

---

---

# Software Testing Introduciton

— QA Automation —

---

---

# Pengenalan

# Apa itu Software Testing?

Pengujian perangkat lunak atau biasa kita sebut software testing adalah proses pengujian program, dengan tujuan mencari error atau kecacatan pada sebuah program sebelum dibagikan kepada user.

Pada pelaksanaan pengujian itu, jika terlaksana dengan sukses maka hasilnya akan terlihat jelas letak kesalahan atau kekurangannya

# Kenapa perlu Software Testing?

Saat pembuatan perangkat lunak pastinya memakan waktu yang lama dan banyak trial and error.

Apalagi mengingat bahwa software itu merupakan perangkat yang penting dalam menjalankan sebuah sistem pada perusahaan. Yang mana jika ada salah satu sistem yang eror maka akan berdampak buruk untuk perusahaan tersebut.

Dengan melakukan software testing maka akan meminimalisir kesalahan pada perangkat lunak tersebut.

# Jenis-jenis Metode Testing

## A. Manual Testing

Merupakan proses pengujian software yang dilakukan dengan tangan untuk mengetahui Apakah fitur dalam sebuah aplikasi berfungsi dengan baik atau tidak

## B. Automation Testing

Automation test menggunakan tools, script, atau software tertentu untuk menggantikan aktivitas manual testing yang dilaksanakan oleh manusia. Pada automation test, seorang tester terlebih dahulu mempersiapkan script yang kemudian akan dilaksanakan secara berulang oleh software.

# Technology

- **Javascript ES6** sebagai Bahasa Pemrograman Utama
- **Mocha JS + Chai JS** untuk Unit Testing
- **Supertest** untuk Rest API Testing
- **WebdriverIO** untuk Web Testing
- **Appium** untuk Mobile Testing
- **K6** untuk Performance Testing

# Test Case

# Apa itu Test Case?

- Test case adalah format tertentu untuk software testing yang diperlukan untuk memeriksa apakah aplikasi/software tertentu berfungsi atau tidak
- Test case terdiri dari serangkaian kondisi yang perlu diperiksa untuk menguji suatu aplikasi/software
- Test Case dibuat berdasarkan dokumen spesifikasi software seperti PRD, BRD, atau FSD

Contoh PRD / FSD: [https://kc.umn.ac.id/21238/5/BAB\\_III.pdf](https://kc.umn.ac.id/21238/5/BAB_III.pdf)



# Contoh Dokumen Test Case

	A	B	C	D	E	F	G	H	I	J
1	<b>TEST SCENARIO</b>									
2										
3										
4										
5	Test Designed by:	Zetha			Test Designed date:	10 May 2023				
6	Device:	Web								
7										
8	STATUS	13								
9	Passed:	13								
10	Pending/Bug Fixing	0								
11	Failed	0								
12	Blank	0								
13										
14	Story	Test Case	Test Case ID	Test Step	Test Type (+/-)	Test Data	Pre - Condition	Expected Result	Actual Result	Status
15		Search dengan semua data valid	SRB_001	1. Go to <a href="https://booking.kai.id/">https://booking.kai.id/</a> 2. Input semua field 3. Click "Cari & Pesan Tiket"	Positive	1. Stasiun Asal = Purwokerto 2. Stasiun Tujuan = Semarang Tawang 3. Tanggal Keberangkatan = 13 Mei 2023 4. Dewasa = 1	Sudah melakukan login.	Search diharapkan berhasil dan beralih ke halaman search.	Search berhasil dan beralih ke halaman search.	Passed
16		Search dengan stasiun awal kosong	SRB_002	1. Go to <a href="https://booking.kai.id/">https://booking.kai.id/</a> 2. Input semua field 3. Click "Cari & Pesan Tiket"	Negative	1. Stasiun Tujuan = Semarang Tawang 2. Tanggal Keberangkatan = 13 Mei 2023 3. Dewasa = 1	Sudah melakukan login.	Search diharapkan gagal dan muncul alert "Error. Stasiun Awal wajib diisi."	Search gagal dan muncul alert "Error. Stasiun Awal wajib diisi."	Passed
17		Search dengan stasiun tujuan kosong	SRB_003	1. Go to <a href="https://booking.kai.id/">https://booking.kai.id/</a> 2. Input semua field 3. Click "Cari & Pesan Tiket"	Negative	1. Stasiun Asal = Purwokerto 2. Tanggal Keberangkatan = 13 Mei 2023 3. Dewasa = 1	Sudah melakukan login.	Search diharapkan gagal dan muncul alert "Error. Stasiun Tujuan wajib diisi."	Search gagal dan muncul alert "Error. Stasiun Tujuan wajib diisi."	Passed

