ChatGPT

世界シミュレーションゲーム サービス要件定義書および基本設計

サービス要件定義

背景・目的

本プロジェクトは、世界の複数の国を舞台にしたシミュレーションゲームを開発し、プレイヤーが国のリーダーとして政策を実行しその結果を体験できるサービスを提供することを目的としています。近年、世界経済や国際情勢に関心を持ち始めたユーザー層に対し、ゲームを通じて**「自分の行動が世界に与える影響」**をシミュレートし、楽しみながら学べる場を提供します。従来のストラテジーゲームでは決められた選択肢による操作が主流でしたが、本サービスでは**生成AI（大規模言語モデル）**を活用し、プレイヤーが**自然言語で入力した自由なアクション**を解析してゲーム内世界の変化を予測・反映します。このアプローチにより、プレイヤーは現実さながらの複雑な世界の因果関係やイベントを体験できるようになります [1] [2]。

背景には、大規模言語モデルの発展によりテキストベースで仮想世界をシミュレーションすることが可能になりつつあるという技術トレンドがあります。例えばStanford大学の研究では、仮想空間上に複数のAIエージェントを配置し、それぞれが自律的に行動・交流することで、**人間社会の文化や行動ダイナミクスをデジタル空間に再現**する試みがなされています [1] [3]。こうした事例は、本ゲームのようにAIが国際社会のシミュレーションを行うコンセプトの実現可能性を示すものです。また、OpenAIのGPT-4など最新の生成AIは極めて豊富な知識を持ち、政策変更が経済や外交に与える影響について**人間らしい推論**を行えることが期待できます。ただし、研究によれば現状の言語モデルをそのまま**ワールドシミュレーター**として用いると不安定な挙動も報告されており [4]、安定したシミュレーションのためには工夫（出力フォーマットの構造化など）が必要とされています [5]。本サービスでは、生成AIの**自由度**と**知見**を最大限に活かしつつ、**構造化データ**で結果を受け取ることでシミュレーションの信頼性を高める設計とします [6]。

以上を踏まえ、本サービスの目的は単なるゲーム提供に留まらず、ユーザーが**世界を動かすシミュレーション**を通じて知的好奇心を満たし、世界経済や国際関係への理解を深めることにあります。将来的には教育的な側面や政策シミュレーションへの応用も視野に入れ、まずは娯楽性の高いゲームとして具現化します。

ターゲットユーザー

本ゲームの主なターゲットは以下のユーザー層です。

- **30代〜40代の大人**：社会人として世界経済やニュースに関心を持ち始めた層。仕事や投資を通じて国際情勢に興味を抱き、「自分なら世界をこう動かすのに」と感じるようなユーザーを想定します。この層はモバイルゲーム全体でも大きな割合を占めており、**35〜44歳がモバイルゲーマー全体の23.1%**を占めるなど存在感があります [7]。平均的なモバイルゲーマー年齢は36歳 [8] であり、30代のプレイヤーにも訴求力のある題材です。

- **10代後半〜20代前半の若年層**：高校生・大学生など、世界史や現代社会を学ぶ中で世界の動きに興味を持った層もターゲットです。モバイルゲーミングでは**16〜24歳が全体の28.3%**を占め [7]、若年層の多くが日常的にスマホゲームをプレイしています。この世代はエンターテインメントとしてゲームを選ぶ傾向が強く [9]、遊びながら世界情勢をシミュレーションできる本ゲームは高い没入感を提供

できるでしょう。特に**Gen　　Zやミレニアル世代は戦略ゲームやサンドボックス系を好む**傾向があり[10]、自由度の高い本ゲームのスタイルとマッチします。

- **世界経済や国際関係に興味を持ち始めた層**：年齢問わず、「ニュースで聞く経済問題を自分で操作してみたい」「歴史上の出来事を自分ならどう変えるか試したい」といった好奇心旺盛な層です。例えば現実の政策実験は倫理的・政治的リスクがありますが、**AIを用いた社会シミュレーションにより仮想社会で政策の影響を試せる**時代が来つつあります[11][12]。本ゲームはそうしたニーズに応え、ユーザーに安全な仮想環境で「もし○○したら世界はどうなるか？」を試行錯誤できる場を提供します。

上記のターゲット層はいずれも**スマートフォンでのゲームプレイに親和性**が高いことがデータから分かっています（例えばGen Z・ミレニアル世代の約70%がモバイルでゲームを遊んでいる[13]）。また「世界を動かす」シミュレーションは学習欲求や自己実現欲求の強いユーザーに刺さりやすく、**ストレス発散や知的興奮**を得る手段として機能すると考えられます[14]。

プラットフォーム

プラットフォームはスマートフォン（モバイル）を第一対象とし、ウェブアプリ（Webアプリ）として提供します。具体的には、モダンなフロントエンドフレームワークを用いた**シングルページアプリケーション（SPA）**として実装し、モバイル端末のブラウザ上で動作させます。これにより、iOS/Android問わず幅広いユーザーがインストール不要でプレイ可能となり、将来的なマルチプラットフォーム展開（タブレットやPCブラウザ対応）も容易です。

SPAを採用する理由は、ページ遷移の少ない**シームレスなユーザ体験**を実現するためです。ゲーム内で頻繁にデータの更新（ターン経過による国情報やイベントの更新）が発生しますが、SPAであればページリロードなしに動的に画面を書き換えられるため、アプリのようにスムーズな操作感を提供できます[15]。特にモバイル環境では回線状況によりページ再読込がストレスとなる場合がありますが、必要データのみを通信するSPAモデルならユーザー体感のパフォーマンス向上が期待できます[16]。

加えて、ウェブ技術ベースで構築することで、将来的に**Progressive Web App (PWA)**の形で提供し、オフライン時の簡易動作やホーム画面へのショートカット追加などネイティブアプリに近い利便性を持たせることも可能です。初期リリース時点ではモバイルブラウザでの最適化を重視し、タッチ操作に適したUIやレスポンシブデザインを採用します。PCからアクセスした場合も動作はしますが、画面レイアウトはスマホ向けを基本とします（必要に応じてPCレイアウトも将来検討）。

バックエンド側はクラウド上にサーバを用意し、ゲームの**APIサーバ**および**AI連携サーバ**として機能させます。ユーザーのアクセスはHTTPS経由でサーバに接続し、ゲームデータの取得やアクション結果の送受信を行います。サービス開始当初はアクセス集中も限定的と見込み、まずはシングルサーバで構築しますが、将来的なユーザー増加に備えてスケーラビリティ（負荷分散やサーバレスアーキテクチャの検討）も視野に入れます。

ゲームコンセプト・概要

本ゲームは、一種の**国家運営シミュレーション＆戦略ゲーム**です。プレイヤーはランダムに生成された架空の国家のひとつを担当し、その国のリーダー（統治者）となって国を発展させたり他国と関わったりします。他にもAIが制御する国が約9ヶ国（合計10カ国程度）が存在し、世界が構成されます。ゲーム開始時点で各国にはそれぞれ**異なる初期条件**が設定されます（例：経済大国だが文化的影響力が弱い国、若い人口構成の新興国、資源に恵まれるも政情不安な国、など多様なプロファイル）。これにより毎回異なる世界が形成され、プレイヤーは新しいシナリオを楽しめます。

プレイヤーの使命は、自国の状況を把握しつつ**国家戦略を自由に立案・実行**していくことです。ゲーム内の行動フェーズでは、プレイヤーは**自由記述のテキスト入力**によって国が起こすアクションを決定します。例えば: -「隣国との貿易協定を締結する」 -「自国通貨の金融緩和を実施する」 -「国営のAI研究所に予算投下する」 -「同盟国と合同で文化交流イベントを開催する」 -「敵対国に対し経済制裁を科す」 -「環境テクノロジーへの大規模投資を行う」

といった具合に、現実さながらの政策・行動を日本語または英語で入力できます。入力形式に明確な制限は設けず、自然文で書かれたものをAIが解釈します。これにより、**メニュー選択式では得られない自由度**と**想像力を掻き立てるゲーム体験**を提供します。

各ターン（アクション入力後）、ゲームシステムは現在の自国および世界の状況とプレイヤーの指示したアクション内容を生成AIに送信し、その結果として**世界に何が起こったかの予測**を取得します。生成AI（大規模言語モデル）は、与えられた状況と行動に基づいて以下を推論・生成します。

