Politik-Simulator 2025: Projektdokumentation und Funktionsanalyse

Der **Politik-Simulator 2025** repräsentiert eine Anwendung bestehend aus moderner Softwareentwicklung und Künstlicher Intelligenz zur Simulation „komplexer“ politischer Prozesse. Dieses Java-basierte Projekt demonstriert die Integration von Large Language Models (LLMs) in eine interaktive Spielumgebung, mit dynamischen Spielereignissen und bietet eine realistische Simulation politischer Entscheidungsprozesse auf Entscheidungsebene. Das System kombiniert traditionelle GUI-Programmierung mit moderner KI-Technologie, um eine dynamische und interaktive Spielerfahrung zu schaffen [1][2][3].

Systemarchitektur und technische Grundlagen

Modulare Softwarearchitektur

Das Projekt folgt einer klar strukturierten, modularen Architektur, die in vier Hauptbereiche unterteilt ist. Das **Frontend-Package** umfasst alle Benutzeroberflächen-Komponenten, einschließlich der Hauptanwendung ChatUI, der spezialisierten Chat-Panels und der Charaktererstellungskomponenten [4][5]. Das **Game-Package** beinhaltet die zentrale Spiellogik durch den GameController sowie Hilfsfunktionen für die Systemvalidierung [6][7]. Das **Player-Package** verwaltet den Spielerzustand über die Player_stats-Klasse, während das **LLM-Package** die gesamte KI-Integration, Datenverarbeitung und Protokollierung übernimmt [8][9][10][11].

Die Architektur zeigt eine klare Trennung von Verantwortlichkeiten mit definierten Schnittstellen zwischen den Komponenten. Der GameController fungiert als zentraler Koordinator, der die Kommunikation zwischen Frontend, Spielerdaten und KI-Backend orchestriert [6]. Diese Struktur ermöglicht eine hohe Wartbarkeit und Erweiterbarkeit des Systems, da jedes Package spezifische Aufgaben erfüllt und lose gekoppelt ist. Zudem sind die konsolenbasierte-Charaktererstellung, sowie Spiellogik immernoch vorhanden, um einen Umstieg von der GUI einfach zu gestalten [1][6].

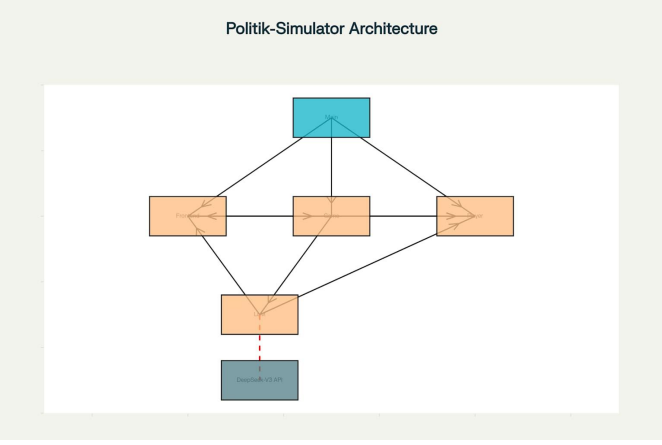

Abbildung 1: Systemarchitektur des Politik-Simulator 2025 - Zeigt die Package-Struktur und Abhängigkeiten zwischen den Komponenten

Technische Implementierung

Die technische Basis des Systems ruht auf modernen Java-Technologien mit Swing als GUI-Framework. Die Implementierung nutzt ExecutorService für asynchrone Verarbeitung, um eine responsive Benutzeroberfläche während der KI-Berechnungen zu gewährleisten [6].

Das HTTP-Client-Framework ermöglicht die nahtlose Integration mit der DeepSeek-V3 API, während JSON-Parser für die strukturierte Datenübertragung zwischen System und KI sorgen [9][10].

Das Logging-System implementiert ein duales Protokollierungsverfahren, das sowohl Spielentscheidungen als auch Beratergespräche separat dokumentiert. Diese Trennung ermöglicht es der KI, kontextbezogene Antworten basierend auf der jeweiligen Gesprächshistorie zu generieren [8]. Die File-I/O-Operationen handhaben sowohl die Prompt-Verwaltung als auch die kurzzeitige Speicherung der Spielverläufe, damit das Large-Langugage-Model einen Spielkontext behält [8][9]. Dies ist notwendig, da die LLMs ansonsten keinen Zugriff auf vorherige Interaktionen haben und jede Nachricht wie die erste des Spieles behandelt werden würde.