# Komponen Umum pada Test Case

- **Test Story / Suite / Module:** Tema secara garis besar tentang fitur yang ingin diuji
- **Test ID:** Identitas unik yang diberikan kepada setiap kondisi
- **Test Description / Test Case:** Deskripsi singkat tentang kondisi yang akan diuji
- **Test Step:** Langkah-langkah yang akan dilakukan untuk mengecek kondisi yang diberikan
- **Test Data:** Data yang akan diinput ketika melakukan pengujian
- **Pre-Condition:** Kondisi yang harus dipenuhi sebelum melakukan pengujian
- **Expected Result:** Hasil/Output yang diharapkan pada akhir pengujian
- **Actual Result:** Hasil/Output sebenarnya yang muncul pada akhir pengujian
- **Test Type:** positif atau negatif
- **Status:** Status dari pengujian. Seperti PASS, FAIL, N/A

# Positive vs Negative Test Case

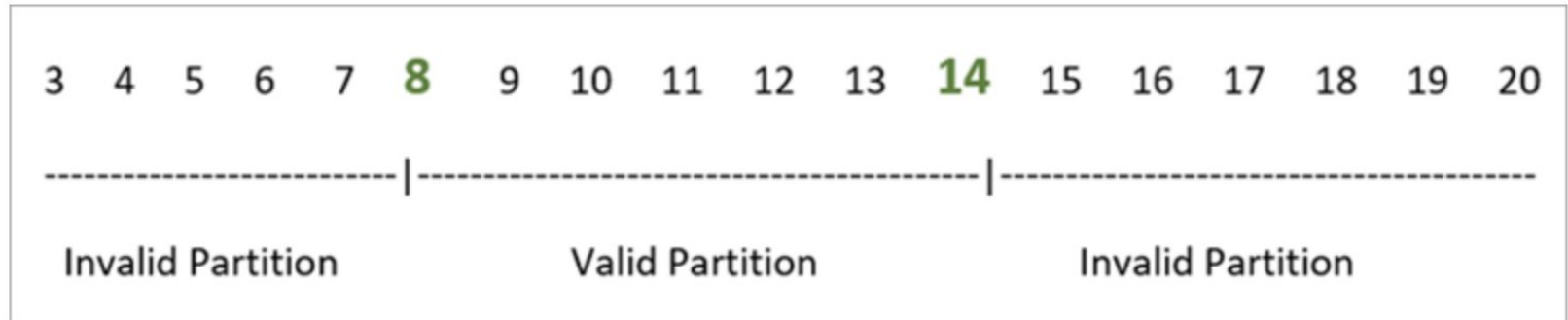
Positive Test	Negative Test
Test yang dilakukan dengan menginput data yang valid	Test yang dilakukan dengan menginput data yang tidak valid
Dilakukan untuk menguji apakah software berfungsi sebagaimana mestinya ketika diberikan input yang valid	Dilakukan untuk menguji apakah software berfungsi sebagaimana mestinya ketika diberikan input yang tidak semestinya / tidak valid
Tujuannya untuk mengecek apakah software berfungsi sesuai dengan yang diharapkan atau tidak	Tujuannya untuk mengecek apakah software tidak rusak dan tetap stabil walaupun diberikan input yang salah

