**Summary Pengenalan Software Engineer**

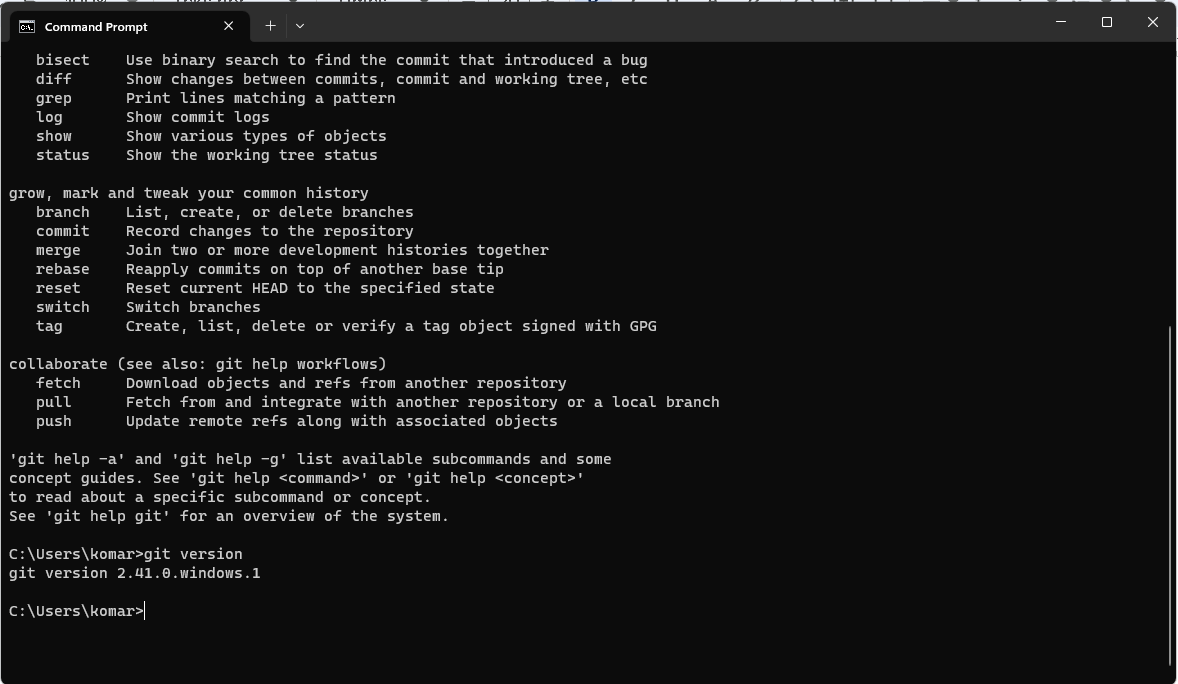
