UX

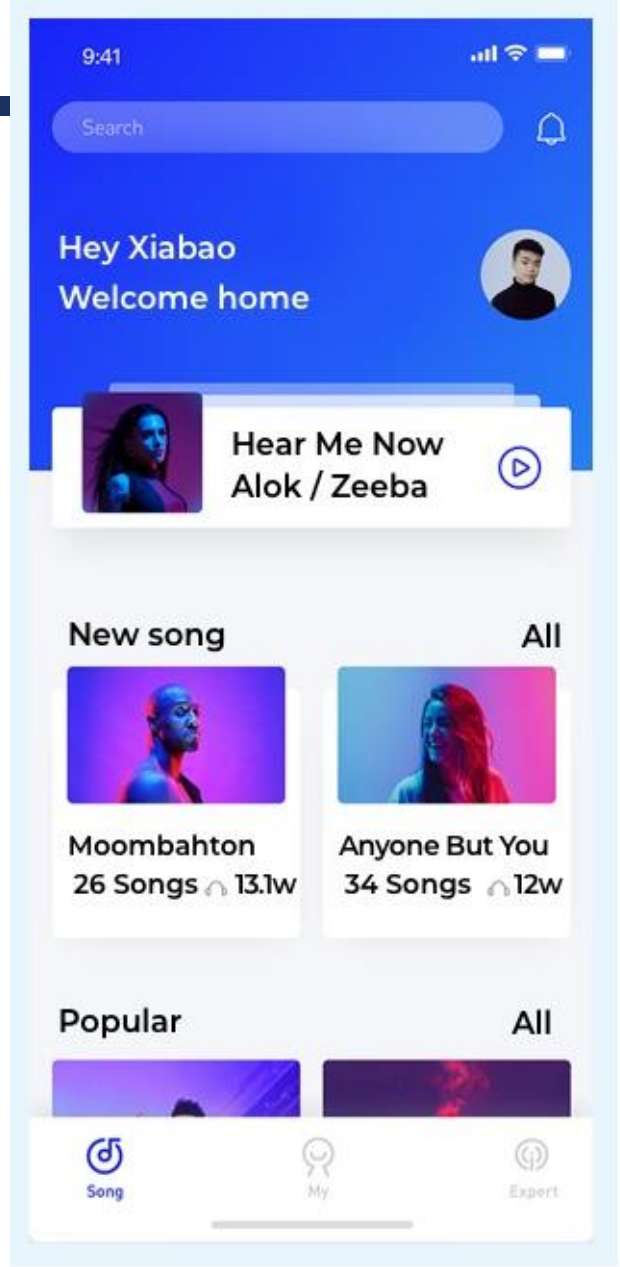
	UI (User Interface)	UX (User Experience)
Tujuan	Membuat tampilan lebih menarik	Memberikan kenyamanan saat memakai produk.
Fokus	Kesan pertama pengguna terhadap keindahan tampilan produk pada ukuran layer yang beragam	Kepuasan dan pengalaman pengguna ketika menggunakan produk.
Pre-Building	Desain mockup	Desain wireframe dan prototype
Elemen	Warna, gambar, video, animasi, tipografi, button, dan visual interaksi lainnya.	Usability, fitur-fitur, struktur desain, interaction design, navigasi, copywriting, hingga branding
Skill yang dibutuhkan	Desain grafis, desain branding dan creative thinking.	Riset, analytical thinking, problem solving, critical thinking, dan creative thinking

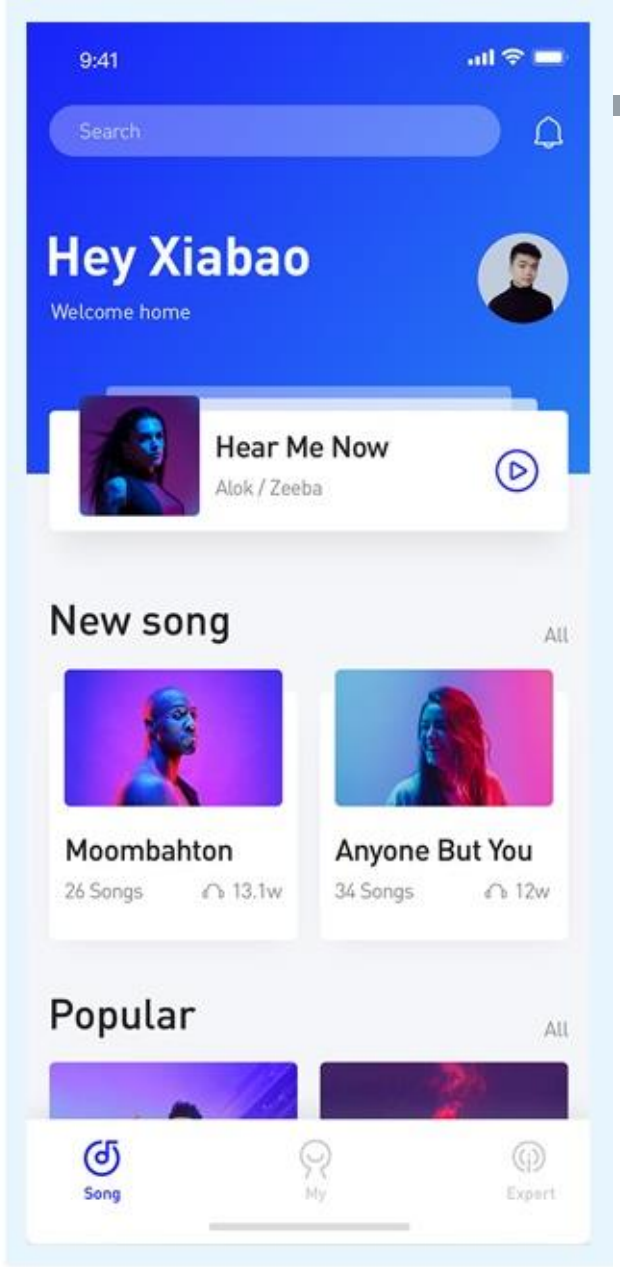

9:41
Blog
16 Best Plants That Thrive In Your Bedroom
Bedrooms deserve to be decorated with lush greenery just like every other room in the...
Plants that Grow without Sunlight
Bedrooms deserve to be decorated with lush greenery just like every other room in the house – but it can be tricky to find a plant that thrives...

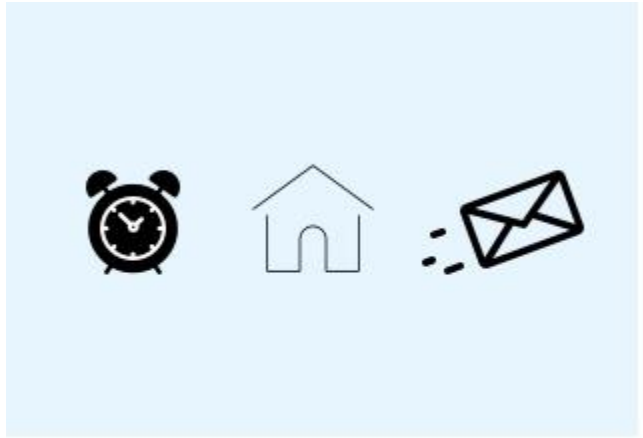

9:41
Blog
16 Best Plants That Thrive In Your Bedroom
Bedrooms deserve to be decorated with lush greenery just like every other room in the house – but it can be tricky to find a plant that thrives...
Plants that Grow without Sunlight
Bedrooms deserve to be decorated with lush greenery just like every other room in the house – but it can be tricky to find a plant that thrives...

