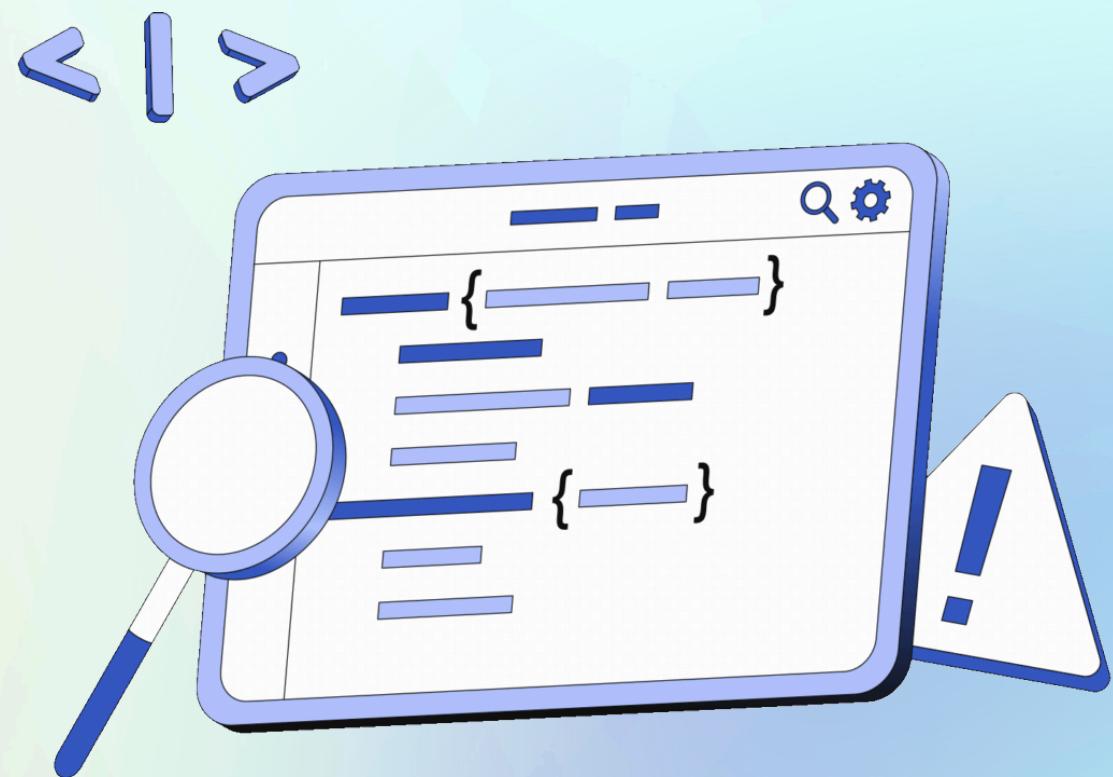




# STRATEGI TESTING





# ANGGOTA KELOMPOK

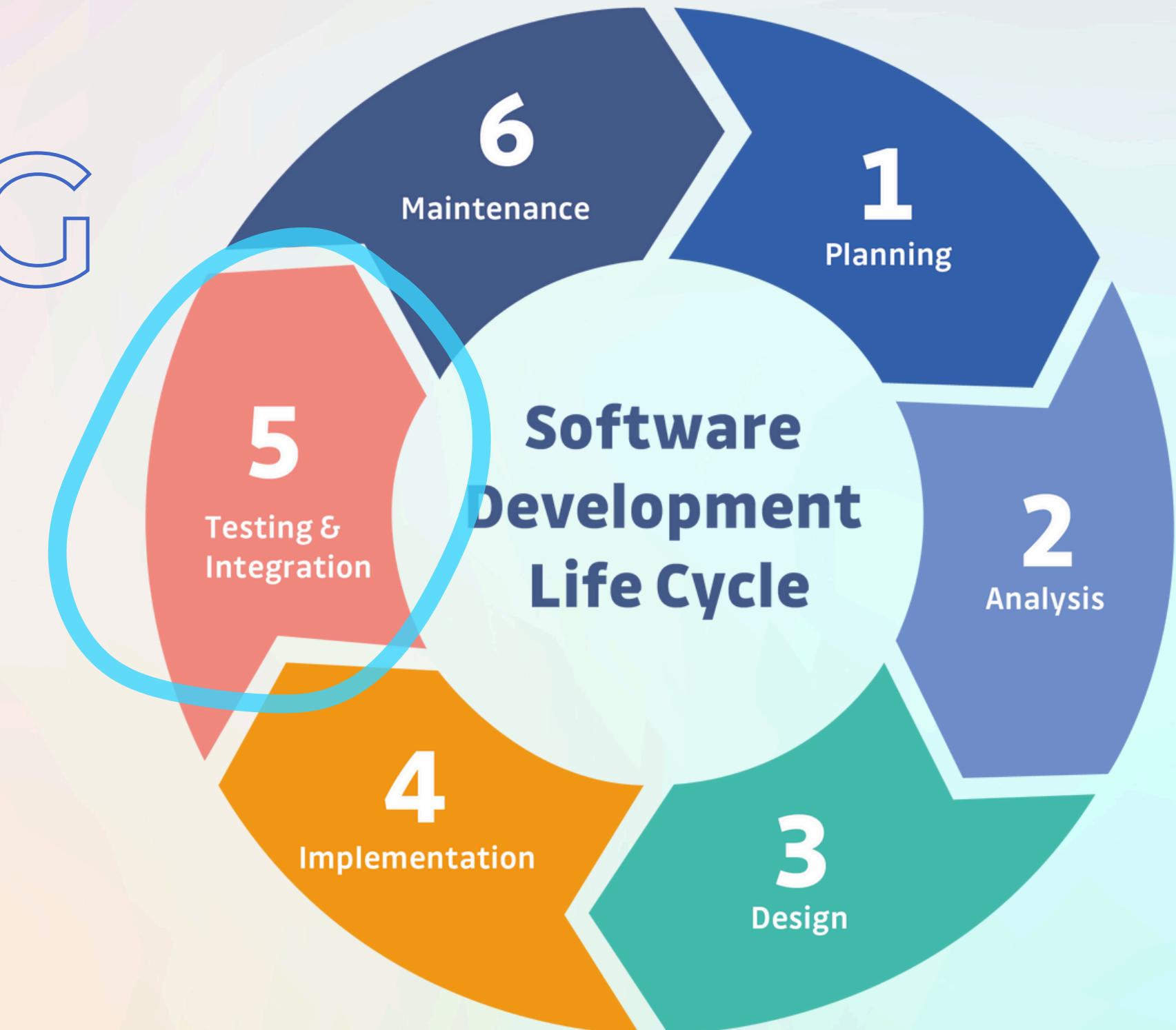


**H071231016**  
**H071231018**  
**H071231020**  
**H071231026**  
**H071231049**  
**H071231063**  
**H071231086**

Athifah Nur Rahman MD  
Fadhilah Meisya Az Zahrah  
Angga  
Zainab Muchsinin  
Syaebatul Hamdi  
Amalia Diah Ramadhani  
Naila Mujadiah

# Fase TESTING

Software (Perangkat Lunak) dikembangkan melalui tahapan bertahap yang dikenal dengan **Software Development Life Cycle (SDLC)**.



# Apa Itu TESTING?

Testing adalah proses mengevaluasi produk perangkat lunak untuk menemukan cacat dan memastikan produk bekerja sesuai kebutuhan baik fungsional maupun non-fungsional.

what ?





## Apa Tujuan dari **TESTING?**

Menemukan kesalahan atau cacat(bug) dalam perangkat lunak atau sistem dan memastikan kualitas sebelum produk dirilis.



# Tujuan Testing

- Verifikasi dan Validasi
- Mengurangi resiko kegagalan
- Meningkatkan kepercayaan stakeholder
- Menjamin kemanan
- Efisiensi biaya
- Meningkatkan Pengalaman Pengguna