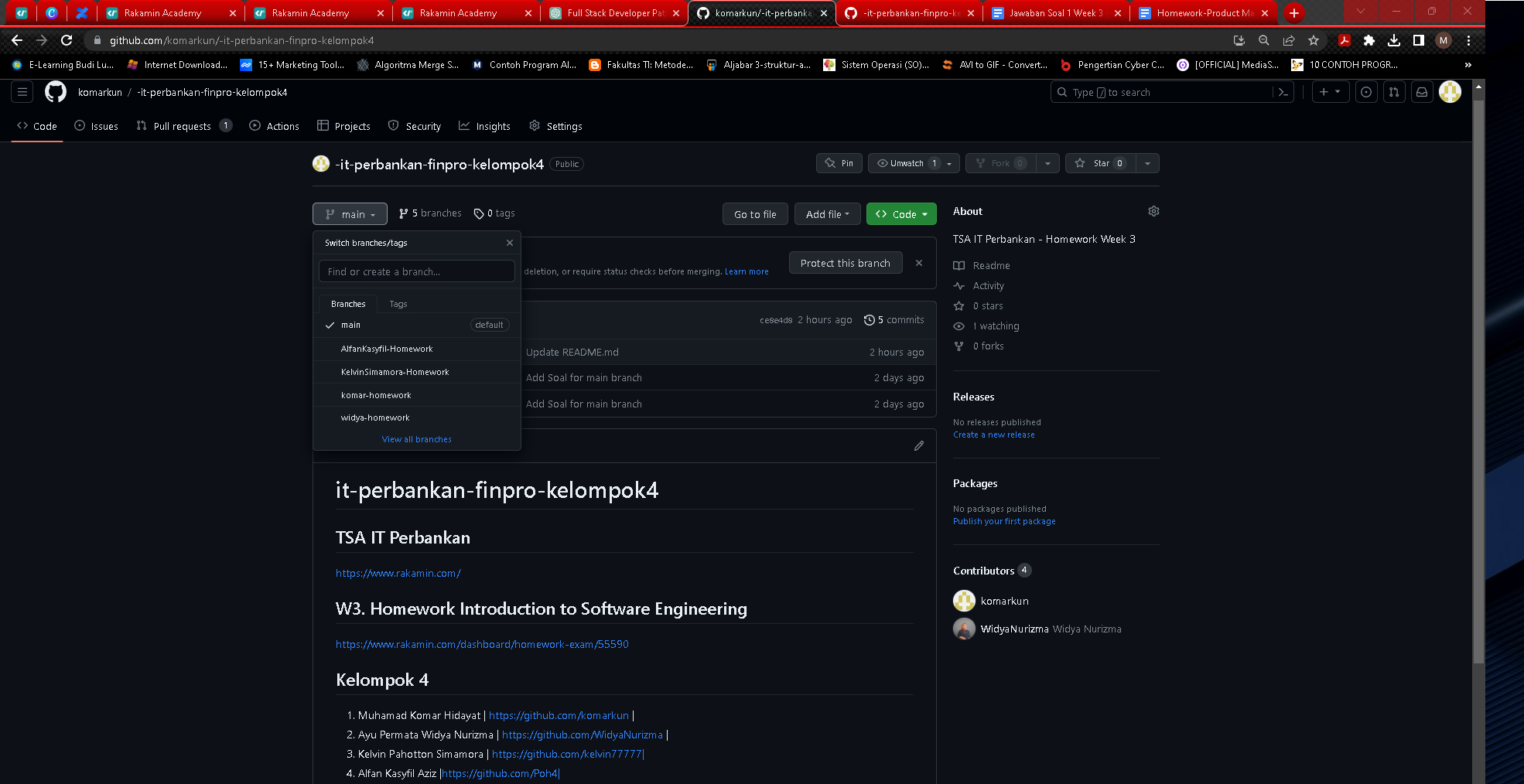
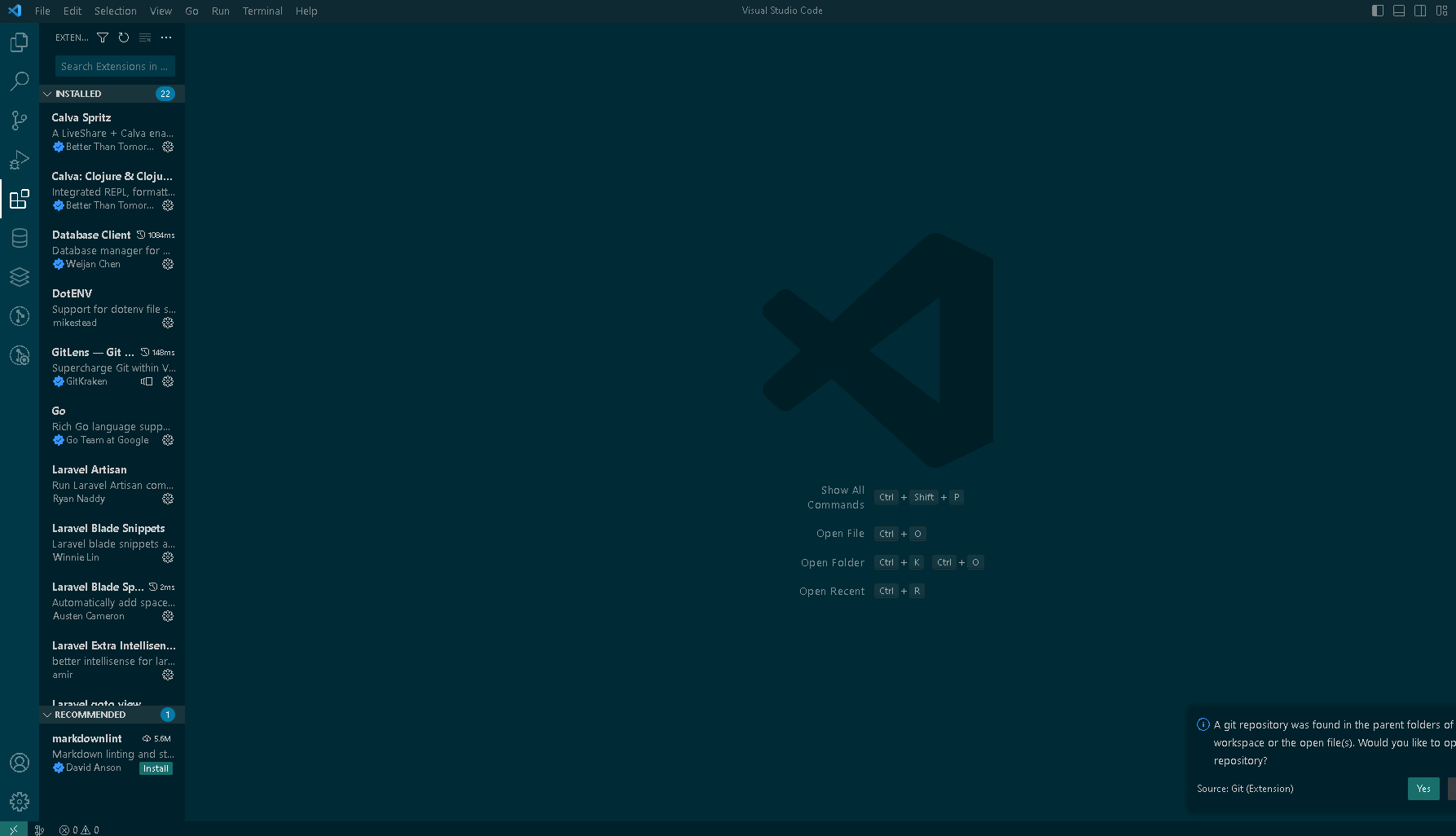
****

