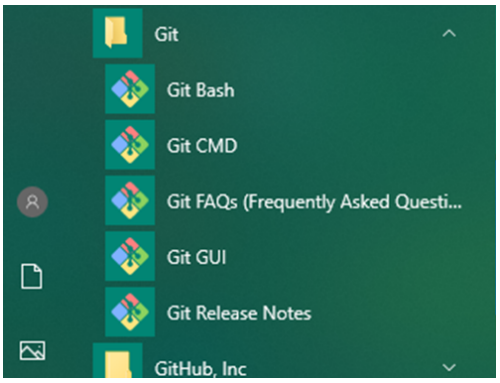
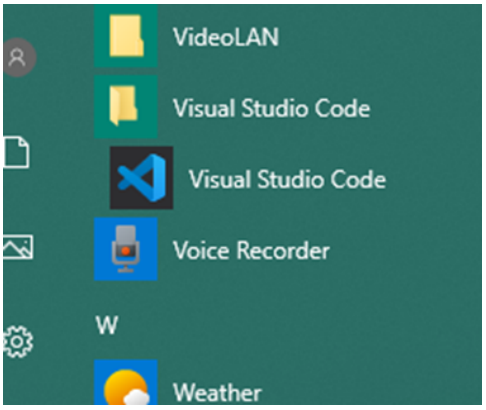
Bukti Instalasi

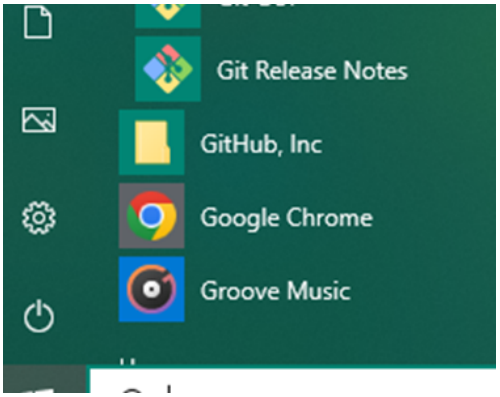
- Git



- Visual Studio Code



- Browser



Summary Introduction to Software Engineering

➢ Introduction Full Stack Web/Mobile Developer

- **Definisi**

Merupakan 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**
  - ➔ Front-End Development

    Membuat tampilan dan interaksi menggunakan HTML, CSS, JavaScript, serta framework seperti React, Angular, atau Vue.js.
  - ➔ Back-end Development

    Menangani logika server dan pengelolaan data dengan bahasa seperti Node.js, Python, Java, atau PHP.
  - ➔ Database Management

    Mengelola penyimpanan data dengan MySQL, PostgreSQL, MongoDB, dll.
  - ➔ Integration of Front-End and Back-End

    Menghubungkan front-end dan back-end melalui API untuk komunikasi data.
  - ➔ Version Control and Collaboration

    Menggunakan sistem pengendalian versi, seperti Git, untuk mengelola perubahan kode dan kolaborasi dalam tim pengembang. Memastikan kode terus berkembang dengan aman dan sesuai dengan tujuan proyek.
  - ➔ Mobile Development

    Beberapa Pengembang Full Stack juga memiliki kemampuan untuk mengembangkan aplikasi mobile menggunakan framework seperti React Native, Flutter.
- **Dasar-Dasar Frontend 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.
  - ➔ HTML

    HTML (HyperText Markup Language) adalah blokbangunan paling dasar dari Web. Ini mendefinisikan arti dan struktur konten web.
  - ➔ CSS

    CSS adalah bahasa yang kami gunakan untuk menata halaman Web. Dan biasanya CSS digunakan untuk kosmetik dari sebuah web yang kita punya.

➔ 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 Backend Web 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.

  ➔ 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.

  ➔ Server Framework

  Framework seperti Express.js untuk Node.js, Flask untuk Python, Ruby on Rails untuk Ruby, Spring untuk Java, dan Laravel untuk PHP.

  ➔ Database Management

  Jenis database yang umum digunakan adalah SQL (MySQL, PostgreSQL, SQL Server) dan NoSQL (MongoDB, Firebase).

