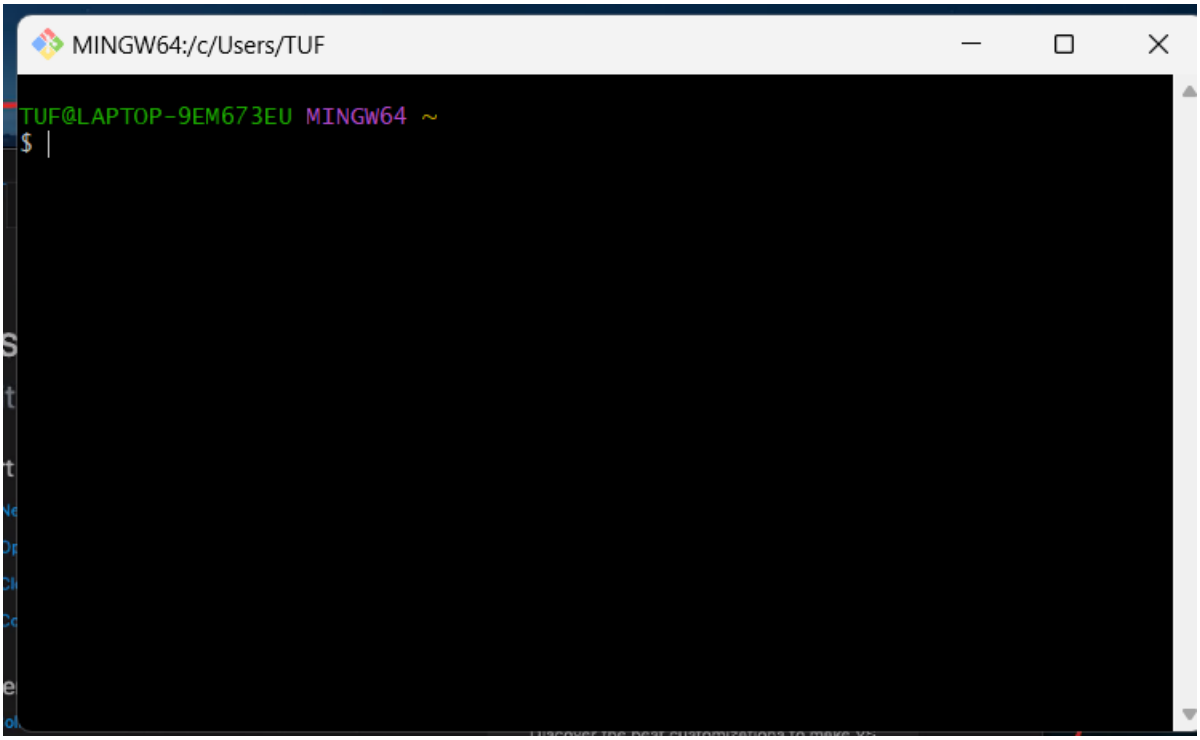
Homework Introduction to Software Engineering