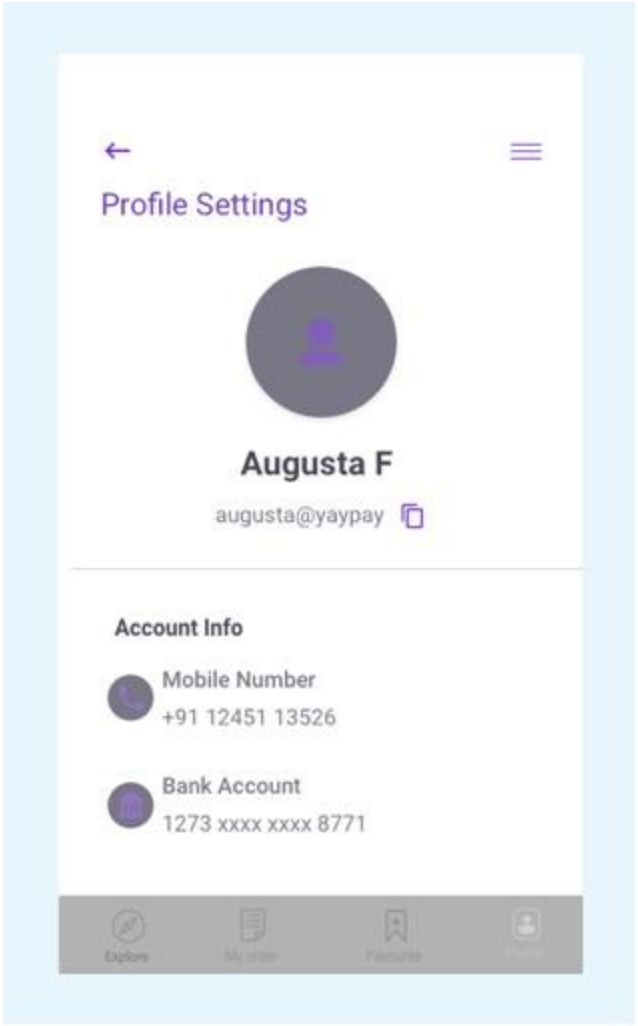

LOG IN
E-mail adress
Password
LOGIN ME
SIGN UP
FORGOT PASSWORD?

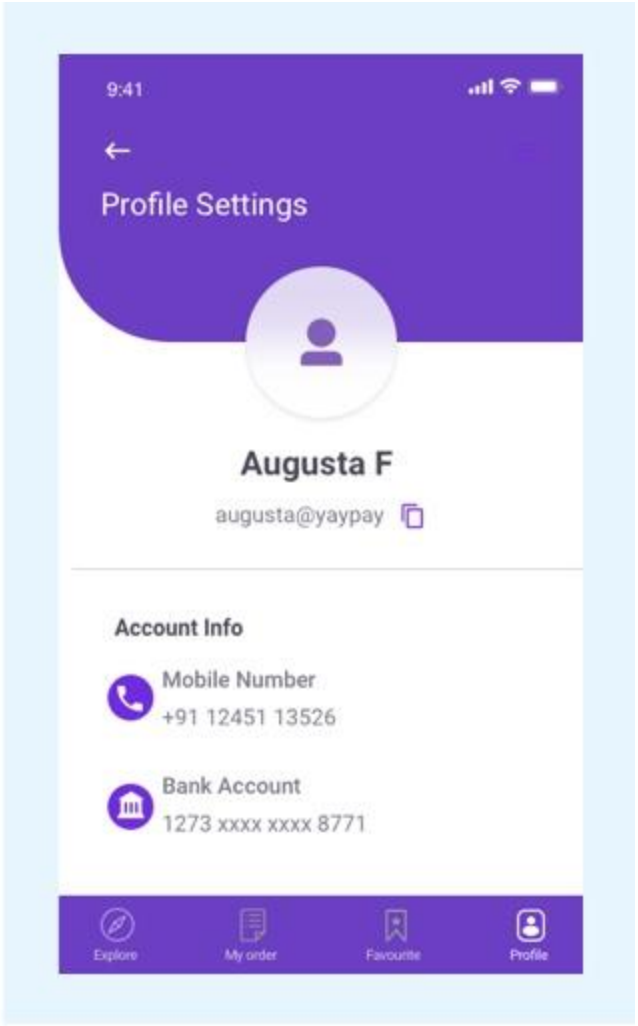

LOG IN
E-mail adress
Password
LOGIN ME
SIGN UP
Forgot Password?

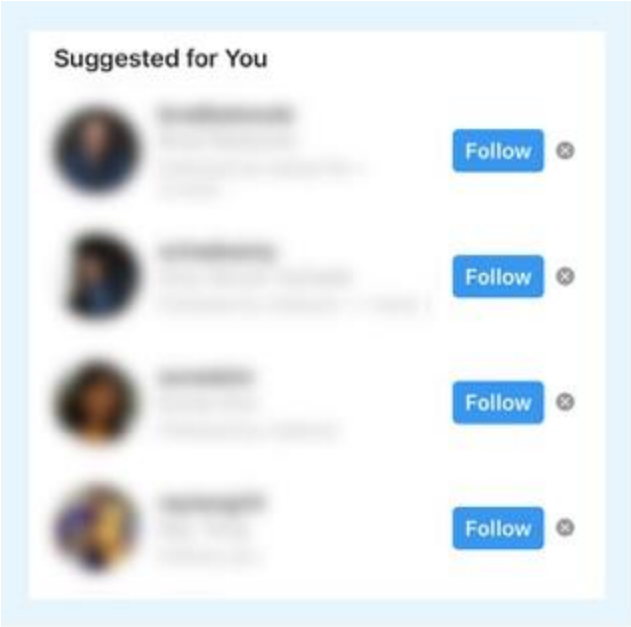

9:41
Search
Hey Xiabao
Welcome home
Hear Me Now
Alok / Zeeba
New song
All
Moombahton
26 Songs 13.1w
Anyone But You
34 Songs 12w
Popular
All
Song
My
Expert

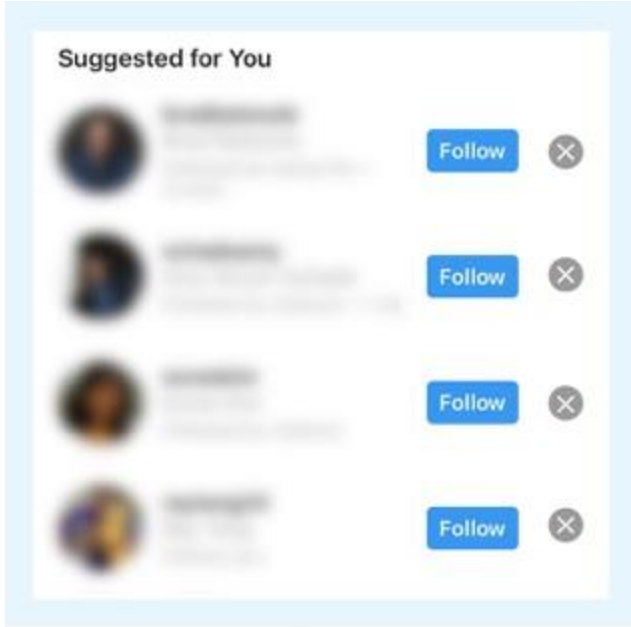

9:41
Search
Hey Xiabao
Welcome home
Hear Me Now
Alok / Zeeba
New song
All
Moombahton
26 Songs 13.1w
Anyone But You
34 Songs 12w
Popular
All
Song
My
Expert




