Nome del progetto: AI System Analyzer

Obiettivo principale: Creare un'intelligenza artificiale locale in grado di analizzare un computer e dare suggerimenti intelligenti per ottimizzazione, sicurezza e prestazioni.

Schema di procedimento dettagliato:

1. Preparazione dell'ambiente
2. Installare Java JDK 17 o superiore.
3. Scegliere un IDE (Eclipse, IntelliJ IDEA o VS Code con plugin Java).
4. Configurare Git per versionamento e creare repository per il progetto.
5. Installare ClamAV e Nmap sui sistemi di test.
6. Verificare la configurazione della GPU per eventuali calcoli ML.
7. Fase 1 - Raccolta dati di sistema (hardware/software)
8. Integrare la libreria OSHI per leggere CPU, RAM, GPU, disco, rete e processi.
9. Creare classi Java dedicate alla raccolta di ciascun tipo di dato:
    - CPUInfo.java: utilizzo, frequenza, core
    - RAMInfo.java: memoria totale, libera, utilizzata
    - GPUInfo.java: modello, memoria, utilizzo
    - DiskInfo.java: spazio totale, libero, SMART status
    - NetworkInfo.java: interfacce, indirizzi IP, velocità
    - ProcessInfo.java: processi attivi, utilizzo CPU/RAM
10. Implementare metodi per esportare i dati in formato JSON o oggetti leggibili dall'IA.
11. Testare ogni modulo separatamente e assicurarsi che i dati siano accurati.
12. Fase 2 - Interfaccia grafica base
13. Creare una GUI minimale con JavaFX.
14. Visualizzare dati raccolti in tabelle o grafici semplici.
15. Implementare una barra di navigazione per passare tra hardware, software e rete.
16. Testare la GUI su diverse risoluzioni e sistemi operativi.
17. Fase 3 - Modulo antivirus
18. Integrare ClamAV tramite ProcessBuilder o libreria pyclamd compatibile Java.
19. Creare classe AntivirusScanner.java per gestire scansioni:
    - Scansione completa o selettiva di cartelle
    - Lettura output e identificazione file/processi sospetti
20. Trasformare i risultati in input leggibile per l'IA.
21. Testare su file e cartelle di prova con malware simulato.
22. Fase 4 - Diagnostica rete
23. Integrare Nmap tramite ProcessBuilder.

24. Creare classe NetworkScanner.java per:
    - Scansione IP locali e subnet
    - Identificazione dispositivi sconosciuti
    - Verifica porte aperte e vulnerabilità base
25. Trasformare i risultati in report leggibile dall'IA.

26. Eseguire test su reti diverse e gestire errori di permessi.

27. Fase 5 - Motore dell'IA (motore di regole)

28. Creare classe AIAnalyzer.java.
29. Definire regole complesse basate sui dati raccolti:
    - Esempio: RAM > 90% per 5 minuti -> suggerire chiusura processi pesanti
    - CPU > 95% continuo -> suggerire controllo task
    - File sospetti da antivirus -> suggerire quarantena
    - Dispositivi sconosciuti in rete -> alert sicurezza
30. Implementare sistema di pesi per priorità dei consigli.

31. Testare con scenari simulati e dati fittizi.

32. Fase 6 - Integrazione moduli

33. Collegare raccolta dati, antivirus, diagnostica rete e motore AI.
34. Creare flusso centralizzato: dati raccolti -> IA -> GUI.
35. Assicurarsi che tutti i moduli comunichino correttamente e gestiscano errori.

36. Testare end-to-end su macchine di prova.

37. Fase 7 - Ottimizzazione e testing

38. Effettuare test su più macchine con configurazioni diverse.
39. Verificare che l'IA dia consigli coerenti e utili.
40. Ottimizzare codice per performance e ridurre consumo risorse.

41. Aggiornare GUI per visualizzare suggerimenti chiaramente.

42. Fase 8 - (Opzionale) Machine Learning / LLM

43. Integrare Deep Java Library (DJL) o modelli ONNX compatibili GPU.
44. Addestrare modello leggero locale per interpretare dati e fornire consigli in linguaggio naturale.

45. Testare prestazioni GPU e compatibilità multipiattaforma.

46. Fase 9 - Versione portable

    - Creare eseguibile standalone con JavaFX incluso.
    - Testare portabilità su diverse macchine, assicurandosi che ClamAV/Nmap siano inclusi o facilmente configurabili.

47. Documentazione completa

    - Creare README dettagliato con istruzioni di setup e utilizzo.

- Documentare tutte le classi, metodi e regole dell'IA.
- Preparare screenshot, demo e esempi di output.
- Redigere note sulle limitazioni e possibili sviluppi futuri.