**Materi Pengembangan Full Stack**

* Pengembangan Full Stack mencakup seluruh aplikasi, dari front-end hingga back-end, dan dalam beberapa kasus, hingga sisi klien.
* Lingkup Penting Full Stack Development: Front-End Development: Membangun antarmuka pengguna menggunakan HTML, CSS, dan JavaScript.
* Back-End Development: Membangun server dan logika bisnis aplikasi menggunakan bahasa pemrograman server-side dan framework.
* Database Management: Mendesain dan mengelola basis data.
* Integrasi Front-End dan Back-End: Menghubungkan komponen front-end dengan layanan back-end melalui API.
* Version Control and Collaboration: Mengelola perubahan kode dan kolaborasi tim menggunakan sistem pengendalian versi seperti Git.
* Mobile Development: Pengembangan aplikasi mobile dengan menggunakan framework seperti React Native atau Flutter.

Dasar-Dasar Frontend Web Development

* Frontend Web Development adalah praktik pembuatan HTML, CSS, dan JavaScript untuk membuat tampilan dan interaksi pada situs web.
* HTML adalah bahasa yang digunakan untuk mendefinisikan struktur konten web.
* CSS digunakan untuk tata letak dan gaya halaman web.
* JavaScript digunakan untuk membuat halaman web menjadi interaktif.

Framework Front-End Populer

* Framework seperti React, Angular, dan Vue.js digunakan untuk mempercepat pengembangan front-end.

Dasar-Dasar Backend Development

* Backend Development bertanggung jawab untuk memproses permintaan dari pengguna, mengelola data di database, dan memberikan respons kepada klien.
* Bahasa pemrograman server-side seperti Node.js, Python, Ruby, Java, PHP, dan C# digunakan.
* Framework seperti Express.js, Flask, Ruby on Rails, Spring, dan Laravel membantu dalam pengembangan backend.

Framework Backend Populer

* Framework seperti Express.js untuk Node.js, Flask untuk Python, Ruby on Rails untuk Ruby, Spring untuk Java, dan Laravel untuk PHP sangat populer.

Dasar-Dasar Database Management

* Database Management melibatkan pengelolaan data dalam aplikasi atau sistem.
* Ada database relasional (SQL) seperti MySQL, PostgreSQL, dan database non-relasional (NoSQL) seperti MongoDB.

Framework Mobile Development

* Pengembang Full Stack dapat mengembangkan aplikasi mobile menggunakan framework seperti React Native atau Flutter.

Version Control and Collaboration

* Penggunaan sistem pengendalian versi seperti Git memungkinkan kolaborasi tim yang efektif dalam pengembangan.

Tools Set Sebagai Full Stack Developer

* Full Stack Developer memerlukan berbagai alat seperti IDE (Code Editor), Version Control (Git), DBMS, API, alat pengujian, alat pengembangan mobile, layanan cloud, CI/CD, alat desain UI/UX, dan lainnya.

**Software Development Life Cycle (SDLC) dan penerapannya dalam Design Thinking.**

* SDLC adalah serangkaian tahap dalam pengembangan perangkat lunak, termasuk perencanaan, analisis, desain, pengembangan, pengujian, penerapan, dan pemeliharaan. Ada berbagai model SDLC seperti Waterfall, Agile, dan lainnya, yang dipilih sesuai dengan jenis proyek dan kebutuhan organisasi.
* Design Thinking adalah pendekatan kreatif untuk memahami kebutuhan pengguna dan menciptakan solusi yang lebih baik melalui tahap empati, definisi masalah, ideasi, pembuatan prototipe, pengujian, dan implementasi.
* Integrasi Design Thinking dalam SDLC dapat membantu menciptakan produk yang lebih berorientasi pada pengguna, intuitif, dan responsif terhadap perubahan kebutuhan pengguna dan pasar.

**Basic Git & Collaborating Using Git**

Git adalah sistem kontrol versi yang digunakan untuk mengelola perubahan dalam kode sumber perangkat lunak.

Diciptakan oleh Linus Torvalds pada tahun 2005 dan sangat populer di kalangan pengembang perangkat lunak. Git membantu dalam melacak perubahan kode sumber dan memungkinkan kolaborasi yang efisien dalam pengembangan perangkat lunak.

Ini memungkinkan tim untuk bekerja bersama secara paralel pada proyek yang sama tanpa konflik.

Mengenal Istilah dalam Git:

* Repository (Repo): Tempat penyimpanan proyek Git.
* Commit: Perubahan kode yang telah disimpan dalam repositori.
* Branch: Cabang independen dari repositori yang memungkinkan pengembang untuk bekerja secara terpisah.
* Pull Request (PR): Permintaan untuk menggabungkan perubahan dari satu cabang ke cabang lain.
* Merge: Proses menggabungkan perubahan dari satu cabang ke cabang lain.
* Clone: Menyalin repositori Git ke mesin lokal.
* Fork: Membuat salinan repositori di akun GitHub pribadi.

Langkah-langkah Dasar dalam Penggunaan Git:

* Inisialisasi Repositori: git init untuk membuat repositori baru.
* Menambahkan Berkas ke Staging Area: git add <nama\_file> untuk memantau perubahan.
* Membuat Commit: git commit -m "Pesan commit" untuk menyimpan perubahan.
* Melihat Riwayat Commit: git log untuk melihat daftar commit.
* Berpindah ke Cabang Lain: git checkout <nama\_cabang> untuk bekerja di cabang yang berbeda.
* Menggabungkan Cabang: git merge <nama\_cabang> untuk menggabungkan perubahan.
* Push ke Repositori Jarak Jauh: git push untuk mengirim perubahan ke repositori di server.

Kolaborasi Menggunakan Git:

* Fork Repository: Buat salinan repositori di akun GitHub pribadi Anda.
* Clone Repository: Salin repositori ke mesin lokal Anda.
* Buat Cabang Baru: Buat cabang baru untuk mengerjakan fitur atau perbaikan.
* Commit & Push: Buat commit dan dorong perubahan ke repositori GitHub pribadi.
* Buat Pull Request: Buat PR untuk mengajukan perubahan ke repositori asli.
* Review & Diskusi: Tim dapat me-review PR dan berdiskusi tentang perubahan.
* Merge Pull Request: Setelah perubahan disetujui, merge PR ke repositori asli.

Tips untuk Penggunaan Git yang Efisien:

* Selalu berikan pesan commit yang jelas.
* Gunakan cabang untuk mengisolasi perubahan.
* Sering commit untuk menghindari commit yang terlalu besar.
* Selalu perbarui repositori lokal Anda sebelum memulai pekerjaan baru.