Profile Settings
Augusta F
augusta@yaypay
Account Info
Mobile Number
+91 12451 13526
Bank Account
1273 xxxx xxxx 8771


9:41
Profile Settings
Augusta F
augusta@yaypay
Account Info
Mobile Number
+91 12451 13526
Bank Account
1273 xxxx xxxx 8771
Explore
My order
Favourite
Profile


Suggested for You
Follow
Follow
Follow
Follow


Suggested for You
Follow
Follow
Follow
Follow

INTERNATIONAL STANDARD FOR INFORMATION TECHNOLOGY (IT)

- Standar internasional yang digunakan sebagai patokan untuk menguji kualitas dari sebuah produk/teknologi/software berbasis sistem informasi
- ISO 25010 : 2011 merupakan standar terbaru dan relevan untuk menguji sistem informasi yang akan dikembangkan. Sekaligus menjadi tolak ukur analisa kualitas sebuah perangkat lunak (software)
- Menggantikan standar sebelumnya yaitu ISO 9126

INTERNATIONAL STANDARD FOR INFORMATION TECHNOLOGY (IT)

- Kriteria ISO 9126 untuk **External & Internal Quality**:
  1. Functionality (Fungsionalitas)
  2. Reliability (Keandalan)
  3. Usability (Kegunaan)
  4. Eficiency (Efisiensi)
  5. Maintainability (Pemeliharaan)
  6. Portability (Portabilitas)
- Kriteria ISO 9126 untuk **Quality in Use**:
  1. Effectiveness (Efektivitas)
  2. Productivity (Produktifitas)
  3. Safety (Keselamatan)
  4. Satisfication (Kepuasan)

INTERNATIONAL STANDARD FOR INFORMATION TECHNOLOGY (IT)

- **Software Product Quality Characteristics** berdasarkan ISO 25010 : 2011 meliputi:
  1. Functional Suitability (Kesesuaian Fungsi)
  2. Reliability (Keandalan)
  3. Performance Efficiency (Efisiensi Kinerja)
  4. Operability (Operasional)
  5. Security (Keamanan)
  6. Compatibility (Kompatibilitas)
  7. Maintainability (Pemeliharaan)
  8. Transferability (Transferabilitas)

JENIS SOFTWARE TESTING

- Manual Test
- Automation Test

DEFINISI
MANUAL TEST & AUTOMATION TEST

- Manual Test adalah tipe software testing test case dijalankan secara manual tanpa bantuan alat automation.
- Automation Test adalah Teknik software testing yang dilakukan dengan software automation testing untuk menjalankan test case / test case suite.

MANUAL TEST VS AUTOMATION TEST > PRO'S

MANUAL TEST

- Semua bisa melakukan test
- Cara paling mudah untuk meningkatkan kualitas
- Cocok untuk orang yang tidak memiliki latar belakang developer (programmer)
- Fokus pada work-flow utama dari produk
- Memungkinkan adanya faktor **human-observation** (**visual feedback**) untuk meningkatkan customer experience

AUTOMATION TEST

- Siklus test berlangsung cepat
- Mampu mengidentifikasi lebih banyak defect dalam waktu yg relatif singkat
- Menghemat pengeluaran biaya untuk skala project besar dan pengetesan dilakukan secara berulang kali (repetition)
- Cocok disandingkan dengan metodologi Agile Development
- Mudah untuk fokus pada semua jenis work-flow yang memungkinkan

MANUAL TEST VS AUTOMATION TEST > CON'S

MANUAL TEST

- Bisa saja tidak mencakup semua test case
- Bisa saja tidak mengidentifikasi semua bugs
- Kualitas produk lebih rendah dikarenakan semakin banyaknya defect yang ditemukan
- Akurasi kurang dipengaruhi oleh faktor human error
- Time consuming, dan hanya cocok untuk test yg dilakukan sekali atau dua kali saja

AUTOMATION TEST

- Test case (script) dibuat oleh scripter (pembuat script testing)
- Membutuhkan manual test case yang harus ditulis terlebih dahulu baru kemudian bisa dilakukan otomatisasinya
- Membutuhkan platform automation testing
- Cost effectiveness rendah jika proses pengujian hanya dilakukan sekali-dua kali saja (skala project kecil)
- Tidak adanya faktor human-observation

KONSEP SOFTWARE TESTING

- Defect & Bug
- Severity & Priority
- Black Box & White Box Test
- Endurance & Stress Test
- Monkey & Gorilla Test
- Perspektif Data Quality

KONSEP SOFTWARE TESTING

DEFECT

- Defect adalah variasi atau penyimpangan dari business requirement aslinya
- Defect = cacat terhadap kualitas dari software yang dihasilkan
- Hasil akhir dari software melenceng atau tidak sesuai dengan spesifikasi asli dari software tersebut

Wajib Untuk Diperbaiki!!!

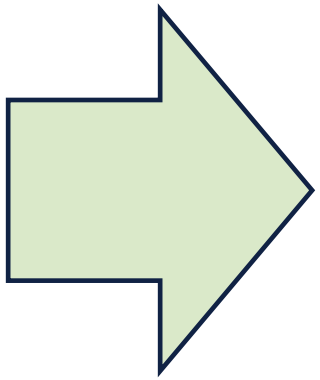
BUG

- Bug adalah sebuah konsekuensi atau dampak dari sebuah kesalahan pada coding
- Umumnya disebabkan oleh kesalahan coding yang dilakukan oleh developer software (individu)

KONSEP SOFTWARE TESTING

JENIS - JENIS DEFECT

- Cacat pada Logic
- Cacat pada Syntax (code)
- Cacat pada operasi Aritmatika
- Cacat Antar-Muka (UI/UX)
- Cacat pada Multi-Threading
- Cacat pada Performa
- Cacat pada System/Manufacturing Process



Bugs pada code

KONSEP SOFTWARE TESTING

SEVERITY

- Severity adalah derajat dampak yang dihasilkan dari defect yang terdapat dalam proses development atau pengoperasian komponen sebuah aplikasi yang sedang diuji
- Dampak atau efek yang dihasilkan diukur dalam derajat severity dengan ukuran tertentu

PRIORITY

- Priority didefinisikan sebagai urutan defect yang harus diperbaiki
- Semakin tinggi level priority, semakin cepat defect harus diselesaikan
- Defect yang menyebabkan software **tidak dapat digunakan** diberikan prioritas **lebih tinggi** dibandingkan defect yang menyebabkan kegagalan fungsi pada software