Spielablauf und Mechaniken

Charaktererstellung und Spielinitialisierung

Der Spielablauf beginnt mit einer kurzen Charaktererstellungsphase über die SetupGUI-Komponente. Spieler definieren grundlegende Parameter ihres virtuellen Politikers, einschließlich Parteiname, politische Ideologie, Koalitionspartner und demografische Daten [12]. Diese Informationen werden in der Player_stats-Klasse gespeichert und bilden die Grundlage für alle weiteren KI-Berechnungen [11].

Das System initialisiert neun Kernattribute mit ausgewogenen Startwerten: Popularität, Vertrauen im Parlament, Medienakzeptanz, Koalitionsstabilität, Gesundheit, Stresslevel, Jahre im Amt, verabschiedete Gesetze und überstandene Krisen. Jeder Wert operiert auf einer Skala von 1 bis 100 und beeinflusst die weiteren Spielmechaniken [11][13].

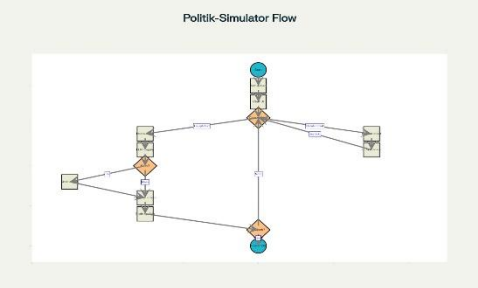

Abbildung 2: Abbildung 2: Spielablauf-Diagramm des Politik-Simulator 2025 -

Hauptspielschleife und Entscheidungsmechaniken

Nach der Initialisierung öffnet sich die ChatUI mit einem geteilten Interface, das zwei primäre Interaktionsmodi bietet. Der Hauptchat ermöglicht politische Entscheidungen in natürlicher Sprache, während der Beraterchat strategische Beratung durch einen KI-Politikberater bietet [4][5]. Diese Dualität spiegelt reale politische Prozesse wider, in denen Entscheidungsträger sowohl handeln als auch beraten werden müssen.

Die Entscheidungsverarbeitung erfolgt über das LLM-Engine-System, das jede Spielereingabe analysiert und realistische Konsequenzen generiert. Ein integriertes Krisensystem fügt mit 50-prozentiger Wahrscheinlichkeit zusätzliche Herausforderungen hinzu, wodurch das Spiel unvorhersehbar und anspruchsvoll bleibt [6][13]. Die Werteveränderungen sind auf ±30 Wertpunkte pro Entscheidung begrenzt, um extreme Schwankungen zu vermeiden und Realismus zu gewährleisten [13].

Wertesystem und Interdependenzen

Das Herzstück der Spielmechanik bildet ein komplexes Wertesystem aus neun miteinander verbundenen Attributen. Diese Werte repräsentieren verschiedene Aspekte politischer Führung: soziale Akzeptanz (Popularität, Medienakzeptanz), politische Effektivität (Vertrauen im Parlament, Koalitionsstabilität), persönliche Faktoren (Gesundheit, Stresslevel) und Erfolgsmessungen (Jahre im Amt, verabschiedete Gesetze, überstandene Krisen) [11][13].

Die Interdependenzen zwischen diesen Werten schaffen ein realistisches Feedback-System, bei dem Entscheidungen weitreichende Konsequenzen haben können. Beispielsweise kann eine Erhöhung des Stresslevels die Gesundheit beeinträchtigen, was wiederum die Popularität und letztendlich die politische Effektivität reduzieren kann [11][13].

