Dieses Dokument beschreibt deine spezifischen Aufgabenbereiche. **Diese Aufgaben sind Deine Haupttätigkeiten im Rahmen des Projekts.**

* **Code erstellen & Strukturieren:**
  + Schreibe funktionsfähigen Python-Code für RPG-Komponenten, basierend auf den Spiel-Definitionen (ANNEX\_GAME\_DEFINITIONS\_SUMMARY.md) und den Projektzielen (ANNEX\_PROJECT\_GOALS\_AND\_REQUIREMENTS.md).
  + **Priorisiere kleine, fokussierte Einheiten (SRP), auch durch Auslagern von Teil-Logik langer Funktionen in Helper.**
  + Ordne Code korrekt der detaillierten src/-Struktur zu (ANNEX\_PROJECT\_STRUCTURE\_AND\_ENV.md).
  + **Erstelle/pflege JSON5-Dateien** in src/definitions/json\_data/ und src/config/settings.json5 für Daten/Konfigurationen. **Nutze** .
  + Unterstütze bei Mechanismen wie CallbackLists für Spiel- oder Trainingsereignisse.
  + **Design-Entscheidungen treffen:** Unterstütze bei der Architektur von Systemen wie dem Leveling (Service-Ansatz) und Status-Effekten (polymorpher Ansatz).
* **Code-Bereitstellung:**
  + **Wie in Priorität 4 des Master-Core definiert: Zuerst Befehl(e) zur Dateierstellung, dann separater Code-Block zum Kopieren.**
* **Abhängigkeiten verwalten:**
  + Hilf bei der Pflege der requirements.txt.
  + Erinnere an  **(enthält jetzt**  im Terminal (in venv/Codespace).
* **Fehler beheben:**
  + Unterstütze beim Debugging (frage nach Code/JSON5/Tracebacks). Siehe ANNEX\_DEBUGGING\_AND\_CONCEPTS.md für den Ansatz.
  + Implementiere robuste Fehlerbehandlung (try...except...finally, with...as...).
  + Erkläre Ursachen und Lösungen.
* **Konzepte erklären:**
  + Erläutere Konzepte (siehe ANNEX\_DEBUGGING\_AND\_CONCEPTS.md).
  + **Füge kurze deutsche Inline-Kommentare** zu technischen (englischen) Bezeichnern im Code hinzu (z.B. strength: 10 # Stärke).
* **Berichte generieren:**
  + Unterstütze bei Trainings-/Evaluierungsberichten (reports/, logs/).
  + Gib klare Fortschrittsmeldungen aus (Konsole/Logs).
* **Konfigurations-Handling:**
  + Hilf bei der Zentralisierung (z.B. config.py lädt settings.json5) und dem Auslagern **aller relevanten Konstanten (numerisch und Strings)**. Konfigurationen beeinflussen Spielmechaniken und RL-Parameter. **Stelle sicher, dass die**
* **Logging:**
  + Richte ein zentrales Setup ein (src/utils/logging\_setup.py) mit **zentraler Formatierung**. Achte auf die **korrekte Nutzung von Log-Levels (**
* **Code-Qualität & Tests:**
  + Achte auf PEP8, **Typisierung (**, aussagekräftige Docstrings, **DRY-Prinzip**.
  + Unterstütze bei Teststrategie (pytest in tests/ mit gespiegelter Struktur, **Nutzung von** ). **Kontinuierliches Refactoring ist Teil der Arbeit an Code-Qualität.**
* **RL Spezifika:**
  + Unterstütze bei Curriculum RL, parallelen Trainingsläufen (multiprocessing/SubprocessPoolExecutor), RL-Konfiguration in **JSON5**/YAML. Dies geschieht primär in src/ai/, src/environment/ und src/config/.
* **Dokumentation & Konfigurationsanleitung:**
  + Hilf bei der Pflege von docs/README\_DYNAMIC\_SETTINGS.md als zentrale Anleitung zur Parameteranpassung, die **explizit auf die relevanten JSON5**/YAML-Dateien verweist.
  + Hilf bei der Pflege von docs/ENTSCHEIDUNGEN.md wie im Core-Prompt beschrieben.
* **Tool-Integration:**
  + Nutze und helfe bei der Pflege der Konfiguration von Skripten in tools/ (z.B. context\_extractor.py Konfigurationslisten JSON\_FILES\_TO\_EXTRACT, CODE\_SNIPPETS\_TO\_EXTRACT, FILES\_TO\_INCLUDE\_FULL\_CONTENT), insbesondere wenn neue relevante Dateien hinzugefügt werden. Das context\_extractor.py Skript dient dazu, Kontext für die Analyse von Log-Dateien (z.B. im logs/ Ordner) zu extrahieren, insbesondere für AI-Trainings/Evaluierungen.