- **自国の変化**：プレイヤーの行動によって自国の経済指標や軍事力、国民の幸福度、政権の安定度などがどう変化したか。例えば「貿易協定締結」によりGDPが成長し失業率が改善するが、自国市場に安価な輸入品が流入し一部産業が打撃を受けた、など具体的な変化が返ります。
- **他国の反応・変化**：自国の行動に影響を受けた他の国々がどのように対応したか、またそれによって各国に何が起きたか。例えば貿易協定を結んだ相手国は経済成長する、第三国がそれを脅威と感じ関税を上げる、文化イベントを開催したことで他国との友好度が上昇する、一方で敵対国は警戒を強め軍拡に動く、等々**世界全体の動き**が予測されます。
- **イベントの発生**：上記の結果として起こるイベントをテキストやデータで示します。イベントとはニュース的な出来事で、例えば「同盟国AとBが貿易協定を締結」「国Cで政変が発生し新政権が成立」「国際スポーツ大会が国Dで開催される」「大規模自然災害が発生し各国が救援対応する」など、多彩な出来事が含まれます。**文化要素**も重視し、単に数値が変わるだけでなく**各国のカルチャーに根ざしたイベント**（祭典や宗教行事、芸術的ブームなど）が発生する可能性があります。これはゲーム内世界の臨場感を高め、ユーザーに「物語」を提供する役割を果たします。

生成AIから返ってくるこれらの結果は、予め定義した**構造化データ形式**（例えばJSON）に従っています[6]。構造化された形式でデータを受け取ることで、ゲームシステムはAIの出力を機械的に処理しやすくなり、各国ステータスの更新やイベントログへの登録を自動化できます。例えば「国AのGDPが+2.5%、国Bの軍事力指数が-1.0%、イベントXが発生」といった具合にデータとして反映されます。**OpenAIのAPI**では2024年8月にリリースされた新機能「ファンクションコーリング / Structured Output」を活用することで、開発者が指定したJSONスキーマに沿った出力を安定して得ることが可能になっています[17]。この技術により、AIの応答が常に予測可能な構造（キー項目やデータ型が保証されたJSONなど）となり、ゲームシステムへの統合が容易になります[17]。

ゲームシステムはAIからの応答データを受け取ると、それを元に**ゲーム内の状態を更新**します。各国のパラメータを新しい値に書き換え、発生したイベントはタイムライン（時系列ログ）に追加されます。この時、ゲーム内の時間も経過します（例えば1ターンごとにゲーム内で半年〜1年が進む想定）。画面上には「◯年◯月：あなたの国が実行した政策『XXX』の結果、〜〜が起こった…」といったレポートやニュースフィードが表示され、プレイヤーは自分の行動の影響を確認できます。

この一連の流れが**ターン制**で繰り返されます。プレイヤーは新たな世界情勢を踏まえて次のアクションを入力し、再度AIによるシミュレーション結果を得て状態更新…というサイクルです。他国は基本的にプレイヤーの行動にリアクションする形で動きますが、場合によっては**プレイヤー行動と無関係に環境由来のイベント**が発生する可能性もあります（例：自然災害や世界的流行イベントなどランダム要素）。これもAIが総合的に判断し「環境駆動の変化」として結果に含めることで実現します[18][19]。

ゲームは**サンドボックス型**の性質が強く、明確な「勝利条件」は初期バージョンでは設定しない想定です。プレイヤーの目標は自主的に「自国を世界一の経済大国にする」「全ての国と平和同盟を結ぶ」「軍事力で

世界を制圧する」等を見出してもらいます。一定ターン経過（例えば50年分）した時点で統計情報を表示し、一旦の区切り（スコア算出など）を出す程度は検討しますが、基本的にはユーザーの創意工夫で**シナリオを創造**できる自由度の高さを重視します。これはAIが予測する **emergent な社会変化**をそのままゲーム展開に取り込むという、これまでにない体験を提供するためです [20]。なお、将来的には「シナリオモード」（例えば冷戦期の世界を再現し特定の目的を達成する等）やマルチプレイヤーモードの実装も検討しますが、まずは**シングルプレイヤーのオープンサンドボックス**として完成度を高めます。

主要機能一覧

サービス要件に基づき、本ゲームが提供すべき主要機能を以下に整理します。

1. **国家プロファイル生成**:　　ゲーム開始時に架空の国家を約10ヶ国自動生成します。各国には国名、人口、経済規模(GDP)、政体（民主主義/独裁など）、文化的特徴、外交姿勢、軍事力、資源量などのパラメータを持たせます。国ごとに簡潔な紹介文や初期状況（例：「資源豊富だが技術力が低い農業国」など）も設定し、プレイヤーが把握しやすいようにします。生成AIを活用してユニークな国設定を自動生成することも検討しますが、初期バージョンでは**固定のテンプレート**やランダム値の組み合わせで十分現実味ある範囲に収めます。

2. **プレイヤー国家の選定・表示**: ランダム生成された国の中から一つをプレイヤーの担当国とし、その国の詳細情報をゲームUIに表示します。プレイヤーは自国の基本ステータス（人口、経済力、軍事力等）や現在起きている主なイベント、他国との関係性などを確認できます。UI上には**ダッシュボード**形式で重要指標を可視化し、必要に応じてグラフ表示（ターンごとの経済推移グラフ等）も行います。

3. **アクション入力（自然言語コマンド）**: プレイヤーがターンごとに行う政策や行動をテキストで入力できるインターフェースを提供します。入力ボックスと送信ボタンを設置し、送信時にその内容をサーバに送ります。自由入力ゆえの**入力支援**機能も用意します（例：よくあるアクション例の提示やテンプレート文の用意、「外交」「経済」などカテゴリ選択で案を出すなど）。また、入力された内容があまりに曖昧な場合はAI側で適切に解釈できるようプロンプトを補強するか、あるいは**入力内容の確認プロンプト**（「〜〜という理解で実行しますか？」）を返すことも検討します。

4. **生成AIによるシミュレーション**: バックエンドにて、現在のゲーム状態とプレイヤー入力したアクション内容をもとに生成AIへの問い合わせ（プロンプト生成とAPIコール）を行います。AIには**世界の状態遷移を予測**するよう促し、結果を**構造化データ**（JSON等）で受け取ります。この機能はゲームロジックの中核であり、AIモデルには高度な推論能力を持つもの（例：GPT-4クラス）を用いて、政治・経済・軍事・文化など多方面にわたる結果を同時に導出します [11]。AI応答の信頼性を高めるため、OpenAI APIの**ファンクションコーリング**機能やJSONスキーマを活用し、可能な限り一貫したフォーマットで結果が得られるようにします [17]。なお、AIが不確かな結果や過激すぎるシナリオ（現実乖離した核戦争乱発など）を出しにくくするため、プロンプト内で**現実的で一貫性のある予測を行うこと**や**極端な表現を避けること**を指示するガイダンスを含めます。

5. **ゲーム状態の更新**: AIから受け取った結果データをもとに、ゲーム内の各種データを更新します。具体的には:

6. 各国オブジェクトのパラメータを変更（増減）し、新しい値を格納。
7. イベントについてはイベントログに新規イベントエントリを追加。各イベントは発生国・日時・内容などを持つデータとして扱います。

8. 世界共通のタイムスタンプ（現在の年/月など）を進める。　更新時には前ターンからの**差分**を明確に把握できるよう、差分データを別途保持したり、更新後の値をUIにハイライト表示するなどの工夫を行います。

9. **結果の表示（イベントタイムライン/レポート）**： プレイヤーに対し、そのターンで何が起こったかを**分かりやすく伝える表示**を行います。重要機能の一つとして、**時系列のイベント表示**があります。ゲーム画面上に「ニュースフィード」「年表」あるいは「ログ」セクションを設け、最新の出来事から順にテキストで表示します。例：「2025年7月 - あなたの国が隣国Bとの自由貿易協定を締結。これにより両国の貿易量が増加し経済成長率が向上しました。他方、競合関係にある国Cは危機感を覚え関税を引き上げました。」といった形で、プレイヤーの行動とその影響を文章で確認できます。可能であれば重要度に応じて**トピック別に色分け**したり、アイコン（経済=グラフアイコン、戦争=爆発アイコン、文化=祭りアイコン等）を付与して視認性を上げます。また、自国に直接関係ない他国間のみのイベントも発生した場合は「世界ニュース」として表示します。

10. **国家情報閲覧（他国の状況確認）**：プレイヤー以外の各国についても、詳細情報や現在の状況を閲覧できるようにします。他国のパラメータ（経済規模、軍事力、友好/敵対関係、政体、文化特色など）や、最近どんなイベントがその国で起きたかをリスト表示します。これは各国の動向を把握し戦略を練る手がかりとなります。UI上では世界地図上の国をタップして詳細パネル表示、または国名一覧から選択して情報表示する形式を想定しています。

