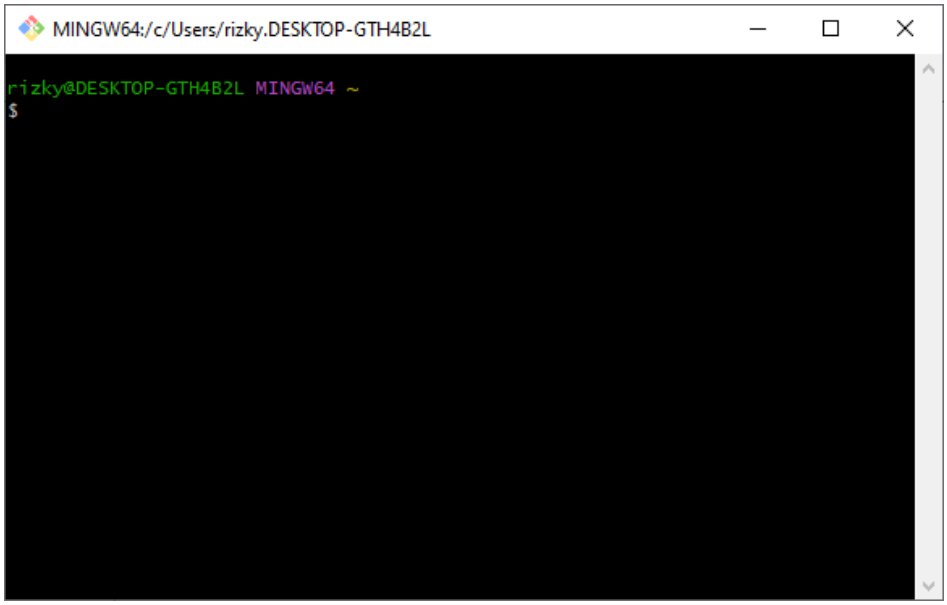
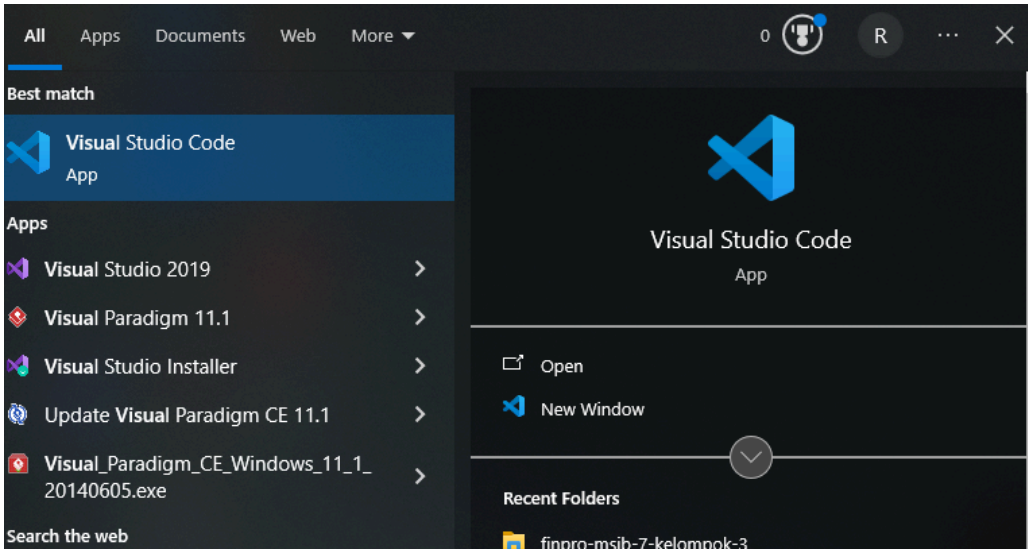
Nama : Muhammad Rizky Ardian
Program : Studi Independen Batch 7