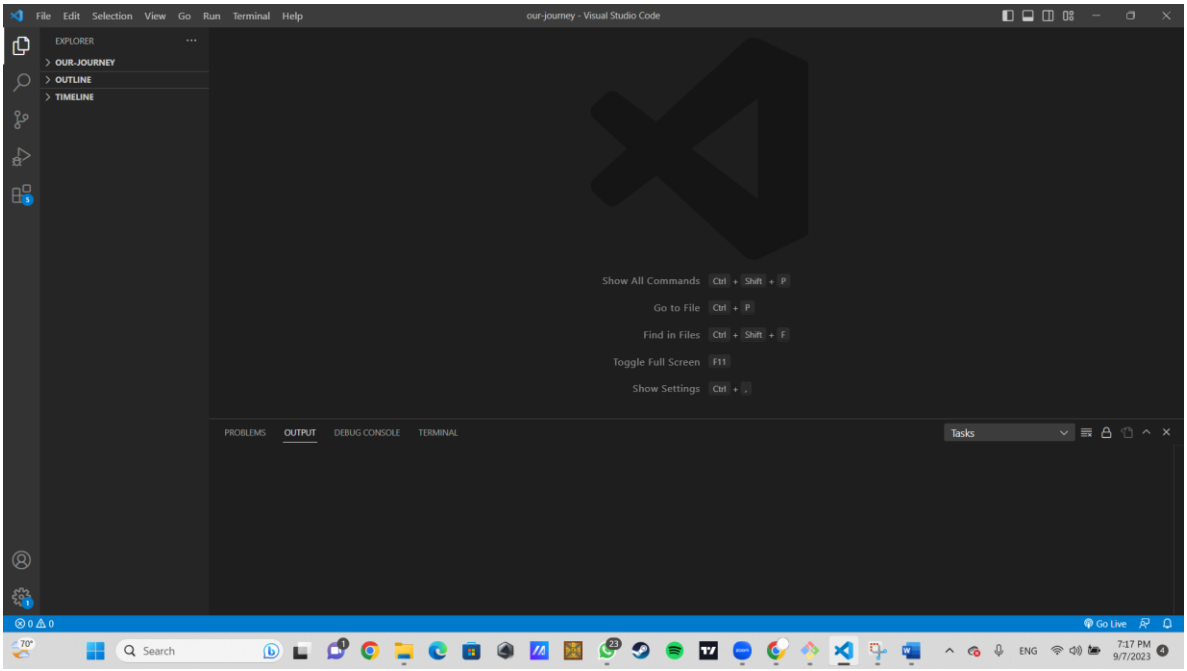
Kevin Cheng

1.

GIT:



Visual Studio Code:



Video 1: Fullstack Developer Career Path

Full Stack development adalah pengembangan seluruh aplikasi secara end-to-end, mencakup front-end, back-end, hingga client-side. Jenis-jenis komponen dalam Full Stack development meliputi Front-End Development, yang bertujuan untuk membangun antarmuka yang menarik dan interaktif; Back-End Development, yang berperan sebagai "otak" dari aplikasi; dan DataBase Management, yang bertujuan untuk mengelola basis data.

Dalam ruang lingkup Full Stack Development, terdapat beberapa aspek penting yang harus diperhatikan, seperti Integrasi antara Front-End dan Back-End, Pengendalian Versi dan Kolaborasi, serta Pengembangan Aplikasi Mobile. Dasar dari Front End Development melibatkan pemahaman dan penguasaan tiga bahasa pemrograman utama, yaitu HTML, CSS, dan JavaScript. Sementara itu, dasar dari Back-End Development melibatkan pengkuasaan bahasa pemrograman server-side, penggunaan server framework, dan manajemen basis data. Untuk aspek DataBase Management, pemahaman tentang Database Management System, jenis-jenis basis data, dan bahasa kueri sangatlah penting. Terakhir, dalam persiapan untuk Mobile Development, diperlukan pemahaman tentang platform mobile yang akan digunakan serta lingkungan pengembangan terpadu (IDE). Semua elemen ini saling terkait dalam mencapai kemahiran Full Stack Development yang komprehensif.

Proses pengembangan aplikasi end-to-end melibatkan beberapa tahap penting. Tahap pertama adalah perancangan dan analisis, di mana tujuan aplikasi, sasaran pengguna, dan penelitian pasar ditetapkan. Tahap desain mengikuti, di mana antarmuka pengguna (UI/UX) yang menarik dirancang. Kemudian, pengembangan front-end dilakukan dengan menggunakan HTML, CSS, dan JavaScript untuk membuat bagian depan aplikasi. Sementara itu, tahap pengembangan back-end berkaitan dengan pengembangan sisi server dan logika bisnis aplikasi menggunakan bahasa pemrograman server-side.

Tahap integrasi dan pengujian penting untuk memastikan bahwa front-end dan back-end dapat berkomunikasi dan berbagi data melalui API (Application Programming Interface) serta untuk memastikan semua fitur berjalan dengan baik. Setelah aplikasi diluncurkan, tahap pemeliharaan dan peningkatan terus berlanjut untuk menjaga dan meningkatkan kinerja dan fungsionalitasnya.