# Pentingnya Testing

dalam Pengembangan Perangkat Lunak



# SIKLUS HIDUP TESTING





# Apa Itu **SOFTWARE TESTING LIFECYCLE?**

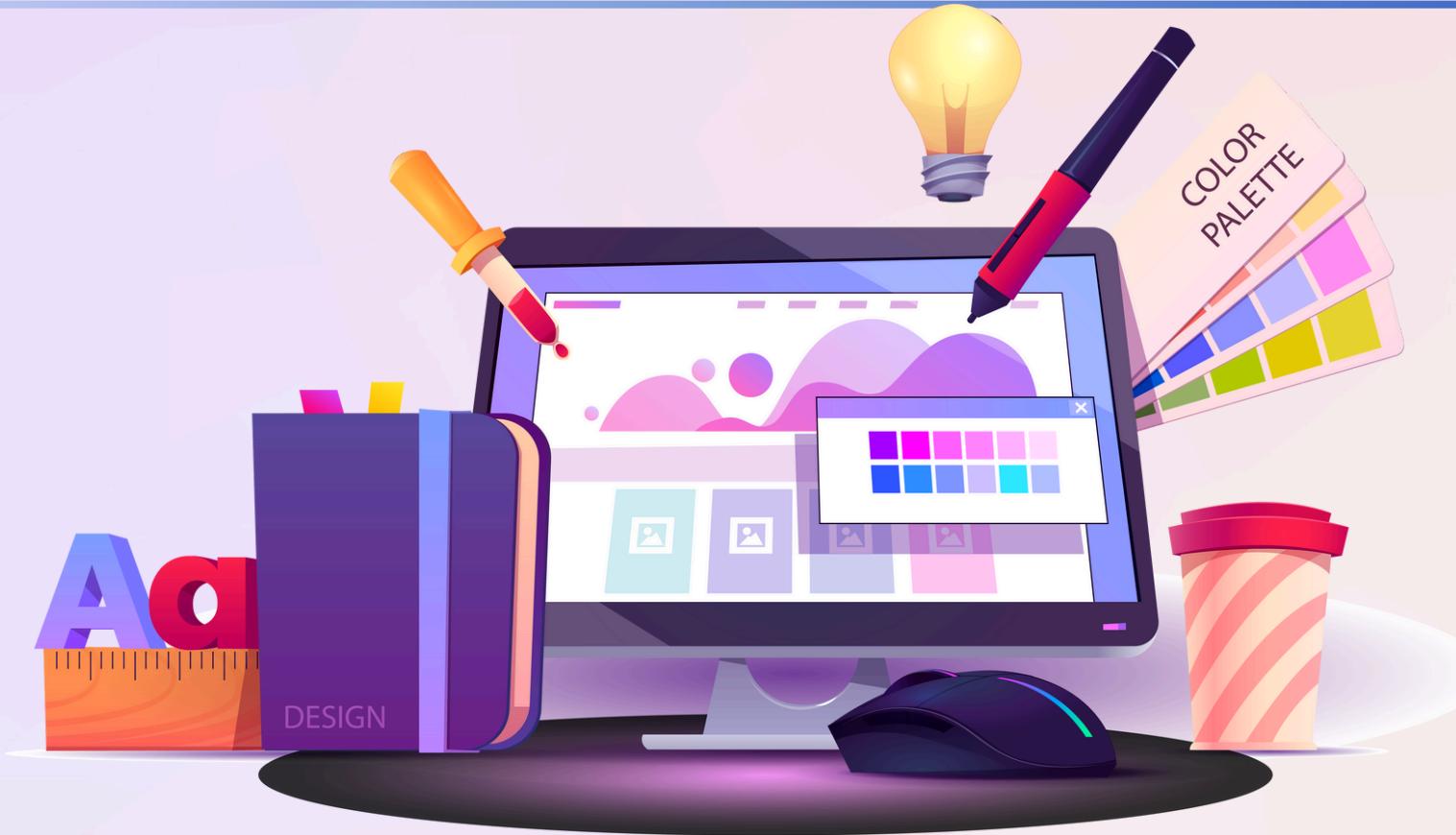
Software Testing Life Cycle adalah pendekatan sistematis untuk menguji perangkat lunak (software) untuk memastikan bahwa software tersebut memenuhi persyaratan dan bebas dari cacat. Ini merupakan proses yang mengikuti serangkaian langkah atau fase, dan setiap fase memiliki tujuan dan hasil yang spesifik.

Proses ini digunakan untuk memastikan bahwa software yang dibuat dan dikembangkan adalah software dengan kualitas terbaik, dapat diandalkan, dan memenuhi kebutuhan pengguna akhir (user).



