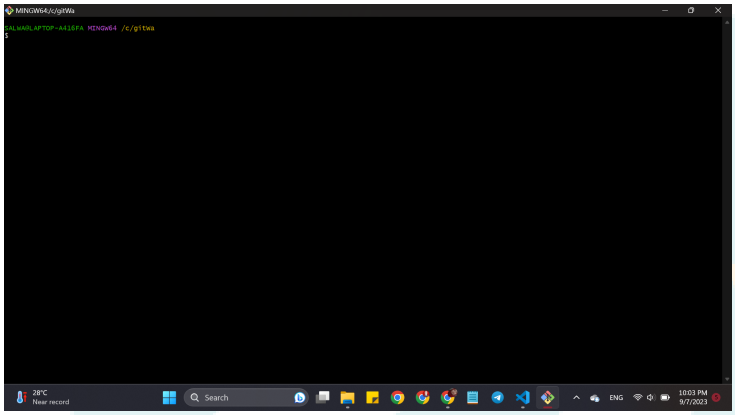
Rakamin Academy

Introduction to Software Engineering

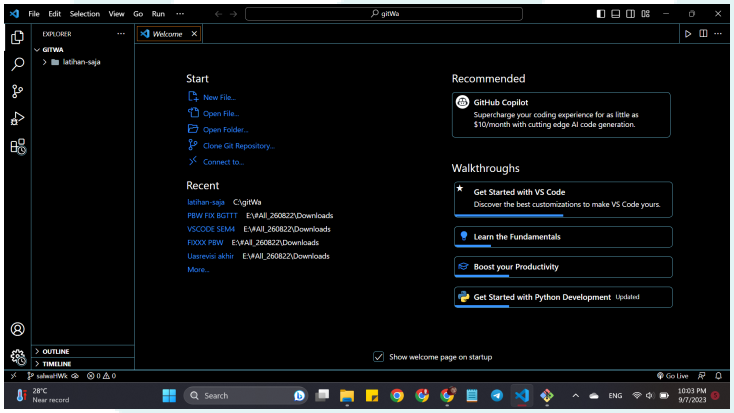
Salwa Salsabila - Kelompok 7
Universitas Singaperbangsa Karawang

Instalasi

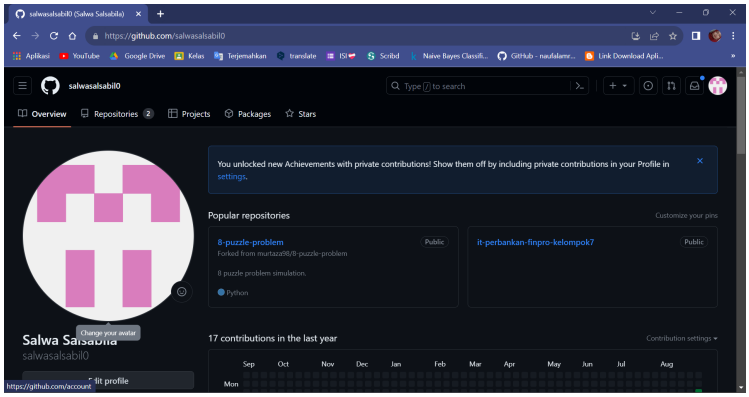
- Git



- Visual Studio Code



- Browser



Rakamin Academy

SUMMARY

A. Full Stack Developer Career Path

Definisi: 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 Penting:

1. Front-End Development, membangun antarmuka menggunakan HTML, CSS, dan JavaScript. Framework seperti React, Angular, Vue.js, atau jQuery.
2. Back-End Development, membangun server dan aplikasi menggunakan bahasa pemrograman server-side seperti Node.js, Python, Ruby, Java, PHP, atau C#. Framework seperti Express.js untuk Node.js, Flask untuk Python, Ruby on Rails untuk Ruby, Spring untuk Java, dan Laravel untuk PHP. Jenis database yang umum digunakan adalah SQL (MySQL, PostgreSQL, SQL Server) dan NoSQL (MongoDB, Firebase).
3. Database Management, mendesain dan mengelola basis data dengan dua tipe database utama yang umum digunakan yaitu SQL atau database relasional dan NoSQL atau database non-relasional.
4. Integration of Front-End and Back-End, menghubungkan komponen front-end dengan layanan back-end melalui API (Application Programming Interface).
5. Version Control and Collaboration, menggunakan sistem pengendalian versi, seperti Git, untuk mengelola perubahan kode dan kolaborasi dalam tim pengembang.
6. Mobile Development, mengembangkan aplikasi mobile menggunakan framework seperti React Native, Flutter.

Tahap tahap pengembangan aplikasi end-to-end: Tahap awal perencanaan dan analisis kebutuhan, perancangan desain UI/UX, pengembangan Front-End, pengembangan Back-End, integrasi dan pengujian aplikasi, tahap terakhir melakukan pemeliharaan dan peningkatan.