11. **ゲーム内時間とターン管理**: ターンが進むごとにゲーム内の年月を管理し、各イベントにタイムスタンプを付与します。例えば初期を2025年とし、1ターン＝半年進む設定なら、2ターン目は2025年下期、3ターン目は2026年上期...のように**暦の概念**を持たせます。これにより「◯年に何が起きたか」を振り返ることができ、ゲームの歴史が蓄積されていく楽しみがあります。UIには現在年次を表示し、年が変わるタイミングで年越しイベントや統計情報（「今年の世界経済成長率は◯%でした」等）を表示する演出も考えられます。

12. **設定およびチュートリアル**: ゲームの導入として、遊び方を説明するチュートリアル機能を用意します。特に自然言語入力に慣れていないユーザー向けに、「どんなことが出来るのか」「有効なコマンド例」等をガイドします。チュートリアルではサンプルシナリオ（例えば「経済政策編」「戦争編」など）を示し、ユーザーが数ターン試して感覚を掴めるよう誘導します。またオプション設定ではゲームスピード（1ターンあたりの期間）、難易度（AIの反応の厳しさやイベント頻度を調整）、言語設定（UIや出力の言語を日本語/英語切替）などを変更可能にします。

13. **データ保存/ロード（将来的機能）**： 初期バージョンでは一回のプレイが完結する前提ですが、将来的にはゲーム状態を保存し後から再開できるようにすることも検討します。その場合ユーザーアカウント管理やサーバ側への状態保存が必要になります。現段階では優先度低めのため基本設計には含めませんが、コードやデータ構造は将来拡張に対応できるよう留意します。

非機能要件

本サービスに求められる非機能面での要件・考慮事項を以下に整理します。

- **パフォーマンス**: 生成AIへの問い合わせには数秒程度の時間を要する可能性があるため、ユーザーにストレスを与えない工夫が必要です。ターン終了後の結果生成待ち時間は目標**5秒以内**（理想は2秒程度）とし、それ以上かかる場合はUI上でロードインジケータやヒントメッセージ（例：「AIが未来を予測しています...」）を表示して不安を軽減します。フロントエンドはSPAとして軽量化し、初回ロードも3G回線でも許容できる範囲（数MB以下）に抑えます [21] 。サーバサイドはノンブロッキングIO等でスケールしやすく実装し、同時に複数ユーザがアクセスしても各自のAIコール処理が極力並行して行えるようにします。

- **スケーラビリティ**: ローンチ直後は限定的なユーザ数を想定しますが、メディア露出等で急増する可能性もあります。バックエンドはステートレスなAPIサーバとして設計し、必要に応じて負荷分散（クラウドのオートスケール機能など）で対応可能にします。生成AI部分は外部APIに依存するため、1秒間

に処理できるターン数に限りがあります。そのためユーザ数が増えた場合は、AI APIキーの追加取得や自前モデルの用意、もしくはターン実行にチケット制・クールダウンを設ける等の対応を検討します。

- **信頼性・一貫性**: AIの出力する結果が毎回構造化されていないとゲームが破綻するため、**構造化データの信頼性**は極めて重要です。OpenAIの機能でかなり高精度にJSON出力を得られる見込みですが、万一JSONのパースに失敗した場合に備え、再試行やエラーハンドリングを実装します。またAIの内容自体の信頼性（物理法則や論理、一貫性）も注意点です。例えば前のターンで経済成長+10%と言っていたのに次のターンで「実は不況でした」と矛盾しないよう、プロンプトに前ターンまでの文脈を常に与える、重要事項をリマインドする等の対策を取ります。それでも**完全な一貫性は保証できない**ため、ユーザーには「予測に過ぎない」ことを理解してもらう文言をゲーム内で案内しつつ、矛盾が生じにくい調整を繰り返します。

- **UX（ユーザビリティ）**： 自然言語入力という柔軟な操作系の反面、何をして良いか分からないという戸惑いが生じないようにします。先述のチュートリアルやガイド機能に加え、ゲーム中も適宜ヒントや例を表示します（例：「経済政策で迷ったら‘増税して財政健全化’など試してみましょう」）。また、シミュレーション結果がテキスト中心になるため**情報量が多くなりすぎないよう**工夫します。段落を分けて読みやすくしたり、重要な数値は強調表示、長文の説明は折りたたんで要約をまず表示し詳細はタップで展開など、スマホ画面でも把握しやすいUIにします。**短い段落と箇条書き**を適切に用いて、一度に読む負担を軽減します（本ドキュメントと同様の手法をゲーム内文章にも活用）。

- **多言語対応**: 現時点ではUI表示と言語モデルの応答を**日本語**中心にすることも可能ですが、ゲームコンセプト的に英語も親和性が高いです。生成AIは英語の方が学習データが豊富なため詳細かつ的確なシミュレーション結果を出す傾向があることが予想されます。そのため**内部的には英語でプロンプト＆結果生成**し、ユーザーには日本語翻訳して見せる、という手法も検討します（翻訳もAIにさせる）。ただし直訳では固有名詞等が崩れる可能性があるので、まずは日本語プロンプトでも十分テストし、必要に応じて内部英語化を決めます。UIテキストは将来的に英語版を作成して海外ユーザ向けリリースも視野に、**i18n対応**（多言語リソース管理）を行っておきます。

- **安全性・モデレーション**: プレイヤーの入力内容やAIの出力内容に対して、不適切なものが含まれないよう配慮します。プレイヤーは自由入力できるため、暴力的・差別的・政治的にセンシティブな内容を入力する可能性があります。またAIも世界シミュレーションの中で紛争や犯罪などセンシティブな事柄を語る場合があります。基本方針としては**過度な不適切表現は禁止**し、AI側でもOpenAIのコンテンツフィルタ等を利用して露骨なヘイトやアダルト内容は出力しないようにします。ゲーム内容上、戦争や経済制裁など避けられないテーマもありますが、それらはあくまで架空の国同士の出来事として扱い、現実の特定の人種・国・政治団体を中傷するような展開は避けます。必要に応じてAIのプロンプトに「倫理・国際法に反する行為は周辺国から強い非難を受ける」といったモラル的ガイドを与えることで、ユーザーが極端な行動を取った際もゲーム内できちんと**ペナルティや反応**が返るようにし、自然に抑止力とします。

- **拡張性**: ゲームデザイン上、後から機能追加・調整が頻繁に起こり得ます。例えば国のパラメータに「環境指数」を追加したい、イベント種類を増やしたい、UIを刷新したい等に備え、コードのモジュール化やデータ駆動設計を意識します。特にAIプロンプト部分はチューニングが鍵となるため、**プロンプト文をサーバサイドでバージョン管理**し、アップデート時には柔軟に変更できるようにします（場合によってはA/Bテスト的に異なるプロンプトを試すなど）。データ構造も、既存フィールドの変更が他へ波及しにくいよう抽象化しておきます。またAIモデル自体の変更（APIから自前モデルへの移行など）にも対応できるよう、AI呼び出し部分をラップしたモジュールにしておきます。

以上がサービス要件定義です。次章では、これら要件を実現するための基本設計について詳細を述べます。

基本設計

アーキテクチャ全体像

本ゲームのシステムは、大きく**フロントエンド（クライアント）**と**バックエンド（サーバ）**に分かれるクライアント・サーバ型アーキテクチャです。さらにバックエンド内で、ゲームロジックとAI連携部分を適切に構成します。全体像を以下に示します。

- **フロントエンド**: モバイル端末のブラウザ上で動作するSPA。UIレンダリング、ユーザー入力受付、結果表示の責務を持つ。バックエンドとはHTTP(S)通信（REST APIもしくはGraphQLなど）でデータをやり取りする。
- **バックエンド**: クラウド上に配置されるWebサーバ/APIサーバ。複数のサブコンポーネントに分かれる。
- **ゲームサーバ（アプリケーションサーバ）**: ゲームのメインロジックを担うサーバサイドモジュール。クライアントからのリクエスト（例:「行動Xを実行」）を受け取り、ゲーム状態を管理・更新する。必要に応じてAIモジュールを呼び出し、結果を受け取って処理する。
- **AI連携モジュール**: 大規模言語モデル（LLM）にリクエストを送り、シミュレーション結果を受信する部分。OpenAI等の外部APIを利用する場合、その通信処理やトークン数管理、APIキー管理を行う。LLMへの入力プロンプト生成と、出力JSONのパースもここで実施する。ゲームサーバから見ると一種のサービスモジュールであり、例えば `simulate_world(state, action) -> result` のようなインタフェースを提供する。
- **データストレージ**: ゲームの永続データを保管するデータベース。または必要に応じ、インメモリで状態を管理する。初期版では単純なセッション内状態のみでDB不要だが、将来的なセーブ機能やユーザ管理を見据えデータストアを設計しておく。国情報のテンプレートやイベントテンプレートなど静的データもここに含む。
- **認証・アカウント管理（必要なら）**: 初期リリースでは不要だが、ユーザごとのプレイ履歴を保持する場合に備え用意だけ検討。今回はシングルプレイかつ匿名利用が前提のため実装しない。