- Link GitHub : https://github.com/Faqihrofiqi/lm-msib-7-kelompok-2.git

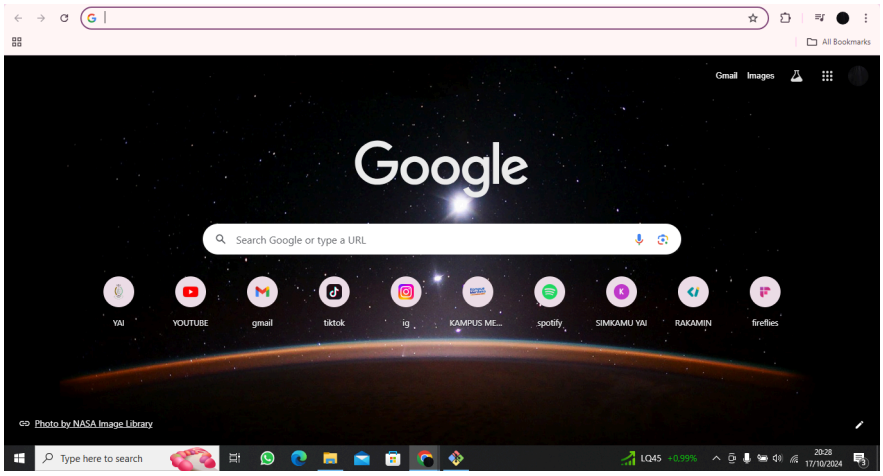
- Install Git



- Install Visual Studio Code



- Browser



Summary Introduction to Software Engineering

➔ Full Stack Developer Career Path

- **Introduction Full Stack Web/Mobile Developer**
  - Pengembangan full-stack mencakup pengembangan aplikasi baik di sisi front-end maupun back-end, memberikan pendekatan yang komprehensif untuk membangun perangkat lunak.
  - Pengembangan front-end melibatkan penggunaan HTML untuk struktur, CSS untuk styling, dan JavaScript untuk interaktivitas untuk membuat antarmuka pengguna yang menarik.
  - Kerangka kerja front-end yang populer seperti React, Angular, dan Vue.js memperlancar proses pengembangan dan meningkatkan responsivitas aplikasi web.
  - Pengembangan back-end fokus pada pemrograman sisi server, mengelola permintaan pengguna, dan memproses data dengan bahasa seperti Python, Ruby, Java, dan PHP.
  - Manajemen basis data sangat penting untuk menyimpan, mengambil, dan memanipulasi data aplikasi, dengan basis data SQL dan NoSQL menjadi teknologi utama.
  - Integrasi menghubungkan komponen front-end dan back-end melalui API, memungkinkan komunikasi yang lancar antara server dan basis data.
  - Sistem kontrol versi seperti Git sangat penting untuk menjaga integritas kode dan memfasilitasi kolaborasi di antara para pengembang.

- Pengembangan mobile berfokus pada pembuatan aplikasi untuk platform seperti Android dan iOS, menggunakan bahasa seperti Java dan Swift.
- Kerangka kerja seperti React Native dan Flutter memungkinkan pengembangan aplikasi mobile lintas platform yang efisien, menyederhanakan proses bagi para pengembang.
- Memilih alat dan kerangka kerja yang tepat sangat penting untuk pengalaman pengembangan yang sukses, mendorong pembelajaran yang efektif dan pelaksanaan proyek.

- **Skillset Full Stack Web/Mobile Developer**
  - Perencanaan dan analisis yang matang sangat penting sebelum memulai proyek pengembangan aplikasi apa pun.
  - Fase desain fokus pada penciptaan antarmuka pengguna (UI) yang menarik dan meningkatkan pengalaman pengguna (UX).
  - Pengembangan frontend biasanya menggunakan HTML, CSS, dan JavaScript untuk membuat antarmuka yang interaktif.
  - Pemrograman backend dapat melibatkan bahasa seperti Python dan PHP, yang fokus pada fungsi sisi server.
  - Integrasi antara frontend dan backend sangat penting untuk memastikan komunikasi yang mulus melalui API.
  - Pengujian yang ketat diperlukan untuk memastikan bahwa semua fitur aplikasi berfungsi dengan baik sebelum diluncurkan.
  - Pemeliharaan pasca peluncuran sangat penting untuk menangani bug dan meningkatkan fitur berdasarkan umpan balik pengguna.
  - Kolaborasi di antara anggota tim dapat ditingkatkan dengan menggunakan sistem kontrol versi seperti Git.
  - Kontrol versi memungkinkan pelacakan perubahan kode, mengelola riwayat proyek, dan menyelesaikan konflik dengan efisien.
  - Pemahaman menyeluruh tentang proses pengembangan dari awal hingga akhir adalah kunci untuk proyek perangkat lunak yang sukses.

- **Tools Full Stack Web/Mobile Developer**
  - Visual Studio Code direkomendasikan sebagai editor kode yang efektif untuk pengembangan full-stack.
  - Pengontrol versi sangat penting, dengan Git, GitHub, dan GitLab menjadi pilihan populer untuk mengelola kode.
  - Memahami berbagai opsi DBMS seperti PostgreSQL, MySQL, Oracle, MongoDB, dan Redis sangat penting untuk manajemen data.

