Introduction Software Engineer
Full Stack Developer Career Path

1. Introduction Full Stack Web/Mobile Developer
2. Skillset Full Stack Web/Mobile Developer
3. Tools Full Stack Web/Mobile Developer

---

1. Introduction Full Stack Web/Mobile Developer

Full Stack Developer : pengembangan seluruh aplikasi end-to-end, dari sisi depan (front-end) hingga sisi belakang (back-end) ada juga beberapa kasus yang hingga sisi client (client-side).

Scope Fullstack Developer

FE Development	**Backend Development**	**Database Management**
❖ Membangun interface user yang menarik dan interaktif menggunakan HTML, CSS dan JS. ❖ Framework, terdiri atas React.jas, Angular, Vue.Js atau JQuery.	❖ Membangun server & aplikasi yang berfungsi untuk menerima request dari FE, memproses data, dan memeberikan respon yang sesuai. ❖ Bahasa terkait server-side terdiri atas Node.Js, Phyton, Ruby, Java, PHP, atau C#.	❖ Mendesain dan mengelola database untuk menyimpan, mengambil, dan memanipulasi data aplikasi. ❖ Teknologi yang digunakan MySQL, PostgreSQL, MongoDB atau Firebase.

Integration of FE da BE	**Version Control and Colaboration**	**Mobile Dev**
❖ Menghubungkan komponen FE dengan layanan BE melalui API (Application Programming Interface) dalam berkomunikasi antara server dengan database. ❖ Menyelaraskan data dan tampilan antara FE dan BE	❖ Menggunakan Git dalam kolaborasi dan mengelola code dalam tim developer. ❖ Memastikan kode berkembang dengan aman dan sesuai tujuan.	❖ Menggunakan aplikasi mobile dengan framework seperti React, Native dan Flutter.

Dasar Dasar Frontend Web Developer
Frontend merupakan sisi client yang praktiknya menggunakan HTML, CSS dan JS dalam aplikasi web, sehingga user dapat interaksi secara langsung.

HTML (Hypertext Markup Language) : blok bangunan yang mendefinisikan arti dan struktur dari konten web.
CSS : berguna untuk menata halaman web.
JavaScript : berguna dalam membuat web yang interaktif.

Dasar Dasar Backend Development
BE bertanggung jawab dalam memproses permintaan dari pengguna, mengelola dan menyimpan data di database, serta memberi respon kepada client (FE) dari permintaannya.

Bahasa pemrogaman server-side : Node.js (Java Script), Python, Ruby, Java, PHP, C#, dll.
Server Framework : Express.js untuk Node.js, Flask untuk Python, Ruby on Rails untuk Ruby, Spring untuk Java, dan laravel untuk PHP.
Database Management : SQL (MySQL, PostgreSQL, SQL Server) dan NoSQL (MongoDB, Firebase).

Dasar Dasar Database Management
Digunakan dalam mengelola data dalam aplikasi atau sistem. Database mengimpan dan mengorganisisr informasi dalam aplikasi.

Database Management System : perangkat lunak untuk mengelola dan mengakases data dalam database. DBMS menyediakan interface untuk berinteraksi dengan database.
Tipe Database : SQL (Structured Query Languange) atau database relasional dan NoSQL (Not Only SQL) atau database Non Relasional.
Bahasa Query : SQL untuk berinteraksi dengan database SQL. Bahasa Query memungkinkan dalam melakukan operasi SELECT, INSERT, UPDATSE dan DELETE.

Dasar Dasar Mobile Development
Mobile Development membuat aplikasi yang dijalankan di mobile (smartphone dan tablet).

Platform Mobile : Android, IOS, dan Windowsphone.
Aplikasi Android dapat dibangun dengan java atau kotlin, IOS dibangun dengan SWIFT dan Objective-C.
IDE (Intergrated Development Enviroment) : software yang menyediakan alat bantu, penyunting code, pengelola proyek, simulator perangkat, fasilitas debugging untuk memudahkan proses pengembangan.

2. Skillset Full Stack Development

Pengembangan Aplikasi End-to-End

Pendekatan yang mencakup keseluruhan siklus pembuatan aplikasi dari tahap perencanaan hingga pengujian dan implementasi. Tujuannya untuk menghasilkan aplikasi yang lengkap, fungsional dan siap digunakan oleh end-user.

