Projektplan: Vom Regelbuch zum Online-Spiel

Dies ist eine allgemeine Übersicht über die Schritte und Komponenten, die erforderlich sind, um ein komplexes Brettspiel wie "Shogun: Gekokujō" als Online-Multiplayer-Webanwendung umzusetzen.

1. Zusammenfassung

Das Ziel ist es, von einem statischen Regelbuch zu einer dynamischen, interaktiven Spielplattform zu gelangen. Dies erfordert zwei Hauptteile, die ständig miteinander kommunizieren:

1. **Das Frontend (Client):** Was jeder Spieler in seinem Browser sieht (das Spielbrett, die Einheiten, die Menüs).
2. **Das Backend (Server):** Ein zentraler "digitaler Spielleiter", der die Regeln durchsetzt, den Spielstatus verwaltet und die Züge an alle Spieler kommuniziert.

2. Die Kernkomponenten

A. Das Backend (Der "Server" oder "Digitale Spielleiter")

Dies ist die wichtigste Komponente für ein Multiplayer-Spiel. Es ist die "Quelle der Wahrheit".

- **Zweck:**
  - Verwaltung des zentralen Spielstatus (z. B. "Spieler A kontrolliert Provinz X", "Es ist Spieler B am Zug", "Einheit Y hat 3 Lebenspunkte").
  - Durchsetzung der Spiellogik (z. B. "Ist dieser Zug legal?", "Wie ist das Kampfergebnis?").
  - Verwaltung von Spiel-Sitzungen ("Lobbys") und Spieler-Authentifizierung.
- **Empfohlene Technologie:**
  - **Firebase Firestore:** Dies ist eine cloud-basierte NoSQL-Datenbank, die perfekt für diesen Zweck geeignet ist. Sie ist von Haus aus "persistent" (speichert Daten) und "real-time" (Echtzeit).
  - **Wie es funktioniert:** Wenn Spieler A eine Einheit bewegt, sendet sein Browser diese Aktion an das Backend. Das Backend prüft die Regel, aktualisiert den Spielstatus in der Firestore-Datenbank, und Firestore sendet diese Änderung *automatisch* und sofort an alle anderen Spieler in der Lobby. Sie sehen den Zug von Spieler A in Echtzeit.

B. Das Frontend (Die "Benutzeroberfläche")

Hier interagiert der Spieler mit dem Spiel. Ihre vorhandenen HTML/CSS-Dateien sind eine hervorragende *visuelle* Vorlage, aber die script.js-Datei müsste komplett neugeschrieben

werden.

- **Zweck:**
  - Anzeige des Spielbretts, der Einheiten, der Handkarten usw. (die "Ansicht").
  - Annahme von Benutzereingaben (z. B. Klicken auf eine Einheit, Ziehen einer Einheit auf eine Provinz).
  - Senden dieser Eingaben an das Backend zur Validierung.
  - Empfangen von Updates aus der Firestore-Datenbank und Aktualisierung der Ansicht (z. B. "Zeige die Bewegung der gegnerischen Einheit an").
- **Empfohlene Technologie:**
  - **HTML/CSS/JavaScript (wie bisher):** Sie können bei diesen Technologien bleiben. Die script.js-Datei wäre jedoch keine "Seiten-Navigation" mehr, sondern eine "Spiel-Engine", die:
    1. Eine Verbindung zu Firestore herstellt.
    2. Das Spielbrett dynamisch aufbaut (<canvas> oder -Elemente).
    3. Event-Listener für Klicks und Züge hinzufügt.
    4. Einen onSnapshot-Listener von Firestore verwendet, um Spiel-Updates in Echtzeit zu empfangen und das Brett neu zu zeichnen.

C. Die Spiellogik (Die "Regel-Engine")

Dies ist der Code, der Ihr Regelbuch digital abbildet. Diese Logik muss *primär auf dem Backend* leben, um Betrug zu verhindern.

- **Beispiele für Funktionen, die Sie schreiben müssten:**
  - function calculateIncome(playerId, gameState)
  - function resolveCombat(attackerUnits, defenderUnits, province)
  - function isValidMove(unitId, targetProvince, gameState)
  - function checkVictoryCondition(gameState)
- **Sicherheit:** Das Backend (z. B. mit Firebase Cloud Functions) muss jeden Zug validieren. Man kann sich nicht darauf verlassen, dass der Browser eines Spielers die Regeln korrekt befolgt.

3. Nächste Schritte (Vereinfacht)

1. **Architektur wählen:** Entscheiden Sie sich für ein Backend (z. B. Firebase Firestore).
2. **Datenmodell entwerfen:** Legen Sie fest, wie Sie den Spielstatus in der Datenbank speichern (z. B. eine games-Sammlung, die players-, provinces- und units-Objekte enthält).
3. **Frontend aufbauen (Schritt 1):** Erstellen Sie eine Webseite, die eine Verbindung zu Firestore herstellt und nur das Spielbrett *anzeigt*.
4. **Logik implementieren (Schritt 1):** Implementieren Sie die einfachste Aktion (z. B. eine Einheit von A nach B bewegen).
   - *Frontend:* Bei Klick, sende (unitId, targetProvince) an das Backend.
   - *Backend:* Empfange die Daten, prüfe isValidMove(), aktualisiere die Datenbank.

    - *Frontend (aller Spieler):* Der Echtzeit-Listener von Firestore wird ausgelöst, die Grafik wird aktualisiert.
5. **Iterieren:** Fügen Sie Regeln und Phasen nacheinander hinzu (Einkommen, Kampf, Siegbedingungen).

Dies ist ein komplexes, aber extrem lohnendes Softwareprojekt, das weit über eine statische Webseite hinausgeht. Während eine statische Webseite lediglich Informationen *darstellt*, muss diese Anwendung den gesamten Spielzustand dynamisch *verwalten* und in Echtzeit synchronisieren. Sie erwecken das Regelbuch zum Leben, indem Sie eine "digitale Spielleitung" (das Backend) erschaffen, die jede Regel validiert, jeden Zug verarbeitet und Betrug verhindert. Die Komplexität liegt in der Verwaltung dieses geteilten Zustands – wer ist am Zug, wem gehört welche Provinz, wie lautet das Kampfergebnis? Der lohnende Aspekt ist das Ergebnis: nicht nur ein Dokument, sondern eine lebendige, interaktive Plattform, auf der Spieler weltweit miteinander verbunden sind und das Spiel gemeinsam erleben können. Dies zu bauen ist eine erhebliche technische Errungenschaft, die ein tiefes Verständnis von Datenbanken, Anwendungslogik und Echtzeit-Kommunikation erfordert.