Project on:

AI Movie Dialogue Generator

Lovely Professional University

Prepared By:

Name: **Bhargav Naidu, Surendra.**

Regno: 12318438, 12319765.

Section: K23GF

ROLL.NO: 1, 2.

Submitted To: Dipen Saini

Table of Contents

1. Introduction

The **AI Movie Dialogue Generator** is a cutting-edge web application designed to empower scriptwriters, filmmakers, and creative enthusiasts by generating authentic, genre-specific movie dialogues. Leveraging advanced natural language processing (NLP) and a user-friendly interface, the system creates concise, emotionally resonant dialogues tailored to user inputs such as genre, characters, scene context, and tone. Unlike generic chatbots, this tool focuses on producing screenplay-ready outputs with minimal stage directions, mimicking professional movie scripts. The project showcases the integration of AI with creative writing, offering a scalable solution for rapid script development.

This report outlines the project's objectives, architecture, functionalities, and development process, providing a comprehensive overview of its implementation and potential impact in the creative industry.

2. Module-Wise Breakdown

The project is structured into the following core modules:

1. **User Input Module**
    - Captures user inputs for genre, character names, scene context, and dialogue tone.
    - Example inputs: "Thriller, Alex and Sophie, tense rooftop confrontation, dramatic tone."
    - Validates inputs to ensure clarity and relevance for AI processing.
2. **Prompt Construction Module**
    - Transforms user inputs into precise, structured prompts for the AI model.
    - Incorporates cinematic constraints (e.g., 1-2 lines per character, minimal stage directions).
    - Ensures genre-specific and emotionally appropriate dialogue generation.
3. **AI Dialogue Generation Module**
    - Sends prompts to an AI API (e.g., Gemini API) to generate movie-style dialogues.
    - Processes and formats AI responses into a clean, script-like structure with character names and brief stage directions.
4. **API Integration Module**
    - Handles API authentication, error management, and response retrieval.
    - Ensures reliable communication with the AI model via secure API calls.
5. **Frontend Interface Module**
    - Delivers a modern, responsive chatbot interface using HTML, Tailwind CSS, and JavaScript.
    - Allows users to input details, view generated dialogues, and modify or regenerate outputs.
6. **Response Display Module**

- Presents AI-generated dialogues in a formatted, screenplay-style layout.
- Offers options to “Regenerate Dialogue,” “Edit Inputs,” or “Save Script.”

7. **Dialogue Enhancement Module (Optional/Future Enhancement)**
   - Suggests alternative dialogue variations based on tone or pacing adjustments.
   - Provides genre-specific tips or references to iconic movie scripts.

3. System Architecture

The system is modular, comprising interconnected components that handle user inputs, AI processing, and output formatting. The architecture is divided into the following modules:

1. **Input Processing Module**
   - Captures user inputs (genre, characters, scene, tone) via a web form.
   - Validates inputs to ensure clarity and compatibility with AI processing.

2. **Prompt Engineering Module**
   - Converts user inputs into structured prompts optimized for AI models.
   - Incorporates cinematic constraints (e.g., brevity, emotional intensity).
3. **AI Dialogue Generation Module**
   - Interfaces with an AI API (e.g., Gemini API) to generate dialogues.
   - Parses AI responses into a screenplay-like format.
4. **API Management Module**
   - Manages secure API calls, error handling, and response caching.
5. **Frontend Interface Module**
   - Renders a dynamic, responsive UI using React and Tailwind CSS.
   - Displays dialogues and provides options for regeneration or export.
6. **Output Formatting Module**
   - Formats dialogues with character names and minimal stage directions.
   - Supports export to text or screenplay software formats.
7. **Analytics Module (Proposed)**

- Tracks user interactions and dialogue preferences for future personalization.

4. Core Functionalities

The AI Movie Dialogue Generator offers the following features:

1. **Dynamic Dialogue Generation**
   - Produces 2-4 line exchanges tailored to user-specified genres and scenes.
   - Example: A tense sci-fi confrontation or a heartfelt romantic confession.
2. **Customizable Parameters**
   - Users can define genre, character names, scene context, and emotional tone.
3. **Real-Time Preview**
   - Displays generated dialogues instantly in a screenplay format.
4. **Regeneration and Editing**
   - Allows users to tweak inputs or request alternative dialogues.
5. **Export Capabilities**
   - Supports copying dialogues or exporting as text/PDF for screenwriting tools.
6. **Responsive Design**
   - Ensures accessibility across devices (desktop, tablet, mobile).
7. **Error Handling**
   - Provides user-friendly error messages for invalid inputs or API issues.
8. **Genre-Specific Tips**
   - Offers suggestions for improving dialogues based on genre conventions (e.g., punchy one-liners for action films).

