Summary

SUBJECT:	Browser, GIT, Visual Studio Code	NAME:	IKRAR BAGASKARA	↓

Fullstack Developer

Fullstack developer is a role where developers have the ability to develop systems like:

1. Front-end or interface dari sebuah sistem, dimana user akan melakukan interaksi dengan sistem tersebut. Beberapa kemampuan yang wajib dimiliki oleh seorang front-end developer adalah HTML, CSS, dan JavaScript. Seiring berkembangnya website, banyak juga framework pendukung untuk membangun website seperti React, Vue.Js, Angular.Js, Stelve, Next.Js, dan lain sebagainya.
2. Back-end atau sistem yang bertanggung untuk memproses permintaan dari pengguna. Bahasa pemrogframan seperti Node.Js, Python, Ruby, dan C# digunakan untuk membangun sistem back-end. Back-end juga memiliki dukungan framework seperti, Ruby on Rails, Laravel untuk PHP, Flash/Django untuk Python, dan lain sebagainya.
3. Database management juga merupakan kemampuan penting yang harus dimiliki oleh seorang fullstack developer, dimana fullstack developer harus paham tipe database yang akan digunakan, model database yang digunakan dan sebagainya untuk memastikan sistem dapat berjalan dengan semestinya

Selain dalam pengembangan wesbite, banyak juga fullstack developer memiliki kemampuan untuk mengembangkan aplikasi mobile menggunakan dukungan framework seperti Kotlin, Flutter, dan React Native.

Dalam pengembangan aplikasi biasanya fullstack developer akan melalui tahapan sebagai berikut:

1. **Perencanaan dan analisis**
   Mengumpulkan kebutuhan dan pemahaman mendalam tentang tujuan aplikasi, sasaran pengguna, dan lingkungan operasional
2. **Desain**
   Merancang user interface dan user experience yang intuitif dan menarik
3. **Pengembangan Front-End**
   Developer akan membuat bagian depan dari aplikasi, menggunakan HTML, CSS, dan JavaScript untuk membuat tampilan dan interaksi menarik bagi pengguna
4. **Pengembangan Back-End**
   Developer akan membuat bagian belakang dari aplikasi guna untuk memproses data yang dimasukan oleh user.
5. **Intergasi dan Pengujian**
   Developer menguji interaksi antara Back-end dengan Front-end menggunakan API (Application Programming Interace) sehingga mereka dapat berkomunikasi dabnberbagi data
6. **Pemeliharaan dan Peningkatan**
   Developer juga bertanggung jawab atas product yang sudah dirilis seperti melakukan penyeusain teknologi yang digunakan, memperbaiki yang muncul, menambahkan fitur yang diperlukan.

Collaboration and Version Control System

Kolaborasi adalah salah satu skill yang harus dimiliki oleh fullstack developer untuk membangun sistem yang baik. Salah satu alat pendukung kolaborasi adalah menggunakan version control seperti Github, GitLab, dan BitBucket. Pihak pengembang dapat memilih menggunakan version control tergantung dari kebutuhan tim pengembang. Dengan berkolaborasi menggunakan version control ada juga keuntungan seperti:

- Adanya history pada setiap perubahan code
- Version control dapat membantu mengidentifikasi bila ada konflik pada code base
- Version control juga menyediakan fitur untuk recovery code ke versi sebelumnya

Berikut adalah beberapa fitur umum yang tersedia di hampir semua version control system:

- Inisialisasi Proyek, tempat dimana semua kode akan disimpan secara lokal
- Pengembangan Pararel, setiap anggota tim mempunyai salinan sendiri pada lokal mesin mereka
- Branching, memungkin tim mengembangkan aplikasi tanpa mengganggu code utama.
- Merge, cabang yang sudah dibuat untuk pengembangan fitur baru dapat digabungkan dengan code utama
- Pull Request, fitur dimana memungkinkan pengembangg mengajukan perubahan pada code mereka untuk di review oleh anggota team lain nya

SDLC & Design Thinking Implementation

SDLC (Siklus Hidup Pengembangan Perangkat Lunak) adalah serangkaian proses dan metode terstruktur yang digunakan untuk mengembangkan perangkat lunak dari awal hingga akhir. SDLC terdiri dari serangkaian langkah yang saling terkait dan dilakukan secara berurutan untuk memastikan proses pengembangan perangkat lunak berjalan lancar dan konsisten dengan kebutuhan dan tujuan yang ditentukan. SDLC juga memiliki siklus seperti berikut:

1. **Perencanaan dan Analisis**
   Pada tahap ini, team mengidenfitikasi masalah dan kebutuhan bisnis sesaui dengan kebutuhan user. Rencana ini juga mencakup alokasi sumber daya, jadwal waktu, dan tanggung jawab anggota tim.

