**Summary Pengenalan Software Enginering**

****

**ILHAM SAPUTRA**

**IT FULL STACK DEVELOPER**

**MSIB 7**

**1. Fullstack Web Developer**

Seorang **Fullstack Web Developer** adalah pengembang serba bisa yang mahir dalam berbagai aspek pengembangan web, baik di sisi **front-end** maupun **back-end**. Dengan kemampuan di kedua sisi, seorang fullstack developer mampu menciptakan aplikasi web secara end-to-end, mulai dari antarmuka pengguna (UI) yang menarik hingga logika bisnis yang kompleks dan pengelolaan database di server.

**a. Front-end Development**

Bagian front-end bertanggung jawab atas apa yang dilihat dan digunakan oleh pengguna. Tugas utamanya adalah memastikan pengalaman pengguna yang intuitif, cepat, dan visual menarik.

* **HTML (HyperText Markup Language)**: Bahasa dasar untuk membangun struktur halaman web. Semua elemen visual yang dilihat pengguna seperti teks, gambar, dan tautan didefinisikan dalam HTML.
* **CSS (Cascading Style Sheets)**: Digunakan untuk memperindah tampilan halaman web. CSS mengatur warna, tata letak, tipografi, dan animasi yang memberikan gaya unik pada sebuah halaman web.
* **JavaScript**: Bahasa pemrograman yang menghidupkan halaman web. Dengan JavaScript, developer bisa membuat elemen dinamis seperti drop-down menu, slideshow, dan berbagai interaksi lainnya.
* **Frameworks dan Libraries**: Seperti **React.js**, **Angular**, dan **Vue.js** yang mempermudah pengembangan aplikasi berbasis JavaScript dengan menyediakan struktur dan komponen yang telah disederhanakan.

**b. Back-end Development**

Di balik layar, bagian back-end menangani logika bisnis, manajemen data, dan keamanan aplikasi. Back-end developer bertanggung jawab atas server, database, dan API yang menghubungkan aplikasi dengan pengguna.

* **Bahasa Pemrograman**: Bahasa seperti **Node.js (JavaScript di server-side)**, **Python (Django, Flask)**, **Ruby on Rails**, dan **PHP** digunakan untuk mengembangkan logika yang memproses permintaan pengguna.
* **Database**: Untuk menyimpan dan mengambil data pengguna, sistem membutuhkan database. Pilihan populer mencakup **MySQL**, **PostgreSQL**, dan **MongoDB**.
* **Server dan API**: Seorang back-end developer bekerja dengan server untuk mengatur hosting dan deployment, serta mengintegrasikan **API (Application Programming Interface)** yang memungkinkan komunikasi antar aplikasi.

**Keuntungan Fullstack Developer**

Keuntungan dari seorang fullstack developer adalah fleksibilitasnya. Mereka bisa bekerja di berbagai lapisan aplikasi, berkolaborasi dengan berbagai tim, serta beradaptasi dengan cepat terhadap perubahan teknologi.

**2. SDLC: Kerangka Kerja Pengembangan Perangkat Lunak**

**Software Development Life Cycle (SDLC)** adalah proses yang terstruktur dan berulang, yang digunakan untuk memastikan bahwa perangkat lunak dikembangkan dengan baik, memenuhi kebutuhan pengguna, dan berfungsi dengan optimal. SDLC terdiri dari beberapa fase kunci yang harus dilalui dalam pengembangan perangkat lunak.

**a. Planning (Perencanaan)**

Tahap awal ini melibatkan diskusi intensif antara pemangku kepentingan (stakeholders) untuk menentukan tujuan, kebutuhan, dan ruang lingkup proyek. Tujuan utamanya adalah mengidentifikasi visi proyek secara jelas dan merumuskan langkah-langkah strategis yang harus diambil. Risiko juga diidentifikasi dan direncanakan untuk diminimalisir.

**b. Requirement Analysis (Analisis Kebutuhan)**

Pada tahap ini, kebutuhan pengguna didefinisikan secara rinci. Baik kebutuhan fungsional (apa yang harus dilakukan oleh perangkat lunak) maupun kebutuhan non-fungsional (seperti performa, keamanan, dan skalabilitas) harus dianalisis secara mendalam.