5. Technology Stack

The project leverages modern web technologies and AI tools to deliver a robust solution.

Programming Languages

- **HTML5**: Structures the web interface.
- **JavaScript (ES6+)**: Handles logic, API integration, and dynamic rendering.
- **CSS**: Styles the UI via Tailwind CSS.

AI and Integration Tools

- **Gemini API**: Generates dialogues using advanced NLP capabilities.

Development Tools

- **Visual Studio Code**: Primary IDE for coding and debugging.

6. Workflow Diagram

The workflow of the AI Movie Dialogue Generator is as follows:

[User Interface]

↓

[User Inputs: Genre, Characters, Scene, Tone]

↓

[React App Validates Inputs]

↓

[Prompt Engine Constructs AI-Compatible Prompt]

↓

[Axios Sends Prompt to Zapier Webhook]

↓

[Zapier Triggers Gemini API]

↓

[Gemini API Generates Dialogue]

↓

[Response Returned to React App]

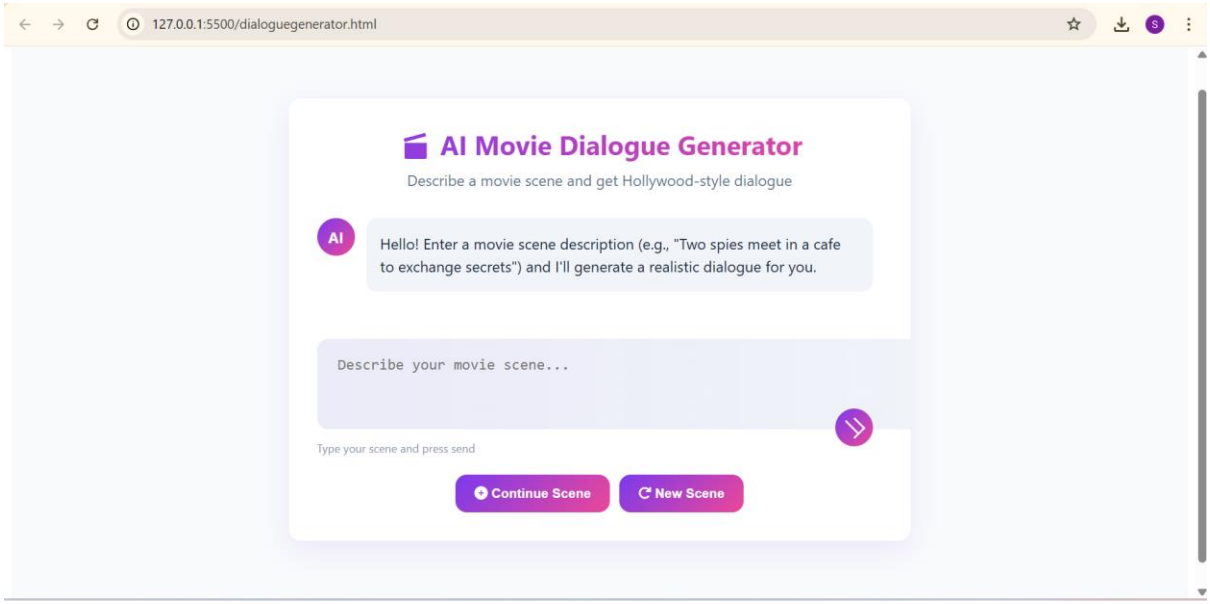
↓

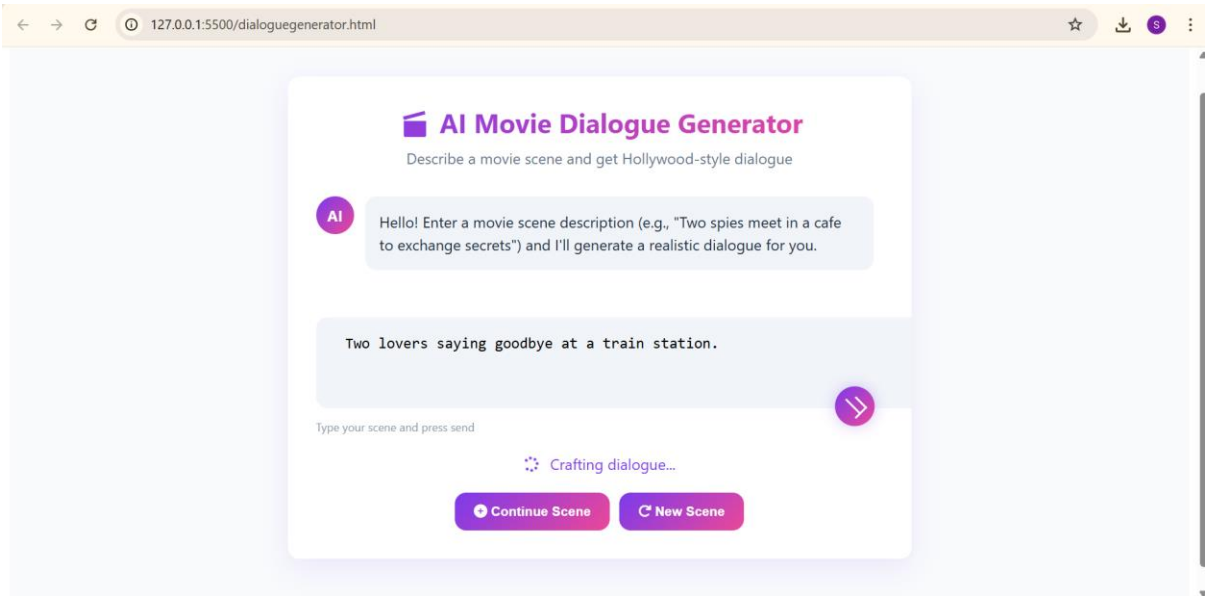
[UI Displays Formatted Dialogue]

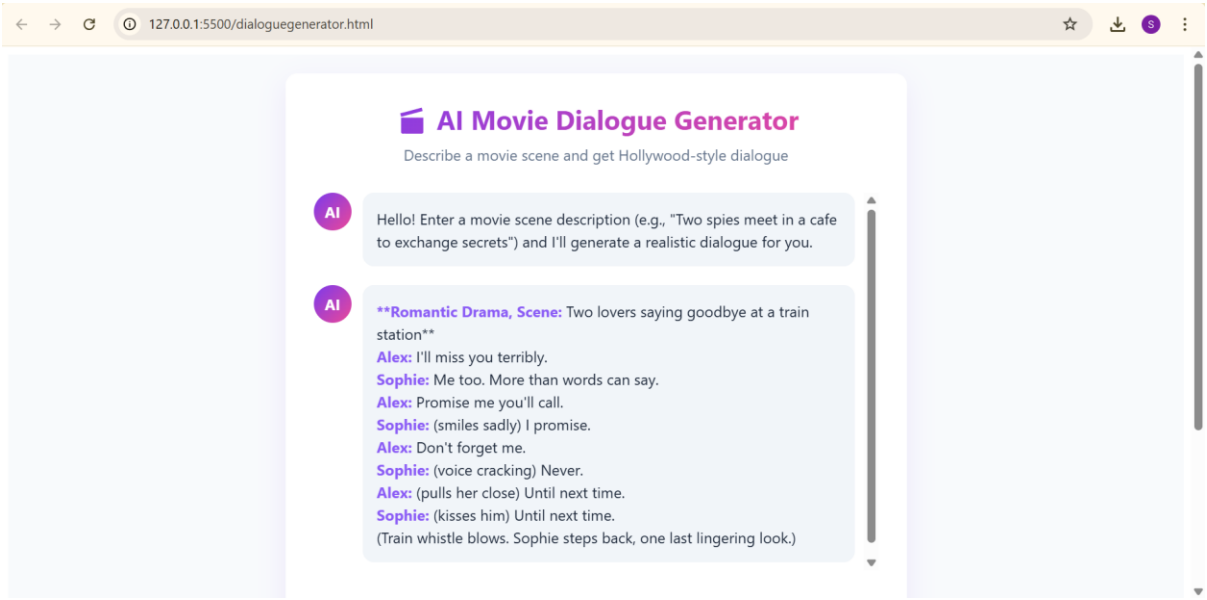
↓

[User Options: Regenerate, Edit, or Export]

Diagrams:







7. Development Process

The project followed an agile development methodology with the following phases:

1. **Requirement Analysis**
    - Identified user needs (e.g., genre flexibility, professional output).
    - Defined technical requirements (e.g., API integration, responsive UI).