- **Dasar-Dasar Database Management**

  Database merupakan bagian kritis dari aplikasi karena menyimpan dan mengorganisir informasi yang diperlukan untuk menjalankan aplikasi dengan benar.

  ➔ Database Management System

  Perangkat lunak yang memungkinkan pengguna untuk mengelola dan mengakses data dalam database. DBMS menyediakan antarmuka untuk berinteraksi dengan database.

  ➔ 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.

  ➔ 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**

Pengembangan aplikasi mobile mencakup beberapa aspek penting untuk memastikan aplikasi dapat berjalan dengan baik dan memberikan pengalaman pengguna yang optimal. Framework yang digunakan biasanya React Native dan Flutter (dapat berjalan di android dan ios).

➔ 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.

➔ 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.

➢ Skillset Full Stack Web/Mobile Developer

- **Pengembangan Aplikasi End to End**

Merupakan pendekatan pengembangan perangkat lunak yang mencakup keseluruhan siklus pembuatan aplikasi, dari tahap perencanaan hingga tahap pangujian dan Implementasi. Tujuannya adalah untuk menghasilkan aplikasi yang lengkap.fungsional, dan siap digunakan oleh pengguna akhir.

Berikut adalah **tahapan-tahapan dalam pengembangan aplikasi end-to-end**:

1. Perencanaan dan Analisis

   Tahap awal ini melibatkan pengumpulan kebutuhan dan pemahaman mendalam tentang tujuan aplikasi sasaran pengguna, den lingkungen operasional. Analisis kebutuhan dan riset pasar dilakukan untuk mengidentifikasi fitur utama yang harus dimasukkan daları aplikasi.

2. Desain

   Proses desain melibatkan merancang antarmuka pengguna (UI) dan pengalaman pengguna (UX) yang intuitif dan menarik. Pengembang juga merencanakan arsitektur aplikasi, termasuk pemilihan teknologi, database, dan framework yang sesuai.

3. Pengembangan Front-End

Pada tahap ini, tim pengembang bekerja pada bagian depan aplikasi, menggunakan HTML, CSS, dan JavaScript untuk membuat tampilan dan interaksi yang menarik bagi pengguna. Pengembang mungkin juga menggunakan framework front-end seperti React, Angular, atau Vue.js untuk mempercepat pengembangan.

4. Pengembangan Back-End

   Tahap ini melibatkan pengembangan sisi server dan logika bisnis aplikasi. Pengembang menggunakan bahasa pemrograman server-side seperti Node.js, Python, Ruby, atau Java, dan menggunakan framework back-end seperti Express.js, Flask, atau Ruby on Rails.

5. Integrasi dan Pengujian

   Bagian depan dan belakang aplikasi harus diintegrasikan melalui API (Application Programming Interface) sehingga mereka dapat berkomunikasi dan berbagi data. Pengujian aplikasi dilakukan untuk memastikan semua fitur berfungsi dengan benar dan mengidentifikasi dan memperbaiki bug yang mungkin ada.

6. Pemeliharaan dan Peningkatan

   Setelah diluncurkan, aplikasi harus dipelihara dengan memperbaiki bug dan menangani perubahan lingkungan atau kebutuhan bisnis. Peningkatan terus menerus dilakukan untuk memperbarui fitur, meningkatkan kinerja, dan memastikan aplikasi tetap relevan dalam waktu yang berlanjut.

- **Version Control**

  Merupakan 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. Yang biasa digunakan adalah Git dan Mercurial.

- **Manfaat Version Control untuk Berkolaborasi**

1. Rekam Perubahan

   Setiap kali pengembang membuat perubahan pada kode, sistem version control merekam detail perubahan tersebut.

2. Pencatatan Riwayat

   Version control memungkinkan tim untuk melinat riwayat lengkap dari semua perubahan yang terjadi pada proyek dari awal hingga saat ini.