# Test Planning

- **Membuat strategi pengujian**
- **Mengidentifikasi lingkungan pengujian**
- **Mengidentifikasi uji kasus**
- **Memperkirakan waktu dan biaya**
- **Mengidentifikasi hasil tes dan capaiannya**
- **Menugaskan peran dan tanggung jawab**
- **Meninjau dan menyetujui rencana testing**

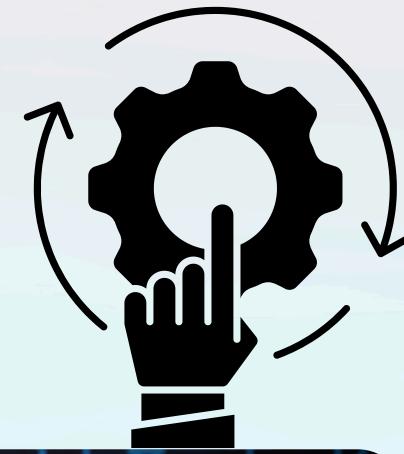


# Test Design

- Mengidentifikasi test case
- Menulis test case
- Membuat data dan skenario pengujian
- Mengidentifikasi hasil yang diharapkan
- Meninjau dan memvalidasi hasil
- Memperbarui dokumen Requirement Traceability Matrix



# Test eksekusi



## BUG :)

### sistem testing

Pengujian terhadap integrasi sub-system, yaitu keterhubungan antar sub-system.

### Acceptance Testing

Pengujian terakhir sebelum sistem dipakai oleh user.  
Melibatkan pengujian dengan data dari pengguna sistem

### Integration Testing

Pengujian kelompok komponen-komponen yang terintegrasi untuk membentuk sub-system ataupun system  
Dilakukan oleh tim penguji yang independent  
Pengujian berdasarkan spesifikasi sistem

### komponen testing

Pengujian komponen-komponen program  
Biasanya dilakukan oleh component developer (kecuali utk system kritis)

# Pelaporan & Analisis Testing

## Ringkasan hasil pengujian

Menyajikan jumlah kasus uji yang berhasil, gagal, atau belum dijalankan.

## Evaluasi kualitas aplikasi

Menilai apakah sistem memenuhi spesifikasi fungsional dan non-fungsional.



## Identifikasi bug dan isu teknis

Mencatat kesalahan yang ditemukan selama pengujian, termasuk tingkat keparahan dan status perbaikannya.

## Rekomendasi perbaikan

Memberikan saran untuk peningkatan performa, keamanan, atau stabilitas sistem.

## Grafik dan data analitik

Menampilkan tren pengujian, seperti peningkatan jumlah kasus uji yang berhasil atau penurunan jumlah bug.

# KLASIFIKASI SOFTWARE TESTING



# Klasifikasi Software Testing

berdasarkan Abstraksi



- **Unit testing**
- **Integration testing**
- **System/End-to-End testing**
- **Acceptance testing**



# Unit Testing

Strategi ini berfokus pada pengujian komponen perangkat lunak terkecil yang dapat diuji secara terpisah, seperti fungsi, metode, atau kelas.



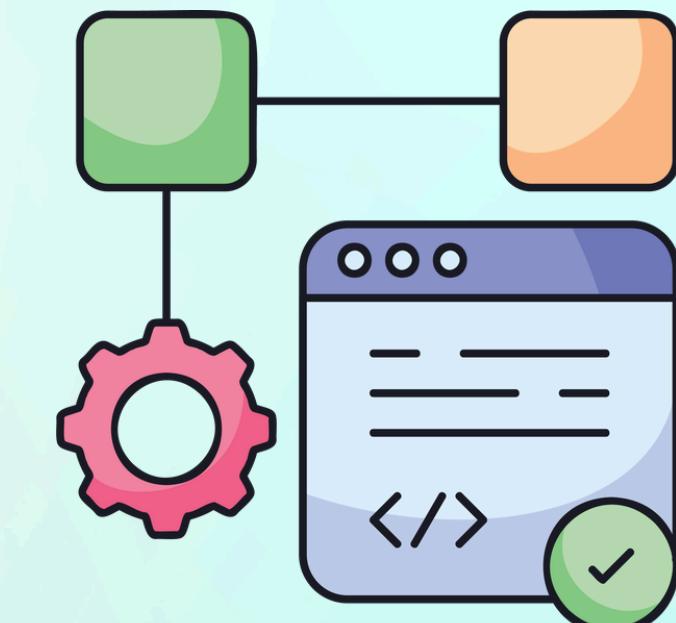
## Tujuan

Memverifikasi fungsionalitas unit kode secara individual.