Tahap-tahap Pengembangan End-to-End

1. Perencanaan dan analisis
    - Mengumpulkan kebutuhan dan pemahaman mendalam mengenai tujuan aplikasi, sasaran pengguna dan lingkungan operasional.
    - Analisis kebutuhan dan riset pasar untuk mengidentifikasi fitur utama yang perlu ada dalam aplikasi.
2. Desain
    - Merancang UI & UX yang intuitif dan menarik.
    - Merencanakan arsitektur aplikasi, termasuk pemilihan teknologi, database, dan framework yang sesuai.
3. Pengembangan FE
    - pembangunan bagian depan aplikasi dengan HTML, CSS dan JS.
    - Framework yang digunakan developer seperti React, Angular atau Vue.js.
4. Pengembangan BE
    - Membangun sisi server dengan logika bisnis aplikasi.
    - Bahasa pemrogaman server-side terdiri atas Node.js, python, ruby, atau java.
    - Framework terdiri atas Express.js, Flask, atau Ruby on Rails.
5. Integrasi dan pengujian
    - Bagian depan dan belakang aplikasi diintegrasikan melalui API, sehingga dapat berkomunikasi dan berbagi data.
    - Pengujian dilakukan untuk memastikan agar fitur berfungsi dengan benar dan mengidentifikasi & memperbaiki bug.
6. Pemeliharaan dan Peningkatan
    - Aplikasi dipelihara dengan memperbaiki bug & menangani perubahan enviroment atau kebutuhan bisnis.
    - Memperbaharui fitur, meningkatkan kinerja & memastikan aplikasi tetap relevan dalam waktu yang berlanjut.

Kolaborasi Efektive

Version Control (pengendali versi) merupakan sistem yang memungkinkan software melacak perubahan pada code sumber aplikasi selama pengembangan. Kolaborasi efisien dilakukan antar anggota tim pada project yang sama. Contoh version control adalah git dan mercurial.

Manfaat Version Control

1. Merekam perubaha, setiap detail perubahan akan direkam oleh version control.
2. Pencatatan riwayat, memungkinkan tim untuk melihat riwayat lengkap dari seluruh perubahan yang terjadi pada proyek.
3. Pemecahan konflik, membantu mengidentifikasi dan menyelesaikan konflik.
4. Pemulihan mudah, memudahkan dalam memulihkan kode ke versi sebelumnya jika ada masalah/bug. Tujuannya untuk mengurangi resiko kehilangan pekerjaan.

Penggunaan Version Control

1. Inisiasi Project
   Pada saat memulai projek, membuat repositori yang akan menyimpan code sumber file dan perubahan yang dilakukan selama pemembangan.
2. Pengembangan Paralel
   Setiap anggota tim akan memiliki salinan repo pada komputernya dan pekerjaan dapat dialkukan paralel dan membuat perubahan.
3. Branching
   Memungkinkan membuat cabang yang terpisah dari kode utama. Memungkinkan tim untuk mengisolasi perubahan dan fitur yang sedang dikembangkan.
4. Merge
   Setelah fitur mengalami perubahan, cabang dapat digabungkan ke cabang utama (merge).
5. Pull Request
   Mekanisme yang memungkinkan pengembang mengajukan perubahan mereka dan ditinjau oleh anggota tim lain sebelum di merge ke cabang utama.

---

3. Tools Fullstack Web/Mobile Dev

IDE → Code Editor (Ex: Visual studio code)
Version Control → Repositori (Ex: Github, Gitlab, Bitbucker)
Version Control → Git Tools (Ex: Sourcetree, Gitlens)
DBMS (Ex: PostgreSQL, MySQL, Oracle, MongoDB, Redis)
API (Ex: Postman, Swagger)
Text dan Debugging (Jest, Mocha, Chai, JUnit, 5)
Mobile Development (React Native, Flutter)
Layanan Cloud (AWS, Google, Cloud, Azure)
CI/CD (Jenkins, CircleCI)
Design UI/UX (Figma, Sketch)

Roadmap

HTML (Basic Text dan Shapes) → CSS (Style HTML) → JS (Interactive Element) → Python (Data Processing) → SQL (Data Manipulation) → Node.Js (Programming Server) → Full Stack Development.

SLDC & Design Thinking Implementation

1. What is SLDC?
2. Model Model SLDC
3. Design Thinking Implementation

---

1. What is SLDC

SLDC merupakan siklus hidup pengembangan software yang prosesnya terstruktur serta merupakan metodologi yang digunakan untuk merancang software dari awal hingga akhir.

Siklus SLDC

1. Perancangan dan analisis
2. Desain
3. Pengembangan
4. Pengujian
5. Penerapan
6. Pemeliharaan

1. Perencanaan dan Analisis

- Mengidentifikasi masalah atau kebutuhan bisnis yang perlu diselesaikan oleh perangkat lunak.
- Stakeholders mengumpulkan requirements dan menentukan project scope.
- Membuat rencana project secara keseluruhan.
- Rencanan melibatkan alokasi sumber daya, jadwal waktu, definisi tugas dan tanggung jawab anggota tim.