**Nama : Muhamad Komar Hidayat**

**Nim : 2011500747**

**Profil Link :** [**https://www.rakamin.com/profile/muhamad-komar-hidayat-b27ttkjac5z47qil**](https://www.rakamin.com/profile/muhamad-komar-hidayat-b27ttkjac5z47qil)

**Installation Success tools**

****

**Full Stack Web/Mobile Developer:**

Seorang Full Stack Web/Mobile Developer adalah seorang profesional di dunia pengembangan perangkat lunak yang memiliki kemampuan untuk bekerja pada seluruh stack teknologi yang digunakan dalam pengembangan aplikasi web atau mobile. Mereka memiliki pengetahuan luas tentang baik sisi frontend (bagian depan) maupun backend (bagian belakang) dari pengembangan perangkat lunak, memungkinkan mereka untuk merancang, mengembangkan, dan memelihara aplikasi dengan tingkat kompleksitas yang beragam.

Skillset Full Stack Web/Mobile Developer:

Seorang Full Stack Web/Mobile Developer perlu menguasai sejumlah keterampilan kunci untuk berhasil dalam karir mereka. Berikut beberapa skillset yang penting:

1. Bahasa Pemrograman: Menguasai beberapa bahasa pemrograman seperti JavaScript, Python, Ruby, atau Java untuk backend, dan HTML, CSS, serta JavaScript untuk frontend.
2. Frontend Development: Kemampuan dalam merancang tampilan dan antarmuka pengguna yang menarik dengan menggunakan teknologi seperti React, Angular, atau Vue.js.
3. Backend Development: Keterampilan dalam membangun server, mengelola basis data, dan mengembangkan logika bisnis menggunakan kerangka kerja seperti Node.js, Ruby on Rails, Django, atau Spring.
4. Database Management: Memahami sistem manajemen basis data (DBMS) seperti MySQL, PostgreSQL, MongoDB, atau Firebase.
5. API Development: Kemampuan untuk membuat dan mengelola API (Application Programming Interface) yang memungkinkan aplikasi berkomunikasi dengan server dan layanan lainnya.
6. Version Control: Pemahaman tentang alat-alat kontrol versi seperti Git untuk mengelola kode sumber.
7. Security: Mengetahui praktik-praktik keamanan web/mobile development, termasuk manajemen otentikasi, otorisasi, dan perlindungan terhadap serangan siber.
8. Deployment and Hosting: Memahami cara mendeploy aplikasi ke server atau cloud dan mengelola hosting.
9. Testing and Debugging: Keterampilan dalam pengujian aplikasi dan penemuan serta perbaikan bug.
10. DevOps and Automation: Memahami konsep DevOps dan kemampuan otomatisasi untuk mempermudah proses pengembangan, pengujian, dan implementasi.