2. **Prototyping**
    - Created wireframes in Figma to design the UI.
    - Tested initial API calls using Postman.
3. **Implementation**
    - Developed the frontend using React and Tailwind CSS.
    - Integrated Gemini API via Zapier for dialogue generation.
4. **Testing**
    - Conducted unit tests for input validation and API responses.
    - Performed user acceptance testing to ensure usability.
5. **Deployment**
    - Hosted the application on Netlify for public access.
    - Maintained code on GitHub for version control.

8. Challenges and Solutions

Challenge 1: Inconsistent AI Outputs

- **Issue**: The Gemini API occasionally produced verbose or off-tone dialogues.
- **Solution**: Refined prompt engineering to include strict constraints (e.g., "2-line max per character, no excessive stage directions").

Challenge 2: Responsive UI Across Devices

- **Issue**: Initial UI layouts broke on smaller screens.
- **Solution**: Adopted Tailwind CSS's mobile-first approach and tested on multiple devices using Chrome DevTools.

Challenge 3: API Rate Limits

- **Issue**: Frequent API calls triggered rate limits during testing.
- **Solution**: Implemented response caching and retry logic using Axios interceptors.

9. GitHub Repository Details

The project's codebase is hosted on GitHub for version control and transparency.

- **Repository Name**: Ai-movie-dialogue-generator
- **GitHub Link**: https://github.com/KUNTIMADDIBHARGAVANAIDU/Ai-movie-dialoue-genrerator

10. Conclusion and Future Enhancements

Conclusion

The AI Movie Dialogue Generator successfully bridges AI technology and creative writing, delivering a practical tool for generating professional movie dialogues. Its modular architecture, powered by React, Tailwind CSS, and Gemini API, ensures scalability and ease of use. The project demonstrates the potential of NLP in creative industries, reducing the time and expertise required for scriptwriting while maintaining high-quality outputs.

Future Enhancements

- **Voice-Activated Input**: Integrate Web Speech API for hands-free scene descriptions.
- **Dialogue Feedback System**: Implement AI-driven analysis of dialogue quality (e.g., pacing, authenticity).
- **Multi-User Collaboration**: Enable real-time scriptwriting sessions for teams.
- **Integration with Screenwriting Tools**: Support direct exports to Final Draft or Celtx.
- **Personalized Recommendations**: Use analytics to suggest genre-specific dialogue tweaks.

11. References

- Gemini API Documentation
- React Documentation: https://reactjs.org/docs
- Tailwind CSS Documentation: https://tailwindcss.com/docs
- GitHub Guides: https://docs.github.com/en/get-started
- Netlify Deployment Guide: https://docs.netlify.com

12. Source Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AI Movie Dialogue Generator</title>
    <style>
        .ai-chat-widget {
            --ai-primary: #7c3aed;
            --ai-primary-light: #8b5cf6;
            --ai-primary-dark: #6d28d9;
            --ai-secondary: #ec4899;
            --ai-bg-color: #f8fafc;
            --ai-text-color: #1e293b;
            --ai-border-radius: 12px;
            --ai-shadow: 0 10px 30px rgba(124, 58, 237, 0.1);
            --ai-transition: all 0.3s ease;
            font-family: "Inter", -apple-system, BlinkMacSystemFont, "Segoe
UI", Roboto, sans-serif;
            background-color: var(--ai-bg-color);
            color: var(--ai-text-color);
            display: flex;
            justify-content: center;
            align-items: center;
            padding: 20px;
            min-height: 100vh;
            margin: 0;
        }
        .ai-chat-container {
            width: 100%;
            max-width: 600px;
            background: white;
            border-radius: var(--ai-border-radius);
            box-shadow: var(--ai-shadow);
            overflow: hidden;
```