KONSEP SOFTWARE TESTING

SEVERITY LEVEL

- **Critical**. Mengindikasikan shut-down total dari proses, tidak bisa melanjutkan proses berikutnya (force close/terminate)
- **Major**. Tingkat tinggi, bisa menyebabkan sistem collapse, akan tetapi beberapa bagian dari system masih bisa berfungsi
- **Medium/Moderate**. Menyebabkan beberapa kejadian yang tidak diharapkan, akan tetapi system masih tetap berfungsi
- **Low/Non Critical**. Tidak menyebabkan kerusakan/gangguan dari system
- **Cosmetic**. Bersifat tampilan/pemanis (UI/UX)

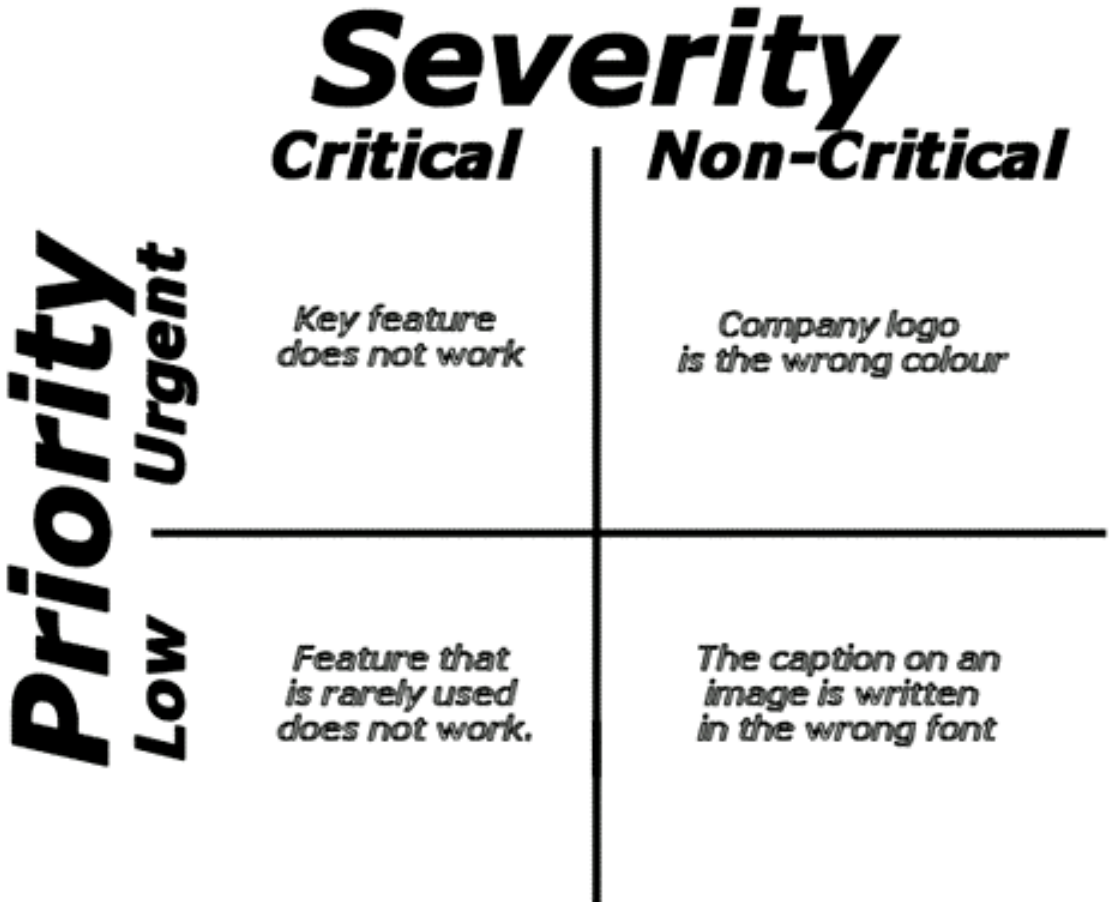
PRIORITY LEVEL

- **Low**. Defect yang ada bersifat mengganggu, tapi proses perbaikan bisa dilakukan setelah defect yang lebih berbahaya selesai diperbaiki
- **Medium**. Defect yang ada dapat diperbaiki selama proses development berlangsung, dan bisa menunggu hingga rilis versi baru dibuat
- **High/Urgent**. Defect yang ada harus diperbaiki sesegera mungkin karena berakibat pada gangguan system sehingga tidak bisa digunakan sebelum defect berhasil diperbaiki

