FULLSTACK WEB DEVELOPER

I. Introduction

A. Pengertian

Pengembangan fullstak (Full Stack development) adalah perembangan seluruh aplikasi secara end to end, dari mulai front end hingga ke backend.

B. Scope penting Full Stack Development

1. Front end Development

Menurut saya, front-end development berfokus pada pengalaman pengguna. Seperti desain antarmuka yang estetis dan responsif, serta interaktivitas. Penguasaan HTML, CSS, dan JavaScript merupakan hal yang standar terutama saat menggunakan framework modern seperti React atau Angular yang memudahkan pembuatan aplikasi web dinamis dan efisien.

2. Back-End Development

Back-end development adalah tulang punggung aplikasi, menangani logika bisnis, otorisasi, dan pemrosesan data. Pengembang back-end bertanggung jawab untuk membangun dan memelihara server, API, serta manajemen data yang aman dan efisien. Bahasa pemrograman seperti Node.js, Python, Ruby, dan Java sering digunakan untuk membangun back-end yang handal.

3. Database Management

Database management mengelola penyimpanan, pengambilan, dan manipulasi data yang efisien. Perannya sangat penting dalam memastikan data tersedia, aman, dan dapat diakses sesuai kebutuhan aplikasi. Pengembang harus menguasai sistem seperti SQL dan NoSQL, serta memastikan integrasi yang baik antara database dan aplikasi.

4. Mobile Development

Mobile development adalah proses pengembangan aplikasi yang dapat berjalan di perangkat mobile seperti smartphone dan tablet. Ini mencakup beberapa aspek penting:

- Platform: Umumnya aplikasi dikembangkan untuk dua platform utama, yaitu Android (menggunakan Java atau Kotlin) dan iOS (menggunakan Swift atau Objective-C).
- Framework: Untuk aplikasi lintas platform, framework seperti React Native dan Flutter memungkinkan pengembangan satu kode dasar yang berjalan di berbagai platform.

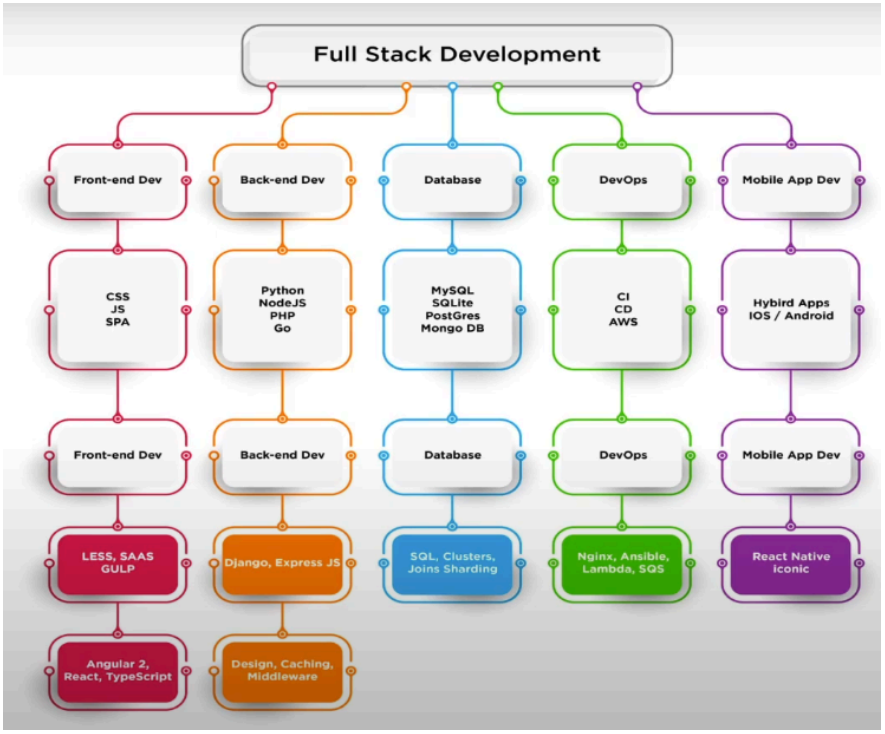
- IDE: Alat yang digunakan untuk pengembangan, seperti Android Studio untuk Android dan Xcode untuk iOS.

