Nama : Egydia Alfariza Ramadhani
Kelas : SIB 7 IT Full Stack Developer

Summary Full Stack Developer Career Path, SDLC & Design Thinking Implementation dan Basic Git & Collaborating Using Git

Full Stack Developer Career Path

Definisi dan Scope Pengembangan Full Stack

Pengembangan Full Stack (Full Stack Development) merujuk pada pengembangan seluruh aplikasi secara end-to-end, dari sisi depan (front-end) hingga sisi belakang (back-end) dan, dalam beberapa kasus, hingga sisi klien (client-side)

Scope Penting Full Stack Development

a. Front-End Development → Membangun antarmuka pengguna yang menarik dan interaktif menggunakan HTML, CSS, dan JavaScript.
b. Back-end Development → Membangun server dan aplikasi yang berfungsi sebagai "otak" dari aplikasi, menerima permintaan dari sisi depan, memproses data, dan memberikan respon yang sesuai.
c. Database Management → Mendesain dan mengelola basis data untuk menyimpan, mengambil, dan memanipulasi data aplikasi.
d. Integration of Front-End and Back-end → Menghubungkan komponen Front-end dengan layanan back-end melalui API (Application Programming Interface) untuk berkomunikasi dengan server dan database.
e. Version Control and Collaboration → Menggunakan sistem pengendalian versi, seperti Git, untuk mengelola perubahan kode dan kolaborasi dalam tim pengembang.
f. Mobile Development → Beberapa pengembang Full Stack juga memiliki kemampuan untuk mengembangkan aplikasi mobile menggunakan framework seperti React Native dan Flutter.

Dasar-dasar Front-end Web Development

dikenal sebagai pengembangan sisi klien adalah praktik pembuatan HTML, CSS, dan Javascript untuk situs web atau Aplikasi Web sehingga pengguna dapat melihat dan berinteraksi dengannya secara langsung.

a. HTML → HTML (HyperText Markup Language) mendefinisikan arti dan struktur konten web.
b. CSS → CSS adalah bahasa yang digunakan untuk menata halaman web.
c. Javascript → JS biasanya digunakan untuk membuat web yang kita punya lebih interaktif.

Dasar-dasar Back-end Web Development

Bagian back-end bertanggung jawab untuk memproses permintaan dari pengguna, mengelola dan menyimpan data di database, serta memberikan respons kepada klien (front-end) berdasarkan permintaan yang diterima.

a. Bahasa Pemrograman Server-Side → Bahasa pemrograman seperti Node.js (Javascript), Python, Ruby, Java, PHP. C# dll. digunakan untuk menulis kode di sisi server.
b. Server Framework → Framework seperti Express.js untuk Node.js, Flask untuk Python, Ruby on Rails untuk Ruby, Spring untuk Java, dan Laravel untuk PHP
c. Database Management → Jenis database yang umum digunakan adalah SQL (MySQL, PostgreSQL, SQL Server) dan NoSQL (MongoDB, Firebase)

Dasar-dasar Database Management

Serangkaian konsep dan teknik yang digunakan untuk mengelola data dalam sebuah aplikasi atau sistem. Database merupakan bagian kritis dari aplikasi karena menyimpan dan mengorganisir informasi yang diperlukan untuk menjalankan aplikasi dengan benar.

a. Database Management System → Perangkat lunak yang memungkinkan pengguna untuk mengelola dan mengakses data dalam database. DBMS menyediakan antarmuka untuk berinteraksi dengan database.
b. Tipe Database → SQL (Structured Query Language) atau database relasional dan NoSQL (Not Only SQL) atau database non-relasional.
c. Bahasa Query → SQL adalah bahasa Query yang digunakan untuk berinteraksi dengan database SQL. Bahasa query memungkinkan pengguna untuk melakukan operasi seperti SELECT, INSERT, UPDATE dan DELETE.

Dasar-dasar Mobile Development

Serangkaian konsep dan teknologi yang digunakan untuk membangun aplikasi yang dapat dijalankan di perangkat mobile, seperti smartphone dan tablet.

a. Platform Mobile → Aplikasi mobile dapat dikembangkan untuk berbagai platform, termasuk android, IOS, dan windows phone.
b. IDE (Integrated Development Environment) → IDE adalah perangkat lunak yang digunakan untuk mengembangkan aplikasi mobile. IDE menyediakan alat bantu, penyunting kode, pengelola proyek, simulator perangkat dan fasilitas debugging untuk mempermudah proses pengembangan.