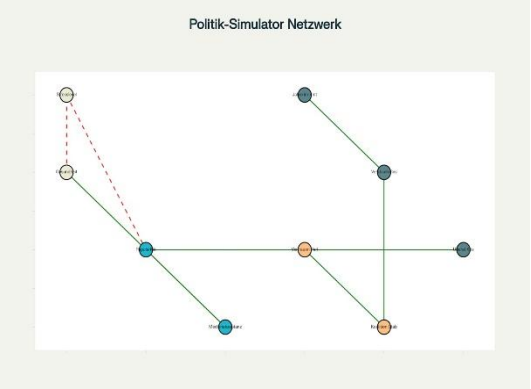

Abbildung 3: Abbildung 3: Werte-Netzwerk des Politik-Simulator 2025 - Zeigt die Interaktionen und Abhängigkeiten zwischen den neun Spielerwerten

KI-Integration und Prompt-Engineering

Dual-LLM-Architektur

Das System implementiert zudem eine Dual-LLM-Architektur mit zwei spezialisierten Prompt-Systemen. Das Engine-Prompt fungiert als objektiver Spielmechanismus, der politische Entscheidungen bewertet, Konsequenzen berechnet und Krisen generiert [13]. Das Advisor-Prompt agiert als subjektiver Politikberater, der strategische Empfehlungen gibt und historische Vergleiche anstellt [14].

Diese Trennung ermöglicht es, zwei verschiedene KI-Persönlichkeiten zu schaffen: eine systemische, die die Spielwelt simuliert, und eine beratende, die dem Spieler hilft. Das Engine-System gibt strukturierte JSON-Antworten zurück, während der Advisor in natürlicher Sprache kommuniziert [9][10][14][13].

Prompt-Design und Kontextualisierung

Die Prompt-Entwicklung ist durch das Trial-and-Error Prinzip über die gesamte Entwicklung hinweg weiter verfeinert worden. Das Engine-Prompt enthält detaillierte Anweisungen für die Bewertung politischer Entscheidungen, einschließlich spezifischer Richtlinien für Krisengenerierung und Werteveränderungen [13]. Es fordert konkrete Szenarien mit spezifischen Akteuren, Orten und Konsequenzen, um Vagheit zu vermeiden und Immersion zu schaffen.

Das Advisor-Prompt implementiert eine persönliche Beraterpersönlichkeit, die sowohl fachlich kompetent als auch emotional unterstützend agiert. Es integriert historische Referenzen und strategische Überlegungen in einen natürlichen Gesprächsfluss [14]. Die Kontextualisierung erfolgt durch die Integration der Spielerhistorie über das Logging-System, wodurch beide KI-Instanzen informierte Entscheidungen treffen können [8][14][13].

Benutzerinterface und Usability

„Moderne“ GUI-Implementierung

Die Benutzeroberfläche implementiert ein modernes, dunkles Design mit Chat-basierter Interaktion. Das System nutzt Java Swing mit umfangreichen Custom-Stylungen für ein zeitgemäßes Aussehen [4][5]. Die Entwicklung der GUI, habe ich selber getätigt um meine Fähigkeiten in die Funktionsweise unter Beweis zu stellen. Hiermit möchte ich Sie darüber informieren, dass ich die Grundlegende Programmierung der grafischen Benutzeroberfläche (GUI) selber vorgenommen habe. Allerdings habe ich die GUI mithilfe mehrerer KI-Tools stilisiert habe. Die zugrundeliegende Programmierung habe ich aber selbst durchgeführt. Die ChatUI verwendet ein Tabbed Interface, das verschiedene Spielbereiche elegant trennt und gleichzeitig eine intuitive Navigation ermöglicht.

Die Chat-Komponenten implementieren Bubble-Design für Nachrichten mit unterschiedlichen Stilen für Benutzer- und System-Nachrichten. Responsive Message Bubbles passen sich automatisch an die Fenstergröße an, während Custom Scrollbars die Benutzerführung verbessern [2][3][5]. Progress Bars zeigen die neun Spielerwerte in Echtzeit an, wodurch Spieler sofortige Rückmeldung über ihre Entscheidungen erhalten [4].

Innovation und technische Besonderheiten

KI-gesteuerte Spielmechaniken

Die Integration von Large Language Models in Spielmechaniken stellt einen Ansatz dar, der über traditionelle regelbasierte Systeme hinausgeht. Anstatt vordefinierte Entscheidungsbäume zu verwenden, generiert das System dynamische, kontextabhängige Reaktionen auf Spielereingaben [9][10][13]. Diese Flexibilität ermöglicht eine nahezu unbegrenzte Variation in Spielszenarien und -verläufen.

