Full Stack Web Developer Career Path

Full Stack Development adalah sebuah pendekatan komprehensif dalam pengembangan aplikasi, di mana seorang developer menguasai seluruh lapisan pengembangan, mulai dari tampilan depan (front-end) hingga bagian belakang (back-end) aplikasi.

Lingkup Keahlian Full Stack Developer:

- Front-end: Membangun tampilan visual dan interaktif aplikasi menggunakan HTML, CSS, JavaScript, dan framework seperti React, Angular, atau Vue.js.
- Back-end: Mengelola logika bisnis aplikasi, berinteraksi dengan database, dan menyediakan API untuk komunikasi dengan front-end. Bahasa pemrograman yang umum digunakan antara lain Node.js, Python, Ruby, Java, dan PHP.
- Database: Mendesain dan mengelola basis data untuk menyimpan data aplikasi. SQL (MySQL, PostgreSQL) dan NoSQL (MongoDB) adalah jenis database yang populer.
- Integrasi: Menghubungkan front-end dan back-end melalui API untuk memastikan data dan tampilan aplikasi sinkron.
- Kontrol Versi: Menggunakan Git untuk melacak perubahan kode dan memudahkan kolaborasi dalam tim.
- Mobile Development: Mengembangkan aplikasi mobile menggunakan framework seperti React Native atau Flutter.

Keterampilan Penting:

- Pengembangan aplikasi end-to-end: Mampu membangun aplikasi lengkap dari awal hingga akhir.
- Version control: Memahami dan menggunakan Git untuk mengelola kode sumber.
- Kolaborasi: Bekerja efektif dalam tim dan memberikan kontribusi pada proyek bersama.
- Problem-solving: Mampu menganalisis dan menyelesaikan masalah teknis.
- Belajar terus-menerus: Teknologi terus berkembang, sehingga seorang full stack developer harus selalu bersedia belajar hal-hal baru.

Tools yang Digunakan:

- IDE: Visual Studio Code
- Version control: GitHub, GitLab, Bitbucket
- Database: PostgreSQL, MySQL, MongoDB
- API: Postman, Swagger
- Testing: Jest, Mocha Chai
- Mobile development: React Native, Flutter
- Cloud: AWS, Google Cloud, Azure
- CI/CD: Jenkins, CircleCI
- UI/UX design: Figma, Sketch

SDLC & Design Thinking Implementation

SDLC (Software Development Life Cycle) adalah suatu metodologi yang digunakan untuk mengembangkan perangkat lunak secara sistematis. SDLC terdiri dari beberapa fase, mulai dari perencanaan hingga pemeliharaan, dengan tujuan menghasilkan perangkat lunak yang berkualitas, sesuai kebutuhan pengguna, dan tepat waktu.

Fase-fase dalam SDLC:

- Perencanaan dan Analisis: Mendefinisikan tujuan, mengumpulkan kebutuhan, dan merencanakan proyek.
- Desain: Membuat rancangan arsitektur, antarmuka, dan database.
- Pengembangan: Menulis kode program.
- Pengujian: Memastikan perangkat lunak berfungsi dengan baik.
- Implementasi: Meluncurkan perangkat lunak.
- Pemeliharaan: Memperbaiki dan meningkatkan perangkat lunak.

Model-model SDLC:

- Waterfall: Pendekatan linear dan berurutan.
- V-Shaped: Fokus pada pengujian pada setiap tahap pengembangan.
- Prototype: Membuat prototipe awal untuk mendapatkan umpan balik pengguna.
- Spiral: Menggabungkan pendekatan inkremental dengan analisis risiko.
- Iterative Incremental: Pengembangan dilakukan secara bertahap.
- Big Bang: Pendekatan tanpa perencanaan yang matang.
- Agile: Pendekatan fleksibel dan kolaboratif.

Design Thinking adalah pendekatan inovatif yang berfokus pada pengguna. Design Thinking dapat diintegrasikan ke dalam SDLC untuk menghasilkan solusi yang lebih relevan dan bernilai bagi pengguna.

Langkah-langkah Design Thinking:

- Empathize: Memahami kebutuhan dan keinginan pengguna.
- Define: Mendefinisikan masalah yang akan dipecahkan.
- Ideate: Menghasilkan ide-ide kreatif untuk solusi.
- Prototype: Membuat prototipe untuk menguji ide.
- Test: Menguji prototipe dengan pengguna.
- Implement: Menerapkan solusi yang telah diuji.

Integrasi SDLC dan Design Thinking memungkinkan tim pengembangan untuk menciptakan perangkat lunak yang tidak hanya memenuhi persyaratan teknis, tetapi juga memenuhi kebutuhan dan ekspektasi pengguna. Pendekatan ini mendorong kolaborasi, inovasi, dan fleksibilitas dalam proses pengembangan perangkat lunak.

Manfaat Menggabungkan SDLC dan Design Thinking:

- Produk yang dihasilkan lebih sesuai dengan kebutuhan pengguna.
- Pengurangan risiko kegagalan dan pemborosan waktu.
- Produk yang dihasilkan lebih berkualitas dan andal.
- Tim pengembangan dapat bekerja sama lebih efektif.

Basic Git & Collaborating Using Git

Terminal adalah antarmuka baris perintah yang digunakan untuk berinteraksi dengan sistem operasi. IDE (Integrated Development Environment) adalah perangkat lunak yang menyediakan berbagai fitur untuk pengembangan perangkat lunak, seperti editor kode, debugger, dan alat build.

Tutorial install git: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Dasar-dasar Command Git:

- git init : menginisialisasi direktori sebagai repositori kosong.
- git clone : menduplikasi repositori git yang sudah ada ke direktori lokal.
- git status : menampilkan status perubahan yang belum di commit di repositori lokal.
- git add : menambahkan perubahan ke area persiapan (staging area) untuk disiapkan menjadi commit.
- git commit : membuat commit dari perubahan yang sudah di-staging dan menambahkan pesan commit.
- git push : mengirimkan commit ke repositori jarak jauh (remote repository).
- git pull : mengambil commit terbaru dari repositori jarak jauh dan menggabungkannya ke repositori lokal.
- git branch : menampilkan daftar cabang yang ada di repositori dan menunjukkan cabang aktif.
- git checkout : beralih ke cabang lain atau ke commit tertentu.
- git merge : menggabungkan perubahan dari satu cabang ke cabang aktif.
- git log : menampilkan daftar commit beserta riwayatnya dalam repositori.
- git remote : menampilkan daftar repositori jarak jauh yang terhubung dengan repositori lokal.
- git fetch : mengambil informasi terbaru dari repositori jarak jauh tanpa menggabungkan perubahan.
- git diff : menampilkan perbedaan antara versi yang sudah di-staging dengan versi sebelumnya.
- git reset : mengembalikan file yang sudah di-staging ke direktori kerja sebelumnya.