A. Fullstack Developer Career Path

Definisi dan Scope Full Stack Developer : 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 Full Stack Developer :

1. Front-End Development = 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).
2. Back-End Development = Membangun server dan aplikasi yang berfungsi sebagai "otak" dari aplikasi, menerima permintaan dari sisi depan, memproses data, dan memberikan respons yang sesuai. Menggunakan bahasa pemrograman server-side seperti Node.js, Python, Ruby, Java, PHP, atau C#.
3. Database Management = Mendesain dan mengelola basis data untuk menyimpan, mengambil, dan memanipulasi data aplikasi. Menggunakan teknologi database seperti MySQL, PostgreSQL, MongoDB, atau Firebase.
4. Mobile Development = Pengembang Full Stack juga memiliki kemampuan untuk mengembangkan aplikasi mobile menggunakan framework seperti React Native, Flutter.

Dasar - Dasar Front-End Web Development : Pengembangan web front-end, juga 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. Dasar - Dasar Frontend Web Development ada 3 macam yaitu ;

1. HTML = HTML (HyperText Markup Language) adalah blokbangunan paling dasar dari Web. Ini mendefinisikan arti dan struktur konten web.
2. CSS = CSS adalah bahasa yang kami gunakan untuk menata halaman Web. Dan biasanya CSS digunakan untuk kosmetik dari sebuah web yang kita punya.
3. JavaScript = JavaScript adalah bahasa pemrograman paling populer di dunia. JavaScript adalah bahasa pemrograman Web. JS biasanya digunakan untuk membuat web kita punya lebih interaktif.

Dasar - Dasar Back-End Development : Bagian backend bertanggung jawab untuk memproses permintaan dari pengguna, mengelola dan menyimpan data di database, serta memberikan respons kepada klien (front-end) berdasarkan permintaan yang diterima. Dasar – Dasar Back-End Development ada 3 macam yaitu ;

1. Bahasa Pemrograman Server-Side = Bahasa pemrograman seperti Node.js (JavaScript), Python, Ruby, Java, PHP, C#, dan lain-lain, digunakan untuk menulis kode di sisi server.
2. Server Framework = Framework seperti Express.js untuk Node.js, Flask untuk Python, Ruby on Rails untuk Ruby, Spring untuk Java, dan Laravel untuk PHP.
3. 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. Dasar – Dasar Database Management ada 3 macam yaitu ;

1. Database Management System = Perangkat lunak yang memungkinkan pengguna untuk mengelola dan mengakses data dalam database. DBMS menyediakan antarmuka untuk berinteraksi dengan database.
2. Tipe Database = Ada dua tipe database utama yang umum digunakan dalam pengembangan aplikasi: SQL (Structured Query Language) atau database relasional dan NoSQL (Not Only SQL) atau database non-relasional.
3. Bahasa Query = SQL adalah bahasa query yang digunakan untuk berinteraksi dengan database SQL. Bahasa query memungkinkan pengguna untuk melakukan operasi seperti SELECT , INSERT , UPDATE DELETE.

Dasar - Dasar Mobile Development : Serangkaian konsep dan teknologi yang digunakan untuk membangun aplikasi yang dapat dijalankan di perangkat mobile, seperti smartphone dan tablet. Pengembangan aplikasi mobile mencakup beberapa aspek penting untuk memastikan aplikasi dapat berjalan dengan baik dan memberikan pengalaman pengguna yang optimal. Dasar – Dasar Mobile Management ada 2 macam yaitu ;

1. Platform Mobile = Aplikasi mobile dapat dikembangkan untuk berbagai platform, termasuk Android, iOS, dan Windows Phone. 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.
2. 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.

B. SDLC & Design Thinking Implementation

Apa itu SDLC : SDLC (Siklus Hidup Pengembangan Perangkat Lunak) adalah rangkaian proses yang terstruktur dan metodologi yang digunakan untuk mengembangkan perangkat

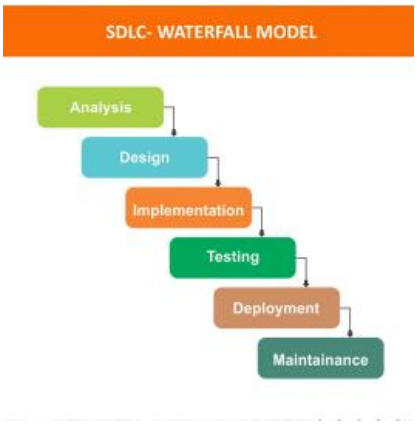
lunak dari awal hingga selesai. SDLC terdiri dari serangkaian tahap yang saling terkait dan dilakukan secara berurutan untuk memastikan bahwa pengembangan perangkat lunak berjalan dengan baik dan sesuai dengan kebutuhan dan tujuan yang ditentukan.