II. Skillset Full Stack Web / Mobile Developer

A. Pengembangan aplikasi End to End

merupakan pndekatan pengembangan perangkat lunak yang mencakup keseluruhan siklus pembuatan aplikasi, dari tahap perencanaan hingga tahap pengujian dan implementasi. Tujuannya adalah untuk menghasilkan aplikasi yang lengkap, fungsional, dan siap digunakan oleh pengguna akhir. Berikut tahapan-tahapan dalam pengembangan aplikasi end-to-end

1. Perencanaan : Identifikasi kebutuhan pengguna dan buat spesifikasi teknis untuk aplikasi.
2. Desain : Kembangkan wireframe dan prototipe UI/UX untuk visualisasi aplikasi.
3. Pengembangan Front-End : Bangun antarmuka pengguna menggunakan HTML, CSS, dan JavaScript atau framework mobile.
4. Pengembangan Back-End : Kembangkan logika bisnis, API, dan pengaturan server untuk aplikasi.
5. Integrasi dan Pengujian : Gabungkan front-end dan back-end, lalu lakukan pengujian fungsional dan non-fungsional untuk memastikan aplikasi berjalan dengan baik.
6. Implementasi : Luncurkan aplikasi ke pengguna, baik melalui App Store atau server produksi.
7. Pemeliharaan dan Peningkatan : Pantau performa aplikasi, perbaiki bug, dan lakukan pembaruan serta peningkatan berdasarkan umpan balik pengguna dan kebutuhan teknologi terbaru.



B. Kolaborasi Efektif

Version control (pengendalian versi) adalah sistem yang memungkinkan pengembang perangkat lunak untuk melacak perubahan pada kode sumber aplikasi selama pengembangan. Contohnya GIT dan mercurial.

Manfaat Version Control untuk Berkolaborasi

1. Rekam Perubahan

   Setiap kali pengembang membuat perubahan pada kocie sistem version control merekam detail perubahan tersebut

2. Pencatatan Riwayat

   Version control memungkinkan tim untuk melihat riwayat lengkap dari semua perubahan yang terjadi pada proyek dari awal hingga saat ini

3. Pemecahan Konflik

   Ketika dua atau lebih pengembang melakukan perubahan pada area kode yang sorna, version control membantu mengidentifikasi dan manyelesaikan konflik

4. Pemulihan Mudah

   Version control memungkinkan pengembang untuk memulihkan kode ke versi sebelumnya jika ada masalah atau bug yang terjad, sehingga mengurangi risiko kehilangan pekerjaan

