



Module Lesson - Perencanaan & Dokumentasi

A. Requirement Gathering

B. Prinsip Teknis Aplikasi

C. Spesifikasi Kebutuhan Perangkat Lunak

Requirement Gathering



Requirement Gathering

Pada tahapan awal dalam pengembangan proyek Teknologi Informasi (IT), ada langkah penting yang harus dilakukan sebelum benar-benar memulai merancang dan membangun perangkat lunak. Langkah ini dikenal sebagai "requirement gathering" atau pengumpulan kebutuhan. Pengertian yang kuat tentang apa yang diperlukan oleh pengguna dan pemangku kepentingan sangat penting, karena membentuk dasar untuk menciptakan solusi yang tepat dan efektif.

Requirement gathering adalah fondasi proyek IT. Tanpa pemahaman yang baik tentang apa yang dibutuhkan, risiko kesalahan dan kegagalan proyek jauh lebih besar. Pada tahap ini, para pemangku kepentingan berkolaborasi untuk mengidentifikasi dan menggambarkan kebutuhan yang spesifik. Ini membantu memastikan bahwa perangkat lunak yang dihasilkan akan memenuhi tujuan bisnis, memecahkan masalah yang dihadapi pengguna, dan memberikan nilai yang diharapkan.

Tahapan requirement gathering dalam proyek IT adalah langkah kritis yang membantu memastikan bahwa proyek memiliki arah yang jelas dan tujuan yang tepat. Dengan memahami kebutuhan pengguna dan pemangku kepentingan secara mendalam, tim pengembangan dapat merancang solusi yang sesuai dan menghasilkan perangkat lunak yang memenuhi harapan. Integrasi tahapan ini dengan siklus pengembangan perangkat lunak memastikan bahwa hasil akhir yang dihasilkan akan berfungsi dengan baik dan memberikan nilai bagi semua pihak yang terlibat.

Prinsip Teknis Aplikasi



Prinsip Teknis Aplikasi (Arsitektur perangkat lunak)

Arsitektur perangkat lunak merujuk pada cara komponen-komponen dalam perangkat lunak diorganisir dan berinteraksi satu sama lain. Penerapan arsitektur yang tepat sangat penting dalam proyek IT karena memastikan bahwa perangkat lunak memiliki struktur yang teratur, skalabilitas yang baik, dan mudah dikelola. Dalam penerapan arsitektur, beberapa konsep penting yang perlu dipertimbangkan meliputi:

- Komponen: Bagaimana komponen-komponen berhubungan satu sama lain dan berinteraksi dalam sistem.
- Pemisahan Kepentingan: Memisahkan bagian-bagian yang berbeda berdasarkan tanggung jawabnya.
- Pengaturan Aliran Data: Bagaimana data mengalir melalui komponen-komponen.

Dalam proyek IT, pemilihan arsitektur yang sesuai, seperti arsitektur berorientasi layanan (SOA) atau arsitektur mikro layanan, dapat memberikan manfaat signifikan dalam pengembangan dan pengelolaan perangkat lunak.

Dalam proyek IT, pemilihan arsitektur yang sesuai, seperti arsitektur berorientasi layanan (SOA) atau arsitektur mikro layanan, dapat memberikan manfaat signifikan dalam pengembangan dan pengelolaan perangkat lunak.

Prinsip Teknis Aplikasi (SOLID Framework)



Prinsip SOLID adalah seperangkat pedoman desain yang membantu membangun kode yang lebih terstruktur, fleksibel, dan mudah dipelihara. Prinsip ini membantu pengembang dalam memutuskan bagaimana merancang kelas-kelas dan komponen-komponen dalam perangkat lunak:

- *Single Responsibility Principle (Prinsip Tanggung Jawab Tunggal)*: Setiap kelas seharusnya hanya memiliki satu tanggung jawab utama. Ini membuat kode lebih fokus dan mudah dipahami.
- *Open/Closed Principle (Prinsip Terbuka/Tertutup)*: Kode harus terbuka untuk perluasan tetapi tertutup untuk modifikasi. Ini mendorong penggunaan warisan (inheritance) atau polimorfisme untuk mengubah perilaku.
- *Liskov Substitution Principle (Prinsip Substitusi Liskov)*: Objek dari kelas turunan harus dapat menggantikan objek dari kelas induk tanpa merusak kebenaran program.
- *Interface Segregation Principle (Prinsip Pemisahan Antarmuka)*: Klien tidak boleh terpaksa mengimplementasikan metode yang tidak relevan bagi mereka. Ini mendorong pembuatan antarmuka yang lebih spesifik.
- *Dependency Inversion Principle (Prinsip Inversi Ketergantungan)*: Kode tingkat tinggi tidak boleh bergantung pada kode tingkat rendah. Prinsip ini mendorong penggunaan injeksi ketergantungan (dependency injection) untuk menghindari ketergantungan langsung.

Manfaat dari penerapan prinsip SOLID termasuk kode yang lebih mudah dikelola, lebih fleksibel untuk perubahan, dan lebih cocok untuk kolaborasi tim.

Spesifikasi Kebutuhan Perangkat Lunak



Spesifikasi Kebutuhan Perangkat Lunak

Komponen-komponen dalam dokumen spesifikasi kebutuhan berkontribusi pada pemahaman yang jelas tentang apa yang akan dibangun dalam proyek IT. Mengidentifikasi aktor dan use case membantu merinci interaksi pengguna dengan sistem. Penggunaan diagram alur use case memvisualisasikan interaksi ini secara lebih terstruktur. Penjelasan persyaratan fungsional dan non-fungsional dengan jelas memberikan panduan yang jelas tentang fitur yang akan diimplementasikan dan standar teknis yang harus dipenuhi. Semua langkah ini mengarah pada pengembangan perangkat lunak yang sesuai dengan ekspektasi dan kebutuhan.