# Test Case Technique

- Equivalence Partitioning
- Boundary Value Analysis
- Decision Table

# Equivalence Partitioning

- Teknik dimana data input dibagi menjadi partisi/grup dengan nilai yang valid dan tidak valid, dan semua partisi harus menunjukkan perilaku yang sama.
- Jika salah satu data dari sebuah partisi lolos test, maka semua data yang ada pada partisi tersebut dianggap lolos/valid



# Boundary Value Analysis


- Teknik ini menguji value pada tepi/batas partisi yang valid dan invalid
- Kondisi pada tepi setiap partisi memiliki kemungkinan salah lebih besar dibandingkan dengan kondisi di tengah partisi
- Input value yang perlu diperhatikan:
  - Tepat di bawah minimum value \*
  - Minimum value \*
  - Tepat di atas minimum value
  - Normal (nominal) value
  - Tepat di bawah maximum value
  - Maximum value \*
  - Tepat di atas maximum value \*

BOUNDARY VALUE ANALYSIS (BVA)		
Invalid Value length (min -1)	Valid Value length (min input length, min+1, nominal input length, max-1, max input length)	Invalid Value length (max + 1)
7 (characters)	8, 9, 11, 13, 14 (characters)	15 (characters)

# Decision Table

- Teknik yang digunakan untuk menguji sistem berdasarkan kombinasi input yang berbeda
- Kombinasi dari beberapa input beserta outputnya dicatat dalam bentuk table





**Log in**

<b>Email (condition1)</b>	T	T	F	F
<b>Password (condition2)</b>	T	F	T	F
<b>Expected Result (Action)</b>	Account Page	Incorrect password	Incorrect email	Incorrect email

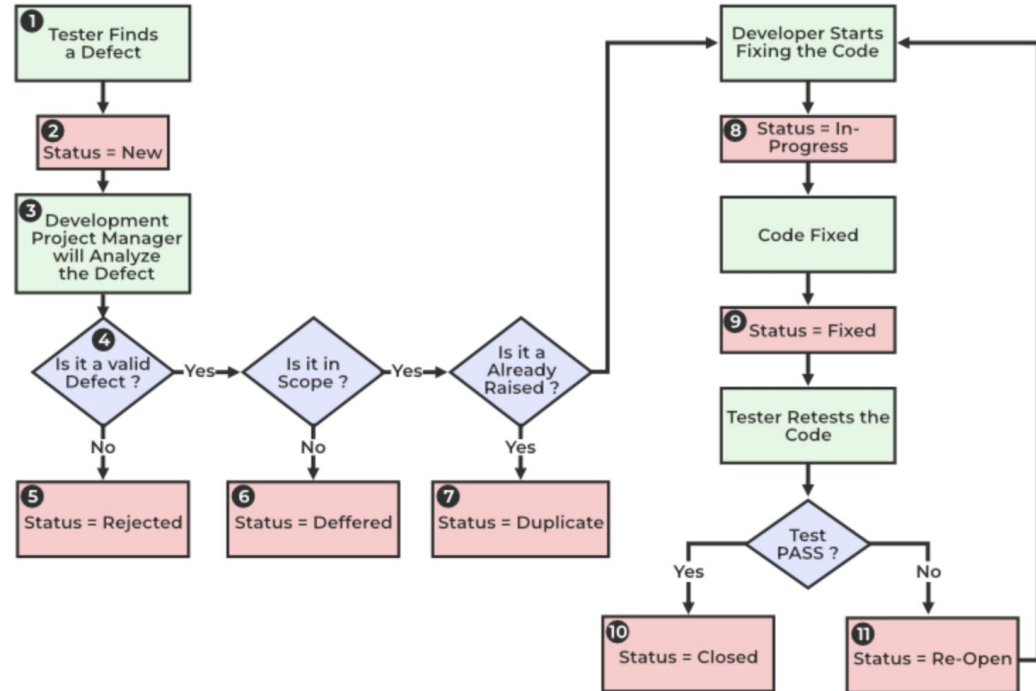
# Bug / Defect

- Bug/Defect adalah kesalahan pada aplikasi yang muncul saat proses pembuatan software, ketika software mulai menunjukkan perilaku tidak normal ketika digunakan
- Salah satu kewajiban tester adalah untuk mencari bug/defect sebanyak-banyaknya untuk menjamin kualitas software dan memenuhi spesifikasi yang telah ditentukan.



