Project Title:

PlayFul AI

Team Name:

AI Alchemist

Team Members:

Team Leader Name : R Vinoothna
Team member 1 : Sanjana Koleti
Team member 2 : Pittala Maharshi Revanth
Team member 3 : Yashwitha Reddy Jaly

Phase-1: Brainstorming & Ideation

Objective:

Develop an AI-powered board game assistant that provides intelligent, adaptive gameplay and strategic guidance for various board games using Google Gemini API and reinforcement learning

Key Points:

1. **Problem Statement:**

- Many board game enthusiasts struggle to find suitable opponents or improve their strategies.
- Casual players need guidance on improving their gameplay.

2. **Proposed Solution:**

- An AI-powered assistant that adapts to different skill levels, providing an engaging gameplay experience.
- Offers strategic insights and guidance using machine learning and reinforcement learning.

3. **Target Users:**

- Board game players looking for challenging AI opponents.
- Casual players wanting to improve their strategic understanding.
- Digital game platforms seeking AI-driven opponents.

4. **Expected Outcome:**

- A functional AI assistant that enhances board game experiences by providing adaptive challenges and real-time strategy insights.

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for the GenAI Board Game Assistant.

Key Points:

1. **Technical Requirements:**

- Programming Language: Python
- Backend: Google Gemini API, LCZero AI
- Frontend: Tkinter for GUI
- Machine Learning: Reinforcement Learning, LCZero Weights

2. **Functional Requirements:**

- AI adapts to user skill levels dynamically.
- Provides strategic insights based on in-game performance.
- Supports multiple board games (e.g., Chess, Go, Checkers).

3. **Constraints & Challenges:**

- Training AI to balance difficulty without being unbeatable.
- Ensuring real-time feedback and recommendations.
- Optimizing AI response time for smooth gameplay**.**

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.

Key Points:

1. **System Architecture:**

- User selects a game and difficulty level.
- AI processes the game state and makes moves using ML models.
- AI provides real-time or post-game strategic feedback.

2. **User Flow:**

- Step 1: User selects a board game and AI difficulty.
- Step 2: AI opponent adapts to the user’s moves.
- Step 3: AI provides strategic recommendations after gameplay.

3. **UI/UX Considerations:**

- Simple, intuitive interface for game selection and AI customization.
- Interactive feedback system for learning and improvement.

Phase-4: Project Planning (Agile Methodologies)

Objective:

Break down development tasks for efficient completion.

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Environment Setup & API Integration	⍰ High	6 hours (Day 1)	End of Day 1	Member 1	Google API Key, Python, Tkinter setup	API connection established & working
Sprint 1	Frontend UI Development	⍰ Medium	2 hours (Day 1)	End of Day 1	Member 2	API response format finalized	Basic UI with input fields
Sprint 2	AI Opponent & Strategy Advisor Implementation	⍰ High	3 hours (Day 2)	Mid-Day 2	Member 1& 2	API response, UI elements ready	AI opponent making adaptive moves
Sprint 2	Error Handling & Debugging	⍰ High	1.5 hours (Day 2)	Mid-Day 2	Member 1&4	API logs, UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	⍰ Medium	1.5 hours (Day 2)	Mid-Day 2	Member 2& 3	API response, UI layout completed	Responsive UI, better user experience
Sprint 3	Final Presentation & Deployment	⍰ Low	1 hour (Day 2)	End of Day 2	Entire Team	Working prototype	Demo-ready project

Sprint Planning with Priorities

Sprint 1 – Setup & Integration (Day 1)

(⍰ High Priority) Set up the **environment** & install dependencies.
(⍰ High Priority) Integrate **Google Gemini API**.
(⍰ Medium Priority) Build a **basic UI with input fields**.

Sprint 2 – Core Features & Debugging (Day 2)

(⍰ High Priority) Implement AI opponent and strategy advisor functionalities.

(⍰ High Priority)Debug AI decision-making & handle errors in queries

Sprint 3 – Testing, Enhancements & Submission (Day 2)

(⍰ Medium Priority) Test API responses, refine UI, & fix UI bugs.
(⍰ Low Priority) Final **demo preparation & deployment**.

Phase-5: Project Development

Objective:

Implement core features of the AutoSage App.

Key Points:

1. **Technology Stack Used:**

- Frontend: Tkinter (Python GUI)
- Backend: Google Gemini API, LCZero AI
- Programming Language: Python

2. **Development Process:**

- Implement AI opponent logic using ML models.
- Integrate reinforcement learning for dynamic adaptation.
- Develop the GUI for seamless interaction.

3. **Challenges & Fixes:**

- Challenge: AI moves too fast or too slow.Fix: Adjust processing time for better user experience.
- Challenge: AI difficulty not well balanced.Fix: Fine-tune reinforcement learning parameters.

Phase-6: Functional & Performance Testing

Phase-6: Functional & Performance Testing
Phase-6: Functional & Performance Testing
Phase-6: Functional & Performance Testing
Phase-6: Functional & Performance Testing
Phase-6: Functional & Performance Testing
Phase-6: Functional & Performance Testing
Phase-6: Functional & Performance Testing
Phase-6: Functional & Performance Testing
Phase-6: Functional & Performance Testing

Phase-6: Functional & Performance Testing
Phase-6: Functional & Performance Testing

Objective:

Ensure that the AutoSage App works as expected.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	AI should adapt to different skill levels	AI adjusts difficulty dynamically	✅ Passed	Tester 1
TC-002	Functional Testing	AI provides strategic tips	Insights displayed after the game	✅ Passed	Tester 2
TC-003	Performance Testing	AI response time under 500ms.	Smooth gameplay experience	⚠ Needs Optimization	Tester 3
TC-004	Bug Fixes & Improvements	Fixed incorrect AI responses.	Data accuracy should be improved.	✅ Fixed	Developer
TC-005	Final Validation	Ensure UI is user-friendly.	Simple and intuitive UI	✕ Failed - UI broken on mobile	Tester 2
TC-006	Deployment Testing	Host the app using Streamlit Sharing	App should be accessible online.	⍰ Deployed	DevOps

---

Final Submission

1. **Project Report Based on the templates**
2. **Demo Video (3-5 Minutes)**
3. **GitHub/Code Repository Link**
4. **Presentation**