Protokoll zur Entwicklung des Projekts MTCG

1. App-Design (Entscheidungen, Struktur) Das Projekt basiert auf einer klaren, modularen Architektur, die nach dem MVC-Pattern (Model-View-Controller) strukturiert ist. Folgende Module sind identifiziert:

- **Domain-Layer**: Beinhaltet die Kernlogik und Modelle, z. B. Card, User, Deck, und deren zugehörige Services.
- **Persistence-Layer**: Datenpersistenz wird durch das DatabaseManager Modul und spezifische Repositories wie UserRepository und CardRepository gewährleistet. (Enthält alle Datenbank relevanten Methoden)
- **Application-Layer**: Zuständig für die Verarbeitung von Requests und deren Weiterleitung an Services. Handlers wie UserActionHandler und GameHandler sorgen für eine saubere Trennung von Verantwortlichkeiten.
- **Presentation-Layer**: Enthält HTTP-spezifische Komponenten wie HTTPRequest, HTTPResponse und HTTPRouter. Diese Komponenten sind die Schnittstelle zum Benutzer oder zu API-Clients über einen HTTP-Server.

---

2. Erkenntnisse aus der Entwicklung Während der Entwicklung wurden mehrere wichtige Lektionen gelernt:

- **Frühe Einbindung von Tests**: Es hat sich als vorteilhaft erwiesen, die Kernfunktionen von Anfang an mit Tests zu begleiten. Dadurch konnten grundlegende Fehler frühzeitig entdeckt und behoben werden.
- **Bedeutung klarer Modulgrenzen**: Die Einhaltung des Single Responsibility Prinzips hat geholfen, die Lesbarkeit und Wartbarkeit des Codes zu verbessern. Beispielsweise wurden Services und Repositories klar voneinander getrennt, was die Fehlersuche erleichtert hat.
- **Fehlerbehandlungsstrategie**: Die zentrale Fehlerbehandlung durch die FailureResponse Klasse hat sich als essentiell erwiesen, um konsistente und nutzerfreundliche Fehlermeldungen zu garantieren.
- **Iterative Verbesserungen**: Regelmäßige Code-Reviews und Refactoring-Schritte haben die Codequalität und -konsistenz merklich gesteigert.

---

3. Unit-Testing-Entscheidungen

- **Breite Testabdeckung**: Alle zentralen Services wie CardService, DeckService, PackageService und UserService wurden mit Unit-Tests versehen. Dies stellt sicher, dass Kernfunktionen erwartungsgemäß arbeiten. Außerdem wurde das GameRepository getestet, da es hier mehrere Ausgangslagen gibt (Win_Player1,Win_Player2 oder DRAW)
- **Testmethodik**: Die Tests wurden in JUnit geschrieben, da dieses Framework robust und leicht in Java-Projekte integrierbar ist.

- **Mocking**: Um externe Abhängigkeiten wie Datenbanken zu simulieren, wurden Mocking-Frameworks verwendet. Dies ermöglichte isolierte Tests, die reproduzierbare Ergebnisse liefern.
- **Edge Cases und Fehlerbedingungen**: Besondere Aufmerksamkeit wurde darauf gelegt, auch Grenzfälle und Fehlersituationen zu testen, um die Stabilität der Anwendung zu erhöhen.
- **Testzyklen**: Tests wurden in jeder Entwicklungsphase regelmäßig durchgeführt, insbesondere nach größeren Codeänderungen, um Regressionen zu vermeiden.

---

4. Einzigartiges Feature

Ein Feature des Projekts ist die Implementierung einer **Logout-Funktion**. Diese Funktion schließt aktive Sitzungen sicher ab, indem sie den zugehörigen Token ungültig macht und so die Sicherheit der Benutzerdaten gewährleistet. Der UserSessionHandler spielt hierbei eine zentrale Rolle.

---

5. Zeitaufwand Die Entwicklung des Projekts wurde über mehrere Wochen verteilt. Der geschätzte Aufwand umfasst:

- **Analyse und Design**: 15 Stunden
- **Implementierung**: 40 Stunden
- **Testing und Debugging**: 10 Stunden
- **Dokumentation**: 1 Stunde

Gesamter Zeitaufwand: **66 Stunden**

---

6. Link zum Git-Repository Das Projekt wurde mit Git versioniert. Der Repository-Link lautet:

https://github.com/JakobPronebner04/MCTG

Zusammenfassung

Das Projekt MTCG ist eine modulare und gut strukturierte Anwendung, die auf bewährten Softwareentwicklungsmethoden basiert. Es bietet eine klare Trennung der Logik und Testabdeckung. Grundsätzlich lässt sich sagen, dass dieses Projekt auf jeden Fall ein guter Einstieg in die Gameentwicklung / API ist (Backendpart). Vor allem die richtige Strukturierung eines Projekts (Layers) als auch die richtige Trennung wurde einem gut nähergebracht. Außerdem wurden einem die Grundlagen einer Serverprogrammierung als auch die Integration von Unit-Tests gut nähergebracht.