Jawaban Tugas Rekayasa Perangkat Lunak

# Bagian A: Pemahaman Dasar

## 1. Apa yang dimaksud dengan Design Model dalam rekayasa perangkat lunak?

Design model adalah representasi teknis dari sistem yang bakal dibangun, yang menjembatani antara kebutuhan pengguna (requirement) dan implementasi kode (coding). Ini kayak cetak birunya software, gengs. Jadi, sebelum kamu ngoding-ngoding santuy, kamu kudu tau dulu mau bikin apa dan kayak gimana tampilannya, strukturnya, alurnya, dan fungsinya.

## 2. Sebutkan dan jelaskan empat komponen utama design model!

🧩 Data/Class Design:  
Desain struktur data dan class yang dibutuhkan sistem. Nentuin gimana data disimpan, diakses, dan direlasikan.  
🧩 Architectural Design:  
Struktur sistem secara keseluruhan. Komponen-komponen utama dan hubungan antarkomponen. Kayak denah rumahnya software.  
🧩 Interface Design:  
Desain antarmuka, baik antarmuka ke user (UI) maupun antar komponen (API). Biar sistemnya user-friendly, gak kayak mantan yang ribet!  
🧩 Component-Level Design:  
Detail isi masing-masing komponen (biasanya dalam bentuk class atau fungsi). Ini udah masuk ke detil logic tiap bagian.

## 3. Mengapa design model sangat bergantung pada hasil dari analysis model?

Karena design model lahir dari hasil analisis. Coba bayangin bangun rumah tanpa tahu kebutuhan si pemilik. Bakal ngawur, kan? Sama juga, tanpa analisis yang jelas, desainnya bisa melenceng jauh dari yang dibutuhin user. Makanya, model analisis itu kayak fondasi yang ngasih tau apa yang dibutuhkan, lalu design model menjawab bagaimana cara memenuhinya.

# Bagian B: Hubungan antara Analysis Model dan Design Model

## 1. Jelaskan bagaimana scenario-based model berkontribusi pada design level!

Scenario-based model (misalnya use case diagram) bantu ngejelasin alur interaksi user dengan sistem. Dari sinilah, kamu bisa:  
- Desain UI yang sesuai kebutuhan.  
- Bikin flow komponen, event handler, dan respon sistem.  
Misalnya: dari use case 'Pesan Tiket', bisa diturunin ke UI tombol 'Pesan Sekarang' dan metode pemrosesan transaksi di komponen backend.

## 2. Apa yang dihasilkan dari class-based model saat diturunkan ke tahap desain?

Class-based model saat diturunin ke desain akan menghasilkan:  
- Blueprint class lengkap dengan atribut dan operasi (method).  
- Relasi antar class: inheritance, aggregation, composition.  
Jadi, ini bantu ngelayout struktur program dalam bentuk object-oriented structure (OOP style).

## 3. Bandingkan penggunaan activity diagram dan data flow diagram (DFD) dalam mendukung desain sistem.

| Aspek | \*\*Activity Diagram (UML)\*\* | \*\*Data Flow Diagram (DFD)\*\* |  
|-----------------------------|-----------------------------------------------|---------------------------------------|  
| Pendekatan | Berorientasi pada proses dan alur aktivitas | Fokus ke aliran data |  
| Fokus | Menjelaskan alur kerja sistem atau user | Menjelaskan bagaimana data diproses dan berpindah |  
| Simbol | Action, decision, swimlane | Proses, data store, data flow, terminator |  
| Kelebihan | Lebih fleksibel & cocok untuk desain UI | Cocok buat sistem berbasis data (misal: bank, inventory) |

# Bagian C: Refleksi Penerapan

## 1. Bayangan Struktur Desain Sistem

Berdasarkan sistem yang dianalisis dalam SKPL tersebut, desain struktur sistem akan dibagi ke dalam beberapa bagian penting:  
  