Das Krisensystem demonstriert besonders innovative KI-Nutzung durch die automatische Generierung plausibler politischer Szenarien basierend auf dem aktuellen Spielzustand. Das System kann spontane, aber logisch konsistente Herausforderungen schaffen, die aus den vorherigen Entscheidungen des Spielers resultieren [13].

1. View.pdf
2. AdvisorChatPanel.java
3. ChatPanel.java
4. ChatUI.java
5. MainChatPanel.java
6. SetupGUI.java
7. GameController.java
8. Startchecker.java
9. LLM_Logger.java
10. LLMClient.java
11. LLMResponseParser.java
12. Player_stats.java
13. advisorprompt.txt
14. engineprompt.txt

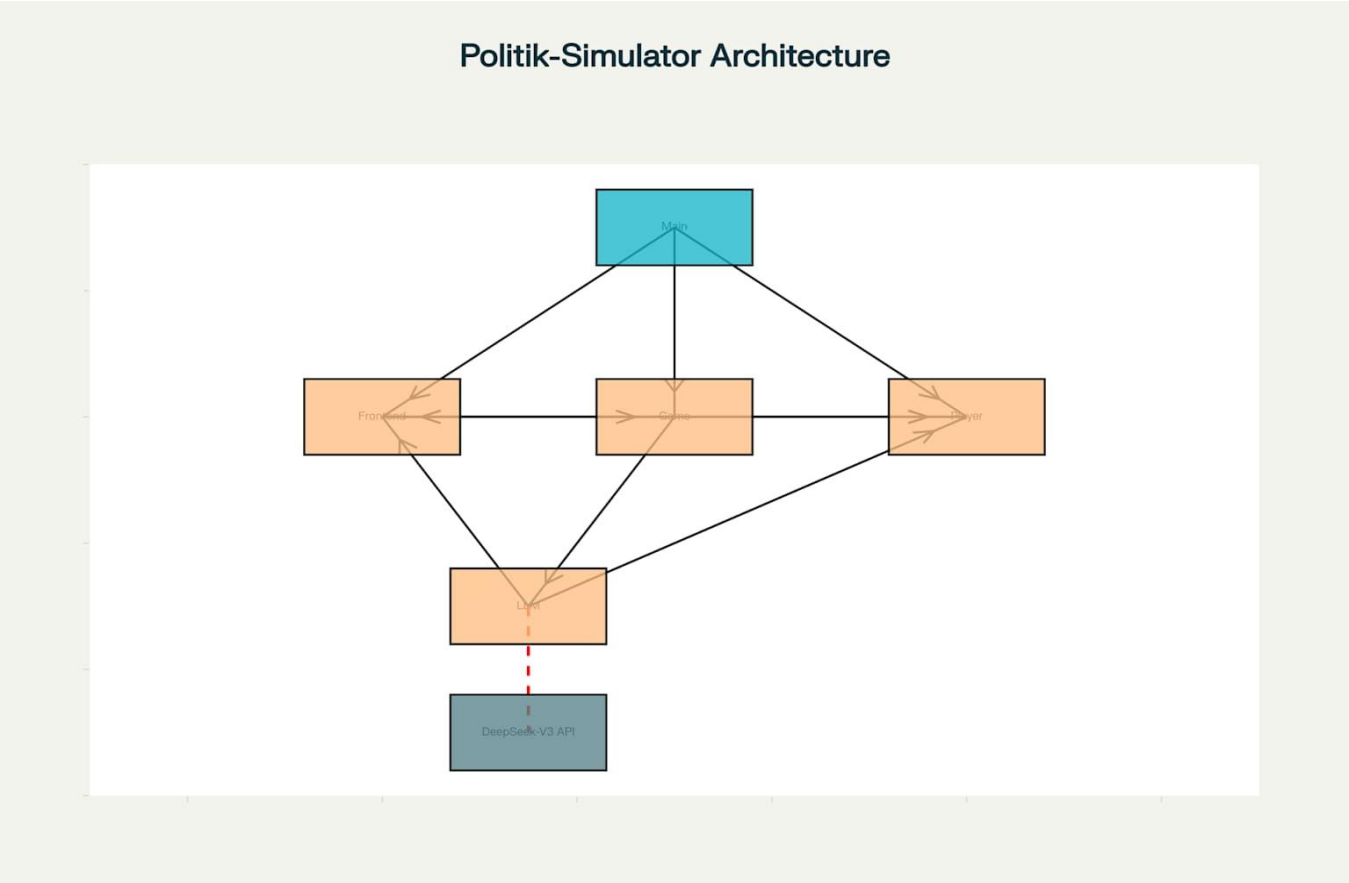

Abbildung 1

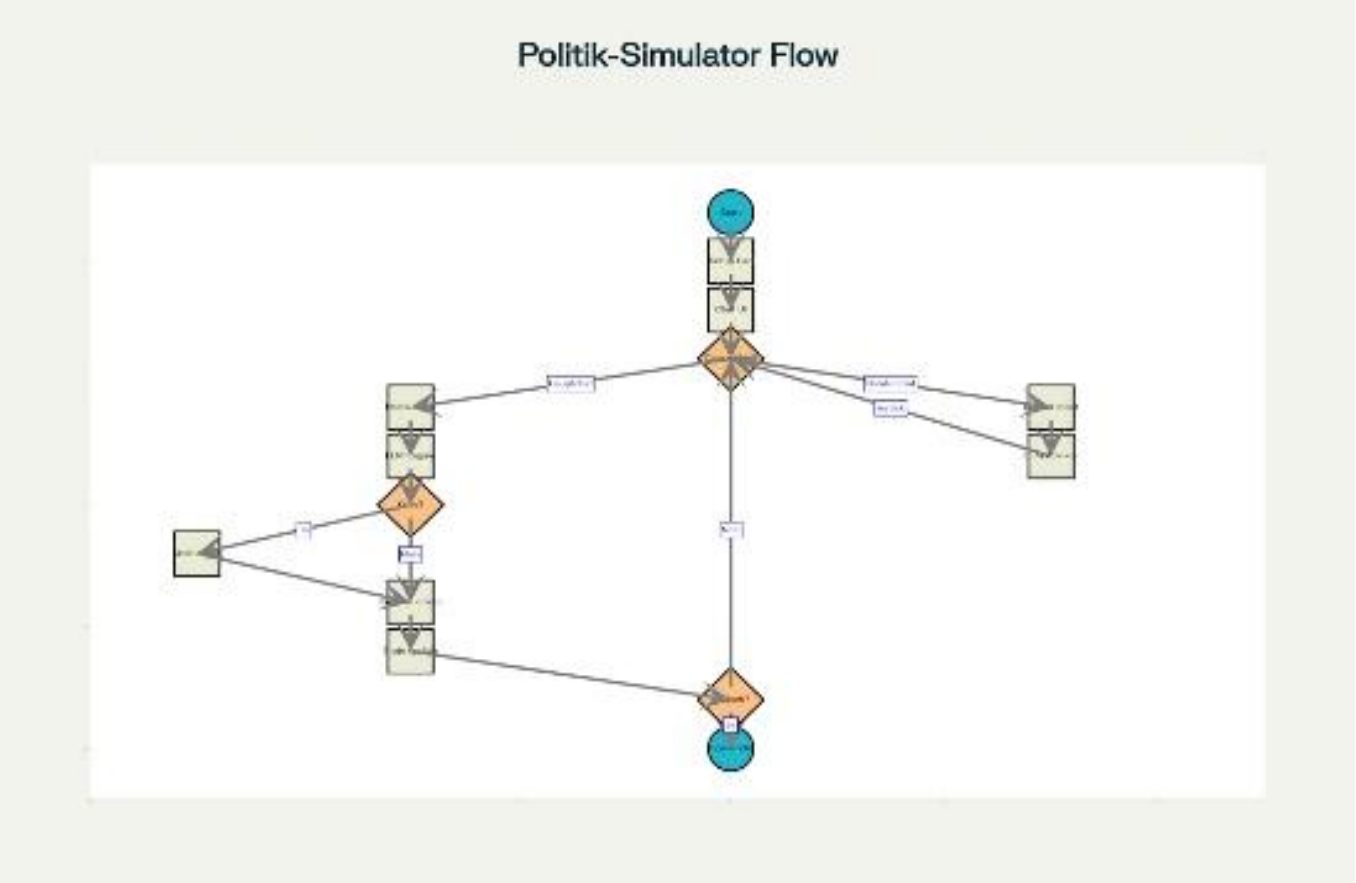

Abbildung 2:

