1. **Introduction Full Stach Web/Mobile Developer**

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

a. Web Development
   - Frontend

Pengembangan web front-end, yang sering disebut sebagai pengembangan sisi klien, adalah tindakan membuat kode HTML, CSS, serta JavaScript untuk situs web atau Aplikasi Web. Ini bertujuan agar pengguna dapat secara langsung menavigasi dan berinteraksi dengan platform tersebut.

   - Backend

Komponen bagian belakang (backend) memiliki tanggung jawab dalam menangani permintaan dari pengguna, mengatur dan menyimpan data di dalam basis data, serta memberikan respons kepada pengguna depan (front-end) sesuai dengan permintaan yang diterima.

Untuk menghubungkan front-end dengan back-end kita dapat melakukannya melalui **API (Application Programming Interface)** untuk berkomunikasi dengan server dan database. Sedangkan untuk mengelola perubahan kode dengan kolaborasi antar tim kita bisa menggunakan sistem pengendalian versi, seperti **GIT**.

b. Mobile

Aplikasi mobile dikembangkan untuk berbagai platform,Setiap platform memiliki bahasa pemrograman dan lingkungan pengembangan yang khas. Misalnya, aplikasi Android dapat ditulis dalam Java atau Kotlin, sedangkan aplikasi iOS menggunakan Swift atau Objective-C. Beberapa Pengembang Full Stack juga memiliki kemampuan untuk mengembangkan aplikasi mobile menggunakan framework seperti React Native, Flutter.

Pengembangan Aplikais Secara End-to-end

Yaitu Pengembangan perangkat lunak secara menyeluruh, dari tahap penrencanaan sampai tahap pengujian, dengna tujuan untuk menghasilkan aplikasi yang lengkap dan fungsional sehingga dapat digunakan oleh pengguna akhir.

Tahap - tahap End-to-end :

a. Perencanaan dan Analisis
   Tahap ini menggunakan pengumpulan kebutuhan tentang tujuan aplikasi, sasaran pengguna dan lingkungan operasional.
b. Desain
   Proses desain melibatkan merancang antarmuka pengguna (UI) dan pengalaman pengguna (UX) yang intuitif dan menarik. Selain itu pengembang juga merencanakan arsitektur aplikasi, termasuk pemilihan teknologi, database, dan framework yang sesuai.
c. Pengembangan Front-End

Pengembangan bekerja pada bagian depan aplikasi, dengan menggunakan bahasa pemrograman seperti HTML, CSS, dan JavaScript, ataupun menggunakan framework.

d. Pengembangan Back-End
   Tahap ini melibatkan pengembangan sisi server dan logika bisnis aplikasi. Pengembangan dapat menggunakan bahasa pemrograman seperti Node.js, Python dan lain-lain.
e. Integrasi dan Pengujian
   Melakukan integrasi antara bagian depan dan belakang aplikasi melalui API (Application Programming Interface), agar keduanya dapat berkomunikasi. Pengujian juga dilakukan untuk memastikan semua fitur berfungsi dengan baik.
f. Pemeliharaan dan Peningkatan
   aplikasi harus dipelihara dengan memperbaiki bug dan menangani perubahan lingkungan atau kebutuhan bisnis. Dengan tujuan untuk memperbarui fitur dan kinerja dari aplikasi.

Version Control yang populer bagi Full Stack Web Development :

- Git
- Mercurial

Manfaat Version Control dalam berkolaborasi :

- Sebagai Rekam Perubahan
  Sistem version control akan merekam semua perubahan code yang terjadi.
- Sebagai Pencatatan Riwayat
  Memungkinkan semua anggota tim dapat meilhat riwayat perubahan yang terjadi.
- Sebagai Pemecahan Konflik
  Version Control dapat mengidentifikasi perubahan yang terjadi.
- Sebagai Pemulihan Mudah
  Dapat memulihkan kode ke versi sebelumnya jika terjadi masalah atau bug.

Penggunaan Version Control dalam berkolaborasi

- Inisialisasi Proyek
  Dimulai denga repositori, yang nantinnya akan menyimpan semua sumber file, dan perubahan yang terjadi.
- Pengembangan Paralel
  Pengerjaan dapat dilakukan secara paralel, yang mana setiap tim melakukan salinan pada repositori komputernya sendiri.
- Branching
  pembuatan cabang (branch) yang terpisah dari kode utama. Ini memungkinkan tim untuk mengisolasi perubahan dan fitur yang sedang dikembangkan
- Merge
  Setelah fitur atau perubahan selesai, cabang dapat digabungkan kembali ke cabang utama.
- Pull Request

Mekanisme yang memungkinkan pengembang untuk mengajukan perubahan mereka untuk ditinjau oleh anggota tim lain sebelum digabungkan ke cabang utama.