## Contoh

Menguji sebuah fungsi yang menghitung diskon untuk memastikan hasilnya akurat.





# Integration Testing

Strategi ini berfokus pada pengujian interaksi dan komunikasi antara dua atau lebih unit yang telah diuji secara terpisah.



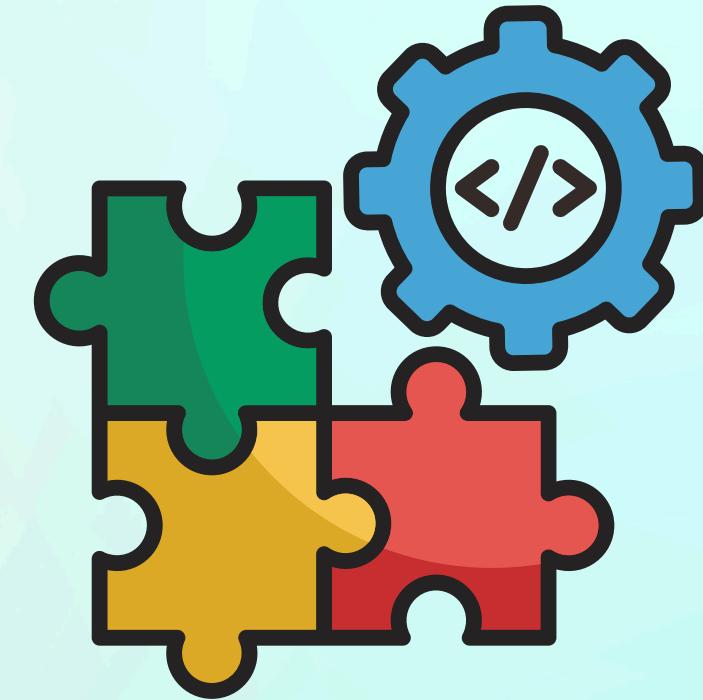
## Tujuan

Memastikan bahwa modul atau unit yang berbeda bekerja sama dengan benar.



## Contoh

Menguji interaksi antara modul login dan modul profil pengguna untuk memastikan data pengguna dapat diakses setelah login berhasil.





# System Testing

Strategi yang berfokus pada pengujian sistem perangkat lunak secara keseluruhan sebagai satu kesatuan yang terintegrasi. Pengujian ini tidak lagi melihat komponen individu, melainkan mengevaluasi apakah sistem memenuhi semua persyaratan fungsional dan non-fungsional yang telah ditentukan.



## Tujuan

Menguji sistem secara holistik terhadap persyaratan yang ada.



## Contoh

Menguji apakah sebuah aplikasi e-commerce dapat menangani 1000 pengguna secara bersamaan (pengujian kinerja) atau apakah semua fitur pembayaran berfungsi dengan baik (pengujian fungsional).

# Acceptance Testing

Strategi ini berfokus pada pengujian sistem dari perspektif pengguna akhir (end-user) atau klien.



## Tujuan

Memvalidasi bahwa sistem dapat diterima oleh pengguna atau klien dan memenuhi kebutuhan bisnis mereka.



## Contoh

Klien menguji sebuah aplikasi mobile baru dan memberikan persetujuan akhir sebelum aplikasi tersebut diluncurkan ke pasar.

# Klasifikasi Software Testing

## berdasarkan Fungsi



- **Fungsional Testing**
- **Non-Fungsional testing**



# Functional Testing

Fungsional testing menguji apakah suatu software atau sistem berfungsi sesuai dengan persyaratan fungsionalnya.



## Tujuan

Memverifikasi bahwa perangkat lunak berfungsi sebagaimana mestinya dan bebas dari bug, serta memvalidasi keluaran yang dihasilkan sesuai dengan harapan pengguna akhir.