Pengembangan Aplikasi End to End

Merupakan pendekatan pengembangan perangkat lunak yang mencakup keseluruhan siklus pembuatan aplikasi, dari tahap perencanaan hingga tahap pengujian dan implementasi. Tujuannya

adalah untuk menghasilkan aplikasi yang lengkap, fungsional dan siap digunakan oleh pengguna akhir. Berikut adalah tahapan-tahapan Pengembangan aplikasi end-to-end:

1. Perencanaan dan Analisis
2. Desain
3. Pengembangan front-end
4. Pengembangan back-end
5. Integrasi dan pengujian
6. Pemeliharaan dan Peningkatan

Kolaborasi Efektif – Version Control

Version Control (Pengendalian Versi) adalah sistem yang memungkinkan pengembang perangkat lunak untuk melacak perubahan pada kode sumber aplikasi selama pengembangan. Beberapa manfaat Version Control adalah dalam Rekam Perubahan, Pencatatan Riwayat, Pemecahan Konflik dan Pemulihan yang Mudah. Penggunaan Version Control untuk Berkolaborasi dimulai dari Inisialisasi Proyek, Pengembangan Paralel, Branching, Merge, lalu Pull Request.

Tools sets Sebagai Full Stack Developer

- IDE - Code Editor : Visual Studio Code
- Version Control - Repository : Github, Gitlab, Bitbucket
- Version Control - Git Tools : Sourcetree, GitLens
- DBMS : PostgreSQL, MySQL, Oracle, MongoDB, Redis
- API : Postman, Swagger
- Test & Debugging : Jest, Mocha, Chai, JUnit
- Mobile Development : React Native, Flutter
- Layanan Cloud : Aws, Google Cloud, Azure
- CI/CD : Jenkins, circleci
- Desain UI/UX : Figma, Sketch

SDLC & Design Thinking Implementation

SDLC (Siklus Hidup Pengembangan Perangkat Lunak) adalah rangkaian proses yang terstruktur dan metodologi yang digunakan untuk mengembangkan perangkat lunak dari awal hingga selesai.

Siklus SDLC – Fase fase pada SDLC

1. **Perencanaan dan Analisis** → identifikasi masalah atau kebutuhan bisnis yang perlu diselesaikan, kumpulkan persyaratan dan tentukan ruang lingkup proyek. Rencana yg dibuat mencakup alokasi sumber daya, jadwal waktu, dan definisi tugas dan tanggung jawab anggota tim

2. **Desain Produk** → perangkat lunak dirancang secara rinci berdasarkan persyaratan yang telah dikumpulkan. Desain mencakup arsitektur sistem, antarmuka pengguna, dan desain database.
3. **Pengembangan Produk** → implementasi rancangan perangkat lunak yang telah disetujui sebelumnya. Para pengembang menulis kode untuk menghasilkan produk perangkat lunak yang berfungsi.
4. **Pengujian Produk** → pengujian dilakukan untuk memastikan bahwa perangkat lunak berfungsi sesuai dengan persyaratan yang telah ditentukan mencakup verifikasi fungsionalitas, kinerja, keamanan, dan kualitas keseluruhan perangkat lunak
5. **Penerapan Produk** → implementasi rancangan perangkat lunak yang telah disetujui sebelumnya. Para pengembang menulis kode untuk menghasilkan produk perangkat lunak yang berfungsi.
6. **Pemeliharaan Produk** → Setelah perangkat lunak diimplementasikan, pemeliharaan dilakukan untuk memperbaiki bug, meningkatkan fitur, dan menjaga perangkat lunak agar tetap sesuai dengan perubahan kebutuhan bisnis.

Manfaat Penggunaan SDLC

- Prediktabilitas dan Pengendalian Proyek
- Peningkatan Kualitas Perangkat Lunak
- Pengelolaan Risiko yang Lebih Baik
- Efisiensi Tim dan Kolaborasi
- Memenuhi Kebutuhan Pengguna
- Penghematan Biaya dan Waktu
- Meningkatkan Pengawasan dan Evaluasi
- Peningkatan Dokumentasi

Model Model SDLC

- Waterfall Model → Waterfall model adalah model SDLC yang linier dan berurutan. Setiap tahap dalam model ini harus selesai sebelum memulai tahap berikutnya. Tahapannya meliputi analisis, perencanaan, desain, pengembangan, pengujian, implementasi, dan pemeliharaan.
- V-Shaped Model → Model V-Shaped adalah model yang terkait erat dengan model waterfall, tetapi menekankan pada pengujian. Tahapan pengujian diwakili oleh garis miring "V", yang berarti bahwa setiap tahap pengembangan memiliki tahapan pengujian yang sesuai.
- Prototype Model → Model Prototype adalah model pengembangan perangkat lunak yang bertujuan untuk menciptakan prototipe atau contoh awal sebelum mengembangkan versi finalnya. Model ini fokus pada pemahaman kebutuhan pengguna dan mengumpulkan umpan balik untuk memastikan bahwa perangkat lunak akhir sesuai dengan ekspektasi dan persyaratan pengguna.