2. Desain Produk

- Perangakat lunak dibangun secara rinci berdasarkan pada persyaratan yang sudah dikumpulkan.
- Desain mencakup arsitektur sistem, interface, dan database.

3. Pengembangan Produk

- Tahap implementasi rancangan software.
- Menuliskan kode untuk menghasilkan produk software yang berfungsi.

4. Pengujian Produk

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

5. Penerapan Produk

- Implementasi rancangan perangkat lunak yang telah disetujui sebelumnya.
- Menulis kode agar menghasilkan produk yang berfungsi.

6. Pemeliharaan Produk

- Memperbaiki bug, meningkatkan fitur, dan menjaga merangkat lunak sesuai dengan perubahan kebutuhan bisnis.

Manfaat Penggunaan SLDC

1. Prediktasbilitas dan pengendalian projek
2. Peningkatan kualitas perangkat lunak
3. Pengelolaan resiko yang lebih baik
4. Efisiensi tim dan kolaborasi
5. Mememnuhi kebutuhan pengguna
6. Penghematan biaya dan waktu
7. Meningktakan pengawasan dan evaluasi
8. Peningkatan Dokumentasi

Kesimpulan

Penggunaan SLDC secara efektif dapat meningkatkan keberhasilan dan efisiensi dalam mengembangkan aplikasi, pengiriman produk yang berkualitas dalam waktu yang tepat dan memberikan nilai yang besar bagi pelanggan dan stakeholders.

---

2. Model Model SLDC

1. Waterfall

- Waterfall adalah SLDC linear dan berurutan, dimana setiap tahapannya perlu diselesaikan terlebih dahulu sebelum dilanjutkan ke tahap selanjutnya.
- Tahapan Waterfall : analisis, perencanaan, desain, development, pengujian, implementasi, dan maintanance.
- Model ini cocok untuk project dengan persyaratan jelas dan stabil.

2. V-Shaped Model

- Model ini berkaitan erat dengan model waterfall, tetapi lebih ditekankan pada pengujian.
- Setiap tahapan pengembangan, memiliki tahap pengujiannya sendiri.
- Cocok untuk project yang berfokus pada kualitas tinggi.

3. Prototype

- model yang menciptakan prototypr atau contoh awal sebelum dilakukan pengembangan.

- Model ini berfokus pada pemahaman kebutuhan pengguna dan mengumpulkan feedback agar sesuai ekspetasi serta persyaratan pengguna.
- Tahapan : Requirements → Quick Design → Build Prototype → User Evaluation → Refining Prototype → Implementing dan Maitain.

4. Spiral Model

- Menggabungkan model spiral dengan pendekatan inkremental, dimana setiap siklus spiral memebangun pada inkrementasi sebelumnya, sehingga menghasilkan perangkat lunak yang semakin berkembang dengan fitur yang lebih banyak tiap siklusnya.
- Model ini cocok untuk project besar dan kompleks dengan resiko yang banyak.

5. Iteratif Incremental Model

- Melibatkan pengulangan siklus pembangunan dan peningkatan software dalam tahapan kecil.
- Setiap iterasi menambahkan lebih banyak fitur, sehingga produk akhir mencapai tingkat kesempurnaan yang sesuai.
- Model ini cocok untuk project dengan waktu dan anggaran terbatas.

6. BigBang Model

- Model yang kurang terstruktur, tahapan pengembangan dilakukan tanpa perencanaan.
- Development dilakukan tanpa melakukan analisis dan perencanaan mendalam.
- Cocok untuk project kecil atau prototyping.

7. Agile Model

- Pendekatan kolaboratif dan iteratif yang berfokus pada pengiriman perangkat lunak secara berkala dan inkremental.
- Tim bekerja dalam sprint (iterasi singkat) dan terbuka untuk perubahan.
- Cocok untuk project dengan lingkungan yang dinamis dan berubah-ubah.