## Contoh

Verifikasi fungsi login aplikasi berhasil dengan kredensial valid, pengujian proses pemulihan kata sandi, validasi transaksi pembayaran di e-commerce

# Non Functional Testing

Non-fungsional testing menguji performa, keamanan, reliabilitas, dan aspek lain dari software yang tidak terkait langsung dengan fungsi spesifiknya. Pengujian ini berfokus pada bagaimana suatu sistem bekerja, bukan apa yang dilakukannya.

## Tujuan

Memastikan bahwa sistem dapat memenuhi tuntutan dunia nyata dan memberikan pengalaman pengguna yang lancar dan berkualitas tinggi.

## Contoh

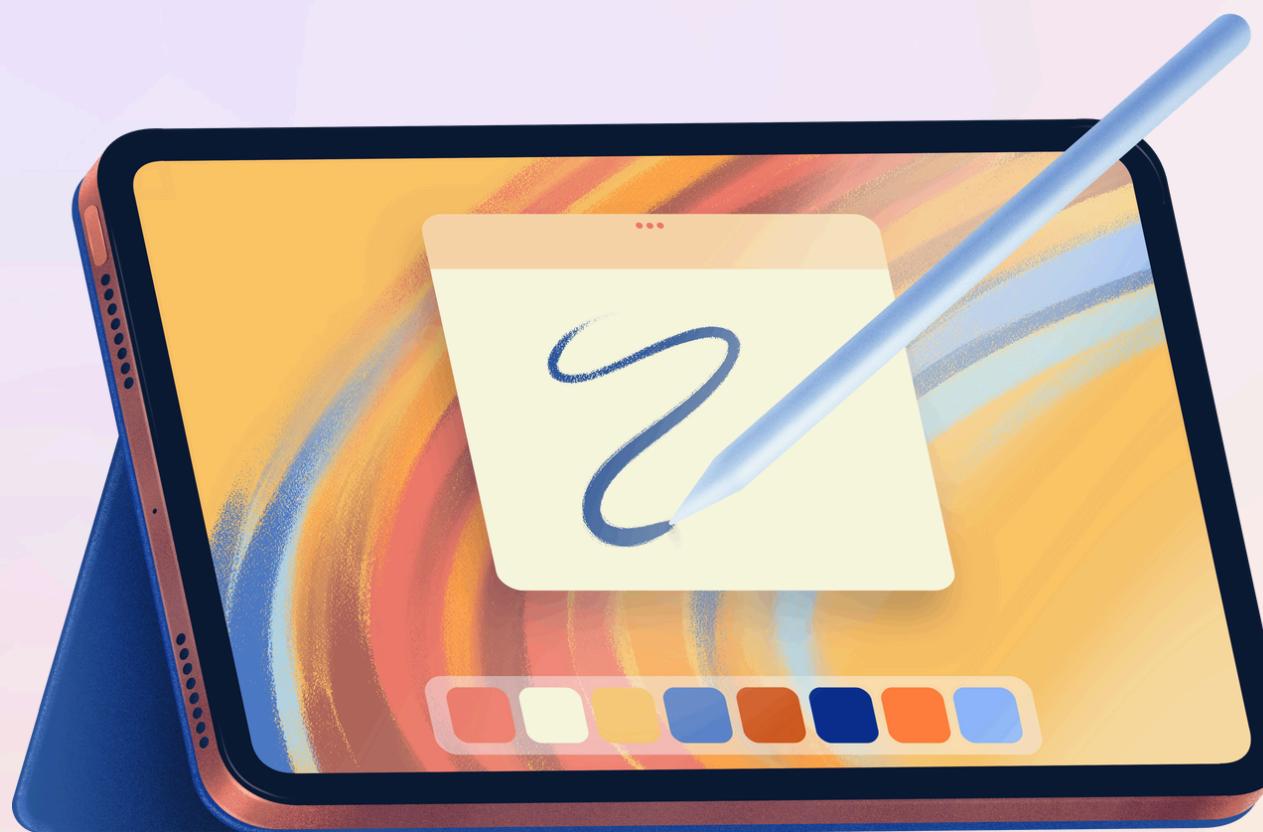
Menguji sebuah website untuk memastikan ia dapat tetap berjalan dengan baik ketika jumlah pengguna melonjak secara bersamaan, seperti saat ada flash sale.