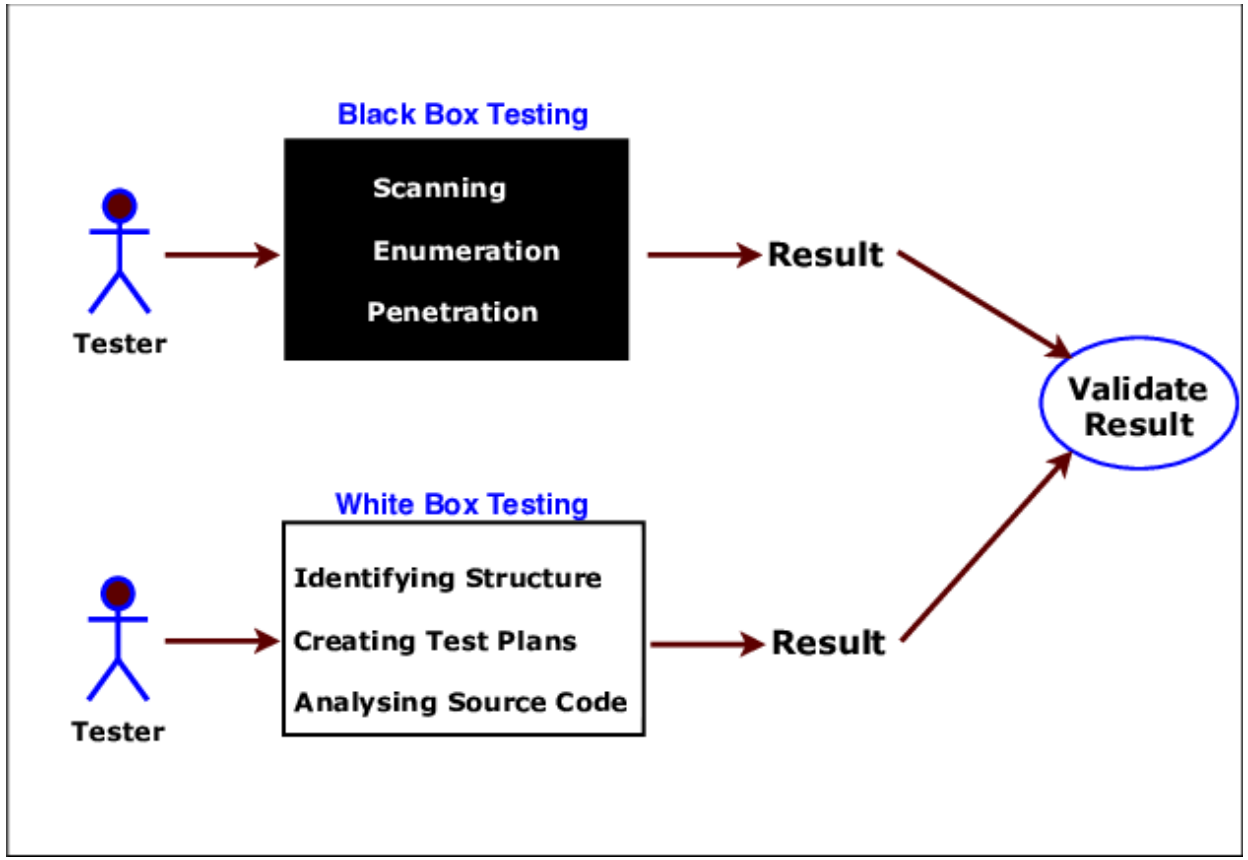
KONSEP SOFTWARE TESTING

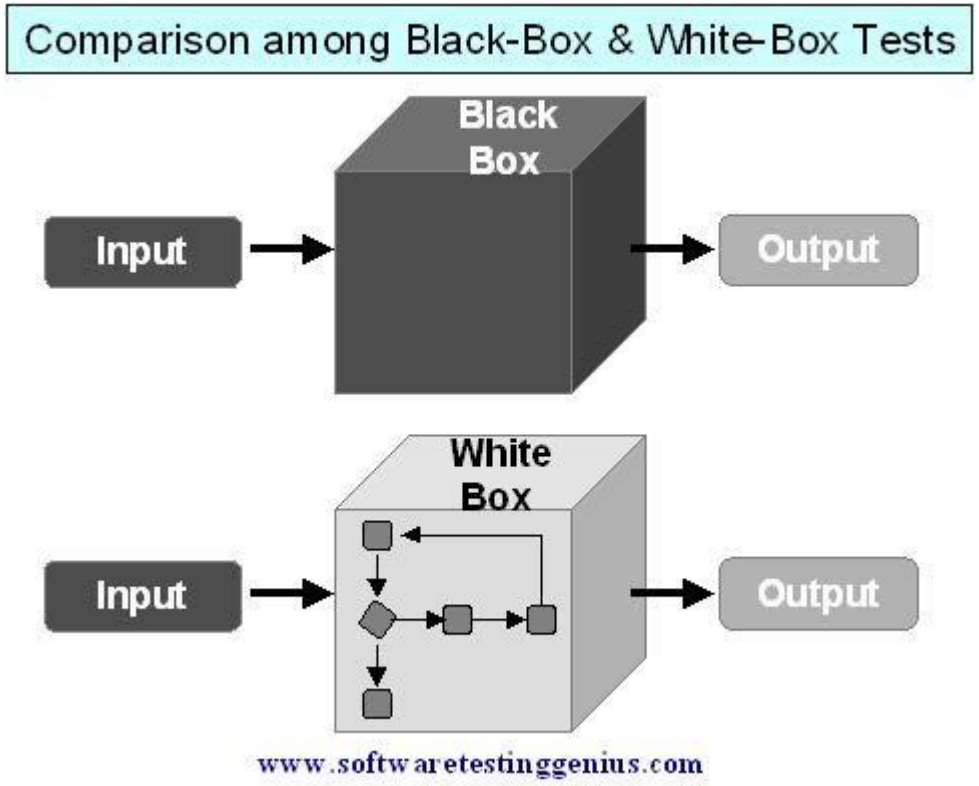
SEVERITY LEVEL

PRIORITY LEVEL



KONSEP SOFTWARE TESTING





KONSEP SOFTWARE TESTING

WHITE BOX TESTING

- White Box Testing adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian.
- Pengujian dilakukan berdasarkan bagaimana suatu software menghasilkan output dari input. Pengujian ini dilakukan berdasarkan kode program.
- Disebut juga struktural testing atau glass box testing

BLACK BOX TESTING

- Black Box Testing adalah pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari software.
- Dianalogikan seperti kita melihat suatu kotak hitam, kita hanya bisa melihat penampilan luarnya saja, tanpa mengetahui ada apa dibalik kotak hitam tersebut.
- Pada pengujian black box, yang di evaluasi hanya dari tampilan luar (interface) & fungsionalitasnya, tanpa mengetahui apa sesungguhnya yang terjadi dalam proses detilnya (hanya mengetahui input dan output).

KONSEP SOFTWARE TESTING

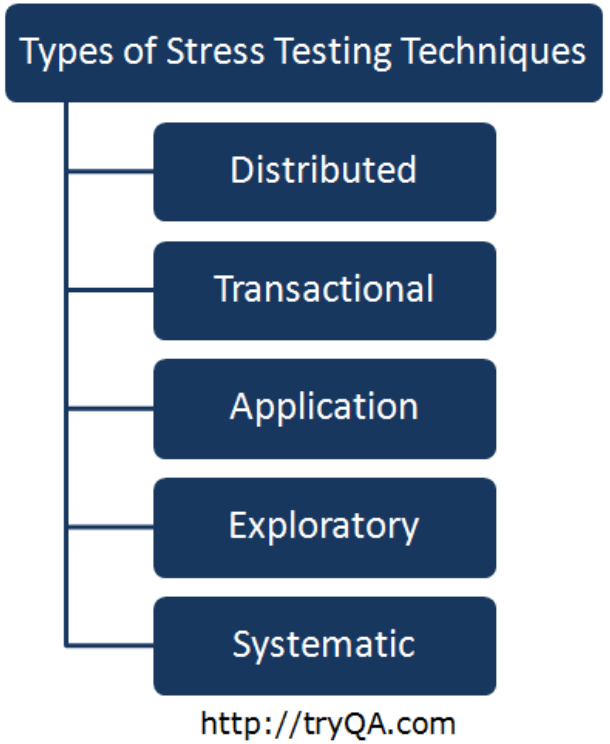
ENDURANCE TEST

- Endurance Test adalah jenis pengujian bersifat non-fungsional.
- Dikenal juga sebagai Soak Test.
- Endurance Test melibatkan pengujian sistem dengan beban signifikan yang diperpanjang selama periode waktu yang signifikan, untuk menemukan bagaimana sistem berperilaku di bawah penggunaan normal yang berkelanjutan.
- Misalnya, dalam pengujian perangkat lunak, sistem dapat berperilaku tepat seperti yang diharapkan ketika diuji selama 1 jam tetapi ketika sistem yang sama diuji selama 3 jam, masalah seperti *memory leak* atau *over-heat* dapat menyebabkan sistem gagal atau berperilaku secara acak.

KONSEP SOFTWARE TESTING

STRESS TEST

- Stress Test adalah jenis pengujian bersifat non-fungsional.
- Stress Test juga disebut Fatigue Test
- Melibatkan pengujian di luar kapasitas operasional normal, seringkali hingga ke titik puncaknya, untuk mengamati hasilnya
- Dilakukan dengan memberi penekanan lebih besar pada ketahanan, ketersediaan dan penanganan kesalahan di bawah beban yang berat, daripada keadaan normalnya.
- Bertujuan untuk memastikan software tidak crash dalam kondisi sumber daya komputasi yang tidak mencukupi (seperti memori, ruang disk, permintaan jaringan dll)



KONSEP SOFTWARE TESTING

MONKEY TESTING

- Monkey Testing adalah jenis pengujian perangkat lunak di mana tester memasukkan input acak ke dalam aplikasi tanpa test case yang telah ditentukan dan memeriksa perilaku sistem.
- Menggunakan teknik eksplorasi.
- Pada Monkey Testing, tester (terkadang developer juga) dianggap sebagai 'Monkey'. Jika monyet menggunakan komputer, maka ia akan *secara acak* melakukan tugas apa pun pada sistem sesuai dengan pemahamannya

GORILLA TESTING

- Gorilla Testing adalah teknik pengujian perangkat lunak di mana satu modul program diuji berulang-ulang kali untuk memastikannya berfungsi dengan benar dan tidak ada bug di modul itu.
- Modul dapat diuji lebih dari seratus kali, dan dengan cara yang sama.
- Gorilla Testing juga dikenal sebagai "Frustrating Testing".