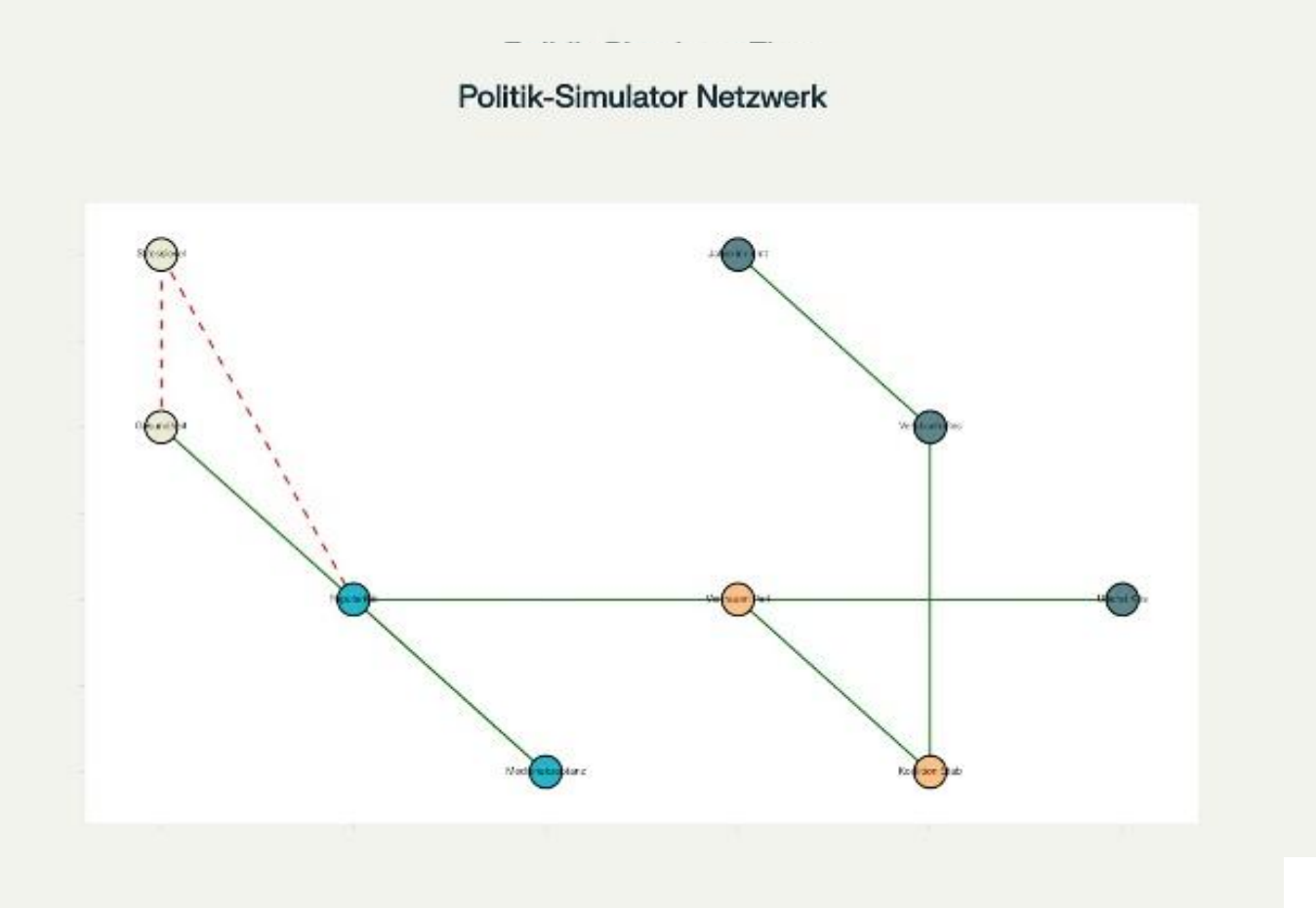

Abbildung 3: