Hackathon Project Phases Template

Project Title:
Adaptive Chess Bot

Team Name:
NextGen Gamers

Team Members:
J. Hitesh
D. Rahul
V. Prashanth
B. Pandu Sagar
B. Kaushik

Phase-1: Brainstorming & Ideation

Objective:
To develop an AI-powered chess bot that not only acts as an opponent but also provides real-time, human-friendly move suggestions, helping players improve their strategy and skills while enjoying the game.

Problem Statement:

- Many chess players, especially beginners, struggle to improve because they lack guidance on move strategies.
- Existing chess bots are either too weak or too strong, making it difficult for players to get an appropriate challenge.
- There is a need for an adaptive AI that provides personalized difficulty and real-time insights into chess moves.

Proposed Solution:

- AI-powered chess bot that adapts to the player's skill level.
- Real-time move suggestions to help players understand better strategies.
- Adaptive difficulty settings that increase or decrease based on player performance.
- User-friendly interface with interactive chessboard and insights.

Target Users:

- Chess beginners looking to improve.
- Enthusiasts who want a challenging AI opponent.
- Anyone who enjoys playing chess and wants guidance on strategies.

Expected Outcome:
A fully functional AI chess bot that provides an engaging, educational, and interactive chess-playing experience with real-time advice.

Phase-2: Requirement Analysis

Technical Requirements:

- **Programming Language:** Python
- **Frontend:** Flask, HTML, CSS, JavaScript
- **Backend:** Stockfish Chess Engine
- **Libraries:** chess, flask, tkinter (for local testing), OpenAI API (for move explanations)

Functional Requirements:

- Interactive chessboard interface for smooth gameplay.
- AI-based adaptive difficulty settings.
- Real-time move suggestions for learning and strategy improvement.
- Undo/Redo moves functionality.
- Game history tracking for performance analysis.

Constraints & Challenges:

- Ensuring fast AI response times.
- Providing clear and human-friendly chess suggestions.
- Developing a simple, intuitive UI.

Phase-3: Project Design

System Architecture:

1. **User Interface Layer:** Displays the chessboard and moves.

2. **AI Engine Layer:** Uses Stockfish for move generation and decision-making.
3. **Advice Module:** Generates human-readable insights using OpenAI API.
4. **Database Layer:** Stores user moves and game history (optional).

User Flow:

1. User starts the game via a web-based interface.
2. Player makes a move on the chessboard.
3. AI responds with a move and provides an optional suggestion.
4. The game continues until checkmate, stalemate, or resignation.
5. User can view game history and performance stats.

UI/UX Considerations:

- Simple interface with an intuitive chessboard.
- Smooth performance across devices.
- Suggestion pop-ups that provide insights without distraction.

Phase-4: Project Planning (Agile Methodology)

Sprint Planning:

Sprint 1 – Setup & Integration (Day 1) ✅ Set up environment & install dependencies.
✅ Integrate Stockfish engine with Flask.
✅ Build a basic UI with a chessboard and move inputs.

Sprint 2 – Core Features & Debugging (Day 2) ✅ Implement AI opponent with adaptive difficulty.
✅ Develop the real-time move suggestion system.

Sprint 3 – Testing, Enhancements & Submission (Day 3) ✅ Test AI move accuracy and suggestion clarity.
✅ Optimize UI for better performance.
✅ Deploy the application for final testing.

Phase-5: Project Development

Technology Stack:

- **Frontend:** HTML, CSS, JavaScript (Flask-based UI)
- **Backend:** Flask (Python), Stockfish (AI Chess Engine)
- **APIs:** OpenAI GPT-4 API for chess insights

Development Process:

1. **Design & Planning:** Defined architecture and UI layout.
2. **Implementation:** Developed AI opponent, chessboard UI, and move suggestions.
3. **Testing & Debugging:** Optimized move accuracy and suggestion clarity.
4. **Deployment:** Hosted the chess bot on a web server.

Challenges & Fixes:

- **AI making weak moves?** → Tuned Stockfish settings for better difficulty adaptation.
- **Slow response time?** → Optimized AI calculations for faster suggestions.
- **Hard-to-understand move tips?** → Improved GPT-4 explanations for clarity.

Phase-6: Functional & Performance Testing

Test Cases:

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	User moves piece on board	Move registers correctly	✅ Passed	Tester 1
TC-002	Functional Testing	AI responds with a move	AI makes a valid move	✅ Passed	Tester 2
TC-003	Performance Testing	AI move response time under 500ms	AI move is generated quickly	⚠ Needs Optimization	Tester 3
TC-004	Functional Testing	Suggestion appears after move	Suggestion is relevant	✅ Passed	Tester 4

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-005	UI Testing	Chessboard scales on mobile	Board adjusts correctly	❌ Failed - Fixing	Tester 5

Final Submission Checklist

- ✅ **Project Report:** (This document)
- ✅ **Demo Video:** (3-5 min walkthrough)
- ✅ **GitHub Repository:** (Contains full source code)
- ✅ **Live Deployment:** (Hosted version of chess bot)
- ✅ **Presentation Slides:** (Explaining project features & challenges)
