**Summary Of Introduction To Software Engineering**

**Nama: Andika Moza Varian**

****

1. **Fullstack Web Developer Career Path**

Pada Materi ini dijelaskan bahwa untuk menjadi Full Stack Developer secara umum harus mempunyai beberapa keterampilan sebagai berikut:

1. **Frontend Developer**

Pada tahap ini dikenal sebagai pengembangan yang bakal terlihat dari sisi klien. Dasar-dasar dari frontend adalah sebagai berikut:

* **HTML**: Struktur dasar halaman web.
* **CSS**: Styling dan layout untuk elemen HTML.
* **JavaScript**: Bahasa pemrograman untuk interaktivitas di halaman web.

Beberapa framework frontend yang populer dan banyak digunakan dalam pengembangan web saat ini:

* **React:** Library JavaScript untuk membangun antarmuka pengguna. Dikembangkan oleh Facebook.
* **Vue.js:** Framework progresif yang fokus pada pengembangan antarmuka pengguna. Mudah diintegrasikan dengan proyek lain.
* **Angular:** Framework yang dikembangkan oleh Google untuk membangun aplikasi web yang dinamis dan kompleks.

1. **Backend Developer**

Pada fase ini lebih berfungsi sebagai penanggung jawab untuk memproses permintaan pengguna, dan mengelola atau menyimpan database. Untuk menjadi backend developer ada hal-hal yang harus dipelajari seperti:

1. **Bahasa Pemrograman**

* **JavaScript (Node.js)**: Digunakan untuk membuat aplikasi backend menggunakan JavaScript.
* **Python**: Dengan framework seperti Django atau Flask.
* **Java**: Banyak digunakan dalam aplikasi enterprise dengan Spring Framework.
* **Ruby**: Terkenal dengan Ruby on Rails untuk pengembangan cepat.
* **PHP**: Digunakan di banyak situs web, terutama dengan Laravel.

### **Database**

### **SQL**: Memahami cara menggunakan database relasional seperti MySQL, PostgreSQL, SQLite, Oracle.

### **NoSQL**: Mengerti database seperti MongoDB atau Firebase untuk data yang tidak terstruktur.

Full Stack Developer juga harus memahami dasar tentang Mobile Development termasuk IDE dan Bahasa Pemrograman Android (Java/Kotlin) serta iOS (Swift/Objective-C).

**B. Tahap-Tahap Pengembangan Aplikasi End-To-End**

Untuk membuat sebuah aplikasi end-to-end maka ada beberapa tahapan yang bisa dilakukan, meliputi:

* **Perencanaan dan Analisis:** Mengumpulkan informasi tentang kebutuhan pengguna dan tujuan aplikasi.
* **Desain**

1. **UI/UX**: Membuat wireframe dan prototipe antarmuka pengguna untuk memastikan pengalaman pengguna yang baik.
2. **Database**: Mendesain skema database dan hubungan antar tabel.

* **Pengembangan**

1. **Frontend**: Membangun antarmuka pengguna menggunakan teknologi seperti HTML, CSS, dan JavaScript.
2. **Backend**: Membangun server, database, dan API menggunakan bahasa pemrograman dan framework yang sesuai.
3. **Integrasi**: Menghubungkan frontend dan backend serta memastikan komunikasi yang lancar.

* **Pengujian**

1. **Fungsional**: Memastikan semua fitur bekerja sesuai spesifikasi.
2. **UX**: Mengumpulkan umpan balik dari pengguna untuk meningkatkan pengalaman.

### **Peluncuran**

1. **Deployment**: Mengunggah aplikasi ke server atau cloud.
2. **Monitoring**: Memantau aplikasi untuk mendeteksi bug atau masalah performa.

### **Pemeliharaan dan Pembaruan**

1. **Perbaikan Bug**: Memperbaiki masalah yang muncul setelah peluncuran.
2. **Pembaruan Fitur**: Menambahkan fitur baru berdasarkan umpan balik pengguna dan kebutuhan pasar.

* **Evaluasi dan Iterasi**

1. **Pengumpulan Umpan Balik**: Mengumpulkan umpan balik dari pengguna untuk perbaikan di masa mendatang.
2. **Iterasi**: Mengulangi proses untuk meningkatkan aplikasi berdasarkan analisis dan umpan balik.