# Bug / Defect Life Cycle

Bug Life Cycle adalah siklus dari bug yang melalui beberapa status selama hidupnya

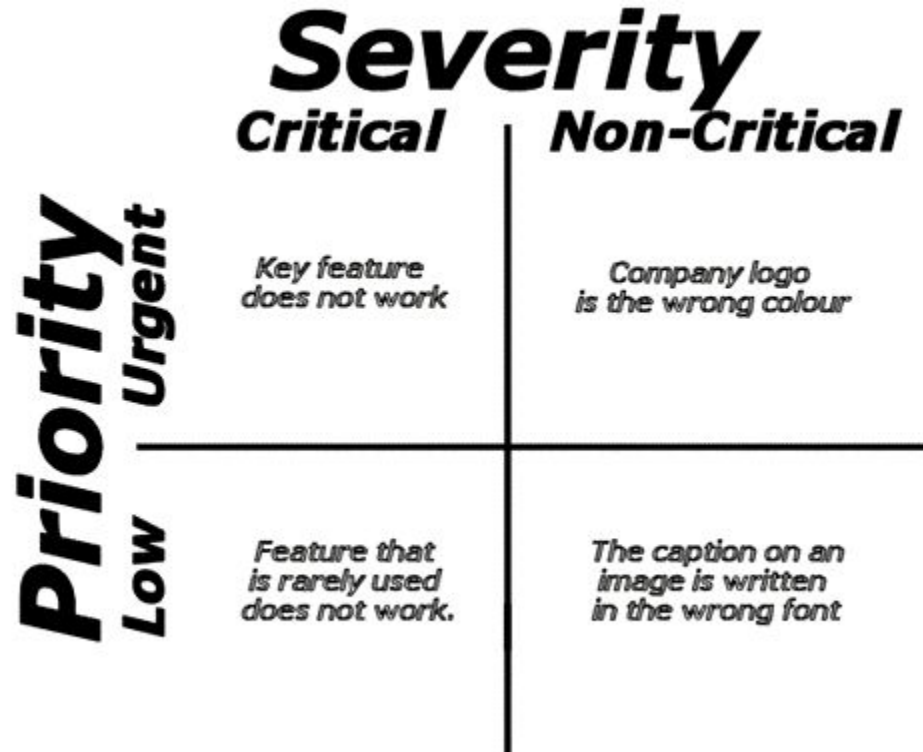


# Severity & Priority

- Setiap Bug yang ditemukan pada tahap pengembangan aplikasi, dapat dikategorikan berdasarkan Severity dan Priority bug tersebut
- Severity adalah parameter seberapa besar dampak suatu bug pada software
- Priority adalah parameter yang menentukan bug mana yang harus diperbaiki terlebih dahulu

Features	Severity	Priority
<b>Definition</b>	Severity is a parameter to denote the impact of a particular defect on the software.	Priority is a parameter to decide the order in which defects should be fixed.
<b>Purpose</b>	Severity means how severe the defect is affecting the functionality.	Priority means how fast the defect has to be fixed.
<b>Relation</b>	Severity is related to the quality standard.	Priority is related to scheduling to resolve the problem.
<b>Categories</b>	Severity is divided into 4 categories: <ul style="list-style-type: none"><li>• Critical</li><li>• Major</li><li>• Medium</li><li>• Low</li></ul>	Priority is divided into 3 categories: <ul style="list-style-type: none"><li>• Low</li><li>• Medium</li><li>• High</li></ul>
<b>Who decides defects?</b>	The testing engineer decides the severity level of the defect.	The product manager decides the priorities of defects.