## Contoh

Sebuah website seharusnya memiliki tampilan yang responsif ketika digunakan pada desktop

# Klasifikasi Software Testing

## berdasarkan Domain



- **Performance testing**
- **Security testing**
- **Usability testing**



# Performance Testing

Berfokus pada menguji kinerja perangkat lunak dari segi kecepatan, responsivitas, dan stabilitas di bawah beban tertentu.



## Tujuan

Memastikan bahwa perangkat lunak mampu menangani beban kerja yang tinggi dan dapat beroperasi dengan baik dalam lingkungan yang berbeda



## Contoh

Menguji sebuah website untuk memastikan bahwa ia masih dapat berjalan dengan baik ketika jumlah pengguna melonjak dalam waktu yang bersamaan.



# Security Testing

Berfokus pada mengidentifikasi celah keamanan dan melindungi data dari ancaman eksternal dan internal.



## Tujuan

Memastikan sistem aman dan data pengguna terlindungi dari serangan.



## Contoh

Menguji apakah aplikasi rentan terhadap serangan seperti SQL injection, Cross-Site Scripting (XSS), atau kebocoran data pengguna.



# Usability Testing

Berfokus pada Mengevaluasi kemudahan penggunaan perangkat lunak oleh pengguna akhir.



## Tujuan

Memastikan bahwa perangkat lunak mudah digunakan serta dapat memenuhi kebutuhan pengguna



## Contoh

Menguji apakah aplikasi mudah dipahami dan digunakan oleh pengguna, misalnya dalam menemukan fitur, menyelesaikan tugas, serta memahami ikon dan navigasi.

# Klasifikasi Software Testing

berdasarkan Struktur



- Black Box Testing
- white Box testing



# Klasifikasi Software Testing berdasarkan Struktur

## Black-Box Testing

Black-box testing adalah metode pengujian perangkat lunak di mana para tester tidak mengetahui struktur internal atau kode program. Fokus utama ada pada fungsi dan output sistem, bukan bagaimana proses di dalamnya bekerja.

### Tujuan

- Memastikan bahwa sistem berfungsi sesuai dengan spesifikasi fungsional.
- Menemukan kesalahan terkait fungsi, antarmuka, kinerja, dan validasi input/output.

# Black-Box Testing

## Kelebihan

- Tidak perlu tahu detail teknis kode.
- Relevan untuk menguji sistem dari perspektif end-user.
- Cocok untuk menguji fungsi besar pada sistem.

## Kekurangan

- Tidak menjamin semua jalur kode diuji.
- Sulit mendeteksi bug tersembunyi di dalam logika program.
- Cakupan pengujian bisa terbatas hanya pada skenario yang didokumentasikan.





# Klasifikasi Software Testing berdasarkan Struktur

## White-Box Testing

White-box testing adalah metode pengujian perangkat lunak di mana tester mengetahui struktur internal dan kode program. Fokus utama ada pada alur logika, algoritma, dan struktur data di dalam program.

### Tujuan

- Memastikan semua alur logika dalam program berjalan dengan benar.
- Memeriksa apakah ada error tersembunyi di dalam struktur kode.
- Menjamin kualitas kode melalui coverage testing.

# White-Box Testing

## Kelebihan

- Menjamin cakupan kode lebih luas.
- Dapat menemukan bug tersembunyi di dalam kode.
- Membantu mengoptimalkan dan meningkatkan kualitas kode.

## Kekurangan

- Membutuhkan pengetahuan mendalam tentang program.
- Membutuhkan waktu lebih lama jika sistem sangat kompleks.
- Tidak cocok untuk menguji fungsi besar dari sisi user (karena terlalu detail teknis).



# Kesimpulan

*software testing* yang merupakan bagian dari **software development life cycle (SDLC)**. Jadi, *testing* merupakan tahapan yang sangat penting untuk diperhatikan ketika mengembangkan sebuah *software*. Ini demi menghasilkan *software* yang baik, bebas dari *bugs*, dan disukai oleh pengguna.



# Terima Kasih

