**Summary Pengenalan Software Enginering**

**Hilham Yulistio**

1. Silahkan teman teman melakukan proses instalasi beberapa tools dibawah ini :

- Git

- Visual Studio Code

- Browser

Setelah melakukan instalasi, buat sebuah summary untuk apa yang sudah dipelajari dalam materi pengenalan software engineer sebelumnya pada google docs.

**Fullstack Web Developer Career Path :**

Seorang **fullstack developer** bertanggung jawab untuk mengembangkan aplikasi dari ujung ke ujung, yaitu dari **frontend** (bagian yang dilihat pengguna), **backend** (logika server dan database), hingga pemahaman dasar tentang **data science**. Mereka memiliki kemampuan yang luas dalam berbagai aspek pengembangan aplikasi web.

1. **Frontend (Bagian Pengguna)**:  
   Ini adalah bagian yang berinteraksi langsung dengan pengguna, seperti tampilan website atau aplikasi. Fullstack developer harus menguasai teknologi seperti **HTML, CSS**, dan **JavaScript** serta framework modern seperti **React** atau **Vue.js** untuk membuat tampilan yang menarik, responsif, dan mudah digunakan.
2. **Backend (Server dan Database)**:  
   Bagian backend berfokus pada logika server, penyimpanan data, dan komunikasi antara server dan frontend. Seorang fullstack developer harus menguasai bahasa pemrograman server seperti **Node.js, Python**, atau **Ruby**, serta memahami cara kerja database seperti **MySQL** (relasional) atau **MongoDB** (NoSQL) untuk menyimpan dan mengelola data.
3. **Data Science**:  
   Meskipun bukan fokus utama, pemahaman dasar tentang **data science** berguna bagi fullstack developer, terutama dalam menangani analisis data atau mengintegrasikan fitur berbasis data. Pengetahuan tentang **Python** untuk analisis data atau menggunakan alat seperti **Pandas** dan **NumPy** bisa membantu dalam mengolah data yang lebih kompleks.

Dengan kemampuan dalam frontend, backend, dan sedikit data science, fullstack developer dapat mengembangkan aplikasi web secara menyeluruh, mulai dari antarmuka pengguna hingga manajemen data di server.

**SDLC & Design Thinking Implementation :**

**SDLC (Software Development Life Cycle)** adalah proses terstruktur untuk mengembangkan perangkat lunak. SDLC membantu memastikan perangkat lunak dibuat dengan efisien dan berkualitas tinggi. Ada beberapa tahapan utama dalam SDLC:

1. **Perencanaan (Planning):** Menentukan tujuan dan kebutuhan proyek.
2. **Analisis (Analysis):** Mengumpulkan dan menganalisis kebutuhan dari pengguna atau klien.
3. **Desain (Design):** Membuat arsitektur dan desain sistem berdasarkan analisis.
4. **Pengembangan (Development):** Menulis kode dan membangun perangkat lunak.
5. **Pengujian (Testing):** Memeriksa perangkat lunak untuk memastikan tidak ada bug dan sesuai dengan kebutuhan.
6. **Implementasi (Implementation):** Menginstal dan menjalankan perangkat lunak di lingkungan pengguna.
7. **Pemeliharaan (Maintenance):** Memperbarui dan memperbaiki perangkat lunak setelah digunakan.

**Manfaat SDLC:**

* Mengurangi risiko kesalahan dalam pengembangan
* Memastikan perangkat lunak sesuai kebutuhan
* Meningkatkan kualitas perangkat lunak
* Meminimalkan biaya dan waktu pengembangan

**Basic Git & Collabortaion Using Git :**

1. **Git dan Cara Kerja di Terminal dan IDE**
   * **Git** adalah sistem kontrol versi (VCS) untuk melacak perubahan kode.
   * Git dapat digunakan melalui **terminal** atau **IDE** (seperti VS Code, JetBrains, dll.).
   * Di **terminal**, gunakan perintah seperti git status, git add, dan git commit untuk mengelola versi.
   * Di **IDE**, biasanya ada integrasi Git yang memudahkan untuk melakukan perubahan tanpa perlu perintah manual.
2. **Instalasi Git**
   * Unduh Git dari situs resmi Git.
   * Setelah instalasi, buka terminal dan ketik git --version untuk memeriksa apakah Git sudah terpasang dengan benar.
3. **Menginisialisasi dan Meng-commit di Git**
   * Menginisialisasi repositori: git init
   * Menambahkan file ke staging area: git add <nama-file>
   * Commit perubahan dengan pesan: git commit -m "Pesan commit"
4. **Kolaborasi Menggunakan Git**
   * Biasanya menggunakan platform seperti GitHub atau GitLab.
   * Untuk bekerja secara kolaboratif:
     + Clone repository: git clone <url-repo>
     + Setelah mengedit, gunakan git add dan git commit untuk menyimpan perubahan.
     + Push perubahan ke server: git push origin <branch>
     + Pull perubahan dari rekan tim: git pull origin <branch>

Dengan Git, pengembang dapat melacak perubahan dan berkolaborasi lebih mudah dalam satu proyek.

1. Soal & Instruksi - Collaborating Using Git

Buatlah repositori GitHub baru dengan nama "finpro-msib-7-kelompok X"

Contoh : finpro msib 7 kelompok 1

\*1 kelompok hanya membuat 1 repositori saja

● Undang teman-teman kelompokmu sebagai kolaborator pada repositori tersebut

● Setiap peserta meng-clone repositori ke mesin lokal masing-masing

● Setiap peserta harus membuat cabang untuk pekerjaannya. Misal : doni-homework, nita-homework, etc

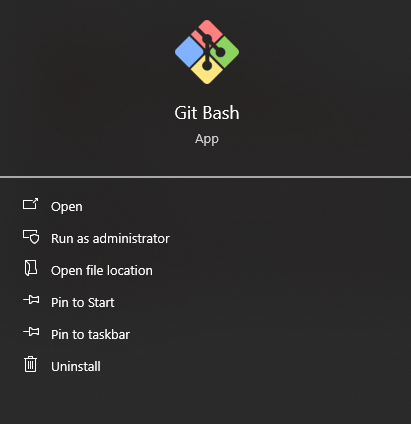
● Masukkan Homework dari minggu sebelumnya ke branch lokalmu.

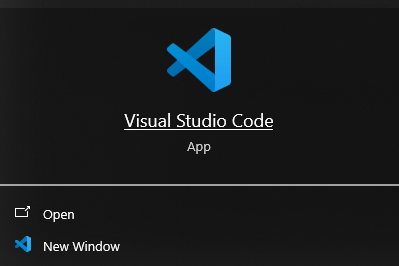
● Setelah melakukan perubahan, teman-teman harus melakukan commit dengan pesan commit yang deskriptif

● Lakukan push and pull ke branch homework dari branch lokalmu.

● Pastikan konflik sudah terselesaikan sebelum melakukan pull request

Setelah selesai, bisa teman teman zip hasil soal pertama dan kedua, lalu upload ke dalam LMS Rakamin.



****

**Link Repo :** <https://github.com/GusSurya031/finpro-msib-7-kelompok-1>