III. Tools Full Stack Web/Mobile Developer

Berikut beberapa tools yang digunakan sebagai full stack developer

1. IDE - Code Editor

2. Version Control - repository

3. Version Control - Git Tools

4. DBMS

5. API

6. Test dan Debugging

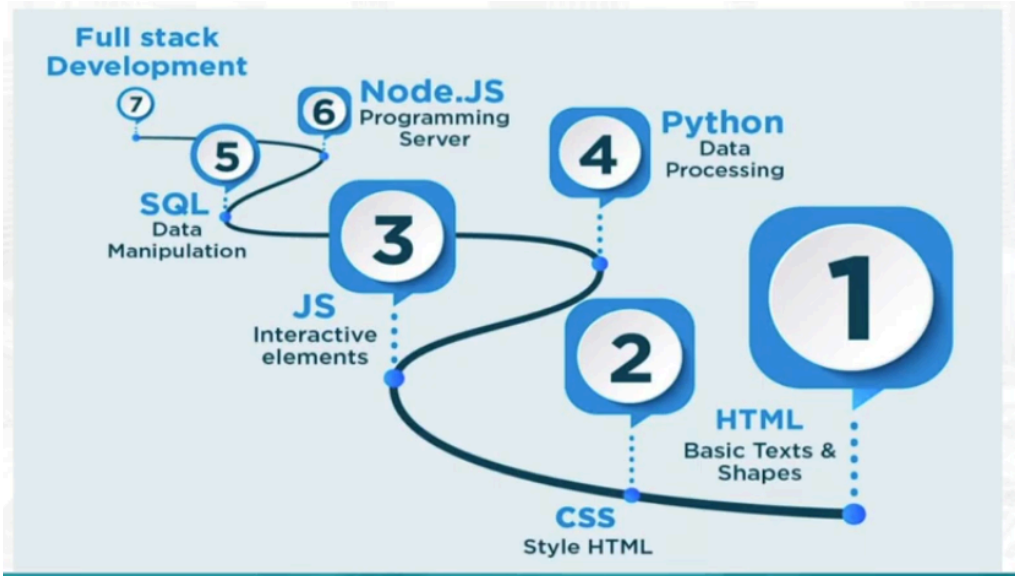
7. Mobile Development

8. Layanan Cloud

9. CI / CD

10. Desain UI/UX

Roadmap Full Stack Development



SDLC & DESAIN THINKING IMPLEMENTATION

I. Pengertian

SDLC (Siklus Hidup Pengembangan Perangkat Lunak) adalah rangkaian proses yang terstruktur dan metodologi yang digunakan untuk mengembangkan perangkat lunak dari awal hingga selesai. Berikut penjelasan singkat tahapan dalam SDLC (Software Development Life Cycle):

1. Perencanaan dan Analisis : Menentukan tujuan dan ruang lingkup proyek. Mengumpulkan dan mendokumentasikan kebutuhan fungsional dan non-fungsional perangkat lunak.
2. Desain Sistem : Membuat arsitektur perangkat lunak dan antarmuka pengguna.
3. Pengembangan : Menulis kode dan membangun perangkat lunak.
4. Pengujian : Memverifikasi bahwa perangkat lunak berjalan dengan benar dan bebas dari bug.
5. Implementasi: Mendistribusikan perangkat lunak kepada pengguna.
6. Pemeliharaan: Melakukan perbaikan dan pembaruan setelah peluncuran.

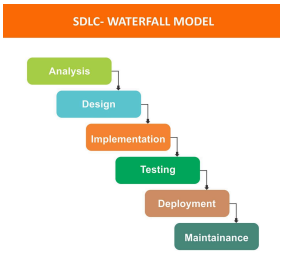
Berikut manfaat dari penggunaan SDLC:

1. Pengelolaan risiko yang llebih baik
2. efisiensi tim dan kolaborasi
3. memenuhi kebutuhan pengguna
4. penghematan biaya
5. peningkatan dokumentas

II. Model SDLC

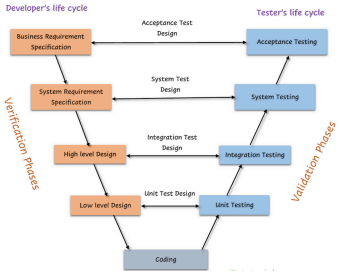
1. Waterfall Model

Cocok



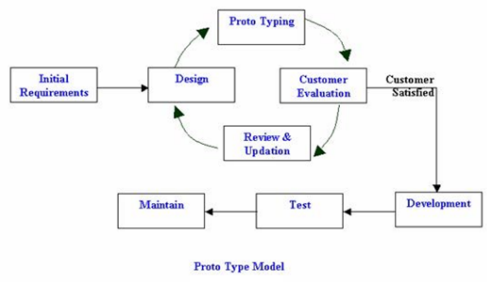
Proses linier di mana setiap tahap harus diselesaikan sebelum tahap berikutnya dimulai. untuk proyek yang terdefinisi dengan baik.

2. V-Model



Varian dari waterfall, dengan pengujian paralel di setiap tahap.

3. Prototype Model



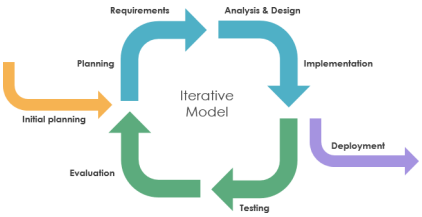
Proto Type Model

Mengembangkan prototipe awal untuk mendapat umpan balik pengguna sebelum pengembangan penuh

4. Spiral model

Berbasis iterasi dengan fokus pada identifikasi dan manajemen risiko. Setiap siklus terdiri dari perencanaan, analisis risiko, pengembangan, dan evaluasi.