Kolaborasi aktif dalam pengembangan aplikasi didukung oleh sistem Version Control, yang memungkinkan tim pengembang untuk melacak perubahan dalam kode sumber aplikasi selama proses pengembangan. Manfaat dari penggunaan Version Control termasuk rekaman perubahan, pencatatan riwayat lengkap, kemampuan pemecahan konflik, dan kemudahan pemulihan kode ke versi sebelumnya jika terjadi masalah atau bug. Dalam penggunaannya, tim menginisialisasi proyek dengan membuat repositori version control, melakukan merge setelah fitur atau perubahan selesai, bekerja secara paralel dalam pengembangan, mengajukan pull request untuk ditinjau oleh anggota tim lain, dan menggunakan branching untuk mengisolasi perubahan

dan fitur yang sedang dikembangkan. Semua ini bertujuan untuk mendukung kolaborasi efisien dalam tim pengembangan aplikasi.

Sebagai seorang Full Stack Developer, alat-alat yang biasa digunakan termasuk editor kode seperti Visual Studio Code, platform manajemen versi seperti GitHub atau GitLab, serta sistem manajemen basis data seperti MySQL atau Oracle. Selain itu, ada alat untuk menguji dan melakukan debugging seperti Jest atau JUnit5, serta alat untuk mengelola API seperti Postman atau Swagger. Dalam pengembangan aplikasi mobile, pilihan umum adalah React Native atau Flutter. Layanan cloud seperti AWS, Google Cloud, dan Azure sering digunakan untuk hosting aplikasi, dan otomatisasi dengan alat seperti Jenkins atau CircleCI mempermudah pengujian dan penyebaran kode. Untuk desain antarmuka pengguna, alat seperti Figma atau Sketch membantu menciptakan UI/UX yang menarik. Keseluruhan alat-alat ini mendukung pengembang Full Stack dalam menghasilkan aplikasi dengan efisien dan profesional.

Video 2: SDLC & Design Thinking Implementation

SDLC, atau Siklus Hidup Pengembangan Perangkat Lunak, adalah serangkaian proses 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 telah ditentukan.

Siklus SDLC dimulai dengan tahap perencanaan dan analisis, di mana masalah atau kebutuhan bisnis yang perlu diselesaikan oleh perangkat lunak diidentifikasi. Para pemangku kepentingan berinteraksi untuk mengumpulkan persyaratan dan menentukan ruang lingkup proyek, sementara rencana keseluruhan proyek disusun dengan alokasi sumber daya, jadwal waktu, dan definisi tugas tim.

Selanjutnya, tahap desain produk melibatkan perancangan rinci perangkat lunak berdasarkan persyaratan yang telah dikumpulkan, termasuk arsitektur sistem, antarmuka pengguna, dan desain database. Tahap pengembangan produk adalah implementasi rancangan perangkat lunak dengan menulis kode untuk menghasilkan produk yang berfungsi.

Setelahnya, tahap pengujian produk dilakukan untuk memastikan bahwa perangkat lunak berfungsi sesuai dengan persyaratan yang telah ditentukan, termasuk pengujian fungsionalitas, kinerja, keamanan, dan kualitas keseluruhan. Penerapan produk melibatkan implementasi rancangan perangkat lunak yang telah disetujui sebelumnya, dan tahap pemeliharaan produk dilakukan setelahnya untuk memperbaiki bug, meningkatkan fitur, dan menjaga perangkat lunak agar tetap sesuai dengan perubahan kebutuhan bisnis.

Penggunaan SDLC memberikan sejumlah manfaat, termasuk prediktabilitas dan pengendalian proyek yang lebih baik, pemenuhan kebutuhan pengguna, peningkatan kualitas perangkat lunak, penghematan biaya dan waktu, pengelolaan risiko yang lebih baik, meningkatkan pengawasan dan evaluasi, efisiensi tim dan kolaborasi yang lebih baik, serta peningkatan dalam dokumentasi proyek.