```
            padding: 30px;
        }
        .ai-chat-header {
            text-align: center;
            margin-bottom: 30px;
        }
        .ai-chat-title {
            font-size: 28px;
            font-weight: 700;
            margin-bottom: 8px;
            background: linear-gradient(135deg, var(--ai-primary), var(--ai-
secondary));
            -webkit-background-clip: text;
            -webkit-text-fill-color: transparent;
            background-clip: text;
        }
        .ai-chat-subtitle {
            color: #64748b;
            font-size: 16px;
        }
        .ai-chat-messages {
            margin-bottom: 30px;
            max-height: 400px;
            overflow-y: auto;
            padding-right: 10px;
        }
        .ai-message {
            display: flex;
            margin-bottom: 20px;
        }
        .ai-avatar {
            width: 40px;
            height: 40px;
            border-radius: 50%;
            background: linear-gradient(135deg, var(--ai-primary), var(--ai-
secondary));
            display: flex;
            align-items: center;
            justify-content: center;
            color: white;
            font-weight: bold;
            margin-right: 12px;
            flex-shrink: 0;
        }
        .ai-message-content {
            background-color: #f1f5f9;
            padding: 15px;
            border-radius: var(--ai-border-radius);
```

```
            flex-grow: 1;
            line-height: 1.5;
        }
        .ai-input-container {
            position: relative;
            margin-top: 20px;
        }
        .ai-input-field {
            width: 100%;
            padding: 16px 60px 16px 20px;
            border: 2px solid transparent;
            border-radius: var(--ai-border-radius);
            background-color: #f1f5f9;
            font-size: 16px;
            resize: none;
            outline: none;
            transition: var(--ai-transition);
            min-height: 60px;
        }
        .ai-input-field:focus {
            border-color: var(--ai-primary-light);
            box-shadow: 0 0 0 4px rgba(124, 58, 237, 0.1);
        }
        .ai-send-button {
            position: absolute;
            right: 10px;
            bottom: 10px;
            width: 40px;
            height: 40px;
            border-radius: 50%;
            background: linear-gradient(135deg, var(--ai-primary), var(--ai-
secondary));
            border: none;
            color: white;
            display: flex;
            align-items: center;
            justify-content: center;
            cursor: pointer;
            transition: var(--ai-transition);
            outline: none;
        }
        .ai-send-button:hover {
            transform: scale(1.05);
            box-shadow: 0 0 15px rgba(124, 58, 237, 0.3);
        }
        .ai-send-button:active {
            transform: scale(0.95);
        }
```

```
        .ai-send-icon {
            position: relative;
            width: 14px;
            height: 14px;
        }
        .ai-send-icon:before {
            content: "";
            position: absolute;
            top: 0;
            left: 0;
            width: 14px;
            height: 14px;
            border-top: 2px solid white;
            border-right: 2px solid white;
            transform: rotate(45deg);
        }
        .ai-send-icon:after {
            content: "";
            position: absolute;
            top: 50%;
            left: -5px;
            width: 20px;
            height: 2px;
            background-color: white;
            transform: translateY(-50%) rotate(45deg);
        }
        .ai-input-info {
            display: flex;
            justify-content: space-between;
            margin-top: 8px;
            font-size: 12px;
            color: #64748b;
        }
        .ai-input-hint {
            opacity: 0.8;
        }
        @keyframes ai-shimmer {
            0% {
                background-position: -100% 0;
            }
            100% {
                background-position: 100% 0;
            }
        }
        .ai-input-field:placeholder-shown {
            background-image: linear-gradient(
                90deg,
                rgba(124, 58, 237, 0) 0%,
```

```
                rgba(124, 58, 237, 0.05) 50%,
                rgba(124, 58, 237, 0) 100%
            );
            background-size: 200% 100%;
            animation: ai-shimmer 3s infinite;
        }
        .ai-button-group {
            display: flex;
            justify-content: center;
            gap: 10px;
            margin-top: 20px;
        }
        .ai-action-button {
            padding: 12px 20px;
            border-radius: var(--ai-border-radius);
            background: linear-gradient(135deg, var(--ai-primary), var(--ai-
secondary));
            color: white;
            font-size: 14px;
            font-weight: 600;
            border: none;
            cursor: pointer;
            transition: var(--ai-transition);
        }
        .ai-action-button:hover {
            transform: scale(1.05);
            box-shadow: 0 0 15px rgba(124, 58, 237, 0.3);
        }
        .ai-action-button:active {
            transform: scale(0.95);
        }
        .ai-loading {
            display: none;
            text-align: center;
            font-size: 16px;
            color: var(--ai-primary);
            margin: 20px 0;
        }
        .ai-loading i {
            margin-right: 8px;
        }
        .dialogue-line strong {
            color: var(--ai-primary-light);
        }
        .dialogue-line span {
            color: var(--ai-text-color);
        }
        @media (max-width: 640px) {
```