Fase – Fase pada SDLC

1. Perencanaan dan Analisis = Tahap pertama ini melibatkan identifikasi masalah atau kebutuhan bisnis yang perlu diselesaikan oleh perangkat lunak. Para pemangku kepentingan berinteraksi untuk mengumpulkan persyaratan dan menentukan ruang lingkup proyek. Rencana keseluruhan untuk proyek perangkat lunak dibuat. Rencana ini mencakup alokasi sumber daya, jadwal waktu, dan definisi tugas dan tanggung jawab anggota tim.
2. Desain = Di tahap ini, perangkat lunak dirancang secara rinci berdasarkan persyaratan yang telah dikumpulkan. Desain mencakup arsitektur sistem, antarmuka pengguna, dan desain database
3. Pengembangan = Tahap ini melibatkan implementasi rancangan perangkat lunak yang telah disetujui sebelumnya. Para pengembang menulis kode untuk menghasilkan produk perangkat lunak yang berfungsi.
4. Pengujian = Setelah perangkat lunak dikembangkan, 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.
5. Penerapan = Tahap ini melibatkan implementasi rancangan perangkat lunak yang telah disetujui sebelumnya. Para pengembang menulis kode untuk menghasilkan produk perangkat lunak yang berfungsi.
6. Pemeliharaan = Setelah perangkat lunak diimplementasikan, pemeliharaan dilakukan untuk memperbaiki bug, meningkatkan fitur, dan menjaga perangkat lunak agar tetap sesuai dengan perubahan kebutuhan bisnis.

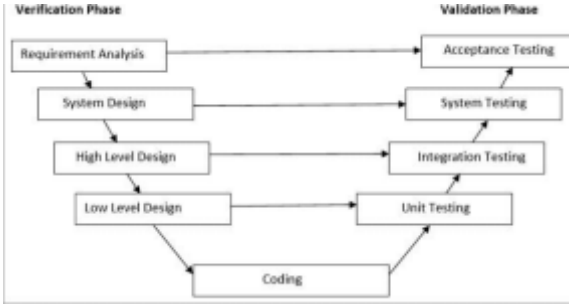
Model – Model SDLC :

1. Watefall Model



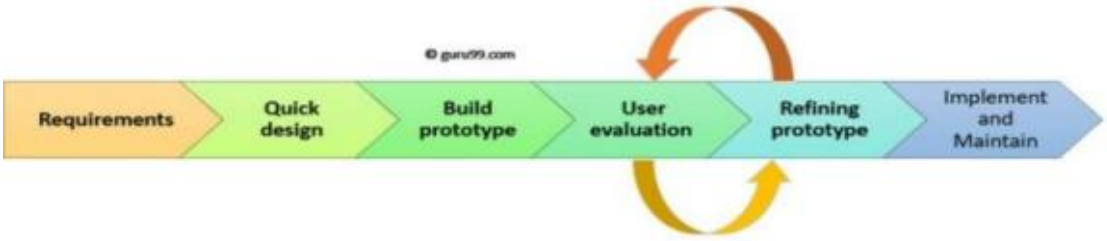
Waterfall model adalah model SDLC yang linier dan berurutan. Setiap tahap dalam model ini harus selesai sebelum memulai tahap berikutnya. Tahapannya meliputi analisis, perencanaan, desain, pengembangan, pengujian, implementasi, dan pemeliharaan. Cocok untuk proyek dengan persyaratan yang jelas dan stabil.

2. V-Shaped Model



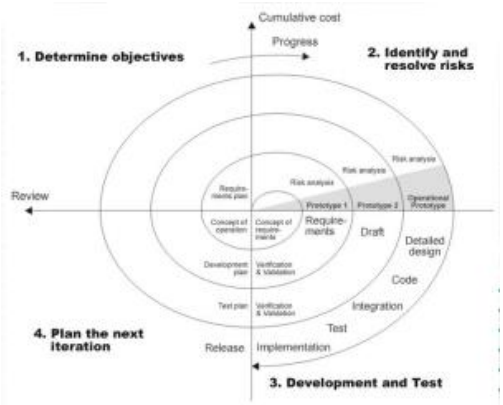
Model V-Shaped adalah model yang terkait erat dengan model waterfall, tetapi menekankan pada pengujian. Tahapan pengujian diwakili oleh garis miring "V", yang berarti bahwa setiap tahap pengembangan memiliki tahapan pengujian yang sesuai. Cocok untuk proyek dengan fokus pada kualitas tinggi.