---

3. Design Thinking Implentation

Tahapan Design Thinking

1. Empathize : memahami kebutuhan user
2. Define : mendefinisikan masalah
3. Ideate : menghasilkan ide
4. Prototype : pembangunan cepat dan solusi iteratif
5. Test : mengumpulkan feedback
6. Implement : develop software

Empathize

- Memahami pengguna akhir, kebutuhan, keinginan dan masalah yang mereka hadapi.
- Memahami user dapat dilakukan dengan cara berikut, yaitu :
    1) User Research, cara ini dapat dilakukan dengan cara wawancara, observasi dan survei dalam mengumpulkan data kuantitatif dan kualitatif mengenai perilaku dan preferensi pengguna.
    2) Empathy mapping, memebuat pemetaan untuk menggambarkan sikap, pemikiran, perasaan dan masalah pengguna.
    3) User Persona, dengan membuat persona pengguna yang mewakili kelompok pengguna. User persona membantu developer dan desainer dalam mengingat target audiens.

Define

- Menganalisis informasi, menentukan dan menetapkan tujuan yang jelas untuk project.
- Kegiatan kunci, dilakukan sebagai berikut.
    1) Problem statement, menyusun persyaratan maslaah yang jelas dan ringkas, membuat arti atau definisi tantangan dari pandangan pengguna.
    2) Stakeholder Aligment, kolaborasi antara stakeholder, pemilik produk, manajer project, desainer dan developer dalam memastikan semua orang sejalan dengan tujuan dan ruang lingkup project.

Ideate

- Fase mendorong pemikiran kreatif fan menghasilkan solusi potensial.
- Kegiatan kunci, dilakukan sebagai berikut.
    1) Brainstorming session, melakukan brainstorming kolaboratif dalam menghasilkan banyak ide tanpa menghakimi siapapun
    2) Idea Consolidation mengelompokkan dan mengkonsolidasikan ide-ide terkait. Saring daftar ide dan ambil solusi yang sesuai dengan tujuan dan batasan project.

Prototype

- Menciptakan representasi nyata dari ide-ide yang dipilih.
- Prototype digunakan untuk mengumpulkan feedback dan validasi asumsi.
- Kegiatan kunci, dilakukan sebagai berikut.
    1) Low Fidelity Prototypes, membuat sketsa, wireframe, dan mockup. Prototype ini cepat dan mudah dibuat, sehingga memungkinkan iterasi yang cepat.
    2) High Fidelity Prototype, tipe detail dari prototype yang dikembangkan atau MVP (minimum Valiable Produk) yang menyerupai tampilan dan fungsionalitas produk akhir.

Test

- Pengumpulan feedback dari pengguna nyata untuk validasi solusi- solusi.
- Kegiatan kunci, dilakukan sebagai berikut.
    1) Usability Testing, 1-on-1 session dengan pengguna dalam mengamati interaksi mereka dengan prototype.
    2) Iterative testing, menggunakan feedback dari usability testing untuk berinteraksi pada desain dengan mengatasi masalah kegunaan yang muncul dan akan muncul.

Implement

- Desain diterjemahkan kedalam kode yang sebenarnya dan diimplementasikan.
- Kegiatan kunci, dilakukan sebagai berikut.
    1) Agile Development, metodologi agile seperti scrum atau kanban memungkinkan development secara bertahap dan continous deliver.
    2) Cross-Functional Colaboration, merupakan kolaborasi antara desainer, pengembang, penguji dan stakeholder untuk memastikan implementasi selaras dengan solusi yang telah dirancang.

Basic Git & Collaborating Using Git

1. Terminal dan IDE
2. Installing, initializing, and commiting GIT
3. Collaborating Using Git

---

1. Memahami Version Control Git

Memahami Version Git Control Git

Control version merupakan metode untuk melacak dan mengelola perubahan dalam kode sumber dan berkas project. Contoh version control yang terdistribusi adalah Git.

1) Sistem Kontrol Versi Terpusat (Central Version Control System), terdapat satu repositori yang berfungsi sebagai "master" dalam menyimpan seluruh project. Setiap pengembang melakukan perubahan pada salinan lokal, kemudian mengirimkan perubahan tersebut ke repositori sentral. Contoh sistem kontrol versi terpusat adalah Subversion (SVN).
2) Sistem Kontrol Versi Terdistribusi (Distributed Version Control System), 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.

Apa itu GIT

Git merupakan sistem konteol versi terdistribusi untuk melacak perubahan dalam kode, kolaborasi antar anggota tim dan mengella revisi code.

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 → Menduplikasi repositori Git yang sudah ada ke direktori lokal.
4. git add → Menduplikasi repositori Git yang sudah ada ke direktori lokal.
5. git commit → Membuat commit dari perubahan yang sudah di-staging dan menambahkan pesan commit.
6. git pull → Mengambil commit terbaru dari repositori jarak jauh dan menggabungkannya ke repositori lokal.
7. git push → Mengirimkan commit ke repositori jarak jauh (remote repository).
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