- API berfungsi sebagai jembatan antara frontend dan backend, dengan alat seperti Postman dan Swagger untuk pengujian dan dokumentasi.
- Kerangka pengujian seperti JUnit (untuk Java) dan Jest (untuk JavaScript) sangat penting untuk pengujian unit dan debugging aplikasi.
- React Native disorot sebagai kerangka utama untuk pengembangan mobile.
- Layanan cloud seperti AWS, Google Cloud, dan Azure penting untuk hosting aplikasi, dengan tingkat gratis tersedia untuk percobaan.
- Alat CI/CD seperti Jenkins dan CircleCI memfasilitasi proses otomatisasi penyebaran dan integrasi.
- Alat desain seperti Figma dan Adobe XD sangat penting untuk membuat antarmuka yang ramah pengguna.
- Peta pembelajaran yang direkomendasikan mencakup HTML, CSS, JavaScript, bahasa backend, basis data SQL, pemrograman server, dan penyebaran aplikasi mobile.

➔ SDLC & Design Thinking Implementation

- **What is SDLC**

  Siklus Hidup Pengembangan Perangkat Lunak (SDLC), yaitu proses terstruktur yang sangat penting untuk pengembangan perangkat lunak yang efektif. Terdapat penjeleasan tahap-tahap kuncinya: Perencanaan dan Analisis, Desain, Pengembangan, Pengujian, Penyebaran, dan Pemeliharaan. Setiap tahap sangat penting untuk memastikan bahwa perangkat lunak memenuhi kebutuhan bisnis dan standar kualitas. Materi ini menekankan manfaat SDLC, termasuk prediksi proyek yang lebih baik, kualitas perangkat lunak yang lebih tinggi, dan manajemen risiko yang lebih baik. Dengan mengikuti SDLC, organisasi dapat meningkatkan efisiensi dan memberikan produk perangkat lunak yang berharga yang memenuhi harapan pemangku kepentingan.

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

  SDLC mencakup berbagai model pengembangan perangkat lunak, masing-masing dengan fitur dan manfaatnya sendiri.

  1) Model Waterfall adalah pendekatan linier yang mengharuskan setiap fase selesai sebelum berpindah ke fase berikutnya, cocok untuk proyek dengan kebutuhan yang jelas dan stabil.
  2) Model V menekankan pengujian di setiap tahap pengembangan, membuatnya ideal untuk proyek yang fokus pada kualitas tinggi tetapi kurang fleksibel terhadap perubahan kebutuhan.

3) Model Prototipe bertujuan untuk mengumpulkan umpan balik pengguna lebih awal dengan membuat prototipe awal, yang membantu mengidentifikasi fitur yang hilang tetapi bisa menyebabkan penambahan ruang lingkup.
4) Model Spiral menggabungkan pengembangan iteratif dengan analisis risiko, memungkinkan perubahan di setiap iterasi, tetapi bisa mahal karena banyak siklus.
5) Model Iteratif Inkremental fokus pada siklus pengembangan kecil, memudahkan deteksi cacat lebih awal tetapi memerlukan rincian kebutuhan yang mendetail.
6) Model Big Bang kurang terstruktur dan fleksibel, cocok untuk proyek kecil, tetapi memiliki risiko tinggi terhadap masalah yang tidak terdeteksi.
7) Model Agile mendorong kolaborasi dan kemampuan beradaptasi terhadap perubahan kebutuhan, menekankan pengiriman cepat fitur fungsional.

Memilih model STLC yang tepat tergantung pada kebutuhan spesifik proyek, termasuk kejelasan kebutuhan dan ukuran proyek.Memahami kekuatan dan kelemahan masing-masing model sangat penting untuk pengembangan dan manajemen proyek yang sukses.

- **Design Thinking Implementation**

  Materi ini menjelaskan penerapan Design Thinking dalam Siklus Hidup Pengembangan Perangkat Lunak (SDLC). Sesi ini menekankan pentingnya memahami kebutuhan pengguna melalui empati, merumuskan pernyataan masalah yang jelas, dan menghasilkan solusi kreatif. Pada hal ini menjelaskan proses iteratif prototyping dan pengujian, mendorong kolaborasi di antara para pemangku kepentingan untuk memastikan produk akhir sesuai dengan harapan pengguna. Dengan menerapkan prinsip-prinsip Design Thinking, tim dapat menciptakan solusi perangkat lunak yang lebih intuitif dan berorientasi pada pengguna yang dapat beradaptasi dengan perubahan permintaan pasar, sehingga akhirnya mencapai tujuan bisnis.