- Spiral Model → Model ini menggabungkan elemen model spiral dengan pendekatan inkremental. Setiap siklus spiral membangun pada inkrementasi sebelumnya, menghasilkan perangkat lunak yang semakin berkembang dengan fitur yang lebih banyak setiap siklusnya.
- Iterative Incremental Model → Model ini melibatkan pengulangan siklus pembangunan dan peningkatan perangkat lunak dalam tahapan-tahapan kecil. Setiap iterasi menambahkan lebih banyak fitur hingga produk akhir mencapai tingkat kesempurnaan yang diinginkan..
- Big Bang Model → Model Big Bang adalah model yang kurang terstruktur, di mana semua tahapan pengembangan dilakukan tanpa perencanaan yang detail. Pengembangan dimulai tanpa melakukan analisis dan perencanaan yang mendalam.
- Agile Model → Model Agile adalah pendekatan kolaboratif dan iteratif yang berfokus pada pengiriman perangkat lunak secara berkala dan inkremental. Tim bekerja dalam sprint (iterasi singkat) dan selalu terbuka untuk perubahan persyaratan pengguna.

Design Thinking Implementation

Steps Design Thinking

1. Empathize: Understand User Needs → fokus pada memahami secara mendalam pengguna akhir dan kebutuhan mereka, keinginan, serta masalah yang dihadapi. Kegiatan Empathize meliputi: User Research, Empathy Mapping, dan User Personas.
2. Define: Define the Problem → informasi yang dikumpulkan selama fase empati dianalisis untuk menentukan masalah dan menetapkan tujuan yang jelas untuk proyek. Kegiatan kunci meliputi: Problem Statement dan Stakeholder Alignment
3. Ideate: Generate Ideas → Fase ideasi mendorong pemikiran kreatif dan menghasilkan berbagai solusi potensial. Kegiatan kunci meliputi: Brainstorming Sessions dan Idea Consolidation.
4. Prototype: Build Quick and Iterative Solutions → fokus pada menciptakan representasi nyata dari ide-ide yang dipilih. Prototype digunakan untuk mengumpulkan umpan balik dan memvalidasi asumsi. Kegiatan kunci meliputi: Low-Fidelity Prototypes dan High-Fidelity Prototypes.
5. Test: Gather User Feedback → Tahap pengujian melibatkan pengumpulan umpan balik dari pengguna nyata untuk memvalidasi solusi-solusi tersebut. Kegiatan kunci meliputi: Usability Testing dan Iterative Testing.
6. Implement: Develop the Software → desain diterjemahkan ke dalam kode yang sebenarnya dan diimplementasikan. Kegiatan kunci meliputi: Agile Development dan Cross-Functional Collaboration.

Kesimpulan:

Dengan menggunakan SDLC secara efektif, organisasi dapat meningkatkan keberhasilan dan efisiensi dalam mengembangkan aplikasi, memastikan pengiriman produk berkualitas tepat waktu, dan

memberikan nilai yang lebih besar bagi pelanggan dan stakeholder. Setiap model SDLC memiliki kelebihan dan kelemahan tergantung pada jenis proyek dan kebutuhan organisasi. Pemilihan model SDLC yang tepat sangat penting untuk mencapai keberhasilan proyek pengembangan perangkat lunak. Dengan mengintegrasikan Design Thinking dalam SDLC, tim pengembangan perangkat lunak dapat menciptakan produk yang lebih berorientasi pada pengguna, intuitif, dan sukses dalam memenuhi kebutuhan pengguna serta tujuan bisnis. Sifat iteratif dari Design Thinking memastikan perangkat lunak terus berkembang dan beradaptasi dengan perubahan kebutuhan pengguna dan dinamika pasar.