上記のバックエンド群は、開始当初は単一サーバプロセス上に同居させます。例えばNode.jsベースでExpressサーバがゲームAPIとAIモジュールを内包し、必要ならDBと接続する構成です。負荷が上がってきたらAIモジュールを別プロセス（マイクロサービス化）してスケールさせる、DBをスケールさせるなど柔軟に対応できるようモジュール間依存を低く保ちます。

データフローとしては以下のようになります。

1. ユーザーがフロントエンドから「アクション入力」→「送信」すると、フロントエンドは現在のゲームIDや自国ID、入力テキスト等を含むリクエストをバックエンドのゲームAPIに送ります（POST `/game/turn` など）。
2. バックエンドのゲームサーバはリクエストを受信すると、自国を含む現在の**世界状態データ**を取得します（メモリ上またはDBから）。そしてAI連携モジュールに対し、「この状態でプレイヤーがXの行動を取った」という指示を渡し、結果を問い合わせます。
3. AI連携モジュールは受け取った状態と行動を元に**プロンプトを生成**します。プロンプトには世界の要約情報（各国の主要データ、直前までの重要イベント）とプレイヤー行動内容を含め、「次に世界がどう変化するかをJSONで出力して下さい」等の指示文を付加します。これを外部のLLM APIに送信します。
4. 外部LLM（例: GPT-4）はプロンプトを受け取り、内部知識と推論に基づいて**世界の次状態**を記述したテキスト（JSONフォーマット）を生成します [22] [23]。OpenAIのfunction calling機能を使う場合、返答は構造化されたJSONデータとなります。
5. AI連携モジュールはLLMからの応答を受け取ります。まず**JSONのパース**を試み、期待するデータ構造に合致するか検証します。問題がなければデータオブジェクトに変換しゲームサーバへ返します。もしJSONが不正（パースエラー、スキーマ不一致など）なら再度LLMにプロンプトを送る（システム

メッセージで「JSONのみ出力せよ」と強制する等してリトライ）か、エラーをゲームサーバに通知します。

6. ゲームサーバは取得した結果データを用いて、内部のゲーム状態を更新します。各国オブジェクトの該当フィールドを書き換え、イベントリストに新イベントを追加し、ゲーム内時間を進めます。この際、更新前のデータと比較して差異を抽出し、フロントエンドに返すレスポンスを組み立てます。レスポンスには例えば「更新された自国データ（全項目または差分）、発生イベント一覧（このターン分）、現在の年月」などが含まれます。
7. フロントエンドはバックエンドからのレスポンスを受け取ると、画面上の状態を書き換えます。具体的には：自国ステータス表示の数値を更新、必要に応じてゲージやグラフをアニメーション更新、イベントログに新項目を追加スクロール表示、など**UIのリアクティブな更新**を行います。ユーザーはこれを確認し、次のターンの行動入力へと進みます。

このように、各ターンごとにクライアント→サーバ→AI→サーバ→クライアントというフローが発生します。SPA内ではAPI通信部分は非同期処理となり、その間ユーザーには前述のローディング指示などを見せます。1ターンの処理が完了するたびに、必要なら内部で**チェックポイント保存**（オートセーブ的な処理）を行う設計としておけば、万一途中でブラウザを閉じても再開できるように拡張可能です。

フロントエンド設計

フロントエンドは**シングルページアプリケーション**として、HTML/JavaScript/CSSで実装します。主要フレームワークは開発生産性と保守性を考慮し、React.js (またはVue.js) + TypeScriptを採用することを提案します。以下、UIの画面構成と各コンポーネントについて記します。

画面構成:
画面は大きく分けて「メインゲーム画面」と「メニュー/設定/ヘルプ画面」に分かれます。起動後すぐメインゲーム画面が表示され、そこからメニュー（チュートリアルや設定）に遷移する形です。SPAのため画面遷移もコンポーネントの表示切替で行います。

- **メインゲーム画面**: ここにゲームプレイ中の主要UI要素が集約されます。
- 上部バー: 現在のゲーム内年月、自国名、重要指標サマリ（例: 人口○千万・GDP○兆円・満足度○%等）を表示するステータスバー。
- 中央メインエリア: 二つのタブ/ビューを切替可能にします。
    1. **「世界ビュー」**: 世界地図あるいは国一覧リストを表示するタブ。地図表示の場合、各国を領土色分けしタップ可能にします。リスト表示の場合は各国名と主要ステータスを一覧できる表形式。ユーザーはここで他国を選択できます。
    2. **「自国ビュー」**: 自国の詳細情報を表示するタブ。国内の経済状況グラフ、国民構成、外交関係一覧、軍事・資源の状況など、項目ごとにカードUIやチャートで見せます。将来的にはこのビューから各省庁ごとの詳細画面に入る等も考えられますが、初期版では一ページに簡潔にまとめます。
- 下部エリア: **アクション入力パネル**。テキスト入力ボックスと送信ボタンがここに配置されます。また、入力の参考としてカテゴリー選択（「経済」「外交」「軍事」「文化」等）の簡易メニューや、過去にプレイヤーが行ったアクション履歴からの再利用機能も検討します。送信ボタン押下時にはすぐに非活性状態+ローディングを表示し、多重送信を防止します。
- 画面右側または別枠: **イベントログ/ニュースフィード**。時系列で最新の出来事を表示するスクロール可能な領域です。最新ターンの結果が一番上に追加され、ユーザーは必要に応じスクロールして過去ログも閲覧できます。各イベントは日時や関係国が分かる短い文章で表示し、詳細閲覧をタップでできるようにする（ポップアップで詳細情報や関連国データを表示）ことも可能です。

- ※スマホ縦画面を想定しているため、上記配置は縦長スクロールになる可能性があります（例えばイベントログは下部タブで切り替えなど）。UIレイアウトはデバイス幅に応じレスポンシブ調整します。タブレットやPCでは横にパネルを並べて同時表示する等。

- **他画面**:

- **設定/メニュー画面**: ゲーム一時停止中に表示。サウンド（本ゲームではBGM/効果音等必要なら）、言語、速度設定など変更可。ゲームに戻るボタンあり。
- **チュートリアル画面**: 初回プレイ時またはユーザーが任意で閲覧可能。基本的な遊び方（テキスト入力例、各画面の説明）をスライド形式で表示。
- **ゲームオーバー/結果画面**: （導入する場合）特定条件でゲーム終了した際に表示。自国の成績統計や達成したことをまとめてフィードバック。再度プレイするか終了するか選択。

コンポーネント: フロントエンドのUIは再利用性と可読性のためコンポーネント指向で構築します。主なコンポーネント例: - `<CountryStatusCard>`: 国の基本ステータスを表示するカード。国旗（架空なので代わりに国名の頭文字アイコンなど）や名称、主要数値（人口、GDPなど数項目）を表示。自国ビューや国一覧で使用。 - `<ActionInput>`: アクション入力パネル。テキストエリアと送信ボタン、およびカテゴリボタン等で構成。送信ボタン押下イベントで親コンポーネントに通知し、API呼び出しをトリガー。 - `<EventLogItem>`: イベントログの1項目。日時、内容テキスト、関連国名（リンク付き）などを含む。イベント種別に応じアイコンや色を変えるロジック含む。複数行にわたる場合折りたたみ処理も持つ。 - `<Timeline>`: イベントログ全体。最新順に `EventLogItem` を並べる。スクロールや「もっと見る」（過去分を読み込む）機能。 - `<WorldMap>`: 世界地図表示コンポーネント。シンプルに10カ国程度であれば抽象化した地図イラストでも良いが、実装コストを考え、初期版では地図は用意せず国一覧テーブルで代替し、地図表示は将来対応にしても良い。 - `<CountryDetailModal>`: 他国詳細表示のモーダル。国一覧やイベントログから国名タップで開き、その国の詳細ステータス・最近の出来事・外交関係などを表示。モーダル内にも `CountryStatusCard` や簡易チャートを含める。