5. Iterative Incremental Model



Model Perangkat lunak dikembangkan melalui pengulangan siklus, memungkinkan peningkatan bertahap.

6. Big Bang Model

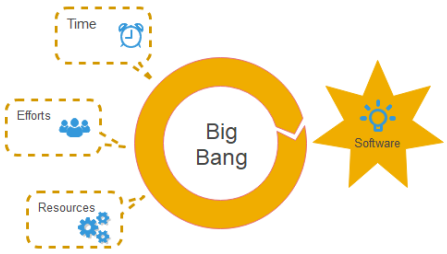
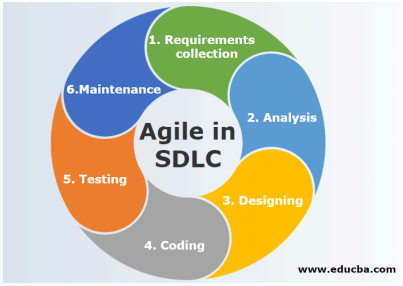


Fig. Big Bang Model

Tanpa perencanaan formal, semua sumber daya digunakan untuk pengembangan langsung hingga produk siap.

7. Agile



Fleksibel dan adaptif, fokus pada pengembangan bertahap melalui sprint. Cocok untuk proyek yang dinamis dan sering berubah.

III. Design Thinking Implementasi

Design Thinking*adalah metode pemecahan masalah yang berpusat pada pengguna, menggabungkan elemen inovasi, kreativitas, dan analisis. Berikut adalah 5 tahapan utama dalam Design Thinking:

1. Empathize (Empati)

Bertujuan Memahami kebutuhan, tantangan, dan perspektif pengguna melalui observasi dan interaksi langsung. Mencakup kegiatan sepert Wawancara pengguna, survei, pengamatan perilaku, dan studi kasus.

2. Define (Definisi)

Berujuan: Merumuskan masalah inti berdasarkan wawasan yang diperoleh dari tahap Empathize. Keluarannya Pernyataan masalah (problem statement) yang jelas dan terfokus pada kebutuhan pengguna.

3. Ideate (Ideasi)

Bertujuan untuk Menghasilkan berbagai ide solusi kreatif untuk masalah yang didefinisikan. Mencakup kegiatan seperti Brainstorming, mind mapping, atau teknik kreatif lainnya untuk mengeksplorasi semua kemungkinan solusi.

4. Prototype (Pembuatan Prototipe)

Bertujuan Membangun versi awal (prototipe) dari solusi yang dapat diuji, dengan fokus pada fungsi inti. Keluarannya model fisik atau digital dari solusi, yang sederhana namun fungsional.

5. Test (Pengujian)

Bertujuan Menguji prototipe dengan pengguna sebenarnya untuk mendapatkan umpan balik dan memperbaiki solusi. Kegiatannya yaitu Pengujian pengguna, observasi, dan iterasi berulang hingga solusi optimal tercapai.

6. Tahap Implementasi

Tahap **Implementasi** dalam kerangka **Design Thinking** merupakan tahap di mana solusi yang telah dikembangkan melalui prototipe dan pengujian diterapkan atau diimplementasikan dalam dunia nyata.

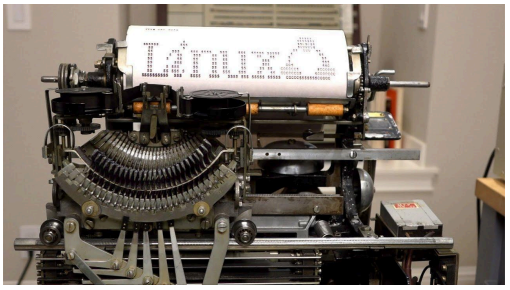
Basic Git & Collaborating Using Git

I. Terminal dan IDE

A. Sejarah

Generasi Terminal telah berkembang pesat seiring waktu:

1. Terminal Generasi Pertama (Teletype)



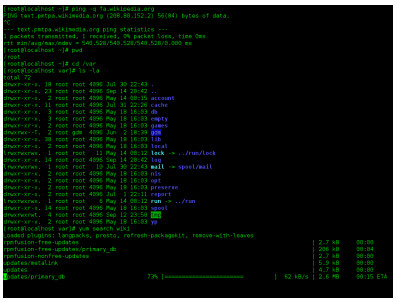
Pada 1920-an hingga 1960-an, terminal awal berupa teletype (TTY), yang mencetak output di atas kertas. Ini bekerja seperti mesin ketik dengan k

Terminal dan IDEabel.