Basic Git & Collaborating Using Git

Memahami Version Control Git

Kontrol versi adalah metode yang digunakan untuk melacak dan mengelola perubahan dalam kode sumber atau berkas proyek. Git merupakan salah satu sistem kontrol versi terdistribusi yang paling populer dan kuat. Berikut adalah langkah-langkah untuk memahami kontrol versi dan Git

- **Sistem Kontrol Versi Terpusat (Centralized Version Control System)** → Dalam sistem ini ada satu repositori sentral yang berfungsi sebagai "master" untuk menyimpan seluruh sejarah proyek. Setiap pengembang melakukan perubahan pada salinan lokal, kemudian mengirimkan perubahan tersebut ke repositori sentral. Contoh sistem kontrol versi terpusat adalah Subversion (SVN).
- **Sistem Kontrol Versi Terdistribusi (Distributed Version Control System)** → setiap anggota tim memiliki salinan sejarah perubahan, tidak hanya salinan terbaru dari seluruh repositori. Contoh sistem kontrol versi terdistribusi adalah Git, Mercurial, dan Bazaar.

Pengertian GIT

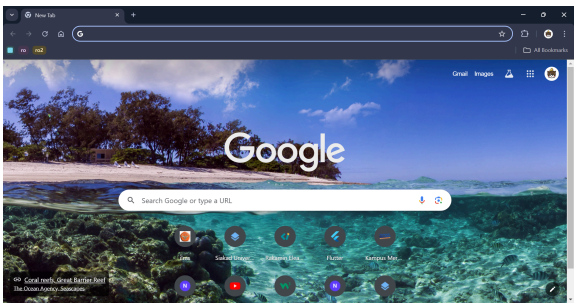
Git adalah sistem kontrol versi terdistribusi yang memungkinkan pengembang perangkat lunak untuk melacak perubahan dalam kode mereka, berkolaborasi dengan anggota tim, dan mengelola revisi kode secara efektif.

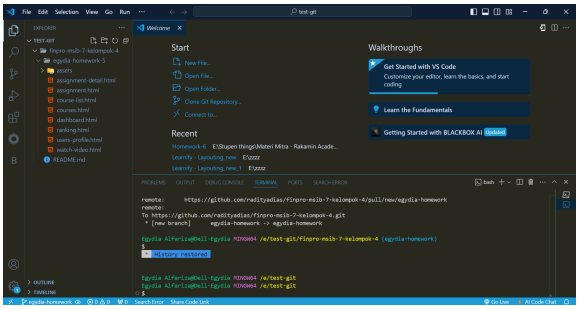
Dasar Dasar Command GIT

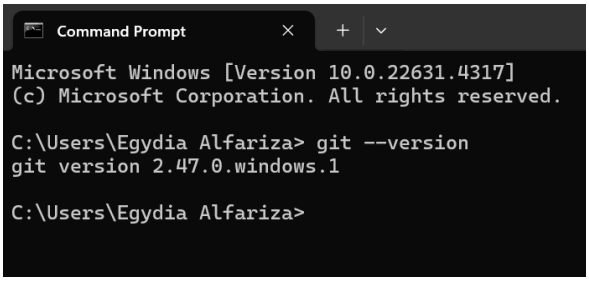
1. git init → Menginisialisasi direktori sebagai repositori Git kosong
2. git clone → Menduplikasi repositori Git yang sudah ada ke direktori lokal.
3. git status → Menampilkan status perubahan yang belum dikomit di repositori lokal.
4. git add → Menambahkan perubahan ke area persiapan (staging area) untuk disiapkan menjadi commit.
5. git commit → Membuat commit dari perubahan yang sudah di-staging dan menambahkan pesan commit.
6. git push → Mengirimkan commit ke repositori jarak jauh (remote repository).

7. git pull → Mengambil commit terbaru dari repositori jarak jauh dan menggabungkannya ke repositori lokal.
8. git branch Menampilkan daftar cabang (branch) yang ada di repositori dan menunjukkan cabang aktif.
9. git checkout → Beralih ke cabang lain atau ke commit tertentu.
10. git merge → Menggabungkan perubahan dari satu cabang ke cabang aktif.
11. git log → Menampilkan daftar commit beserta riwayatnya dalam repositori.
12. git remote → Menampilkan daftar repositori jarak jauh yang terhubung dengan repositori lokal.
13. git fetch → Mengambil informasi terbaru dari repositori jarak jauh tanpa menggabungkan perubahan
14. git diff → Menampilkan perbedaan antara versi yang sudah di-staging dengan versi sebelumnya.
15. git reset → Mengembalikan file yang sudah di-staging ke direktori kerja sebelumnya

>> Screenshot instalasi tools







>> Link Github Kelompok 4 : https://github.com/radityadias/finpro-msib-7-kelompok-4.git

>> Link Gdocs Summary : https://docs.google.com/document/d/1EC88dODUylJ60GmneknoHzSi211QPjQlrUjA_x9xeJ4/edit?usp=sharing