3. Prototype Model



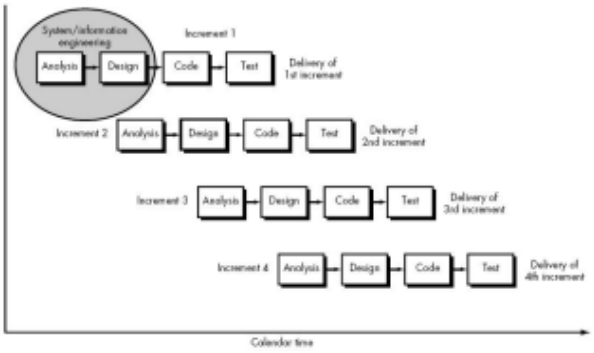
Model Prototype adalah model pengembangan perangkat lunak yang bertujuan untuk menciptakan prototipe atau contoh awal sebelum mengembangkan versi finalnya. Model ini fokus pada pemahaman kebutuhan pengguna dan mengumpulkan umpan balik untuk memastikan bahwa perangkat lunak akhir sesuai dengan ekspektasi dan persyaratan pengguna.

4. 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. Cocok untuk proyek besar dan kompleks dengan banyak risiko.

5. Iterative Incremental Model

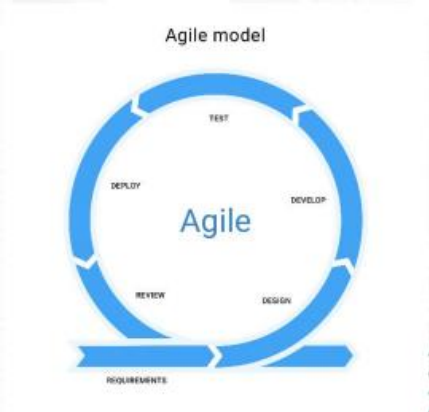


Model ini melibatkan pengulangan siklus pembangunan dan peningkatan perangkat lunak dalam tahapan-tahapan kecil. Setiap iterasi menambahkan lebih banyak fitur hingga produk akhir mencapai tingkat kesempurnaan yang diinginkan. Cocok untuk proyek dengan waktu dan anggaran yang terbatas.

6. 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. Cocok untuk proyek kecil atau prototyping.

7. 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. Cocok untuk proyek dengan lingkungan yang dinamis dan persyaratan yang berubah-ubah.

Design Thinking :

Empathize = Understand User Needs Pada 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 Persons = Buat persona pengguna fiktif yang mewakili berbagai kelompok pengguna.

Define = Define the Problem Pada 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.

Ideate = Generate Ideas Fase ideasi mendorong pemikiran kreatif dan menghasilkan berbagai solusi potensial. 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.

Prototype = Build and Iterative Solutions Pada 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.

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.

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.

C. Basic Git & Collaborating Using Git

Version Control : Version control (pengendalian versi) adalah sistem yang memungkinkan pengembang perangkat lunak untuk melacak perubahan pada kode sumber aplikasi selama pengembangan. Ini memungkinkan kolaborasi yang efisien di antara anggota tim, terutama ketika banyak orang bekerja pada proyek yang sama. Full Stack Web Development Git & Mercurial

Manfaat Version Control :

- Rekam Perubahan = Setiap kali pengembang membuat perubahan pada kode, sistem version control merekam detail perubahan tersebut.
- Pencatatan Riwayat = Version control memungkinkan tim untuk melihat riwayat lengkap dari semua perubahan yang terjadi pada proyek dari awal hingga saat ini.
- Pemecahan Konflik = Ketika dua atau lebih pengembang melakukan perubahan pada area kode yang sama, version control membantu mengidentifikasi dan menyelesaikan konflik.
- Pemulihan Mudah = Version control memungkinkan pengembang untuk memulihkan kode ke versi sebelumnya jika ada masalah atau bug yang terjadi, sehingga mengurangi risiko kehilangan pekerjaan.

Penggunaan Version Control untuk Berkolaborasi :

- Inisialisasi Proyek = Tim memulai proyek dengan membuat repositori version control. Repositori ini akan menyimpan semua kode sumber, file, dan perubahan yang dilakukan selama pengembangan.
- Pengembangan Paralel = Setiap anggota tim akan memiliki salinan repositori pada komputernya sendiri. Mereka dapat bekerja secara paralel, membuat perubahan.
- Branching = Version control memungkinkan 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 (biasanya disebut sebagai "merge").
- Pull Request = Di beberapa platform version control seperti GitHub, GitLab, dan Bitbucket, pull request adalah mekanisme yang memungkinkan pengembang untuk mengajukan perubahan mereka untuk ditinjau oleh anggota tim lain sebelum digabungkan ke cabang utama.

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.