状態管理: フロントエンドではゲームの状態（自国データ、他国リスト、イベントログなど）を適切に状態管理します。React+ReduxやVuex等、状態管理ライブラリを用いてグローバルなゲーム状態ストアを持ち、各コンポーネントはそれを参照して描画します。API通信後にストアを更新すれば、自動的に関連コンポーネントが再描画される仕組みにします。これにより状態の一元管理とデータフローの見通しを良くします。

通信処理: フロントエンドからバックエンドへの通信は、Axios等のHTTPクライアントを用いてREST APIを呼び出す想定です（GraphQLの場合はクエリ/ミューテーションの呼び出し）。通信処理はサービス層を作り、例えば `GameApiService.submitAction(actionText)` のように抽象化して扱います。レスポンスを受け取ったら状態ストアを更新し、ローディング状態を解除します。エラー時にはユーザーにエラーメッセージ（「ネットワークエラー」や「AI応答エラー」等）を表示します。

ビジュアル/演出: ゲーム性を高めるため、単なる数値変化も可能な限りビジュアルで演出します。例えば経済指標が上がったら緑色で数値がフラッシュする、人口減少したら矢印↓アイコンを一瞬表示、同盟が結ばれたら握手アイコンを表示、戦争勃発なら画面が一瞬点滅する等、小さな演出で雰囲気を出します。テキスト中心のゲームですがUIデザインには近代的で直感的なスタイルを採用し、**「ニュースアプリ＋戦略マップ」**のようなイメージで構築します。

バックエンド設計

バックエンドはNode.js（JavaScript/TypeScript）かPythonのいずれかで実装することを考えています。Node.jsであればフロントと言語を揃えやすくリアルタイム処理にも強みがあり、PythonであればAI関連のライブラリ活用や計算に強みがあります。ここでは仮に**Node.js + Express**による実装を前提に説明します。

主要モジュール: 1. **ゲームAPIコントローラ**: Expressのルーティングとして、`POST /api/action` のようなエンドポイントを用意します。リクエストには `gameId`（セッション識別子）や `actionText` 等が含まれます。コントローラは該当ゲームセッションを管理するGameManager的なクラスに処理を委譲します。 2. **GameManager / GameState**: 各ゲームの状態を管理するクラス/モジュールです。GameManagerは例えば `GameManager.get(gameId)` でそのゲームのGameStateオブジェクトを取得できます。GameStateは内部に現

在のターン数、各国のデータ、イベントログ等を保持します。`GameState.applyAction(actionText)` のようなメソッドで、アクション適用→AI結果反映→状態更新までを一貫して行います。　3. **AIクライアントモジュール**: OpenAIなど外部AIサービスと通信するモジュールです。例えば `AIClient.predictOutcome(gameState, action)` を呼ぶと、内部で前述のプロンプト作成→API呼び出し→レスポンス受取→JSONパースを行い、結果のオブジェクト（ゲーム結果DTO）を返します。OpenAIのAPIキーやエンドポイントURLは設定ファイルで管理し、このモジュールがそれらにアクセスします。応答JSONのバリデーションにはJSON Schemaを用いて、必須フィールドが揃っているか等をチェックします [17] 。 4. **データモデル**: Country（国）、Event（イベント）、WorldState（世界全体）などゲーム内要素を表すデータクラス（あるいはTypeScript型定義）を用意します。例えばCountryオブジェクトは以下のようなフィールドを持つ想定です。

```
class Country {
  id: string;
  name: string;
  population: number;
  gdp: number;
  governmentType: string;  // 政体
  culture: string;      // 文化のキーワードや説明
  stability: number;    // 国内安定度0-100
  military: number;     // 軍事力指標
  relations: Map<string, number>; // 他国との関係度（-100〜100など）
  // ...必要に応じて追加
}
```

Eventクラスは例えば

```
class Event {
  id: string;
  year: number;
  description: string;
  relatedCountries: string[]; // 関係した国ID
  type: string; // "economic", "war", "culture" 等分類
}
```

といった構造です。WorldStateは国一覧、現在年月、ターン数、グローバルパラメータ（例: 世界総GDP、技術水準平均など必要なら）を持ちます。

これらモデルはAIからの出力JSONとも対応しています。AIには各国の情報をテキストで与えますが、返すときは例えば国名キーで連想配列を返すようなフォーマットを想定します。そのためパース後にCountryオブジェクトにマッピングしやすいよう、キー名とモデル属性を揃えておきます。

1. **プロンプト生成/結果解釈**: AIClient内で行う処理ですが特に重要なので記します。プロンプト生成は**ひな型テンプレート**を用意し、そこに動的に世界の情報とアクションを埋め込みます。例えばPseudo Prompt:

   ```
   以下は架空の世界の状況です。
   国一覧:
   - 国A: 人口1億人, GDP=5兆円, 民主主義国家, 同盟関係: Bと友好, Cと敵対, ...
   - 国B: 人口5000万人, GDP=2兆円, ...
   ```

```
 ...(他国情報)
（以上の情報は2025年時点）

プレイヤー（国A）の行動: "隣国Bとの自由貿易協定を締結する"

質問: 上記の行動により、この後世界にはどのような変化が起こりますか？以下のJSON形式で出力してください：
```json
{
  "countries": {
    "国A": { ...変化内容... },
    "国B": { ...変化内容... },
    "国C": { ...変化内容... },
    ...
  },
  "events": [
    { "description": "...", "relatedCountries": ["国A","国B"], "type":"economic" },
    ...
  ]
}
```

```
のように...
```

（実際のプロンプトでは英語の方が良い可能性がありますが、このような内容）

このテンプレートで重要なのは、**余計な例文や説明を極力減らし、必要フォーマットを厳格に示す**ことです。JSONのフォーマット例を直接プロンプトに含めているのは、モデルが構造を崩さず出力するための工夫です [6] 。また、各国情報もシンプルな列挙に留め詳細な理由付けは与えません（モデルが自前の知識で因果推論する余地を残す）。行動内容はプレイヤー入力をそのまま埋め込みますが、日本語入力を英語プロンプトに載せる場合は適宜翻訳するか、そのままでもGPT-4なら対応可能なため状況次第です。

結果の解釈では、モデルから返ってきたJSON文字列をまずパースします。Node.jsなら `JSON.parse` でオブジェクト化し、期待する構造と合致するかチェックします。例えば全10カ国分のキーが存在するか、イベント配列があるか、各数値が想定範囲か等を検証します。不備があればログ出力し、ひとまず無視するか再プロンプトします。正常なら各国データを対応するCountryオブジェクトに反映し、Eventも生成してリストにします。**差分更新**の場合、AIから**「差分」**だけ受け取るという手もあります [24] 。例えば「GDP:+5%」のように変更点のみ返させ、それを元に現在値を更新する方式です。こちらの方が出力が冗長でなく効率的ですが、実装複雑度が上がるため初期版では**フル状態を返す**方式で実装し、安定してきたら差分方式に移行を検討します。

1. **データ永続化**: 初期版では必須ではありませんが、ユーザが長時間プレイすることを考えるとサーバメモリ上だけでは不安もあります。クラッシュ対策やゲーム再開の要望に備え、GameStateを定期的にJSONシリアライズしてDB（例えばMongoDBやRedis）に保存する仕組みを用意できます。コストとの兼ね合いで、当初はバックアップ用ログ出力程度に留めるかもしれません。いずれにせよ、GameStateやEventログをJSON化できるシリアライズ機構は作っておきます。

API設計: - `POST /api/action` : プレイヤーの行動を送信。リクエストボディに `{ gameId, action: "...(テキスト)..."}`。レスポンスは `}, events: [ ...今回発生イベント... ]` のような形。フロントはこれを受け取り画面更新。
- `GET /api/state` : （必要に応じて）現在の全世界状態を取得。フロント再接続時やデバッグ用。
```

- `GET /api/country/:id`：国個別の詳細情報取得API。イベントログや詳細ステータスを返す。これは一括状態で足りていれば省略可能。

API通信はHTTPSで行い、セキュリティのためCORS設定や認証（トークンによるゲームID保護など）も実施します。ただ単体ゲームであれば簡易的にゲームIDをGUIDにして推測困難にする程度で良いでしょう。

データ設計

データ構造について、ゲーム内で扱う情報を整理します。

国データ: 各国はCountryモデルで表現され、以下のフィールドとします（上記Countryクラス参照）。パラメータはゲーム性とAIへの説明のために適度な粒度に設定します。あまり細かすぎてもユーザーが理解しづらくAIも扱いづらいため、例えば経済関連はGDPと成長率/失業率くらい、軍事は軍事力スコア1~100くらい、文化は定性的なタグ（「西洋的」「保守的」など）と影響力スコア、など**数値＋キーワード**の組み合わせで表現します。

