PROJECT REPORT

Playful AI: Intelligent Board Game Opponents and Advisors

TEAM MEMBERS:

Ryaan Bansal-22BCE3164
Puneeth Velakaturi-22BCT0268
Archi dhoot-22BCI0086
Riya Prasant Koradwar-

SUBMITTED TO:
Smart InternZ Team

1.0 Problem Statement:

Board games are widely enjoyed for fun, strategy, and bonding. However, creating fresh narratives, characters, and settings for these games is often challenging and repetitive. This project aims to develop an AI-powered tool that generates unique, creative content for board games based on user input, making each session exciting and fresh.

<u>2.0 Requirements:</u>

2.1 Functional Requirements

- Accept user input in a prompt-based interface.
- Generate narrative content based on the prompt.
- Display AI-generated results in real-time.
- Provide sample content based on predefined game scenarios.

2.2 Non-Functional Requirements

- Should respond within 2–3 seconds.
- Must maintain a stable and secure connection with the Gemini API.
- Should be lightweight and easy to deploy (Web App using Streamlit).
- Ensure readability and formatting of the output for user experience.
- Minimal memory and CPU usage.

<u>3.0 Project Planning and Scheduling:</u>

Day	Activity
1	Planning & Setup
2	Backend Development
3	UI Development
4	Integration
5	Testing, Debugging & Deployment
6	Feedback & Documentation

4.0 Code Overview:

Pre-trained model program:

```
import streamlit as st
import google.generativeai as genai

api_key = "API KEY"//keep ur own key
genai.configure(api_key=api_key)
```

```
def create_model(model_name="gemini-1.5-
flash", temperature = 1, top_p = 0.95,
top_k = 64, max_output_tokens = 8192):
    generation_config = {
        "temperature": temperature,
        "top_p": top_p,
        "top_k": top_k,
        "max_output_tokens":
max_output_tokens,
        "response_mime_type":
"text/plain"
    }
    model = genai.GenerativeModel(
        model_name=model_name,
        generation_config=generation_conf
ig
    )

    return model
model = create_model()
if model:
    print(f"Model creation successful")
```

Project-playful-ai program:

- This is the main web application using Streamlit.
- Takes user input. Initializes a chat session with a defined history.
- Sends the prompt to Gemini.
- Outputs a response in real time on the page.

```python
import streamlit as st
import google.generativeai as genai

api_key = "API KEY"//keep ur own key
genai.configure(api_key=api_key)
def create_model(model_name="gemini-1.5-
flash", temperature = 1, top_p = 0.95,
top_k = 64, max_output_tokens = 8192):
    generation_config = {
        "temperature": temperature,
        "top_p": top_p,
        "top_k": top_k,
        "max_output_tokens":
max_output_tokens,
        "response_mime_type":
"text/plain"
    }
    model = genai.GenerativeModel(
        model_name=model_name,
        generation_config=generation_conf
ig
    )

    return model
# Function to start a chat session with
optional history
def start_chat_session(model, history):
    chat =
model.start_chat(history=history)
    return chat
# Function to send a message to the chat
session
```

```
def send_message(chat_session,
user_input):
    response =
chat_session.send_message(user_input)
    return response
# Streamlit app
def main():
    st.title("Playful game ai")
    # Input field for user text
    user_input = st.text_area("Enter your
prompt here:", height=150)
    # Define the initial chat history
    initial_history = [
        {
            "role": "user",
            "parts": [
                "write a unique narrative
characters, and scenarios based on the
given board games and give me 20 lines of
content.\n",
            ],
        },
        {
            "role": "model",
            "parts": [
                "## Board Games
Unleashed:\n\n**1. Clue: The Mystery of
the Missing Manuscript**\n\nThe esteemed
Professor Elmsworth has vanished, and his
priceless manuscript is missing. The
players must navigate through a series of
intricate puzzles and challenges to
uncover the truth behind the
disappearance. Each character has unique
abilities that can aid in solving the
mystery, but they must work together to
piece together the clues and find the
manuscript before time runs out.\n",
            ],
        },
    ]
    # Button to generate response
    if st.button("Generate"):
        if user_input:
            # Create the model
            model = create_model()
            # Start the chat session
            chat_session =
start_chat_session(model, initial_history)
            # Send the user input to the
model and get the response
            response =
send_message(chat_session, user_input)
            # Display the response
            st.subheader("Generated
Response:")
            st.write(response.text)
        else:
            st.warning("Please enter a
prompt to generate a response.")
if __name__ == "__main__":
    main()
```