2. Video Display Terminal (VDT)

Pada 1970-an, terminal berkembang menjadi tampilan visual, dengan layar yang dapat menampilkan teks, menggantikan teletype.

3. Command-Line Interfaces (CLI)



Pada 1980-an dan seterusnya, terminal menjadi perangkat lunak dengan shell seperti Bash atau DOS, memungkinkan pengguna mengirimkan perintah langsung ke sistem operasi.

4. Terminal Modern

   Saat ini, terminal adalah aplikasi perangkat lunak berbasis teks yang berjalan di sistem operasi modern seperti macOS, Linux, dan Windows, memungkinkan tugas-tugas administratif, scripting, dan pengelolaan sistem secara efisien.

Command Line Dasar

Perintah	Windows	Linux/macOS
Menampilkan direktori saat ini	`cd` *(change directory)*	`pwd` *(print working directory)*
Berpindah direktori	`cd [path]` *(change directory)*	`cd [path]` *(change directory)*
Naik satu direktori	`cd ..` *(change directory)*	`cd ..` *(change directory)*
Masuk ke direktori home	`cd` *(change directory)*	`cd ~` *(change directory)*
Menampilkan isi direktori	`dir` *(directory)*	`ls` *(list directory contents)*
Menampilkan isi secara detail	`dir /q` *(directory with owner info)*	`ls -l` *(list with long format)*
Membuat folder	`mkdir [nama_folder]` *(make directory)*	`mkdir [nama_folder]` *(make directory)*
Menghapus file	`del [nama_file]` *(delete)*	`rm [nama_file]` *(remove)*

	Menghapus folder	`rmdir [nama_folder]` *(remove directory)* (kosong) atau `rmdir /s [folder]` (isi)	`rm -r [nama_folder]` *(remove recursively)*
	Menyalin file	`copy [sumber] [tujuan]` *(copy)*	`cp [sumber] [tujuan]` *(copy)*
	Menyalin folder	`xcopy [sumber] [tujuan] /e` *(extended copy)*	`cp -r [sumber] [tujuan]` *(copy recursively)*
	Memindahkan file/folder	`move [sumber] [tujuan]` *(move)*	`mv [sumber] [tujuan]` *(move)*
	Membuat file baru	`type nul > [nama_file]` *(type nul)*	`touch [nama_file]` *(touch/create file)*
	Menampilkan isi file	`type [nama_file]` *(type)*	`cat [nama_file]` *(concatenate and display)*
	Menampilkan sebagian file	`more [nama_file]` *(more)*	`less [nama_file]` *(less)*
II. Installing GIT	Menampilkan 10 baris pertama	-	`head [nama_file]` *(head)*
	Menampilkan 10 baris terakhir	-	`tail [nama_file]` *(tail)*

Git adalah sistem kontrol versi (version control system) yang digunakan untuk melacak perubahan dalam kode sumber selama pengembangan perangkat lunak. Berikut cara instal GIT di Windows

1. Download dan install GIT pada link https://git-scm.com/downloads
2. Buka yang sudah di download. lalu, tekan next saja samapai terinstal
3. Lalu buka cmd, lalu ketik “git –version” . jika sudah ada tampilan version berapa itu tandanya git sudah berhasil di instal

Berikut adalah dasar-dasar **command Git** yang sering digunakan:

1. Inisialisasi Repositori (Membuat repositori baru)

```
git init
```

2. . Mengelola File

a. Menambahkan file ke staging area:

```
git add [nama_file]
```

b. Untuk menambahkan semua file yang telah dimodifikasi:

```
git add-
```

c. Menghapus file

```
git rm [nama_file]
```

3.Melakukan Commi

Menyimpan perubahan dengan pesan

```
git commit -m "Pesan commit"
```

4. Melihat Status dan Riwayat

a. Melihat status repositori

```
git status
```

b. Melihat riwayat commit

```
git log
```

5. Branching

a. Membuat branch baru:

```
git branch [nama_branch]
```

b. Beralih ke branch yang ada:

```
git checkout [nama_branch]
```

c. Membuat dan langsung beralih ke branch baru

```
git checkout -b [nama_branch]
```