3. Pemecahan Konflik

Ketika dua atau lebih pengembang melakukan perubahan pada area kode yang sama, version control membantu mengidentifikasi dan menyelesaikan konflik

4. Pemulihan Mudah

   Version control memungkinkan pengembang untuk memulihkan kode ke versi sebelumnya jika ada masalah atau bug yang terjadi, sehingga mengurangi resiko kehilangan pekerjaan.

- **Penggunaan Version Control untuk Berkolaborasi**

1. Inisialisasi Proyek

   Tim memulai proyek dengan membuat repositori version control. Repositori ini akan menyimpan semua kode sumber, file, dan perubahan yang dilakukan selama pengembangan.

2. Pengembangan Paralel

   Setiap anggota tim akan memiliki salinan repositori pada komputernya sendiri. Mereka dapat bekerja secara paralel, membuat perubahan.

3. Branching

   Version control memungkinkan pembuatan cabang (branch) yang terpisah dari kode utama. Ini memungkinkan tim untuk mengisolasi perubahan dan fitur yang sedang dikembangkan.

4. Merge

   Setelah fitur atau perubahan selesai, cabang dapat digabungkan kembali ke cabang utama (biasanya disebut sebagai "merge").

5. 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.

➢ Tools Full Stack Web/Mobile Developer

Sebagai Pengembang Full Stack Web dan Mobile, Anda akan memerlukan kombinasi dari berbagai alat dan teknologi untuk secara efisien membangun dan mengelola aplikasi. Berikut adalah beberapa **set alat yang penting yang mungkin digunakan oleh Pengembang Full Stack: (**Visual Studio Code) IDE-Code Editor, (GitHub, Git Lab, Bitbucket) Version Control-Repository, (Sourcetree, GitLens) Version Control-Git Tools, (PostgreSQL, MySQL, Oracle, MongoDB, redis) DBMS, (Postman, Swagger) API, (Jest, Mocha, Chai, JUnit5)

Test dan debugging), (React Native, Flutter) Mobile Development, (AWS, Google Cloud, Azure) Layanan Cloud, (Jenkins, circleci) CI/CD, (Figma, Sketch) Desain UI/UX.

➢ SDLC & Design Thinking Implementation

- **Definisi**

  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.

- **Siklus SDLC (Fase pada SDLC)**

  ➔ 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.

  ➔ Desain Produk

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

  ➔ Pengembangan Produk

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

  ➔ Pengujian Produk

  Setelah perangkat lunak dikembangkan, tahap pengujian dilakukan untuk memastikan banwa perangkat lunak berfungsi sesuai dengan persyaratan yang telan ditentukan. Pengujian mencakup verifikasi fungsionalitas, kinerja, keamanan, dan kualitas keseluruhan perangkat lunak

  ➔ Penerapan Produk

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

➔ Pemeliharaan Produk

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

- **Manfaat Penggunaan SDLC**

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.

➔ 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-Model Software Development Life Cycle (SDLC)**

➔ Waterfall 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.

➔ 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.

➔ 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.

- 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.

- 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.

- 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.

- 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 Implementation (Step Design Thinking)**

  ➔ Empathize: Understand User Needs

  Pada tahap ini, fokus pada memahami secara mendalam pengguna akhir dan kebutuhan mereka, keinginan, serta masalah yang dihadapi. Adapun tahapan-tahapan empathize:

  - User Research

    Lakukan wawancara, observasi, dan survei untuk mengumpulkan data kualitatif dan kuantitatif tentang perilaku dan preferensi pengguna. Hal ini membantu memahami konteks di mana pengguna akan menggunakan perangkat lunak.

- Empathy Mapping

  Buat pemetaan empati yang menggambarkan sikap, pemikiran, perasaan, dan masalah pengguna. Latihan ini membantu tim pengembangan lebih memahami kebutuhan pengguna.

- User Personas

  Buat persona pengguna fiktif yang mewakili berbagai kelompok pengguna. Persona membantu desainer dan pengembang selalu mengingat audiens target selama proses pengembangan.

➔ 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. kan Ini membantu tim pengembangan tetap fokus pada pemecahan masalah yang tepat.