Requirements.txt:

```
streamlit
google-generativeai
```

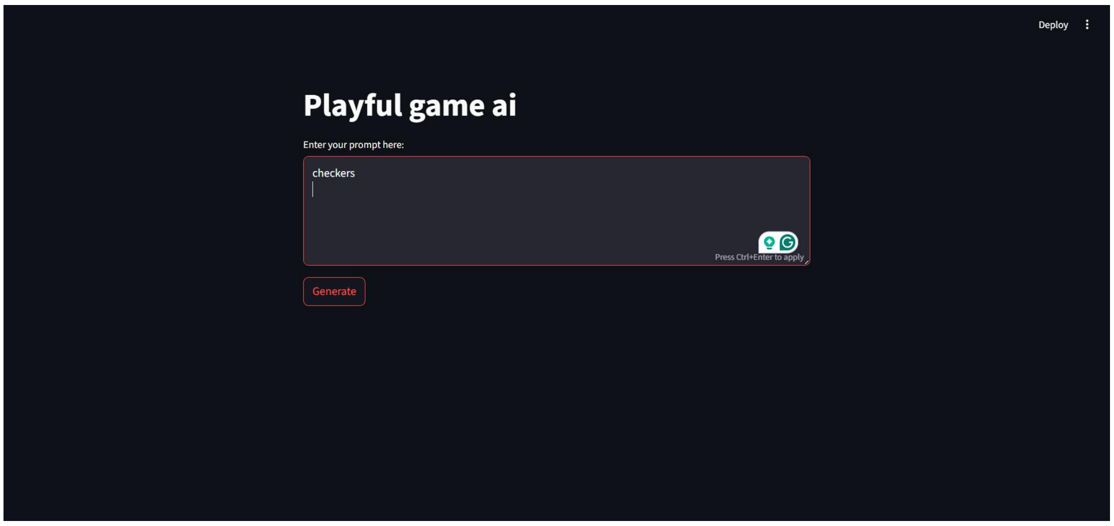
5.0 Testing:

The application was tested to ensure that:

- It produces relevant and coherent narratives.
- It handles various input cases robustly.
- The integration between the frontend (Streamlit) and back end (Gemini API) works seamlessly.

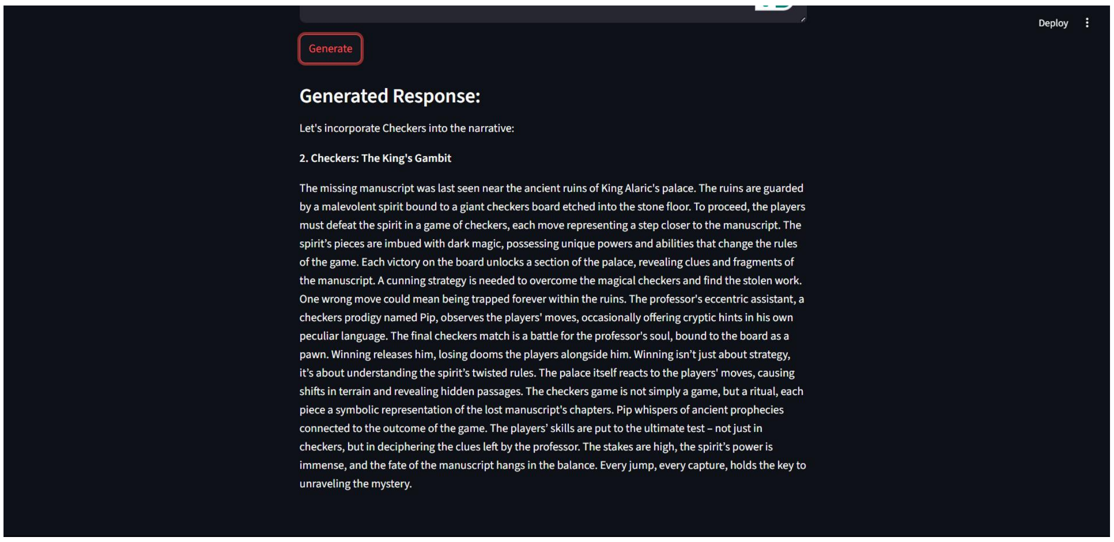
Screen-shots:

Input:



checkers

Output:



Generated Response:

Let's incorporate Checkers into the narrative:

2. Checkers: The King's Gambit

The missing manuscript was last seen near the ancient ruins of King Alaric's palace. The ruins are guarded by a malevolent spirit bound to a giant checkers board etched into the stone floor. To proceed, the players must defeat the spirit in a game of checkers, each move representing a step closer to the manuscript. The spirit's pieces are imbued with dark magic, possessing unique powers and abilities that change the rules of the game. Each victory on the board unlocks a section of the palace, revealing clues and fragments of the manuscript. A cunning strategy is needed to overcome the magical checkers and find the stolen work. One wrong move could mean being trapped forever within the ruins. The professor's eccentric assistant, a checkers prodigy named Pip, observes the players' moves, occasionally offering cryptic hints in his own peculiar language. The final checkers match is a battle for the professor's soul, bound to the board as a pawn. Winning releases him, losing dooms the players alongside him. Winning isn't just about strategy, it's about understanding the spirit's twisted rules. The palace itself reacts to the players' moves, causing shifts in terrain and revealing hidden passages. The checkers game is not simply a game, but a ritual, each piece a symbolic representation of the lost manuscript's chapters. Pip whispers of ancient prophecies connected to the outcome of the game. The players' skills are put to the ultimate test – not just in checkers, but in deciphering the clues left by the professor. The stakes are high, the spirit's power is immense, and the fate of the manuscript hangs in the balance. Every jump, every capture, holds the key to unraveling the mystery.

The programs are working perfectly fine.

6.0 Advantages:

- Enhances creativity in board games with zero manual effort.
- Lightweight and browser-based; no installation needed
- Uses Google's Gemini model—ensures quality text generation.
- Modular structure enables easy upgrades (themes, history, memory).

7.0 Disadvantages:

- Requires internet and API key (dependency on external service).
- Limited by the quota and pricing of the Google Generative AI.
- Cannot generate images or media—text-only application.
- Lacks game-specific rule customization (e.g., for RPG mechanics).

8.0 Conlcusion:

This project successfully integrates Generative AI with a simple web UI to produce creative board game narratives. It demonstrates how LLMs can augment traditional entertainment by generating personalized content. Future enhancements could include saving narratives, support for different genres, and exporting game scripts.