2. **SDLC**

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

Fase pada SDLC

a. Perencanaan

Tahap pertama ini melibatkan identifikasi masalah atau kebutuhan bisnis yang perlu diselesaikan oleh perangkat lunak. Rencana ini mencakup alokasi sumber daya, jadwal waktu, dan definisi tugas dan tanggung jawab anggota tim.

b. Desain

Di tahap ini, perangkat lunak dirancang secara rinci berdasarkan persyaratan yang telah dikumpulkan. Desain mencakup arsitektur sistem, antarmuka pengguna, dan desain database.

c. Pengembangan

Implementasi rancangan perangkat lunak yang telah disetujui sebelumnya. Pengembangan berupa menulis kode untuk menghasilkan produk yang sesuai.

d. Pengujian

Tahap pengujian dilakukan untuk memastikan bahwa perangkat lunak berfungsi sesuai dengan persyaratan yang telah ditentukan. Pengujian mencakup verifikasi fungsionalitas, kinerja, keamanan, dan kualitas keseluruhan perangkat lunak.

e. Penerapan

Tahap ini melibatkan implementasi rancangan perangkat lunak yang telah disetujui sebelumnya. Para pengembang menulis kode untuk menghasilkan produk perangkat lunak yang berfungsi.

f. Pemeliharaan

pemeliharaan dilakukan untuk memperbaiki bug, meningkatkan fitur, dan menjaga perangkat lunak agar tetap sesuai dengan perubahan kebutuhan bisnis.

Manfaat penerapan SDLC

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

Model - Model SDLC

3. **Waterfall**, 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.
4. **V-Shaped**, 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.
5. **Prototype,** 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.
6. **Spiral,** 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.
7. **Iterative Incremental Model**, 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.
8. **Big Bang Model**, model yang kurang terstruktur, di mana semua tahapan pengembangan dilakukan tanpa perencanaan yang detail. Pengembangan dimulai tanpa melakukan analisis dan perencanaan yang mendalam.
9. **Agile Model**, 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

3. DESIGN THINKING

Tahapan Design Thinking

a. Empathize (Understand User Needs)

Tahap ini, fokus pada memahami secara mendalam pengguna akhir dan kebutuhan mereka, keinginan, serta masalah yang dihadapi. Anda dapat melakukan berbagai kegiatan, seperti :

- **User Research**, Lakukan wawancara, observasi, dan survei untuk mengumpulkan data kualitatif dan kuantitatif tentang perilaku dan preferensi pengguna.
- **Empathy Mapping**, Buat pemetaan empati yang menggambarkan sikap, pemikiran, perasaan, dan masalah pengguna.
- **User Personas**, Buat persona pengguna fiktif yang mewakili berbagai kelompok pengguna.

b. Define, (Define the Problem)

Tahap ini, informasi yang dikumpulkan selama fase empati dianalisis untuk menentukan masalah dan menetapkan tujuan yang jelas untuk proyek. Kegiatan kunci meliputi:

- **Problem Statement**, Susun pernyataan masalah yang jelas dan ringkas yang mengartikulasikan tantangan dari sudut pandang pengguna.

- **Stakeholder Alignment**, Kolaborasi dengan pemangku kepentingan, termasuk pemilik produk, manajer proyek, desainer, dan pengembang, untuk memastikan semua orang sejalan dengan tujuan dan ruang lingkup proyek.

c. Ideate (Define the Problem)

Tahap ini, informasi yang dikumpulkan selama fase empati dianalisis untuk menentukan masalah dan menetapkan tujuan yang jelas untuk proyek. Kegiatan kunci meliputi:

- **Brainstorming Sessions**, Lakukan sesi brainstorming kolaboratif dengan tim lintas fungsi untuk menghasilkan banyak ide tanpa menghakimi.
- **Idea Consolidation**, Setelah sesi brainstorming, kelompokkan dan konsolidasikan ide-ide terkait. Saring daftar hingga sekumpulan solusi yang dapat diwujudkan dan sesuai dengan tujuan dan batasan proyek.

d. Prototype (Build and Iterative Solutions)

Tahap ini, 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**, Bangun prototipe rendah seperti sketsa kertas, wireframe, atau mock-up. Prototipe ini cepat dan mudah dibuat, memungkinkan iterasi yang cepat.
- **High-Fidelity Prototypes**, Kembangkan prototipe yang lebih detail atau bahkan Minimum Viable Product (MVP) yang menyerupai penampilan dan fungsionalitas produk akhir.

e. Test (Gather User Feedback)

Tahap pengujian melibatkan pengumpulan umpan balik dari pengguna nyata untuk memvalidasi solusi-solusi tersebut. Kegiatan kunci meliputi:

- **Usability Testing**, Lakukan sesi satu lawan satu dengan pengguna untuk mengamati interaksi mereka dengan prototipe.
- **Iterative Testing**, Gunakan umpan balik dari pengujian ketergunaan untuk beriterasi pada desain dan mengatasi masalah ketergunaan atau kekhawatiran.

f. Implement (Develop the Software)

Pada tahap ini, desain diterjemahkan ke dalam kode yang sebenarnya dan diimplementasikan. Kegiatan kunci meliputi:

- **Agile Development**, Manfaatkan metodologi pengembangan agile seperti Scrum atau Kanban untuk memungkinkan pengembangan secara bertahap dan pengiriman berkelanjutan.
- **Cross-Functional Collaboration**, Tingkatkan kolaborasi antara desainer, pengembang, penguji, dan pemangku kepentingan lainnya untuk memastikan implementasi yang selaras dengan solusi yang telah dirancang.

4. 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 kontrol versi terpusat, 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)**

  Dalam sistem kontrol versi terdistribusi, setiap anggota tim memiliki salinan lengkap dari seluruh repositori. Ini berarti setiap pengembang memiliki salinan lengkap sejarah perubahan, tidak hanya salinan terbaru. Contoh sistem kontrol versi terdistribusi adalah Git, Mercurial, dan Bazaar.