Terdapat berbagai model SDLC (Siklus Hidup Pengembangan Perangkat Lunak) yang digunakan dalam pengembangan perangkat lunak, masing-masing dengan pendekatan dan karakteristiknya sendiri.

Model Waterfall adalah model linier dan berurutan di mana setiap tahap harus selesai sebelum memulai tahap berikutnya. Ini cocok untuk proyek dengan persyaratan yang jelas dan stabil. Model V-Shaped adalah varian dari Waterfall yang menekankan pengujian, dengan setiap tahap pengembangan memiliki tahapan pengujian yang sesuai.

Model Prototype bertujuan untuk menciptakan prototipe sebelum mengembangkan versi final, fokus pada memahami kebutuhan pengguna. Model Spiral menggabungkan elemen model spiral dengan pendekatan inkremental dan cocok untuk proyek besar dan kompleks. Model Iterative Incremental melibatkan pengulangan siklus pembangunan dan peningkatan dalam tahapan kecil hingga mencapai produk akhir yang diinginkan.

Model Big Bang adalah kurang terstruktur, di mana pengembangan dimulai tanpa perencanaan yang detail, cocok untuk proyek kecil atau prototyping. Terakhir, Model Agile adalah pendekatan kolaboratif dan iteratif yang fokus pada pengiriman perangkat lunak secara berkala dan inkremental, cocok untuk lingkungan yang dinamis dan persyaratan yang berubah-ubah. Setiap model SDLC memiliki penggunaan dan situasi yang sesuai sesuai dengan kebutuhan proyek yang berbeda.

Design Thinking adalah pendekatan kreatif dalam mengatasi masalah dan pengembangan solusi yang berfokus pada pemahaman mendalam terhadap pengguna akhir. Langkah-langkah dalam Design Thinking meliputi: Pertama, tahap Empathize, di mana fokusnya adalah memahami pengguna akhir secara mendalam, termasuk kebutuhan, keinginan, dan pengalaman mereka. Kemudian, tahap Define, di mana informasi yang dikumpulkan selama fase empati dianalisis untuk mengidentifikasi masalah yang akan diselesaikan dan menetapkan tujuan yang jelas untuk proyek. Selanjutnya, tahap Ideate mendorong pemikiran kreatif untuk menghasilkan berbagai solusi potensial untuk masalah yang diidentifikasi. Tahap Prototype berfokus pada menciptakan representasi nyata dari ide-ide yang telah dipilih, yang digunakan untuk mengumpulkan umpan balik dan memvalidasi asumsi-asumsi yang ada. Pada tahap Test, solusi-solusi tersebut diuji dengan pengguna nyata untuk memvalidasi apakah mereka memenuhi kebutuhan dan harapan pengguna. Akhirnya, tahap Implement mengartikan desain yang telah dikembangkan ke dalam kode yang sebenarnya dan mengimplementasikannya. Dengan langkah-langkah ini, Design Thinking membantu dalam menghasilkan solusi yang lebih baik dan lebih sesuai dengan pengguna akhirr.

Video 3: Basic Git & Collaborating Using Git

Sejarah terminal menunjukkan bahwa meskipun teknologi dan perangkat lunak terus berkembang, terminal tetap menjadi alat penting bagi para pengembang perangkat lunak, administrator sistem, dan pengguna teknis lainnya. Meskipun antarmuka grafis semakin canggih dan populer, terminal tetap memberikan fleksibilitas dan kekuatan untuk melakukan tugas-tugas khusus dan otomatisasi dalam lingkungan komputer modern. Selanjutnya, dijelaskan cara melakukan instalasi Git, beberapa dasar dari Git, dan cara berkolaborasi dalam Git. Git adalah

sistem pengontrol versi yang penting dalam pengembangan perangkat lunak yang memungkinkan tim untuk bekerja sama dalam mengelola kode sumber proyek secara efisien.