# Severity & Priority Example



# Test Case Best Practices

1. Test Case harus sederhana dan jelas
2. Buat Test Case dengan mempertimbangkan Calon Pengguna
3. Hindari pengulangan Test Step
4. Jangan Berasumsi
5. Pastikan 100% Sesuai dengan Dokumen Spesifikasi
6. Harus dapat diidentifikasi (Berikan ID)
7. Menerapkan Teknik Pengujian
8. Kembalikan ke keadaan semula setelah testing
9. Test Case harus independen dan dapat diulang dengan output yang sama
10. Kolaborasi dengan teman satu tim

STLC

# Apa itu STLC ?

- Software Testing Life Cycle (STLC) adalah tahap-tahap proses pengujian yang dilaksanakan secara sistematis dan terencana
- Tujuannya untuk memastikan bahwa software memenuhi requirement dan bebas dari bug
- STLC memiliki beberapa fase, dan setiap fase memiliki objektif dan deliverable

Learn More: <https://www.geeksforgeeks.org/software-testing-life-cycle-stlc/>

# Fase STLC

1. Requirement Analysis
2. Test Planning
3. Test Case Development
4. Test Environment Setup
5. Test Execution
6. Test Closure



SDLC



# Apa itu SDLC ?

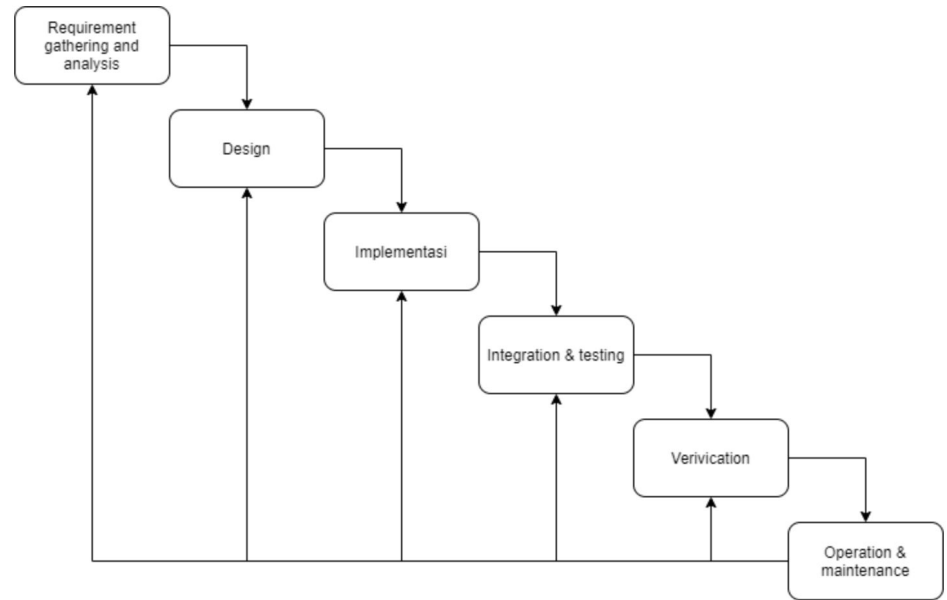
- Software Development Life Cycle (SDLC) adalah sebuah kerangka kerja yang menetapkan tahapan yang dilalui dalam pengembangan software.
- SDLC mencakup secara detail rencana pembuatan, deployment, dan maintenance software