- `id`: 識別子（内部用ユニークID）。例: "C1","C2",...。
- `name`: 国名。ユニークで発音しやすい架空名（AI生成も検討）。
- `population`: 人口（万人単位などざっくり）。
- `gdp`: GDP規模（100億ドル単位など架空通貨換算も検討）。
- `growthRate`: 経済成長率（%）。経済トレンドを見る指標として。
- `governmentType`: 政体（"民主制", "専制国家", "社会主義"等テキスト）。AIに国性格を伝える要素。
- `stability`: 政治安定度（0-100数値）。政権の安定や国内秩序を表す。
- `culture`: 文化的特徴・価値観（"自由主義", "宗教的", "軍国的", "商業的"等キーワード）。
- `cultureInfluence`: 文化的影響力指標（数値）。高いとソフトパワーが強くイベントで有利など。
- `military`: 軍事力指数（数値）。軍備や兵力の規模感。
- `relations`: 他国との関係マップ。キーは他国id、値は友好度（-100〜100）。同盟国なら高、敵対なら低。AIには近隣国との関係のみ提示するなど量を調整。
- `resources`: 資源・産業特性（例: "石油大国","穀物輸出国" 等タグ）。
- `leader`: （余裕あれば）指導者名や性格。AIが判断に使う可能性もあるが複雑になるので省略可能。

これらのうちAIプロンプトに含めるのは主要なもの（人口規模、経済規模、政体、友好敵対関係、特徴キーワードなど）です。実際の数値そのままではなく、「大国/小国」「先進国/途上国」等の表現に置き換えて与える方がAIには理解しやすいかもしれません（例: 人口1億なら"very large population"など）。このあたりは調整ポイントです。

イベントデータ: Eventモデルは上記の通り、`description`（文字列）と`relatedCountries`（関係国ID配列）、`type`（分類）を基本とします。AIにはイベント内容の文章まで生成させますが、構造化データとしては**イベントリスト**を持たせ、それを組み立てて表示します。例えばAIからは「イベント: 国Aと国Bが自由貿易協定を締結」といった文が返るので、それをそのままログ表示に使いつつ、relatedCountriesにA,B、typeに"economic"を割り当て内部保存します。イベントはタイムスタンプ（ゲーム内年月）も記録し、複数ターンまたいでの分析（「過去5年間で戦争タイプのイベント何件発生」等）も可能にします。

世界全体データ: WorldStateには現在の年月日時刻（ターン数から算出）、現在のターン番号、国リスト、グローバルイベントリストなどが含まれます。グローバルイベントとは特定の国に紐付かないイベント（例えば「巨大隕石が地球に接近したが逸れた」など）ですが、基本大半は国関連なのでevent.relatedCountriesが空かどうかで判定します。

AI応答データ形式: 開発段階でJSONスキーマを定義します。例えば:

```
{
 "countries": {
  "<国名>": {
   "population": 10000,
   "gdp": 530,
   "stability": 80,
   "...": "...",
   "note": "<国に関する出来事の要約>"
  },
  "別の国": { ... },
  ...
 },
 "events": [
  { "description": "国Aと国Bは自由貿易協定を締結した。", "related": ["国A","国B"], "type": "economic" },
  { "description": "国Cで大規模な地震が発生し被害が出た。", "related": ["国C"], "type": "disaster" }
 ]
}
```

といった構造です。countriesオブジェクト内に各国の新しい値を記載させ、events配列にイベントを列挙させます。実際には国名をキーにすると日本語キーでJSONパース面倒なので、国IDをキーにさせるか、あるいは配列で返させるかは調整します。上記スキーマをOpenAI APIのfunctionとして登録し、確実にキーや型を遵守させる予定です [17]。AIには各値がどの程度変化すべきかは明示しませんが、常識の範囲（例えばGDPが一度に倍増はしない等）はモデルの知識に委ねます。必要ならプロンプトで「変化は現実的な範囲に」と補足します。

AI活用設計

AI連携は本ゲームの鍵となる部分であり、その設計方針をまとめます。

- **使用モデル**: 現状、最有力は**OpenAI　GPT-4**モデルです。高い推論能力と複雑な指示への応答品質でゲームの要求に応えられると期待できます。コスト面ではAPI呼び出しごとにトークン料金が発生しますが、1ターンあたりのプロンプト・応答が数千トークン程度なら十分実用範囲と判断します（必要に応じてGPT-3.5　Turboなども検討）。将来的には**ファインチューニング**や**専用モデル**の学習も視野に入れますが、まずは汎用モデル＋プロンプト工夫で実装します。

- **プロンプトデザイン**: 先述の通り、プロンプトは世界の状態とアクションを説明し、出力形式を厳格に指定します。モデルに対しては「あなたは高度な世界シミュレーションAIです」というシステムメッセージで役割を与え、以下の点を指示します:

  - プレイヤーの行動を踏まえ、論理的かつ因果関係の通った予測を行うこと。
  - 結果はポジティブ・ネガティブ両面を検討し、偏らないこと（例: 常に成功ばかりでなくリスクも出す）。
  - 出力は指定のJSONフォーマットのみで行うこと（余計な説明やフォーマット外テキストは不要）。
  - 変化には適度な大きさ・期間を考慮すること（例: 1ターンで国が滅亡するような極端な変化は特殊な行動時のみ）。
  - 各国の文化・政体も考慮し、リアクションに反映すること（例: 民主国家は急進的な行動をとりにくい、宗教的国家は文化イベントで大きく士気上昇する等）。

プロンプト内では**直前数ターンの重要イベントも箇条書き**で与えます。こうすることで、モデルが過去の流れを忘れず**ストーリーの一貫性**を保てるようにします（類似研究でもエージェントに記憶を持たせることで社会的振る舞いの継続性が生まれています [25] [26]）。ただしプロンプトが肥大化しすぎるとトークン超過や応答遅延の問題が出るため、入れる履歴は直近数件に限定したり、要約して簡潔にします。

- **安定性のための措置**: LLMはオープンエンドな質問に対し時に想定外の回答をすることがあります。研究ではLLMは依然としてすべての状態変化を正しくシミュレートできるほどではないと指摘されています [4]。そこで、**システム補正**の仕組みを設けます。具体的には:
- **事前チェック**: プレイヤーの入力内容を解析し、明らかな無理筋（「太陽を爆破する」など極端な行動）はプロンプト内でトーンダウンさせます（あるいは「それはできません」とゲーム内で却下する）。
- **事後チェック**: AIの応答結果を検証し、明らかな矛盾（例: 国Aの人口が負の値になる、同じイベントが二重に記載される 等）は検知して修正 or 無視します。
- **リトライ**: 応答がフォーマットエラーの場合、自動で再度プロンプト送信（温度パラメータを下げる等の工夫も）して安定するまで試みます。OpenAI APIのfunction callingはほぼ確実にJSONを返すとされていますが [17]、念のための処理です。

- **ログ**: AIの出力とそれに対する結果適用を詳細にログに残し、デバッグしやすくします。予想外の出力はプロンプトの改善に役立てます。

- **AIレスポンス内容**: 数値の更新だけでなく、**簡潔な解説**も付けるか検討します。例えば各国データに `"note": "輸出が増え経済成長"` のような要約を入れさせると、それをUIでツールチップ表示してユーザーが理解しやすくできます。ただテキスト量が増えるため、まずはイベント記述に集約し、数値は無言で変わる形にします。イベント文はユーザー向けの自然文（日本語）なので、そのままUI表示に使います。

- **Smallville等からの示唆**: 先行するSmallville実験ではエージェントの自発的行動で社会が展開しました [1]。本ゲームはプレイヤー主導ですが、AIに周辺国の自発的アクションも起こさせることで世界の厚みが増します。例えば「他国同士が独自に戦争を始める」「第三国が技術革新する」といったイベントも折り込むのが望ましいです。これをプロンプトで「プレイヤー行動以外にも、世界には他の動きが起こり得る」ことを伝えることで、AIに包括的な変化を考えさせます。実際、マルチエージェントシミュレーションの試みでは**数千のAIエージェントが政策変更の波及効果をシミュレート**するプロジェクトもあるほどで [27]、本ゲームも将来的にはAI同士の相互作用をもっと増やしていく方向です。