6. Menggabungkan Branch

Menggabungkan branch lain ke branch saat ini

```
git merge [nama_branch]
```

7. Remote Repository

a. Menambahkan remote repository

```
git remote add origin [URL_repo]
```

b. Mengambil perubahan dari remote repository

```
git pull origin [nama_branch]
```

c. Mengirim perubahan ke remote repository:

```
git push origin [nama_branch]
```

8. Mengembalikan Perubahan

a. Mengembalikan file dari staging area ke working directory

```
git reset [nama_file]
```

b. Mengembalikan commit terakhir

```
git revert HEAD
```

9. Stash

a. Menyimpan perubahan yang belum di-commit

```
git stash
```

b. Mengambil kembali perubahan yang di-stash

```
git stash apply
```

10. Mendapatkan Bantuan

a. Mendapatkan bantuan untuk perintah Git:

```
git help
```

b. Atau untuk perintah tertentu:

```
git help [perintah]
```

III, Collaborating Using GIT

Berikut adalah langkah-langkah untuk berkolaborasi menggunakan Git di GitHub:

Step 1 : Membuat Repositori di GitHub

1. Login ke Akun GitHub dan Masuk ke akun GitHub Anda.
2. Buat Repositori Baru:

- Klik ikon `+` di pojok kanan atas.

- Pilih **New repository**

- Berikan nama repositori dan deskripsi (opsional).

- Pilih opsi publik atau privat.

- Klik **Create repository**

Step 2. Clone Repositori ke Lokal

1. Salin URL repositori dari GitHub:
2. Buka terminal dan jalankan:

```
git clone [URL_repositori]
```

Step 3 : Beralih ke Direktori Repositori

Pindah ke direktori repositori yang baru saja di-clone:

```
cd [nama_repositori]
```

Step 4: Buat Branch Baru untuk Fitur

Sebelum membuat perubahan, buat branch baru:

```
git checkout -b [nama_branch]
```

Step 5: Lakukan Perubahan

Buat perubahan pada kode atau file proyek sesuai kebutuhan.

Step 6: Menambahkan dan Meng-commit Perubahan

1. Tambahkan perubahan ke staging area:

```
git add [nama_file]  # atau gunakan git add .
```

2. Commit perubahan dengan pesan yang deskriptif:

```
git commit -m "Deskripsi perubahan yang dilakukan"
```

Step 7 : Mengambil Perubahan Terbaru dari Remote

Sebelum mengirim perubahan, ambil update terbaru dari branch target (misalnya `main`):

```
git pull origin main
```

Step 8: Mengirim Perubahan ke Remote

Kirim branch baru yang sudah diubah ke remote:

```
git push origin [nama_branch]
```

Step 9 : Membuat Pull Request (PR) di GitHub

1. Buka Repositori di GitHub

   - Setelah push, buka repositori di GitHub.

2. Buat Pull Request:

   - Anda akan melihat notifikasi untuk **Compare & pull request.** Klik tombol tersebut.
   - Berikan deskripsi yang jelas tentang perubahan yang Anda lakukan.
   - Klik **Create pull request.**

Step 10: Review dan Diskusi

- Kolaborator lain akan melakukan review terhadap PR.

- Diskusikan jika ada umpan balik atau perubahan yang perlu dilakukan.

Step 11: Merge Pull Request

- Setelah disetujui, merge PR ke branch utama:
- Klik tombol **Merge pull request**
- Konfirmasi dengan mengklik **Confirm merge.**

Step 12: Menghapus Branch Setelah Merge

- Setelah merge, Anda dapat menghapus branch yang sudah tidak diperlukan:

- Klik **Delete branch** setelah merge selesai.

Step 13: Sinkronisasi dengan Branch Utama

Beralih ke branch utama dan ambil perubahan terbaru:

```
git checkout main

git pull origin main
```