# Model SDLC Yang Populer

- Waterfall Model
- V-Model
- Prototype Model
- Agile

# SDLC: Waterfall Model

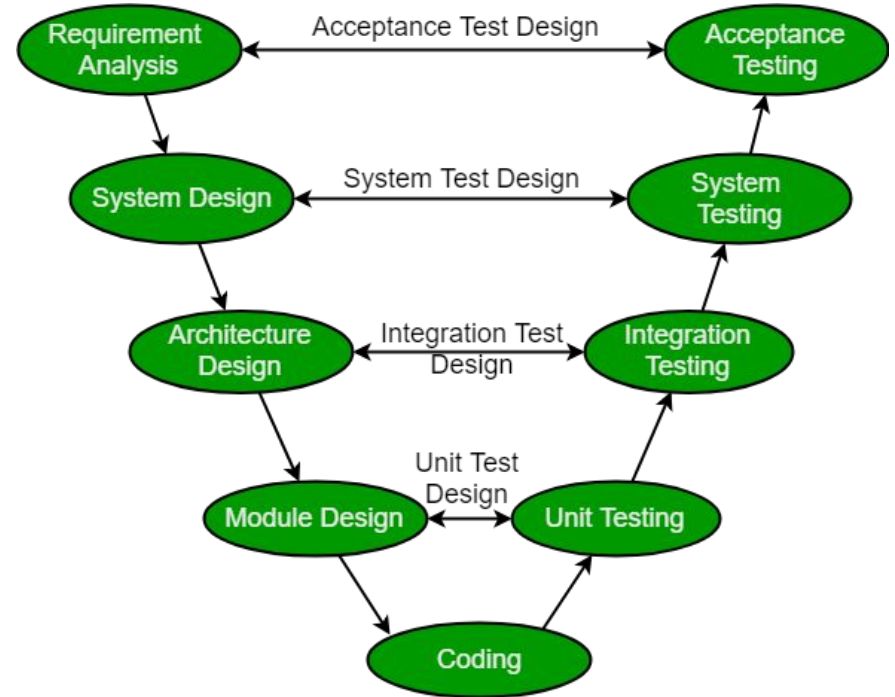
waterfall adalah metode kerja yang menekankan fase-fase yang berurutan dan sistematis. Disebut waterfall karena proses mengalir satu arah “ke bawah” seperti air terjun. Metode waterfall ini harus dilakukan secara berurutan sesuai dengan tahap yang ada.



# SDLC: V-Model

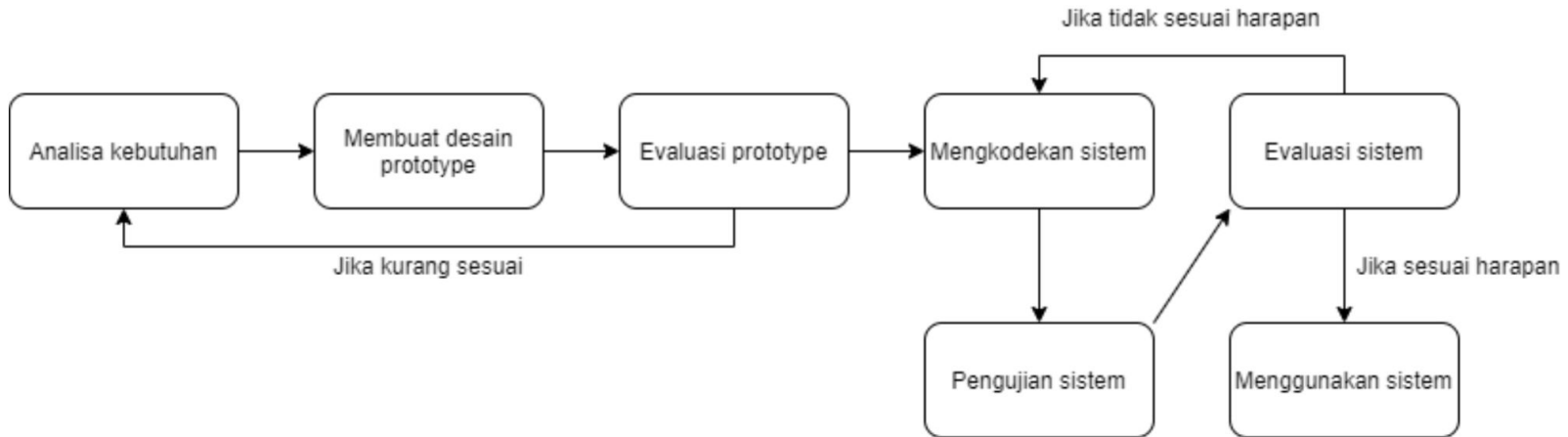
V-Model adalah model SDLC dimana pelaksanaan proses yang terjadi secara berurutan dalam bentuk V.