- **コストと速度**: GPT-4 APIは高性能ですがコストが高く応答も数秒かかります。場合によっては**高速だがやや精度低いGPT-3.5**で下ごしらえ（下位モデルで大まかな結果を出し上位でブラッシュアップ）する方法も考えます。しかし複雑な世界モデルでは不正確な結果はゲーム性を損なうので、初期版は多少重くてもGPT-4一本で行きます。OpenAIの継続的なモデル改良や、他社の強力モデル（例: Anthropic Claude2等）もウォッチし、最適なものを採用します。

ゲームフロー例

ここでは実際のゲームプレイの流れを一例示し、設計の具体的なイメージを補強します。

ゲーム開始〜初期状態:
ユーザーがゲームを開始すると、バックエンドは10個の国家を生成します。例えば: - 国A: 人口8000万人, GDP 2.1兆, 民主国家, 資源: 工業, 文化: 自由主義, 同盟: 国Bと友好... - 国B: 人口1.5億人, GDP 5.5兆, 共産国家, 資源: 資源大国(石油), 文化: 集団主義, ... - ...（国C〜国J）

プレイヤーはこの中から国Aを担当するとします。画面には国Aの概要（中規模先進国・民主制・同盟あり等）が表示されます。他の国についても一覧や地図で存在を把握できます。

ターン1: プレイヤーはまず「経済成長を促すため、隣国Bと自由貿易協定を締結する」と入力しました。バックエンドは現在のA,B両国の経済規模や関係性をプロンプトに含めてAIに問い合わせます。AIは以下のような予測を返したとします（要約）: - 国A: GDP +3%、消費者物価 -1%、国民満足度 +5 上昇（安価な輸入品で物価安定と経済成長）。しかし一部産業が競争に晒され、失業率 +0.2%。 - 国B: GDP +2%、自国の輸出が増え潤う。両国関係 +10 好転。 - 国C（敵対国）: 国AとBの接近に警戒。関税を引き上げ経済を部分遮断。国Cの物価 +2%上昇、国C->A関係 -5 悪化。 - 他の国: 大きな影響なし。 - イベント: 「国Aと国Bは自由貿易協定を締結した（経済協力が強化）。」「国Cはこれに反発し、自国市場を保護するため関税障壁を強化した。」

ゲームはこの結果を受け取り、各国ステータスを更新します。国AのGDP値・満足度が上がり失業率微増、国BもGDP上昇、国Cはインフレ悪化等の変化を適用します。イベントログには上記2件が記録されます。プレイヤーには「自由貿易協定締結」のニュースとそれに続く「国Cが対抗策」のニュースが表示され、世界の反応を実感します。

ターン2: プレイヤーは次に「国Cの反発を抑えるため、国Cに外交使節を送り関係改善を図る」アクションを入力しました。しかし国Cは強硬な軍事独裁政権（文化: 強硬路線）であり、AIの予測は: - 国A: 外交団派遣により緊張緩和を図ったが効果は限定的。国民からは和平努力として評価され満足度+2。 - 国C: 表向きは使節団を受け入れたが態度は冷淡。関係改善 +2 程度とわずか。むしろ国C内部では「弱腰」との批判が政権に向けられ、政権安定度 -5。 - 国B他: 特筆すべき変化なし。 - イベント: 「国Aは国Cとの関係改善のため外交使節を派遣した。しかし国C政府は形式的に受け入れただけで、関係はほとんど改善しなかった。」

ログにこのイベントが追加されます。プレイヤーは国Cの反応が鈍いことを知り、別のアプローチ（例えば軍事圧力か、放置か）を検討するでしょう。また副次効果で国C政権の弱体化が起きたことから、今後国Cで内乱や政変の可能性も出てくるかもしれません（AIはこうした伏線も張る可能性があります）。

ターン3: プレイヤーが想定していない事態として、AIは内在イベントを発生させることもあります。例えば「国Dで大地震発生」のようなランダムイベントです。これはプレイヤー行動に関係なく突然起こりえます。この場合、AIはターン2の結果として以下も返したとします: - 国D: 大地震により甚大な被害。GDP -1%（インフラ破壊）、人口 -0.2%（死者）、国民幸福度 -10（一時的に混乱）。 - 周辺国: 人道支援イベント発生。国Aは支援金拠出（もしプレイヤー国がそういう性格なら自発的に）し、国A->D関係 +5。 - イベント: 「国DでM7.8の大地震が発生し、大きな被害が出た。他国は救援隊や援助金を送り始めている。」

このようにAIが判断して環境イベントを紛れ込ませることも想定します。ゲームとしては予期せぬイベントで緊張感が高まります。プレイヤーは緊急対応として「国Dに追加援助を送る」など新たな行動を選ぶかもしれません。

ターン4以降: 世界はプレイヤーの行動とAIのシミュレーションによって刻々と変化していきます。例えば戦争も起こり得ます。仮にプレイヤーが「国Cが軍備を増強しているとの情報を得た。先制攻撃を検討する」と入力すれば、AIは大きな戦争イベントを生成するでしょう。結果: - 国A: 国Cへの奇襲攻撃を実行。軍事力 -10（消耗）、国内世論は分裂（安定度 -20）。短期的に国Cの軍事施設破壊成功。 - 国C: 大きな被害を受け軍事力 -30。しかし直ちに報復を宣言。徴兵を開始し軍事力徐々に回復中。国Aへの敵対心最大に。 - 国B（同盟国）: 国Aを公式非難（民主国家なので先制攻撃を支持できず）。関係悪化 -20。 - 国E（国際社会）: 国連緊急会合が開かれ、国Aへの経済制裁案が検討され始める。 - イベント: 「突如、国A軍が国Cの軍事拠点に先制攻撃を実施。国Cは宣戦布告し報復を誓った。世界は緊張の度合いを急激に高めている。」

このようにかなり劇的な展開も起こせます。ただしプレイヤーには相応のリスクが返ってきます。ゲームとしてはここからさらに次のターンで和平交渉に動くか全面戦争か、といったドラマが続きます。

以上のフローは一例ですが、基本設計で描いたコンポーネントやモジュールがこれらを実現するよう連携します。重要なのは**プレイヤーの予想を上回るようなEmergentな出来事**が起きることで、ゲームに没入感と深みが生まれる点です [20]。AIのシミュレーションを介することで、そのような予測不能な展開も自然に発生させることができます。

UI/UXデザイン上のポイント

UX面で特に重視すべき点を補足します。

- **視覚的フィードバック**: 例えば経済協定を結んだ際には、国AとBの間に矢印（交易路）を地図上に表示し点滅させる、戦争なら爆発アイコンを国境に表示する、といった視覚効果があると理解が早まります。ただ初期はテキスト中心でも十分遊べるため、これらリッチな表現は徐々に追加します。
- **情報量の制御**: AIが生成するイベント文が長すぎる場合、ログ表示には最初の1文だけ載せ「続きを読む...」で詳細を隠す等します。短文ならそのまま表示。プレイヤーは自分に関係深いイベントだけ詳細を追えばよく、興味のない他国のニュースは読み飛ばせます。
- **入力の自由度とガイド**: 入力欄にはplaceholderで「例：隣国と科学技術協定を結ぶ」など表示しておきます。また、送信後にその入力が実際どう解釈されたかをAI結果に反映させる（例えばイベントとして「国A政府は科学技術協定提案を行った」等）ことで、ユーザーは自分の指示が認識されたことを確認できます。
- **誤入力のフォロー**: 万一タイプミスや曖昧表現でも、AIがそれなりに解釈してくれるはずですが、全く不明な場合はゲーム側で「その行動の意図を理解できませんでした。もう一度簡潔に入力してください。」といったメッセージを返します。これもUXとして必要です。
- **レスポンシブ**: スマホ縦持ちが主ですが、横持ちやタブレット、PCでも極端にレイアウトが崩れないようCSSグリッドやフレックスで調整します。メディアクエリでfontサイズやパネル配置を切り替え、特にPCでは横並びに要素を多く出して一覧性を高めます。
- **操作性**: タップ領域は指でも押しやすいよう十分な大きさを確保（ボタンや地図領域など）。スクロールも多用されるので、誤タップを避ける配置にします。リアルタイムアクションは無いので操作のタイミングで焦らせる要素はありませんが、**ターン終了**後に画面が自動更新されるので、ユーザーがログを読んでいる途中で次ターンボタンを押してしまうなど無いよう、結果表示完了後に次の入力を受け付ける設計にします。