**Tools Full Stack Web/Mobile Developer:**

Seorang Full Stack Web/Mobile Developer menggunakan berbagai alat dan teknologi untuk mengejar karir mereka dengan sukses. Berikut beberapa tools yang umum digunakan:

1. Code Editor/IDE: Contoh alat termasuk Visual Studio Code, IntelliJ IDEA, atau Sublime Text.
2. Version Control System: Git dan platform hosting seperti GitHub atau GitLab.
3. Frontend Frameworks: React, Angular, atau Vue.js untuk pengembangan aplikasi frontend.
4. Backend Frameworks: Node.js, Ruby on Rails, Django, atau Spring untuk pengembangan aplikasi backend.
5. Database Management: MySQL Workbench, MongoDB Compass, atau aplikasi terkait sesuai dengan basis data yang digunakan.
6. API Development: Postman atau Insomnia untuk menguji dan mengelola API.
7. Deployment and Hosting: Heroku, AWS, DigitalOcean, atau platform lainnya untuk mendeploy dan menjalankan aplikasi.
8. Containerization: Docker untuk mengelola kontainer aplikasi.
9. Continuous Integration/Continuous Deployment (CI/CD): Jenkins, Travis CI, atau CircleCI untuk otomatisasi proses pengujian dan implementasi.
10. Monitoring and Analytics: Tools seperti Google Analytics, New Relic, atau Grafana untuk memantau kinerja aplikasi.

Dengan skillset dan tools yang tepat, seorang Full Stack Web/Mobile Developer dapat menjadi ahli dalam mengembangkan aplikasi web atau mobile yang kuat dan inovatif. Mereka juga memiliki fleksibilitas untuk bekerja pada berbagai proyek yang melibatkan teknologi frontend dan backend.

**Software Development Life Cycle (SDLC) & Design Thinking Implementation**

**What is SDLC (Software Development Life Cycle):**

SDLC adalah serangkaian langkah dan proses yang digunakan dalam pengembangan perangkat lunak untuk merencanakan, merancang, mengembangkan, menguji, dan mengelola perangkat lunak dari awal hingga akhir. Ini adalah kerangka kerja sistematis yang memastikan bahwa perangkat lunak yang dihasilkan berkualitas tinggi, dapat diandalkan, dan sesuai dengan kebutuhan pengguna

Model-Model Software Development Life Cycle (SDLC):

Ada beberapa model SDLC yang berbeda yang digunakan dalam pengembangan perangkat lunak, termasuk:

1. Waterfall Model: Model ini mengikuti pendekatan berurutan di mana setiap fase pengembangan dilakukan secara berurutan dan fase baru dimulai setelah fase sebelumnya selesai. Ini cocok untuk proyek-proyek yang memiliki persyaratan yang stabil.
2. Agile Model: Model ini menekankan kolaborasi tim, responsif terhadap perubahan, dan pengembangan beriterasi. Tim bekerja dalam siklus pengembangan yang singkat, menghasilkan perangkat lunak yang dapat dirilis lebih cepat.
3. Scrum: Salah satu kerangka kerja Agile yang paling populer, Scrum membagi proyek menjadi iterasi pendek yang disebut sprint. Setiap sprint menghasilkan fitur atau fungsi yang dapat dirilis.
4. Kanban: Model ini fokus pada visualisasi aliran kerja pengembangan perangkat lunak dan mengelola pekerjaan yang sedang berlangsung dengan menggeser tugas dari satu tahap ke tahap berikutnya.
5. DevOps: Menggabungkan pengembangan perangkat lunak (Dev) dan operasi (Ops) dengan tujuan meningkatkan pengiriman perangkat lunak dan kualitasnya melalui otomatisasi dan kolaborasi yang erat antara tim.

Design Thinking Implementation:

Design Thinking adalah pendekatan inovatif yang digunakan dalam pengembangan perangkat lunak dan produk lainnya. Ini melibatkan pemahaman mendalam terhadap masalah atau tantangan yang ingin dipecahkan dan menciptakan solusi yang fokus pada pengguna. Berikut langkah-langkah implementasi Design Thinking:

1. Empathy (Empati): Mulailah dengan memahami pengguna dan masalah mereka. Ini melibatkan wawancara, pengamatan, dan berinteraksi langsung dengan pengguna untuk mendapatkan wawasan yang dalam.
2. Define (Definisikan): Terjemahkan wawasan yang diperoleh menjadi pernyataan masalah yang jelas dan terfokus. Ini membantu dalam merumuskan tujuan yang jelas.
3. Ideate (Ideasi): Dorong kolaborasi tim untuk menghasilkan sebanyak mungkin ide kreatif untuk menyelesaikan masalah. Gunakan metode seperti brainstorming.
4. Prototype (Prototipe): Buat model atau prototipe dari solusi yang diajukan. Ini memungkinkan pengujian cepat dan iterasi.
5. Test (Uji): Uji prototipe dengan pengguna yang sebenarnya dan peroleh umpan balik. Gunakan umpan balik ini untuk memperbaiki dan memodifikasi prototipe.
6. Iterate (Iterasi): Lanjutkan untuk mengulangi langkah-langkah ini sebanyak yang diperlukan untuk menghasilkan solusi yang optimal.
7. Implement (Implementasi): Setelah solusi final ditemukan, lakukan implementasi dalam pengembangan perangkat lunak, memastikan bahwa solusi yang dihasilkan memenuhi kebutuhan pengguna.

Design Thinking mempromosikan inovasi, pengembangan produk yang lebih baik, dan respons terhadap perubahan dengan berfokus pada kebutuhan pengguna dan pemahaman yang lebih baik tentang masalah yang dihadapi. Ini dapat diintegrasikan ke dalam berbagai model SDLC, terutama dalam kerangka kerja Agile, untuk menghasilkan perangkat lunak yang lebih relevan dan efektif.

**Basic Git & Collaborating Using Git**

**Terminal and IDE:**

Git dapat digunakan baik melalui terminal (baris perintah) maupun di dalam Integrated Development Environment (IDE) seperti Visual Studio Code atau IntelliJ IDEA.

Menggunakan terminal memungkinkan pengguna untuk menjalankan perintah-perintah Git secara langsung, sementara IDE menyediakan antarmuka yang lebih ramah pengguna untuk berinteraksi dengan Git.

**Installing, Initializing, and Committing Git:**

1. Installing Git:

* Mulai dengan mengunduh dan menginstal Git sesuai dengan sistem operasi yang digunakan (Windows, macOS, atau Linux).
* Setelah instalasi selesai, buka terminal atau IDE yang sudah terpasang.

1. Initializing a Git Repository:

Untuk memulai penggunaan Git dalam sebuah proyek, jalankan perintah git init di direktori proyek tersebut. Ini akan membuat repository Git kosong di direktori tersebut.

1. Adding and Committing Changes:

* Untuk menambahkan file ke dalam staging area (persiapan untuk commit), gunakan perintah git add <nama\_file> atau git add . untuk menambahkan semua file yang telah diubah.
* Setelah file-file yang diinginkan telah ditambahkan, lakukan commit dengan perintah git commit -m "Pesan commit".
* Pesan commit sebaiknya menjelaskan perubahan yang dilakukan.

Collaborating Using Git:

* Git sangat berguna dalam kolaborasi tim dalam pengembangan perangkat lunak. Beberapa aspek kolaborasi dengan Git termasuk:

1. Cloning a Repository:

Untuk bergabung dengan proyek yang ada, gunakan perintah git clone <URL\_repository> untuk menduplikasi repository ke komputer lokal.

1. Branching:

Tim dapat bekerja secara paralel dengan membuat cabang (branch) yang terpisah dari branch utama (biasanya master). Gunakan perintah git branch <nama\_branch> untuk membuat branch baru, dan git checkout <nama\_branch> untuk beralih ke branch tersebut.

1. Pushing and Pulling:

Setelah melakukan perubahan di branch lokal, gunakan perintah git push untuk mengirimkan perubahan ke repository pusat (biasanya di platform seperti GitHub atau GitLab).Untuk mendapatkan perubahan dari repository pusat, gunakan perintah git pull.

1. Merging and Resolving Conflicts:

Setelah perubahan di branch sudah siap, pengguna dapat menggabungkan (merge) branch tersebut dengan branch utama melalui pull request atau perintah git merge. Konflik mungkin terjadi jika ada perubahan yang bertentangan, dan konflik ini harus diatasi secara manual.

1. Pull Requests (PRs) and Code Reviews:

Dalam platform kolaborasi seperti GitHub, pengguna dapat membuat pull request untuk mengajukan perubahan ke repository pusat.

1. Tim dapat melakukan code review terhadap perubahan yang diajukan sebelum melakukan merge.

Git adalah alat yang kuat untuk mengelola versi perangkat lunak, memungkinkan kolaborasi yang efisien dan mengikuti perubahan di seluruh proyek. Dengan pemahaman dasar tentang Git dan praktik kolaborasi yang baik, tim pengembangan dapat bekerja lebih efektif dan menjaga kualitas perangkat lunak yang tinggi.