Übung 5: Software- und Architekturmetriken für Codequalität und Architekturoptimierung

Ziel: In dieser Übung analysieren Sie die Qualität des Codes, den Sie im Übungsblatt 4 erstellt haben, mithilfe von Metriken und gezielten LLM-Einsätzen zur Unterstützung. Sie setzen auf den Erkenntnissen des letzten Übungsblatts auf, um gezielte Verbesserungen durchzuführen und erste Frontend-Komponenten zu entwickeln. Der Schwerpunkt liegt auf der Anwendung von Metriken und Tools sowie dem Nutzen von LLM für den Entwicklungsprozess.
Relevante Lerneinheiten: MET + CCD

1. **Überblick und Anwendung einfacher Metriken:** Analysieren Sie die grundlegenden Metriken Ihres Codes, wie Methodenanzahl und Parameteranzahl pro Methode, um erste Schwachstellen aufzudecken. Nutzen Sie ein Metrik-Tool wie SonarQube oder ein IDE-Plugin (z. B. IntelliJ Metrics), um die wichtigsten Werte zu erfassen. Ziel: Erste Erkenntnisse über komplexe Codebereiche gewinnen.
   LLM-Einsatz: Lassen Sie sich mit einem LLM-KI-Tool mögliche Optimierungen oder Vereinfachungen vorschlagen, die durch die Metriken aufgefallen sind (z. B. Methoden zu modularisieren).

2. **Test Coverage erweitern und Code Coverage verbessern:** Erhöhen Sie die Testabdeckung gezielt für kritische Bereiche Ihrer Anwendung und erweitern Sie bestehende Tests, um eine höhere Testabdeckung (z. B. 80 %) zu erreichen. Verwenden Sie dazu ein Coverage-Tool wie JaCoCo, um Abdeckungswerte zu identifizieren und entwickeln Sie spezifische Tests für komplexe Bereiche. Ziel: Erhöhte Testabdeckung und Sicherstellung der Qualität zentraler Logikkomponenten.
   LLM-Einsatz: Fragen Sie das LLM nach Testideen oder Edge-Case-Vorschlägen für besonders komplexe Methoden.

3. **Technical Debt und Regelverletzungen mit LLM analysieren:** Analysieren Sie mithilfe eines Tools wie SonarQube die technische Schuld und Regelverletzungen in Ihrem Code. Ermitteln Sie dazu technische Schulden in den Bereichen Code-Duplizierung oder veraltete Abhängigkeiten und kategorisieren Sie diese nach Schweregrad. Besprechen Sie mit einem LLM die Ergebnisse und lassen Sie sich Empfehlungen zur Schuldenreduktion geben. Ziel: Reduktion technischer Schulden und Verbesserung der Codequalität.
   Output: Dokumentation der aktuellen technischen Schuld und Maßnahmen zur Reduzierung, basierend auf LLM-Vorschlägen.

4. **Frontend-Entwicklung und Erweiterung der Anwendung:** Entwickeln Sie eine einfache Benutzeroberfläche für Ihr System, beispielsweise zur Erfassung und Verwaltung von Patientendaten und Terminen. Erstellen Sie eine kleine Benutzeroberfläche mit JavaFX (Desktop) oder JavaScript/HTML (Web) und binden Sie die wichtigsten Backend-Funktionen ein. Testen Sie UI-Komponenten und binden Sie das LLM ein, um UI-Design-Ideen oder Code-Verbesserungen zu erhalten. Ziel: Einfache, funktionale Benutzeroberfläche, die zentralen Backend-Funktionalitäten zugänglich macht.

5. Reflexion zum Einsatz von Metriken und LLM: Reflektieren Sie über den Einfluss von Metriken und den Einsatz des LLM auf Ihre Codequalität und Frontend-Entwicklung. Schreiben Sie eine kurze Reflexion über die gewonnenen Erkenntnisse und den Mehrwert des LLM-Einsatzes für Ihren Entwicklungsprozess. Ziel: Vertieftes Verständnis für Metriken und KI-gestützte Entwicklung. Output: Kurze Reflexion (200–300 Wörter) über Ihre Erfahrungen mit Metriken und dem LLM.

Hinweise:
- Verwenden Sie regelmäßig das LLM, um Fragen zu Verbesserungen und alternativen Lösungsansätzen zu stellen.
- Halten Sie Ihre Reflektion zum LLM-Einsatz fest und dokumentieren Sie, bei welchen Aufgaben das LLM die Entwicklung am meisten unterstützt hat.
- Arbeiten Sie weiterhin mit einem In-Memory-Repository, um Daten vorübergehend zu speichern. Eine persistente Datenbankintegration, Build-Management sowie das Deployment werden in den kommenden Wochen behandelt.

Abgabe (optional): Git-Repository bis zum **17.11.25, 23:55 Uhr** bereitstellen und via Moodle Kommentar einreichen.