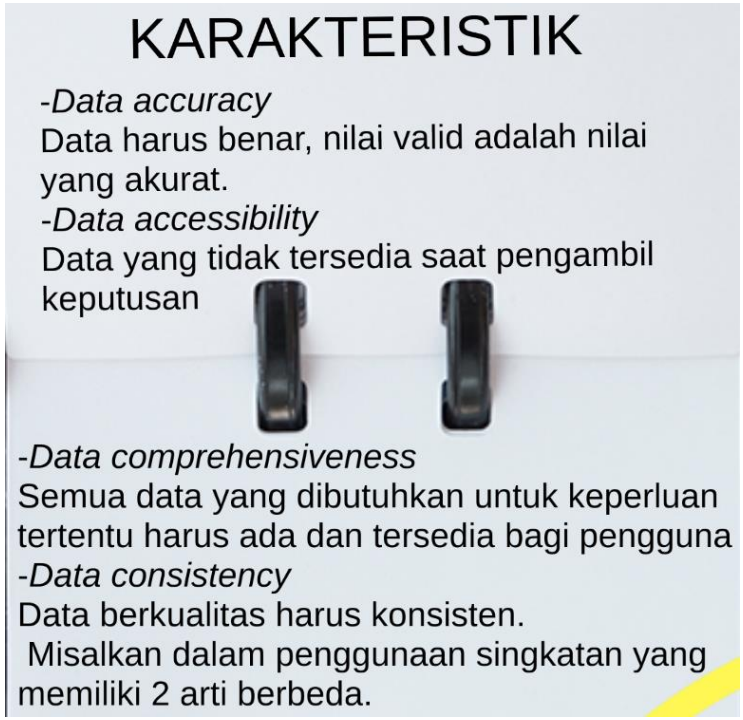
KONSEP SOFTWARE TESTING

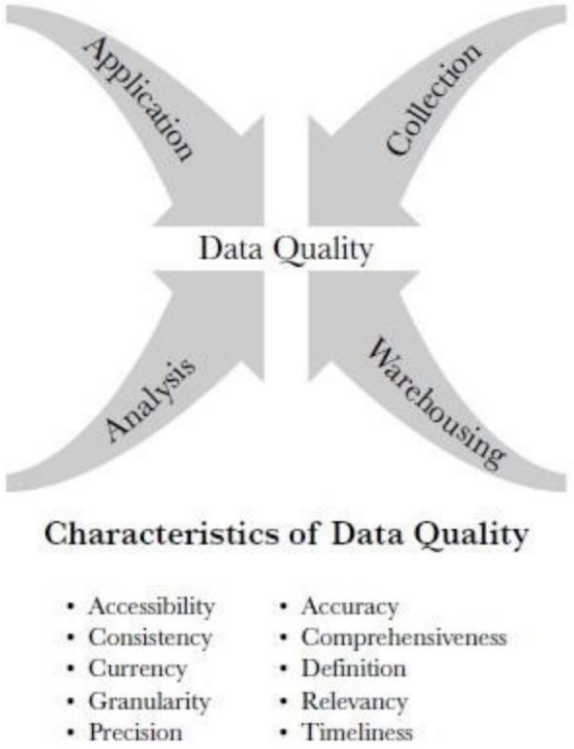
Monkey Testing	Gorilla Testing
Monkey Testing is performed randomly with no specifically predefined test cases	It is neither predefined nor random
Monkey Testing is performed on entire system can have several test cases	Gorilla Testing is performed on specifically few selective modules with few test cases
The objective of Monkey Testing is to check for system crash	Objective of Gorilla testing is to check whether the module is working properly or not

Once get cleared with this difference have a look towards next;

PERSPEKTIF DATA QUALITY

- Data Quality adalah level data yang menyatakan data tersebut akurat (**accurate**), lengkap (**complete**), timely (**update**), konsisten (**consistent**) sesuai dengan semua kebutuhan peraturan bisnis dan relevan. [1]





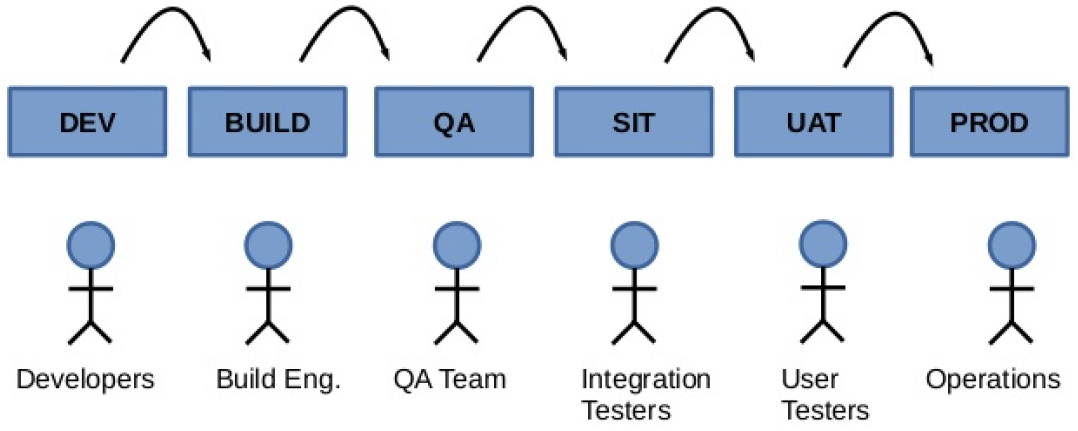
[1] Mark Mosley (2008), dalam bukunya "Dictionary of Data Management"

SOFTWARE DEPLOYMENT

- Software Development & Deployment Environment
- Unit Test
- System Integration Test (SIT)
- User Acceptance Test (UAT)
- Sanity, Smoke, & Regression Tes

FASE
SOFTWARE DEVELOPMENT & DEPLOYMENT ENVIRONMENT

- Software Development Environment adalah ruang lingkup pengembangan sebuah software, meliputi hardware & software yang digunakan.
- Software Deployment Environment adalah ruang lingkup penyebaran/pendistribusian software



UNIT TEST

- Unit Test adalah metode verifikasi perangkat lunak di mana developer menguji suatu unit program layak untuk tidaknya dipakai.
- Unit testing ini fokusnya pada verifikasi pada unit yang terkecil pada desain perangkat lunak (komponen atau modul perangkat lunak). Karena dalam sebuah perangkat lunak banyak memiliki unit-unit kecil maka untuk mengujinya (biasanya dibuat program kecil atau main program) untuk menguji unit-unit perangkat lunak.
- Unit-unit kecil ini dapat berupa prosedur atau fungsi, sekumpulan prosedur atau fungsi yang ada dalam satu file (Structural Programming), atau class, bisa juga kumpulan class dalam satu package (OOP).
- Pengujian unit biasanya dilakukan saat kode program dibuat (fase Development).
- Yang melaksanakan adalah **Developer**.

