Summary Materi Software Engineer

A. Full Stack Web Developer Career Path

- Belajar Dasar Pemrograman : Mulailah dengan mempelajari dasar-dasar pemrograman, seperti HTML, CSS, dan JavaScript untuk front-end. Untuk back-end, bisa belajar bahasa seperti Python, Java, atau Node.js. Ini penting untuk memahami bagaimana komponen dasar sebuah aplikasi bekerja.

- Menjadi Front-End Developer : Langkah berikutnya bisa fokus pada pengembangan bagian antarmuka pengguna (front-end). Pelajari framework seperti React atau Angular, dan bagaimana membuat tampilan yang menarik dan responsif.

- Belajar Back-End Development : Setelah paham front-end, pelajari back-end. Ini melibatkan kerja dengan server, database, dan API. Pelajari framework seperti Express (untuk Node.js) atau Django (untuk Python). Anda juga harus mengerti cara kerja database seperti MySQL atau MongoDB.

- Menguasai Full Stack Tools : Setelah memahami front-end dan back-end, gabungkan pengetahuan tersebut. Pelajari tools yang digunakan dalam pengembangan full stack, seperti Git untuk version control, Docker untuk containerization, dan CI/CD untuk otomatisasi deployment.

B. Basic Git & Collaborating Using Git

1. Apa itu Git?
   Git adalah sistem version control yang membantu melacak perubahan pada kode atau file dalam proyek. Ini sangat berguna untuk bekerja secara tim, karena memungkinkan banyak orang untuk mengerjakan satu proyek tanpa kehilangan atau merusak pekerjaan orang lain. Setiap perubahan dicatat, jadi kita bisa kembali ke versi sebelumnya jika diperlukan.

2. Dasar-Dasar Git Repository (Repo): Tempat di mana semua file proyek dan riwayat perubahan disimpan. Repo bisa berada di komputer lokal atau di server seperti GitHub.
   Commit: Tindakan menyimpan snapshot perubahan kode ke dalam repo. Setiap commit memiliki pesan yang menjelaskan apa yang telah diubah.
   Branch: Cabang dari proyek utama yang memungkinkan pengembangan fitur atau perbaikan secara terpisah. Branch bisa di-merge kembali ke branch utama setelah selesai.
   Merge: Proses menggabungkan perubahan dari satu branch ke branch lainnya.

3. Perintah Dasar Git
   git init: Membuat repo baru di folder saat ini.
   git clone [URL]: Mengunduh (menyalin) repo dari server, misalnya dari GitHub.
   git add [nama_file]: Menandai file yang akan di-commit.

git commit -m "pesan": Menyimpan perubahan yang telah di-add dengan pesan deskriptif.
git status: Mengecek status repo, apakah ada perubahan yang belum di-commit.
git push: Mengirim commit dari repo lokal ke repo di server (GitHub, GitLab, dll).
git pull: Menarik perubahan terbaru dari repo di server ke repo lokal.

4. Kolaborasi dengan Git
   Kolaborasi menggunakan Git sangat berguna untuk tim yang bekerja pada proyek bersama. Berikut adalah langkah-langkah umum saat bekerja sama menggunakan Git:
   a. Cloning Repo
      Setiap anggota tim mulai dengan clone repo dari server (misalnya GitHub) ke komputer lokal masing-masing menggunakan git clone.

   b. Membuat Branch
      Untuk mengerjakan fitur baru atau memperbaiki bug, kita membuat branch baru menggunakan git checkout -b nama_branch. Ini memisahkan pekerjaan kita dari branch utama (main atau master).

   c. Menambah dan Commit Perubahan
      Saat selesai mengerjakan sesuatu, tambahkan file yang diubah ke staging menggunakan git add lalu simpan perubahan dengan git commit -m "pesan perubahan".

   d. Push Perubahan ke Server
      Setelah commit, kita bisa mengirim perubahan ke repo di server dengan git push origin nama_branch.

   e. Pull Request (PR)
      Jika menggunakan layanan seperti GitHub, kita bisa mengajukan pull request (PR). Ini adalah permintaan untuk menggabungkan perubahan kita dari branch yang sedang dikerjakan ke branch utama. Tim lain akan memeriksa PR sebelum menggabungkannya ke main branch.

   f. Menggabungkan (Merge) Perubahan
      Setelah PR disetujui, perubahan bisa di-merge ke branch utama. Gunakan git merge untuk menggabungkan branch secara lokal atau melalui antarmuka GitHub.
   g. Menarik (Pull) Perubahan Terbaru
      Saat anggota tim lain melakukan perubahan, kita perlu memperbarui repo lokal dengan perubahan mereka menggunakan git pull. Ini penting agar semua anggota tim bekerja pada kode yang paling baru.

5. Mengatasi Konflik

Terkadang, saat dua orang mengubah bagian yang sama dari kode, bisa terjadi konflik. Git akan meminta kita untuk memilih perubahan mana yang akan dipertahankan. Untuk mengatasinya, buka file yang bermasalah, lihat perubahan yang bertentangan, lalu edit manual sebelum melakukan commit ulang.

Contoh Alur Kolaborasi:

- Clone repo: git clone [URL_repo]
- Buat branch baru: git checkout -b fitur-baru
- Lakukan perubahan, lalu:
- git add [file]
- git commit -m "menambahkan fitur baru"
- Push branch: git push origin fitur-baru
- Buat Pull Request di GitHub dan tunggu review dari tim.
- Setelah review, merge ke main branch.