Setiap tahapan pada model ini dihubungkan dengan test yang terkait.



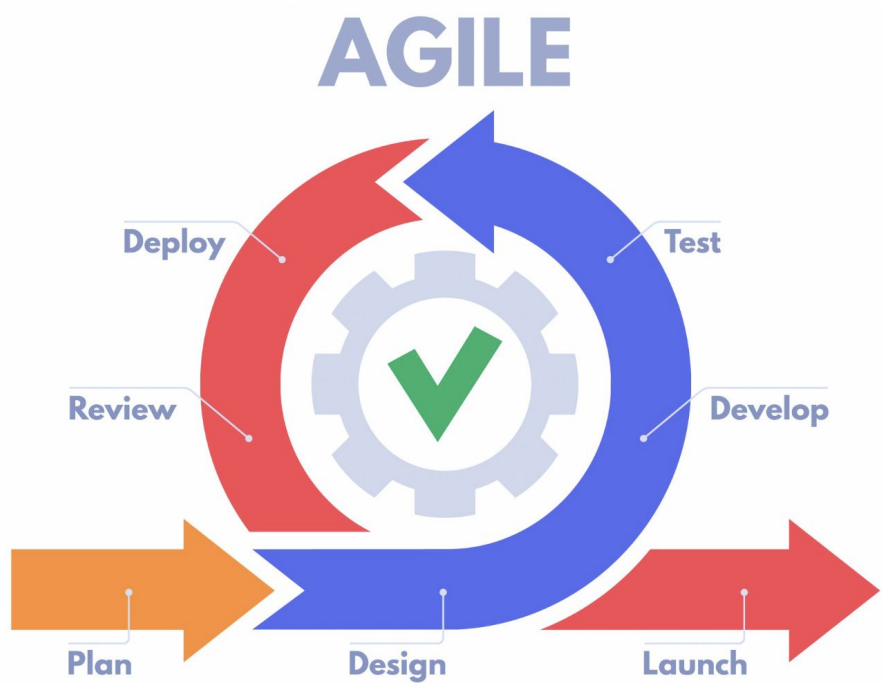
# SDLC: Prototype Model

Metode prototype adalah metode yang memungkinkan pengguna atau user memiliki gambaran awal tentang perangkat lunak yang akan dikembangkan, serta pengguna dapat melakukan pengujian di awal sebelum perangkat lunak dirilis.



# SDLC: Agile

Agile merupakan model yang membagi seluruh proses pengembangan software menjadi beberapa cycle atau sprint kecil. Setiap sprint biasanya memiliki durasi satu hingga tiga pekan.



# Testing Methodologies

# Black Box vs White Box

No.	Perbedaan	White Box	Black Box
1.	Pengetahuan Kode Struktur Internal	Orang yang menjalankan <i>white box testing</i> bisa mengetahui kode dan struktur internal program	Orang yang menjalankan <i>black box testing</i> tidak bisa mengetahui kode dan struktur internal program
2.	Tujuan	<ul style="list-style-type: none"><li>• Untuk melihat struktur aplikasi</li><li>• Memastikan komponen sudah ada di tempat yang seharusnya</li><li>• Memastikan komponen sudah berjalan dengan lancar</li></ul>	<ul style="list-style-type: none"><li>• Untuk menguji sebuah program yang dikembangkan</li><li>• Metode UAT (<i>User Acceptance Testing</i>) untuk mengetahui fungsi aplikasi sudah sesuai ketentuan atau belum</li><li>• Fokus utamanya perspektif <i>end-user</i></li></ul>
3.	Pihak Tester	<i>Software developer</i>	<i>Software tester</i> , bisa dari tim <i>developer</i> baik dari pihak internal perusahaan atau tim tertentu
4.	Pengetahuan Dasar	Diperlukan pengetahuan <i>programming</i> maupun implementasi	Tidak dibutuhkan pengetahuan <i>programming</i> maupun implementasi
5.	Kegiatan Tester	Melihat kode program > membuat <i>test case</i> untuk mencari <i>bugs</i> dan <i>errors</i>	Membuat <i>test case</i> untuk menguji fungsi pada aplikasi > membuat <i>test case</i> untuk menguji kesesuaian alur program > mencari <i>bugs</i> atau <i>errors</i> dari <i>interface</i>