- 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. Dorong ide-ide liar dan tidak konvensional untuk merangsang kreativitas.

- 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 Quick and Iterative Solution

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 Prototype

  Bangun prototipe rendah seperti sketsa kertas, wireframe, atau mock-up. Prototipe ini cepat dan mudah dibuat, memungkinkan iterasi yang cepat.
- High-Fidelity Prototype

  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. Kumpulkan data kualitatif tentang pengalaman pengguna, masalah, dan kepuasan.
- 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.

- **Basic Git & Collaboration Using Git**
  - ➔ Pengenalan Terminal dan Git
    - Sejarahnya dari tahun 1960 hingga perkembangan modern. Terminal tetap menjadi alat penting bagi pengembang perangkat lunak dan administrator sistem, meskipun antarmuka grafis semakin populer.
    - Terminal memberikan fleksibilitas dan kekuatan dalam melakukan tugas-tugas tertentu serta otomatisasi.
  - ➔ Version Control & Git
    - Pentingnya kontrol versi dalam melacak dan mengelola perubahan dalam kode sumber atau berkas proyek. Terdapat dua jenis sistem kontrol versi: terpusat dan terdistribusi.
    - Dalam sistem kontrol versi terpusat, terdapat satu repositori sentral yang menyimpan seluruh sejarah proyek, sedangkan dalam sistem terdistribusi, setiap anggota tim memiliki salinan lengkap dari repositori.
    - Git diperkenalkan sebagai salah satu sistem kontrol versi terdistribusi yang paling populer, memungkinkan pengembang untuk melacak perubahan, berkolaborasi, dan mengelola revisi kode secara efektif.
    - Proses instalasi Git dijelaskan secara rinci untuk berbagai sistem operasi, termasuk Windows, Linux, dan MacOS, dengan langkah-langkah untuk memverifikasi instalasi dan mengatur variabel lingkungan PATH.
  - ➔ Perintah Dasar Git
    - git init: Menginisialisasi direktori sebagai repositori Git kosong.
    - git clone: Menduplikasi repositori yang sudah ada ke direktori lokal.
    - git status: Menampilkan status perubahan yang belum dikomit.
    - git add: Menambahkan perubahan ke area persiapan untuk disiapkan menjadi commit.
    - git commit: Membuat commit dari perubahan yang sudah di-staging dengan pesan commit.
    - git push: Mengirimkan commit ke repositori jarak jauh.
    - git pull: Mengambil commit terbaru dari repositori jarak jauh dan menggabungkannya ke repositori lokal.

- Dan perintah-perintah lainnya seperti git branch, git checkout, git merge, git log, git remote, git fetch, git diff, dan git reset.

➔ Perintah Dasar Git

- Simulasi kolaborasi menggunakan Git melibatkan tiga orang (Alice, Bob, dan Charlie) yang bekerja bersama dalam sebuah proyek. Proses kolaborasi dimulai dengan membuat repositori baru di GitHub dan mengundang kolaborator.
- Setiap peserta mengklon repositori ke mesin lokal mereka dan membuat cabang untuk pekerjaan masing-masing, misalnya "alice-feature," "bob-feature," dan "charlie-feature."
- Setelah membuat perubahan pada berkas proyek, mereka harus melakukan commit dengan pesan yang deskriptif dan mengirimkan cabang mereka ke repositori jarak jauh.
- Materi ini juga menjelaskan bagaimana mengelola konflik yang mungkin terjadi saat dua atau lebih peserta melakukan perubahan pada berkas yang sama. Peserta harus menyelesaikan konflik secara manual dan melakukan commit perubahan setelahnya.
- Setelah semua peserta selesai bekerja, mereka membuat pull request dari cabang mereka ke cabang utama di GitHub. Proses ini melibatkan peninjauan pull request oleh peserta lain dan penggabungan jika semuanya baik.
- Terakhir, peserta perlu memperbarui repositori lokal mereka untuk mencerminkan perubahan yang dilakukan di cabang utama dan menggabungkan perubahan tersebut ke dalam cabang fitur masing-masing.