**c. Design (Perancangan)**

Desain sistem mencakup pembuatan **arsitektur sistem**, pemilihan teknologi, serta rancangan **antarmuka pengguna (UI/UX)**. Pembuatan diagram alur, model database, dan pemetaan interaksi sistem juga dilakukan di sini.

**d. Development (Pengembangan)**

Ini adalah fase di mana kode mulai ditulis. Front-end dan back-end developer bekerja sesuai dengan perancangan sebelumnya. Pengembangan sering dilakukan secara **iteratif** dengan penggunaan metode **Agile** atau **Scrum**, di mana fitur-fitur dirilis dalam tahapan kecil.

**e. Testing (Pengujian)**

Pada fase ini, perangkat lunak diuji untuk memastikan bahwa ia bekerja dengan benar. Pengujian melibatkan **unit testing**, **integration testing**, dan **user acceptance testing (UAT)**. Semua bug dan masalah yang ditemukan diperbaiki sebelum rilis final.

**f. Deployment (Penerapan)**

Setelah pengujian selesai, perangkat lunak siap diimplementasikan di lingkungan produksi. Proses ini bisa dilakukan secara bertahap atau sekaligus, tergantung pada ukuran proyek dan kebutuhannya.

**g. Maintenance (Pemeliharaan)**

Setelah dirilis, perangkat lunak akan terus dipantau dan diperbarui. Masalah-masalah yang muncul dari penggunaan oleh pengguna akhir akan diperbaiki, dan fitur-fitur baru akan dikembangkan seiring waktu.

SDLC membantu tim pengembang memastikan bahwa setiap langkah dalam pengembangan perangkat lunak dilakukan dengan cermat, meminimalkan risiko dan meningkatkan kualitas perangkat lunak secara keseluruhan.

**3. Basic Git: Alat Kolaborasi yang Wajib Dikuasai**

Dalam proyek pengembangan perangkat lunak yang melibatkan banyak pengembang, **Git** adalah alat yang sangat penting untuk mengelola kode dan berkolaborasi. Git adalah sistem **version control** yang memungkinkan pengembang melacak perubahan pada kode, bekerja bersama tanpa konflik, serta menjaga kode tetap terstruktur.

**a. Repository**

**Repository (repo)** adalah tempat penyimpanan seluruh kode proyek. Git memungkinkan pengembang untuk menyimpan versi kode secara lokal di mesin mereka, sekaligus menyinkronkan dengan repo **remote** (misalnya, di GitHub atau GitLab) yang digunakan untuk berbagi dengan tim lain.

**b. Branching dan Merging**

Branching memungkinkan pengembang untuk bekerja secara independen di fitur atau perbaikan tertentu tanpa mempengaruhi kode di branch utama. Setelah selesai, branch dapat digabungkan (merge) kembali ke branch utama, biasanya melalui **pull request** yang memastikan kode telah diuji dan siap untuk digabungkan.

**c. Commit**

Setiap kali perubahan signifikan dibuat pada kode, pengembang melakukan **commit** untuk merekam perubahan tersebut. Commit dilengkapi dengan pesan yang menjelaskan perubahan, memudahkan untuk menelusuri sejarah kode.

**d. Push dan Pull**

* **Push** digunakan untuk mengunggah perubahan dari repo lokal ke repo remote.
* **Pull** digunakan untuk mengambil perubahan terbaru dari repo remote ke lokal.

**e. Collaborative Work**

Git memungkinkan banyak pengembang bekerja pada proyek yang sama dengan menjaga versi kode terpisah, dan kemudian menggabungkannya tanpa konflik. Ini adalah kunci bagi proyek-proyek besar yang memerlukan kolaborasi intensif.

**f. Revert dan Rollback**

Jika terjadi kesalahan, Git memungkinkan pengembang untuk kembali ke versi kode sebelumnya dengan mudah, tanpa harus mengulang pekerjaan dari awal. Ini memberikan keamanan ekstra dalam pengembangan perangkat lunak.