Nama : Restu Wibisono
NPM : 2340506061

1. Peran manajement dalam pengembangan perangkat lunak

   Manajement proyek perangkat lunak memiliki peran yang krusial dalam berbagai hal seperti olanning, resource allocation, risk management, dan pengawasan kualitas. Keberhasilan proyek sangat ditentukan oleh seberapa baik manajer proyek dalam mengelola tim, menyatukan tim, dan menjaga komunikasi.

   Contoh: Kasus "FBI Virtual Case File" (VCF) gagal karena manajemen proyek tidak mampu mengidentifikasi risiko teknis dan memperbaiki scope creep. Sebaliknya, Spotify berhasil membangun ekosistem perangkat lunak yang skalabel melalui manajemen agile squads yang mendukung inovasi dan iterasi cepat.

2. Menentukan model siklus hidup perangkat lunak (SDLC) yang tepat

   Faktor yang perlu dipertimbangkan:

   - Kebutuhan pengguna: Apakah sudah jelas atau masih berubah-ubah?
   - Tingkat risiko dan kompleksitas proyek
   - Ukuran tim dan struktur organisasi
   - Kebutuhan dokumentasi dan regulasi
   - Ketersediaan waktu dan anggaran

Waterfall	Proyek dengan kebutuhan tetap dan regulasi ketat (misalnya sistem avionik)
Agile/Scrum	Proyek dengan perubahan cepat dan feedback rutin dari pelanggan
Spiral	Proyek besar dengan risiko tinggi yang butuh iterasi dan evaluasi
V-Model	Lingkungan dengan pengujian dan verifikasi kuat seperti perangkat medis

3. Mengatasi tantanan dalam menghadapi scrum di tim tradisional

   Tantanga :

   - Perubahan budaya kerja
   - Kurangnya pengalaman dengan peran seperti Product Owner atau Scrum Master
   - Ketidakpastian dengan iterasi pendek dan perubahan backlog

   Solusi :

   - Pelatihan dan mentoring

- Mulai dengan pilot project:
- Gunakan pendekatan hybrid sementara
- Komunikasi intensif

4. Kapan kanban lebih efektif daripada scrum

   Kanban Lebih Efektif Saat:

   - Tidak ada sprint yang jelas; kebutuhan datang secara terus-menerus (misal: tim maintenance)
   - Fokus pada visualisasi dan pengelolaan work in progress
   - Tim bersifat service-based seperti helpdesk atau DevOps

   Metodologi Hybrid: Scrumban

   - Menggabungkan struktur Scrum (daily stand-up, retrospective) dengan alur kerja fleksibel Kanban (board, limitasi WIP)
   - Cocok untuk tim yang ingin lebih fleksibel tetapi tetap ingin menjaga ritme sprint

5. Memastikan kebutuhan pengguna yang valid dengan pendekatan berbasis skenario
   1. Interview dan observasi pengguna
   2. Buat user persona dan skenario (use case)
   3. Lakukan prototyping cepat dan uji coba
   4. Validasi dengan user story mapping dan acceptance criteria

   Contoh: Untuk aplikasi e-commerce, skenario:

   "Sebagai pelanggan, saya ingin dapat memfilter produk berdasarkan harga agar saya dapat menemukan barang sesuai budget."

   Dari skenario ini akan lahir user story, acceptance test, dan dapat divalidasi langsung melalui prototype antarmuka filter produk.

6. Dampak prinsip desian perangkat lunak terhadap efisiensi

   Prinsip SOLID (contoh):

   - Single Responsibility Principle: Setiap class hanya menangani satu tanggung jawab → memudahkan pemeliharaan.
   - Open/Closed Principle: Mudah ditambahkan fitur tanpa mengubah kode lama → mendukung scalability.

Contoh: Pada proyek pengembangan sistem reservasi hotel, penerapan prinsip "Dependency Inversion" memungkinkan tim mengganti metode pembayaran dari PayPal ke Stripe tanpa mengubah logika utama aplikasi

7. Peran continous integration (CI) dan continus testing (CT)

   Manfaat CI:

   - Kode dari berbagai developer digabung secara berkala → menghindari konflik besar
   - Deteksi error lebih dini melalui build otomatis

   Manfaat CT:

   - Setiap perubahan kode diuji secara otomatis
   - Menghindari regresi (fitur lama rusak akibat fitur baru)

   Contoh: Pada pengembangan produk SaaS seperti GitLab, pipeline CI/CD mereka melakukan:

   - Otomatisasi testing unit, integration, dan performance
   - Deployment otomatis ke staging → mempercepat time-to-market