# Functional Testing

# Apa itu Functional Testing?

Pengujian fungsional adalah jenis pengujian yang bertujuan untuk menentukan apakah setiap fitur aplikasi berfungsi sesuai requirement pada dokumen spesifikasi

# Jenis-jenis Functional Testing

- Unit Testing
- Integration Testing (SIT)
- E2E Testing
- System Testing
- User Acceptance Testing
- Smoke Testing
- Regression Testing
- Sanity Testing

# Unit Testing

- Unit Testing adalah tipe testing yang menguji bagian kecil dari kode untuk mengetahui apakah kodenya berfungsi sebagaimana mestinya.
- Tujuannya adalah untuk mengecek bug pada tahap awal dalam development lifecycle
- Unit Test pada umumnya adalah White Box Testing

# Integration Testing

- Integration Testing adalah tipe test yang menguji hubungan/komunikasi antar unit/komponen berfungsi sebagaimana mestinya
- Komponen yang diuji bisa berjumlah 2 atau lebih, bahkan bisa seluruh aplikasi
- Integration Testing bisa jadi Black Box atau White Box

# E2E Testing

- End-to-end testing menguji apakah software berfungsi ketika dipakai dari awal sampai akhir dari sebuah user flow.
- E2E Testing meng-simulasi-kan perilaku user step by step
- E2E menggunakan data dan environment yang mirip dengan production agar simulasi yang dilakukan mirip dengan apa yang ada di production nantinya
- E2E biasanya Black Box

# System Testing

- System testing adalah tipe testing yang menguji software yang telah selesai untuk memastikan software tersebut telah memenuhi requirement yang disediakan
- System Testing biasanya melibatkan functional testing, performance testing, security testing , dll.

# User Acceptance Testing

- User Acceptance Testing (UAT) adalah proses pemeriksaan apakah sebuah software sudah sesuai dengan kebutuhan user.
- Testing dilaksanakan oleh End User / Client
- Memiliki dokumen test case tersendiri
- UAT bertipe Black Box

- Developers have included features on their "own" understanding

- Requirements changes "not communicated" effectively to the developers



# Smoke Testing

- Smoke testing adalah tipe testing yang menguji fitur utama dari sebuah software.
- Smoke testing dilakukan untuk memastikan tidak ada bug yang menghalangi pengujian lanjutan.
- Smoke testing bertipe Black Box

# Regression Testing

- Regression Testing digunakan untuk menguji fitur yang mungkin tersenggol ketika ada update atau perbaikan bug
- Tujuannya untuk memastikan tidak ada bug baru setelah update dilakukan

# Sanity Testing

- Sanity Testing digunakan untuk mengecek fitur atau fungsi yang spesifik berfungsi sebagaimana mestinya setelah ada update atau perbaikan bug.
- Tujuannya sama dengan regression testing, untuk memastikan tidak ada bug baru pada suatu fitur setelah adanya update

# Non-Functional Testing

# Apa itu Non-Functional Testing?

- Non-Functional Testing adalah tipe testing yang digunakan untuk menguji aspek non-fungsional dari software testing
- Contohnya seperti Performance, Usability, Reliability, dll.

# Jenis-jenis Non-Functional Testing

- Performance / Load / Stress Testing
- Compatibility Testing
- Installation Testing
- I18N Testing
- dll

# 7 Software Testing Principles

# 7 Software Testing Principles

1. Testing shows presence of defects
2. Exhaustive testing is not possible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context dependent
7. Absence of errors fallacy