技術要素と選定

フロントエンド: - フレームワーク: **React**を第一候補（開発者リソース豊富、SPA実績多数）。もしくはVue3（直感的だがReactに比べ人材や拡張で劣る場合あり）。チームのスキルに合わせ決定。 - 言語: **TypeScript**。バグ予防と保守性のため。また、データモデルをフロントと共有しやすくなる（バックエンドもTSなら型共通化可）。 - UIライブラリ: 必須ではないが、コンポーネントUIフレームワーク（Material-UIやAnt Design）を使うか検討。デザインカスタムしたければTailwind CSS + 自前コンポーネントでもOK。 - チャートライブラリ: 経済推移などグラフ表示に **Chart.js** や **ECharts** を検討。 - 地図表示: 簡易には国を矩形やリスト表示で代替するが、導入するなら **D3.js** か専用地図ライブラリ(Mapbox等は過剰か)を使う。 - その他: SPAルーティングにReact Router、状態管理にRedux Toolkit or Zustand、API通信にAxios等を予定。

バックエンド: - ランタイム: **Node.js 18+** (安定性と最新機能のため) with **Express** or **Fastify** (パフォーマンス重視なら後者)。 - 言語: こちらも **TypeScript**。フロントとの型共有で開発効率向上。 - 外部API: **OpenAI API** (GPT-4/GPT-3.5)。function calling/JSON modeを使用 [17]。APIキー管理にdotenvなど使用し、キーはサーバサイドのみ保持。 - データベース: 初期は**インメモリ**でOK。必要に応じて**Redis**（高速KVS）や**MongoDB**（ドキュメント指向DB）を導入。ゲームログなど分析したければSQL系も考えるが、柔軟なスキーマ変更考えるとNoSQLが良さそう。 - ホスティング: 小規模開始なら**Heroku**や**Railway**で手軽に。将来的にはAWS/GCP/Azureでスケール基盤に移行。 - AIモデルホスティング: オープンAI利用時は不要。ただ将来自前モデル使うならサーバにGPU環境が必要になるため、その時は別途設計（推論サーバ構築等）。

テスト: - 単体テスト: ビジネスロジック（GameState更新、AI応答パースなど）に**Jest**等でテストを書く。AI部分はモックを使い、想定JSONを返すようにしてゲーム進行が正しく行われるか検証。 - 統合テスト: シナリオを通したテスト。いくつか決め打ちのプレイヤー入力列と、それに対するAIモック結果で、ゲーム状態が意図通り変化するかを確認。 - AIの出力テスト: 実際にOpenAIにいくつかプロンプトを送り、返ってきたJSONが期待通りか、人が読んで妥当かをチェック。これはチューニングに活用。 - 負荷テスト: ターンを高速で100回回した場合メモリリークがないか等チェック。ユーザ数分の並列リクエストをシミュレートし、応答が許容範囲か確認。

今後の拡張・ロードマップ

基本設計を踏まえ、将来的に考えられる発展もまとめます（本要件定義の範囲を超えますが、視野に入れておきます）。

- **現実世界データとの連動**: 架空の国ではなく実在の国データ（人口や経済統計）を初期値に用いるモード。現実の地球をシミュレートし、例えば「自分が○○国の大統領になったら」という体験に。AIモデルには現実知識があるので対応しやすいが、政治的にセンシティブなため注意。
- **マルチエージェントの高度化**: 他国の挙動もすべてAIに委ねると混沌とする可能性があるので、ある程度ルールベースのAIを持たせる検討（例えば毎ターン各AI国家が目的関数に沿って最適行動をとり、それをプレイヤーと同様AIで結果算出するなど）。ただし本プロジェクトの醍醐味はLLMの自由なストーリー生成なので、バランスを見つつ。
- **マルチプレイヤー**: 複数プレイヤーがそれぞれ異なる国を担当し、同じ世界で競う/協力するモード。ターンごとに全員の行動入力を集め、一斉に結果反映する必要があり、ゲーム進行が同期型になる。システム的にもリアルタイム通信や同期処理の難易度が上がるため、中長期課題。
- **UIの充実**: 3D地球儀表示や詳細な地図表示、各国の新聞風イベント表示など、ビジュアル面の強化。特に**世界地図**は要望が高いと予想されるので、国数が増えれば避けられない追加。
- **シナリオエディタ**: ユーザが初期設定（国の数・特性、世界情勢）をカスタマイズできる機能。歴史上の状況を再現したり、極端な世界（全国家が独裁など）のシミュレーションなど遊びの幅が広がる。
- **チューニングと学習**: ユーザのプレイログ（どういう行動を入力し、AIが何を返し、ユーザはゲームを続けたか等）を分析し、AIモデルやプロンプトを改善。場合により専門領域特化のファインチューニングモデルを作成し使用する。例えば経済予測は得意だけど軍事は誤りがち等あれば、補強データを学習させる。
- **収益化**: サービス運営には収益も必要になるため、ゲーム内課金やプレミアム機能も検討。例えば無料版はターン数制限や簡易モデル使用、有料サブスクで無制限＆高性能モデル使用など差別化。または純粋に広告モデルで、ゲーム内にネイティブ広告（架空企業の広告等と絡めるのも面白い）を表示する案も。

まとめ

以上、サービス要件と基本設計を包括的に示しました。本ゲームは最先端の生成AI技術を取り入れることで、これまでにない自由度と深みを持った世界シミュレーション体験を提供しようとするものです。ターゲットユーザーが興味を持つであろう世界経済・国際関係の動きを、ゲームという形で楽しみつつ学べるよう設計しました。要件定義では自由入力や文化イベントなどユニークな特徴を盛り込み、基本設計ではそれを実現するシステム構成とデータフローを詳細に検討しています。

実装段階では、ここに挙げた設計指針を元にプロトタイプを構築し、AI応答の質やゲームバランスを検証しながら調整を繰り返すことになります。特にAIの挙動については未知数の部分もあるため、柔軟にプロンプトやロジックを改良し、ユーザーにとって納得感のあるシミュレーション結果が得られるようにすることが肝要です。

最後に、本プロジェクトを通じてユーザーが「世界を動かす面白さ」を存分に味わい、同時に現実の世界への洞察も深められるような、価値あるサービスとなることを目指します。そのためにチーム一丸となって開発を進めていきたいと思います。

参考文献・出典: 本書で言及した技術・市場データ・研究事例については以下を参照しています。

- Wangら (2023) 「Can Language Models Serve as Text-Based World Simulators?」: LLMを世界シミュレーションに用いる際の課題とJSON構造化出力による精度向上 [5] [28]
- OpenAI APIアップデート (2024) Structured Outputs: JSONスキーマに沿った信頼性の高い出力機能 [17]
- Stanford大学 (2023) 「Generative Agents: Interactive Simulacra」: 仮想町Smallvilleで25人のAIエージェントが自律的に社会生活を営んだ実験 [1] [2]
- Medium記事 (2025) 「AI Is Learning to Simulate Society...」: GPT系マルチエージェントによる社会シミュレーションの展望 [11] [27]
- モバイルゲーム市場統計 (2025) Udonis社ブログ: モバイルゲーマーの年齢層分布やプレイ傾向に関するデータ [7] [10]

---

[1] [3] [11] [12] [27] AI Is Learning to Simulate Society: Toward a New Era of Policy-by-Experiment | by Usable Service Design | May, 2025 | Medium
https://medium.com/@usable/ai-is-learning-to-simulate-society-toward-a-new-era-of-policy-by-experiment-406abd716e4b

[2] [20] [25] [26] Simulate the behavior of 25 AI agents in a virtual space city! | AI-SCHOLAR | AI: (Artificial Intelligence) Articles and technical information media
https://ai-scholar.tech/zh/articles/large-language-models/generative-agents

[4] [5] [6] [18] [19] [22] [23] [24] [28] Can Language Models Serve as Text-Based World Simulators?
https://arxiv.org/html/2406.06485v1

[7] [8] [9] [10] [13] [14] 200+ Mobile Gaming Market Statistics [2025 Report]
https://www.blog.udonis.co/mobile-marketing/mobile-games/mobile-gaming-statistics

[15] [16] Are SPAs (Single Page Applications) suitable for sites aimed for ...
https://stackoverflow.com/questions/20229862/are-spas-single-page-applications-suitable-for-sites-aimed-for-mobiles

[17] OpenAI’s Structured Outputs: A Developer’s Game-Changer | by Shobhit Agarwal | AI Advances
https://ai.gopubby.com/openais-structured-outputs-a-developer-s-game-changer-81e5204c7910

[21] The Main Challenges of the Single-Page Application (SPA) Model
https://trangotech.com/blog/challenges-with-single-page-application-model/