Version Control: sistem yang memungkinkan pengembang perangkat lunak untuk melacak perubahan pada kode sumber aplikasi selama pengembangan. Manfaat version control adalah merekam detail perubahan, melihat riwayat lengkap, membantu mengidentifikasi dan menyelesaikan konflik, memulihkan kode ke versi sebelumnya. Penggunaan version control yaitu membuat repositori, dapat bekerja secara paralel, pembuatan cabang (branch) yang

terpisah, cabang dapat digabungkan kembali ke cabang utama, memungkinkan pengembang untuk mengajukan perubahan.

Tools sets sebagai Full Stack Developer: IDE - Code Editor (Visual Studio Code), Version Control - Repository (GitHub, GitLab, Bitbucket), Version Control - Git Tools (Sourcetree, GitLens), DBMS (PostgreSQL, MySQL, Oracle, mongoDB, redis), API (Postman, swagger), Tests dan Debugging (Jest, mocha chai, JUnit 5), Mobile Development (React Native, Flutter), Layanan Cloud (aws, Google Cloud, Azure), CI/CD (Jenkins, Circle CI), Desain UI/UX (Figma, Sketch).

B. SDLC (Siklus Hidup Pengembangan Perangkat Lunak) & Design Thinking Implementation

Definisi: rangkaian proses yang terstruktur dan metodologi yang digunakan untuk mengembangkan perangkat lunak dari awal hingga selesai.

Fase SDLC: *Planning, Analysis, Design, Development, Testing&Integration, Maintenance*.

Manfaat: prediktabilitas dan pengendalian proyek, peningkatan kualitas perangkat lunak, pengelolaan risiko yang lebih baik, efisiensi tim dan kolaborasi , memenuhi kebutuhan pengguna, penghematan biaya dan waktu, meningkatkan pengawasan dan evaluasi, peningkatan dokumentasi

Model SDLC:

1. Waterfall Model, model SDLC yang linier dan berurutan. Tahapannya meliputi analisis, perencanaan, desain, pengembangan, pengujian, implementasi, dan pemeliharaan. Cocok untuk proyek dengan persyaratan yang jelas dan stabil.
2. V-Shaped Model, 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 pengembangan perangkat lunak yang bertujuan untuk menciptakan prototipe atau contoh awal sebelum mengembangkan versi finalnya. Fokus pada pemahaman kebutuhan pengguna dan mengumpulkan umpan balik.
4. Spiral Model, model yang menggabungkan elemen model spiral dengan pendekatan inkremental. Setiap siklus spiral membangun pada inkrementasi sebelumnya, cocok untuk proyek besar dan kompleks dengan banyak risiko.

5. Iterative Incremental Model, model yang melibatkan pengulangan siklus pembangunan dan peningkatan perangkat lunak dalam tahapan-tahapan kecil. Setiap iterasi menambahkan lebih banyak fitur hingga mencapai tingkat kesempurnaan yang diinginkan.
6. Big Bang Model, model yang kurang terstruktur, di mana semua tahapan pengembangan dilakukan tanpa perencanaan yang detail. Cocok untuk proyek 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 selalu terbuka untuk perubahan persyaratan pengguna. Cocok untuk proyek dengan lingkungan yang dinamis dan persyaratan yang berubah-ubah.

Steps Design Thinking

1. Empathize: Understand User Needs
2. Define: Define the Problem
3. Ideate: Generate Ideas
4. Prototype: Build Quick and Iterative Solutions
5. Test: Gather User Feedback
6. Implement: Develop the Software

C. Basic Git & Collaborating Using Git

Terminal and IDE

Terminal adalah antarmuka teks yang digunakan untuk berinteraksi dengan sistem operasi melalui baris perintah. Merupakan alat yang kuat untuk mengelola proyek perangkat lunak, mengedit berkas, menjalankan perintah, dan lainnya. Beberapa IDE (Integrated Development Environment) memiliki integrasi dengan terminal, memungkinkan pengembang untuk menjalankan perintah terminal dari dalam IDE.

Installing, Initializing, and Committing GIT

Git adalah sistem kontrol versi terdistribusi yang memungkinkan pengembang melacak perubahan dalam kode mereka. Instalasi Git dapat dilakukan di berbagai platform seperti Windows, Linux, dan macOS. Perintah dasar Git termasuk ‘git init’ (menginisialisasi repositori), ‘git clone’ (menduplikasi repository), ‘git add’ (menambahkan perubahan ke staging area), ‘git commit’ (membuat commit), dan lainnya.

Collaborating Using Git

Git memungkinkan kolaborasi antara pengembang dalam pengelolaan proyek perangkat lunak. Langkah-langkah kolaborasi melibatkan mengundang kolaborator, membuat branch, membuat perubahan, mengelola konflik, dan mengirimkan permintaan tarik (pull request) untuk menggabungkan perubahan. Kolaborator dapat bekerja di cabang (branch) yang berbeda untuk menghindari konflik langsung dengan perubahan yang dilakukan oleh orang lain.