**C. Kolaborasi Efektif Menggunakan GIT**

Sebelum melakukan kolaborasi menggunakan GIT ada beberapa hal yang harus dipahami, yaitu: Terminal, dan Terminal Command nya.

**Terminal** adalah antarmuka baris perintah yang memungkinkan pengguna untuk berinteraksi dengan sistem operasi melalui perintah teks.

Berikut adalah beberapa **command terminal** sederhana yang sering digunakan dalam pengembangan dan manajemen sistem:

* pwd: Menampilkan direktori saat ini.
* ls: Menampilkan daftar file dan direktori di dalam direktori saat ini.
* cd <nama\_direktori>: Masuk ke direktori yang ditentukan.
* cd ..: Kembali ke direktori sebelumnya.
* mkdir <nama\_direktori>: Membuat direktori baru.
* rmdir <nama\_direktori>: Menghapus direktori kosong.
* rm <nama\_file>: Menghapus file.
* cp <sumber> <tujuan>: Menyalin file dari sumber ke tujuan.
* mv <sumber> <tujuan>: Memindahkan atau mengubah nama file/direktori.

**Pengertian GIT**

**Git** adalah tools *version control system* (VCS) yang digunakan untuk mengelola perubahan kode selama pengembangan perangkat lunak.

**GitHub** adalah platform berbasis web yang digunakan untuk pengelolaan kode sumber dan kolaborasi dalam pengembangan perangkat lunak.

**Cara Install GIT**

Instalasi Git di Windows memang tidak seperti di Linux yang ketik perintah langsung terinstal. Harus mendownload terlebih dahulu, dengan cara membuka website resminya Git ( [git-scm.com](https://git-scm.com/)). Kemudian unduh Git sesuai dengan arsitektur komputer. Kalau menggunakan 64 bit, unduh yang 64 bit. kemudian melakukan ritual *next>next>finish*.

# **Command GIT**

# git init: Membuat repositori Git baru di direktori saat ini.

* git clone <url>: Mengkloning repositori dari URL ke dalam direktori lokal.
* git status: Menampilkan status file yang telah dimodifikasi, ditambahkan, atau dihapus.
* git config --global user.name "<nama>": Mengatur nama pengguna untuk semua repositori.
* git config --global user.email "<email>": Mengatur email pengguna untuk semua repositori.
* git add <file>: Menambahkan file ke staging area untuk persiapan commit.
* git commit -m "<pesan>": Mencatat perubahan yang ada di staging area dengan pesan deskriptif.
* git commit --amend: Mengedit commit terakhir (baik menambah perubahan atau mengganti pesan).
* git branch: Menampilkan daftar semua branch yang ada.
* git branch <nama\_branch>: Membuat branch baru.
* git checkout <nama\_branch>: Beralih ke branch yang ditentukan.
* git merge <nama\_branch>: Menggabungkan branch yang ditentukan ke branch saat ini.
* git branch -d <nama\_branch>: Menghapus branch yang ditentukan.
* git remote -v: Menampilkan daftar remote yang terhubung dengan repositori.
* git fetch <remote>: Mengambil perubahan terbaru dari remote tanpa menggabungkannya.
* git pull: Mengambil dan menggabungkan perubahan dari remote ke branch saat ini.
* git push <remote> <branch>: Mengunggah commit yang ada di branch saat ini ke remote.
* git log: Menampilkan riwayat commit untuk branch saat ini.
* git diff: Menampilkan perubahan yang belum di-stage.
* git diff --staged: Menampilkan perubahan yang sudah di-stage tetapi belum di-commit.
* git mergetool: Membuka alat untuk membantu menyelesaikan konflik merge.
* git rebase: Menerapkan perubahan dari satu branch ke branch lain secara berurutan, menjaga riwayat lebih bersih.
* git tag <nama\_tag>: Membuat tag untuk menandai titik tertentu dalam riwayat.
* git push origin --tags: Mengunggah semua tag ke remote.