2. **Desain**
   Merancang perangkat lunak secara rinci termasuk arsitektur sistem, antarmuka pengguna, dan desain database.

3. **Pengembangan**
   Implementasi dari produk yang sudah dirancang dan disepakati sebelumnya dengan tujuan untuk menghasilkan produk yang berfungsi.

4. **Pengujian**
   Memastikan produk yang di kembangkan berfungsi sesuai dengan kebutuhan, pengujian juga meliputi fungsionalitas, kinerja, keamanan, dan kualitas seluruh produk.

5. **Penerapan & Integrasi**
   Penerapan dan integrasi dalam SDLC adalah langkah penting untuk menggabungkan komponen perangkat lunak dan memastikan kualitas serta kinerja yang optimal.

6. **Pemeliharaaan**
   Melakukan pemeliharaan terhadap bug yang muncul, serta meningkatkan fitur sesuai dengan kebutuhan.

   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.

Model Model SDLC

- **Waterfall Model**

  Model SDLC yang berurutan, setiapm tahap harus selesai sebelum memulai taha berikutnya, cocok untuk proyek dengan persyaratan yang jelas dan stabil

- **V-Shaped Model**

  Memiliki kesamaan dengan Waterfall model tetapi menekankan pada pengujian, garis miring “V” berarti pengembangan memiliki pengujian yang sesuai. Cocok pada proyek dengan fokus kualitas tinggi

- **Prototype Model**

  Bertujuan untuk menciptakan contoh awal sebelum mengembangkan versi finalnya. Dengan fokus kebutuhan pengguna dan mengumpulkan umpan balik

- **Spiral Model**

  Setiap siklus membanung pada inkremental sebelumnya, menghasilkan perangkat lunak yang semakin berkembang dengan fitur yang lebih banak, cocok untuk proyek besar dan kompleks.

- **Iterative Incremental Model**

  Pengulangan dan peningkatan produkl dalam tahapan kecil. Setiap iterasi menambahkan lebih banyak fitur hingga selesai sesuai dengan tujuan, cocok untuk proyek dengan watku dan anggarana terbatas

- **Big Bang Model**

  Model dimana tahapan pengembangan dilakukan tanpa perencanaan dan analisis mendalam. Cocok untuk proyek kecil

- **Agile Model**

  Pendekatan kolaboratif dan iteratif berfokus pada pengembangan fitur secar berkala. Tim bekerja dalam sprint atau waktu. Cocok untuk proyek dengan lingkungan dan persyaratan dinamis

Design Thinking Implementation

1. **Empathize: Understand User Needs**
   Memahami kebutuhan, masalah, dan keinginan pengguna secara mendalam
   Key points: User Research, Empathy Mapping, User Personas

2. **Define: Define the Problem**
   Menganalisa informasi dari tahap empathy kemudian menetapkan tujuan dari proyek.
   Key points: Problem Statement, Stakeholder Alignment

3. **Ideate: Generate Ideas**
   Melakukan brainstorming unuk mendapatkan solusi dari masalah yang muncul.
   Key points: Brainstorming Sessions, Idea Consolidation

4. **Prototype: Build and Iterative Solutions**
   Membuat reprensetasi nyata dari ide-ide yang sudah di dipilih.
   Key Points: Low Fidelity, High Fidelity

5. **Test: Gather User Feedback**
   Mengumpulkan umpan balik dari pengguna.
   Key points: Usability Testing, Iterative Testing

6. **Implement: Develop the Software**
   Membangun perangkat lunak dari desain yang sudah dibuat.
   Key points: Agile Development, Cross-Functional Collaboration

Basic Git & Collaborating Using Git

Sejarah Singkat Terminal

Dengan perkembangan teknologi dan perangkat lunak, terminal tetap menjadi alat penting bagi para pengembang perangkat lunak, administrator sistem, dan pengguna teknis lainnya. Meskipun antarmuka grafis semakin canggih dan populer, terminal tetap memberikan fleksibilitas dan kekuatan untuk melakukan tugas-tugas khusus dan otomatisasi dalam lingkungan komputer modern.

Version Control Git

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

- **Sistem Kontrol Versi Terpusat (Centralized Version Control System)**

  Dengan sistem terpusat, dimana satu repository berperan sebagai **‘master’** untuk menyimpan proyek. Kemudian setiap pengembang mempunyai salinan proyek tersebut secara lokal, kemudian mengirimkan perubahan kepada repo **‘master’** tersebut

- **Sistem Kontrol Versi Terdistribusi (Distributed Version Control System)**

  Setiap anggota tim pengembang memiliki salinan lengkap dari seluruh repository. Dan tidak ada memiliki sistem repositori **‘master’**