➔ Basic Git & Collaborating Using Git

- **Terminal and IDE**

  Terminal adalah antarmuka berbasis teks yang sangat penting untuk berinteraksi dengan komputer, terutama bagi para pengembang dan administrator sistem. Sejarah terminal dimulai pada tahun 1960-an, ketika pengguna mengakses komputer mainframe melalui terminal dasar dengan layar dan keyboard. Pengenalan terminal VT 100 pada tahun 1970-an membawa fitur signifikan seperti gerakan kursor dan dukungan untuk berbagai protokol komunikasi. Pada tahun

1980-an, emulator terminal menjadi populer, memungkinkan komputer pribadi untuk mensimulasikan terminal fisik untuk interaksi pengguna. Terminal modern di abad ke-21 memiliki fitur canggih, termasuk penyorotan sintaks, antarmuka tab, dan akses ke layanan eksternal.

- Perintah dasar seperti 'dir' (Windows) dan 'ls' (Linux) digunakan untuk menampilkan daftar file dalam direktori.
- Perintah 'cd' sangat penting untuk berpindah direktori, sedangkan 'mkdir' digunakan untuk membuat direktori baru.

Pengguna harus berhati-hati dengan perintah kuat seperti 'rm' (Linux) dan 'del' (Windows) untuk menghindari penghapusan file secara tidak sengaja.
Keterampilan dengan perintah dasar terminal meningkatkan efisiensi dalam pengembangan perangkat lunak dan manajemen sistem. Terminal tetap relevan meskipun ada peningkatan antarmuka pengguna grafis, menawarkan fleksibilitas dan kontrol untuk tugas-tugas yang lebih maju.

- **Installing, Initializing and committing GIT**

Git adalah sistem pengendalian versi terdistribusi yang memungkinkan kolaborasi yang efisien dan pelacakan perubahan di dalam proyek. Memahami perbedaan antara Sistem Pengendalian Versi Terpusat (CVCS) dan Sistem Pengendalian Versi Terdistribusi (DVCS) sangat penting untuk manajemen proyek yang efektif. Git dapat diinstal di berbagai sistem operasi, termasuk Windows, Linux, dan macOS, dengan langkah-langkah khusus untuk setiap platform.

- Perintah 'git init' digunakan untuk menginisialisasi direktori sebagai repositori Git, menandai awal pengendalian versi.
- 'git clone <repo-url>' memungkinkan pengguna untuk membuat salinan lokal dari repositori Git yang sudah ada.
- 'git status' memberikan gambaran tentang keadaan saat ini dari direktori kerja dan perubahan yang telah disiapkan.
- Gunakan 'git add <file>' untuk menyiapkan perubahan sebelum melakukan komit, memastikan Anda mengontrol apa yang akan disertakan dalam komit berikutnya.
- Selalu sertakan pesan yang berarti dengan 'git commit -m "<message>"' untuk mendokumentasikan perubahan untuk referensi di masa mendatang.
- 'git push' sangat penting untuk mengirim komit lokal ke repositori jarak jauh, menjaga proses kolaborasi tetap sinkron.
- 'git pull' mengambil dan menggabungkan perubahan dari repositori jarak jauh ke cabang lokal, memastikan salinan lokal Anda selalu terbaru.

- **Collaborating Using Git**

  Dalam materi ini, memperkenalkan dasar-dasar kolaborasi Git, mendemonstrasikan cara menyiapkan repositori di GitHub dan mensimulasikan kerja tim di antara tiga pengguna: Alice, Bob, dan Charlie. Sesi ini mencakup perintah Git penting untuk mengkloning repositori, membuat cabang, dan membuat perubahan kolaboratif. Ini menekankan pentingnya resolusi konflik ketika beberapa pengguna mengedit file yang sama, memandu melalui proses penggabungan dan cara menangani konflik secara efektif. Materi ini membekali pengguna dengan keterampilan untuk mengelola proyek secara kolaboratif menggunakan Git, meningkatkan kemampuan pengkodean dan kontrol versi mereka.

  Collaborating Using Git mempelajari langkah-langkah penting kolaborasi menggunakan Git, dengan fokus pada pembuatan pull request dan penggabungan cabang. Prosesnya termasuk membuat perubahan di masing-masing cabang, mengirimkan permintaan pull untuk ditinjau, dan menangani konflik penggabungan. Menekankan kerja tim, materi ini mengilustrasikan cara mengelola pembaruan secara efektif di repositori bersama, memastikan semua kontributor tetap disinkronkan. Pada akhirnya, materi ini dibekali dengan keterampilan praktis untuk menavigasi fitur kolaboratif Git, meningkatkan kemampuan mereka untuk mengerjakan proyek tim secara efisien.