SYSTEM INTEGRATION TEST (SIT)

- SIT adalah test yang dilakukan untuk menguji apakah fungsi-fungsi yang ada dari perangkat lunak adalah sebuah solusi yang **terintegrasi**.
- SIT dilakukan untuk mengkonfirmasi apakah modul yang diuji secara individual dapat bekerja bersama untuk memberikan fungsionalitas yang diperlukan. Modul yang diuji secara individual mungkin dapat bekerja dengan baik, tetapi ketika mereka terintegrasi satu sama lainnya, beberapa masalah mungkin dapat terjadi.
- SIT dilakukan dengan menguji ketergantungan antar modul melalui transfer data dari satu modul ke modul lainnya.
- Integrasi sistem dimulai pada tingkat modul dimana unit-unit diintegrasikan bersama membentuk **sub-sistem** dan pada akhirnya integrasi antar sub-sitem akan menjadi suatu **sistem keseluruhan**.
- Yang melaksanakan adalah **Tester**.
- Ada dua pendekatan utama untuk SIT: ***Top-Down*** *Integration Approach* & ***Bottom-Up*** *Integration Approach*.

SMOKE TESTING

- Smoke Testing adalah pengujian yang berfokus pada perangkat lunak secara keseluruhan, namun hanya berfokus pada main function, biasanya mengutamakan positif case, namun tidak menutup kemungkinan dilakukan pada negatif case yang sifatnya critical.
- Tujuan dari smoke testing ini adalah untuk reject aplikasi yang sudah rusak sejak awal development, sehingga tim QA tidak banyak membuang-buang waktu menginstal dan menguji aplikasi perangkat lunak.

SANITY TESTING

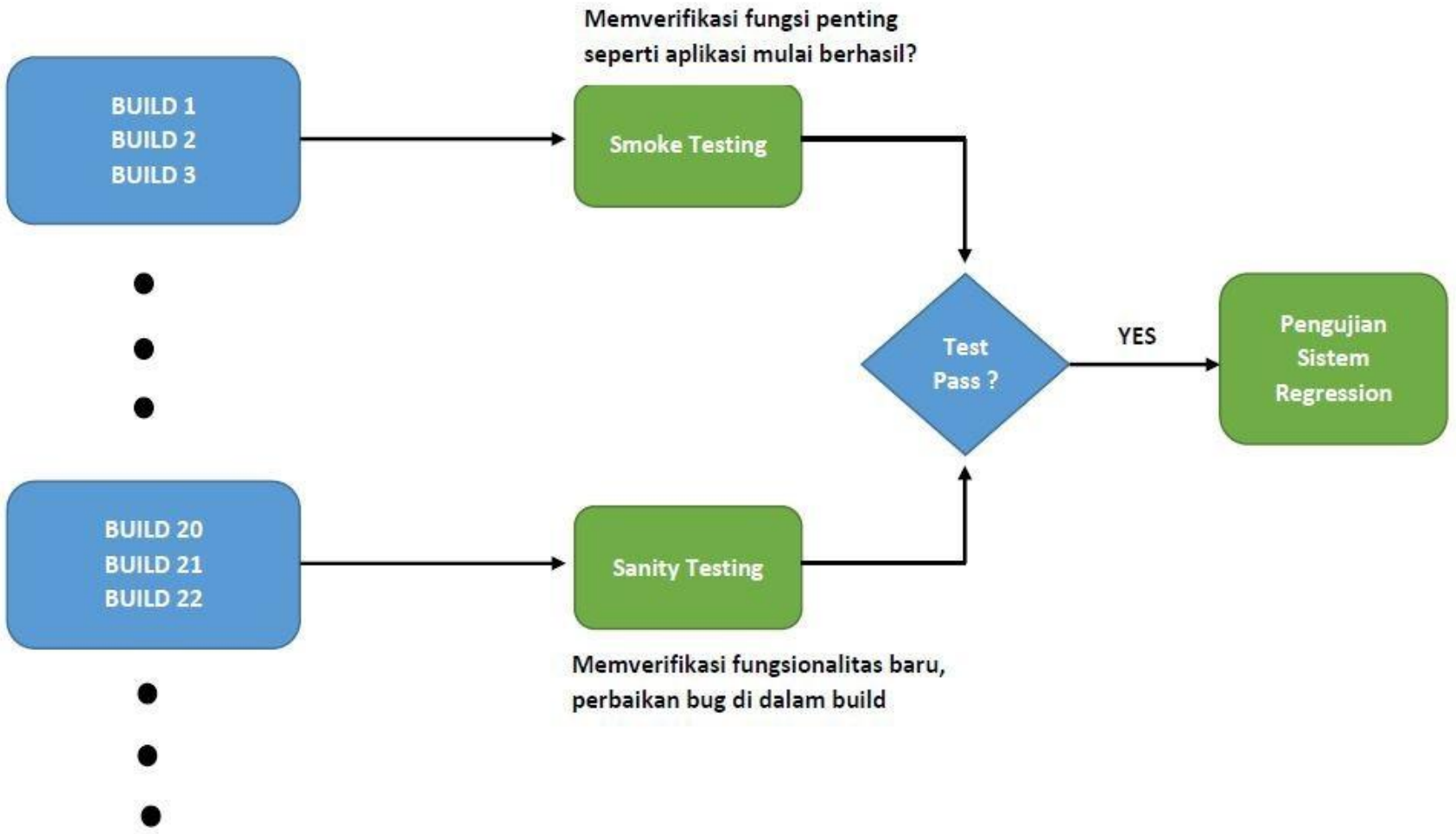
- Sanity Testing adalah pengujian yang dilakukan terhadap sebuah perangkat lunak dan hanya berfokus pada feature atau fungsi baru atau fungsi yang dilakukan perubahan, baik perubahan itu merupakan perubahan flow bisnis atau perubahan karena ditemukannya kesalahan atau bugs pada perangkat lunak versi sebelumnya.
- Tujuan dari sanity testing ini tentunya untuk memastikan bahwa bug-bug yang telah di perbaiki pada saat smoke testing sudah selesai diperbaiki dan tidak ada masalah lebih lanjut serta untuk menentukan bahwa fungsi yang diinginkan bekerja seperti yang diharapkan.