- \*\*UI (User Interface):\*\*  
 - \*\*Pelanggan\*\*: Antarmuka pengguna yang sederhana dan intuitif, dapat diakses melalui aplikasi mobile. Fitur utama meliputi halaman login, daftar menu, halaman pemesanan, konfirmasi pesanan, riwayat pesanan, dan notifikasi status. Pengguna akan dapat memilih menu dan memverifikasi pesanan dengan mudah.  
 - \*\*Admin\*\*: Aplikasi berbasis web, memungkinkan staf kantin untuk mengelola menu, memperbarui stok, memantau status pesanan, dan melihat laporan transaksi. Dashboard akan menampilkan data yang jelas tentang transaksi dan status pesanan secara real-time.  
  
- \*\*Data Class:\*\*  
 - \*\*Pelanggan\*\*: Atributnya meliputi ID, nama, nomor telepon, dan email. Setiap pelanggan dapat melakukan lebih dari satu pesanan, yang akan direkam dalam sistem.  
 - \*\*Menu\*\*: Atribut yang mencakup ID menu, nama menu, harga, dan status ketersediaan. Ini adalah data yang terus diperbarui dan dikelola oleh admin.  
 - \*\*Pesanan\*\*: Atributnya meliputi ID pesanan, tanggal pemesanan, status, dan total pembayaran, yang akan disinkronkan dengan menu yang dipesan oleh pelanggan.  
 - \*\*Notifikasi\*\*: Atribut mencakup ID notifikasi, status, dan waktu notifikasi yang digunakan untuk memberi tahu pelanggan tentang status pesanan.  
  
- \*\*Arsitektur Sistem:\*\*  
 - \*\*Client-Server Architecture:\*\* Sistem ini akan menggunakan pendekatan client-server, di mana aplikasi pengguna akan berkomunikasi dengan server melalui RESTful API. Backend menggunakan framework seperti Laravel (PHP) atau ExpressJS (Node.js).  
 - \*\*Database:\*\* Basis data relasional (MySQL atau PostgreSQL) digunakan untuk menyimpan data pesanan, menu, pelanggan, dan transaksi. Semua transaksi dikendalikan dan diproses melalui server.  
 - \*\*Notifikasi Real-Time:\*\* Menggunakan Firebase Cloud Messaging (FCM) untuk memastikan notifikasi status pesanan dikirimkan secara instan kepada pelanggan.

## 2. Tantangan Menerjemahkan Analisis ke Desain

Beberapa tantangan yang mungkin muncul saat menerjemahkan model analisis ke desain, antara lain:  
- \*\*Keterbatasan dalam implementasi fitur:\*\* Misalnya, sistem ini tidak mencakup pembayaran elektronik atau pengelolaan stok bahan baku di tahap awal, yang bisa menjadi hambatan saat mengembangkan fitur lebih lanjut.  
- \*\*Menjaga konsistensi data:\*\* Mengelola stok dan status pesanan harus dipastikan sinkron antara database, server, dan aplikasi client (pelanggan dan admin). Salah kelola bisa berakibat pesanan tidak dapat diproses dengan benar.  
- \*\*Skalabilitas:\*\* Sistem ini dirancang untuk lingkungan kampus, jadi peningkatan pengguna bisa menjadi tantangan jika diimplementasikan di area yang lebih luas, seperti sekolah dengan banyak cabang.

## 3. Kelebihan Pendekatan Berorientasi Objek vs Proses

| Aspek | \*\*Berorientasi Objek (OOP)\*\* | \*\*Berbasis Proses (DFD)\*\* |  
|-----------------------------|-----------------------------------------------|---------------------------------------|  
| Modularitas | Tinggi (class reusable) | Rendah |  
| Maintainability | Mudah dikelola & dikembangkan | Sulit jika sistem besar |  
| Representasi dunia nyata | Sangat natural (objek = entitas nyata) | Abstrak, lebih fokus ke data |  
| Cocok untuk | Aplikasi kompleks (web, mobile) | Sistem batch processing, legacy system |