**Berkolaborasi Menggunakan Git & GitHub**

Berikut adalah langkah-langkah untuk berkolaborasi menggunakan GitHub:

### **1. Membuat atau Mengkloning Repositori**

* **Buat Repositori Baru**: Buat repositori baru di GitHub.
* **Kloning Repositori**: Jika ingin berkolaborasi pada repositori yang sudah ada, gunakan perintah: git clone <url\_repositori>

### **2. Membuat Branch**

**Buat Branch Baru**: Selalu kerjakan fitur atau perbaikan bug di branch terpisah.  
gunakan perintah: git checkout -b <nama\_branch>

### **3. Melakukan Perubahan dan Commit**

* **Menambahkan Perubahan**: Tambahkan file yang diubah ke staging area.  
  gunakan perintah: git add <file>
* **Commit Perubahan**: Simpan perubahan dengan pesan yang deskriptif.  
  gunakan perintah: git commit -m "Deskripsi perubahan"

### **4. Push Branch ke Remote**

* **Mengunggah Branch**: Setelah commit, unggah branch baru ke repositori remote.  
  gunakan perintah: git push origin <nama\_branch>

### **5. Membuat Pull Request (PR)**

* **Buka GitHub**: Kunjungi repositori di GitHub.
* **Buat Pull Request**: Klik tombol "Compare & pull request" untuk branch yang baru saja di-push.
* **Deskripsikan Perubahan**: Tambahkan deskripsi yang jelas tentang apa yang telah diubah dan mengapa.
* **Ajukan PR**: Kirim pull request untuk ditinjau oleh anggota tim lain.

### **6. Ulasan dan Diskusi**

* **Ulasan Kode**: Tim lain dapat mengulas kode, memberikan komentar, dan meminta perubahan jika diperlukan.
* **Diskusikan Umpan Balik**: Diskusikan umpan balik dan lakukan perubahan jika diperlukan. Commit perubahan baru di branch yang sama.

### **7. Menggabungkan Pull Request**

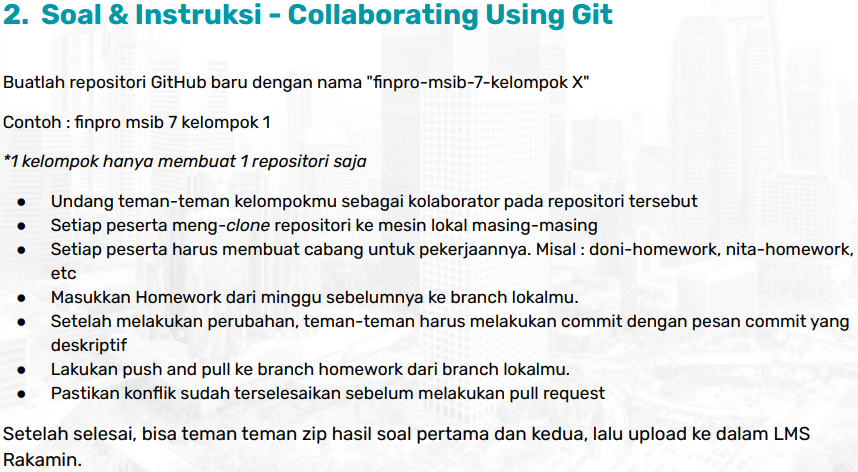
* **Merge PR**: Setelah mendapatkan persetujuan, kamu atau pemilik repositori dapat menggabungkan PR ke branch utama (biasanya main atau master).
* **Hapus Branch**: Setelah digabungkan, kamu bisa menghapus branch yang tidak lagi diperlukan, baik di lokal maupun remote.

### **8. Sinkronisasi dengan Branch Utama**

**Ambil Perubahan Terbaru**: Secara teratur ambil perubahan terbaru dari branch utama untuk memastikan bahwa kode tetap sinkron.  
gunakan perintah: git pull origin main

### **9. Komunikasi**

* **Gunakan Isu (Issues)**: Gunakan fitur isu untuk melacak bug, permintaan fitur, dan diskusi lainnya.
* **Gunakan Alat Kolaborasi**: Pertimbangkan untuk menggunakan alat komunikasi seperti Slack, Discord, atau Trello untuk koordinasi tim.



Link Repository Kelompok 2 Final Projek: <https://github.com/yulia30359/finpro-msib-7-kelompok-2.git>