REGRESSION TEST

- Regression Test didefinisikan sebagai jenis pengujian software untuk mengkonfirmasi bahwa perubahan program atau kode yang baru dilakukan tidak mempengaruhi fitur yang sudah ada.
- Regression Test tidak lain adalah seleksi secara sebagian ataupun menyeluruh dari test case yang sudah dieksekusi sebelumnya yang kemudian dieksekusi kembali untuk memastikan fungsionalitas yang ada berjalan dengan baik.
- Pengujian ini dilakukan untuk memastikan bahwa perubahan kode baru seharusnya tidak memiliki efek samping pada fungsi yang ada. Hal ini untuk memastikan bahwa kode lama masih berfungsi setelah perubahan kode terbaru dilakukan.
- Regression Test diperlukan jika terjadi hal-hal sebagai berikut:

Perubahan persyaratan dan kode dimodifikasi sesuai dengan kebutuhan
Fitur baru ditambahkan ke perangkat lunak
Memperbaiki cacat
Memperbaiki masalah kinerja

RELASI SMOKE, SANITY DAN REGRESSION TESTING



USER ACCEPTANCE TEST (UAT)

- UAT adalah tahap akhir pengujian sebelum sistem **diterima** oleh pengguna operasional.
- Pengguna akhir (end-users) melakukan UAT berdasarkan spesifikasi kebutuhan pengguna untuk mengkonfirmasi apakah suatu perangkat lunak memenuhi persyaratan, sebelum pada akhirnya disebarluaskan.
- UAT bertujuan untuk mengetahui tingkat kepuasan pengguna.
- Ada dua jenis utama untuk UAT yaitu: ***Alpha Testing*** & ***Beta Testing***.

- Alpha Testing dilakukan oleh pengguna di bawah kendali pengembang didalam sistem yang telah disediakan oleh developer. Alpha Testing dilakukan setelah SIT selesai.
- Beta Testing dilakukan oleh pengguna secara mandiri tanpa kendali developer pada perangkat yang dimiliki sendiri oleh pengguna. Beta Testing hanya bisa dilakukan jika Alpha Testing selesai.

SOFTWARE TESTING TOOLS

- Automation Tools
- Manual Tools

SOFTWARE TESTING TOOLS

AUTOMATION TOOLS

- Selenium
- Appium
- Cypress.io
- Katalon Studio
- Ranorex
- Test Complete
- QMetry Automation Studio
- Jenkins
- dll

MANUAL TOOLS

- Apache JMeter
- Postman
- SoapUI
- Swagger
- Mantis Bug Tracker
- TestLink
- Firebug
- dll

TEST CASE

- Definisi Test Case
- Contoh Test Case
- Jenis Test Case
- Tips Pembuatan Test Case

DEFINISI TEST CASE

- **Test Case** adalah **dokumen** yang terdiri dari **serangkaian kondisi** atau tindakan yang dilakukan pada aplikasi perangkat lunak untuk **memverifikasi** fungsionalitas fitur yang diharapkan.
- Test Case adalah dokumen yang menggambarkan input, tindakan, atau peristiwa dan respon yang diharapkan, untuk menentukan apakah fitur dari aplikasi bekerja dengan benar.
  Sebuah kasus uji harus berisi keterangan seperti **identifier test case**, **nama test case**, **tujuan**, **kondisi pengujian / setup**, **persyaratan input data**, **langkah-langkah**, dan **hasil yang diharapkan**.

CONTOH TEST CASE

Test case ID	Test case description	Prerequisites	Test steps	Test data	Expected Result	Actual Result	Status	Created By	Date of creation	Executed By	Date of execution
TC001	The objective of this test case is to verify the 'Login' of Gmail account	1. User is authorized 2. Has an account in Gmail	1. Enter valid username 2. Enter valid password 3. Click on 'Login' button	1. User account should be present in Gmail	1. User should be able to login his Gmail account with his valid credentials 2. 'Invalid username or password' should get displayed if the username and password are not valid	1. If the valid credentials are entered then the user will be able to login his / her account 2. If invalid credentials are entered then nothing happens(the expected message is not displayed)	Fail	Rajesh	1/1/2016	Umesh	1/2/2016

KOMPONEN TEST CASE

- **Test Case ID**: ID dari setiap kasus uji
- **Test Case Description**: tujuan atau ringkasan dari kasus uji
- **Prerequisites**: Segala prasyarat yang perlu dijalankan sebelum memulai pengujian
- **Test Steps**: Prosedur untuk menjalankan pengujian
- **Test Data**: Data Uji yang diperlukan saat menjalankan pengujian
- **Expected Result**: Hasil tes yang diharapkan
- **Actual Result**: Hasil aktual dari tes (fakta)
- **Status**: '**Lulus**', '**Gagal**', '**Tidak Dieksekusi**' saat uji coba tidak dieksekusi, dan '**Diblokir**' saat bug tingkat keparahan tinggi ditemukan
- **Created By**: Nama orang yang menulis kasus uji
- **Date of Creation**: Tanggal di mana kasus uji ditulis
- **Executed By**: Nama orang yang menjalankan atau mengeksekusi kasus uji
- **Date of Execution**: Tanggal di mana kasus uji dieksekusi

JENIS TEST CASE

POSITIVE TEST

- Positive Test dilakukan dengan menentukan software berfungsi sesuai dengan yang diharapkan
- Jika ditemukan kesalahan selama Positive Test berlangsung, maka pengujian dianggap gagal (software tidak memenuhi kriteria yang ditentukan)

NEGATIVE TEST

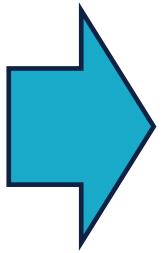
- Negative Test dilakukan dengan memastikan software dapat berfungsi dengan baik meskipun diberikan input yang invalid atau dari perilaku pengguna yang tidak terduga
- Tujuan dari Negative Test adalah untuk mendeteksi situasi tak terduga dan mencegah software crash (tidak berfungsi)
- Negative Test dapat meningkatkan kualitas dari software dengan cara menemukan titik kelemahannya

TIPS PEMBUATAN TEST CASE

- **Customer Requirement**: Test case harus dibuat dengan memperhatikan persyaratan pelanggan.
- **Straightforward, Simple & Clear**: Test case harus tegas, sederhana dan jelas. Juga sangat mudah dieksekusi tidak peduli oleh siapapun dan tetap memberikan output yang sama.
- **Do NOT Presume**: Jangan menebak fungsi atau fitur pada software. Selalu tulis test case sesuai dokumen spesifikasi persyaratan.
- **Unique**: Setiap test case harus memiliki nama unik yang membantu dalam mengklasifikasikan test case, sambil melacak bug atau meninjau persyaratan apa pun pada tahap selanjutnya.
- **Should NOT be Repeated**: Test Case yang dituliskan tidak boleh diulang. Jika salah satu test case mengharuskan untuk melakukan langkah-langkah yang sama (dari test case lainnya) daripada menulisnya lagi, gunakan ID test case-nya di kolom prasyarat.

TIPS PEMBUATAN TEST CASE (2)

- **Assure 100% Test Coverage**: Pastikan semua persyaratan pelanggan dipenuhi saat menulis test case. Sesuai spesifikasi pelanggan, semua persyaratan telah dicakup.
- **Peer Review of Test Case**: Test case harus selalu ditinjau oleh rekan kerja. Jika ada prasyarat atau kondisi apa pun yang terlewatkan saat menulis kasus pengujian, maka hal itu dapat dicakup sesuai umpan balik rekan.
- **Implementation of Testing Techniques**: Saat menulis test case, tidak mungkin untuk mencakup semua kondisi dari software. Dengan bantuan teknik pengujian tertentu maka dapat ditemukan bug yang lebih banyak.



Boundary Value Analysis (BVA)
Equivalence Partitioning (EP)
State Transition Technique
Decision Tables
Error Guessing Technique
Exploratory Testing

THANK YOU

ERIC.AGUSTIAN@XSIS.CO.ID