```
            .ai-chat-container {
                padding: 20px;
            }

            .ai-chat-title {
                font-size: 24px;
            }

            .ai-input-field {
                padding: 14px 50px 14px 16px;
            }
        }
    </style>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
</head>
<body>
    <div class="ai-chat-widget">
        <div class="ai-chat-container">
            <div class="ai-chat-header">
                <div class="ai-chat-title">🎬 AI Movie Dialogue Generator</div>
                <div class="ai-chat-subtitle">Describe a movie scene and get Hollywood-style dialogue</div>
            </div>

            <div class="ai-chat-messages">
                <div class="ai-message">
                    <div class="ai-avatar">AI</div>
                    <div class="ai-message-content">
                        Hello! Enter a movie scene description (e.g., "Two spies meet in a cafe to exchange secrets") and I'll generate a realistic dialogue for you.
                    </div>
                </div>
            </div>
            <div class="ai-input-container">
                <textarea class="ai-input-field" id="situation" placeholder="Describe your movie scene..."></textarea>
                <button class="ai-send-button" onclick="generateDialogue()">
                    <div class="ai-send-icon"></div>
                </button>
                <div class="ai-input-info">
                    <span class="ai-input-hint">Type your scene and press send</span>
                </div>
            </div>
            <p id="loading" class="ai-loading">
```

```
                <i class="fas fa-spinner fa-spin"></i> Crafting dialogue...
            </p>
            <div class="ai-button-group">
                <button class="ai-action-button" onclick="continueDialogue()">
                    <i class="fas fa-plus-circle"></i> Continue Scene
                </button>
                <button class="ai-action-button" onclick="newScene()">
                    <i class="fas fa-redo"></i> New Scene
                </button>
            </div>
        </div>
    </div>
    <script>
        // Gemini API configuration
        const API_URL =
'https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-
flash:generateContent';
        const API_KEY = 'AIzaSyD5fpkAyZfzLfs6XRpDWE_5OUpWVWbtBjM'; // WARNING:
Exposed in client-side code
        let previousDialogue = '';

        async function generateDialogue() {
          const situation = document.getElementById('situation').value.trim();
          if (!situation) {
            alert('Please enter a scene description!');
            return;
          }
          document.getElementById('loading').style.display = 'block';
          const messagesContainer = document.querySelector('.ai-chat-
messages');
          try {
            // Construct prompt for concise movie-style dialogue with stage
directions
            let prompt = `Generate a movie-style dialogue script for this
situation:\n\n${situation}\n\n`;
            if (previousDialogue) {
              prompt += `Continue the dialogue script, maintaining the same
characters and style:\n${previousDialogue}\n\n`;
            }
            prompt += `The dialogue must be formatted as a script with
character names (e.g., Alex, Sophie) followed by a colon, and their spoken
lines limited to 1-2 short sentences. Include brief stage directions in
parentheses (e.g., (teary-eyed), (whispers)) where appropriate to enhance the
scene, but keep them minimal. Ensure the dialogue is realistic, engaging, and
matches the genre and tone of the situation. Avoid lengthy exchanges or
excessive narrative descriptions. Example output for "Romantic Drama, Scene:
Two lovers saying goodbye at a train station":
    Alex: "Stay. Just for one more minute."
```

```
    Sophie: (teary-eyed) "If I stay one more minute, I'll never leave."
    Alex: (whispers) "Then don't."
    (Train whistle blows, Sophie slowly steps back, eyes locked with his.)`;

            // Prepare request body
            const requestBody = {
              contents: [
                {
                  parts: [
                    { text: prompt }
                  ]
                }
              ]
            };

            // Send request to Gemini API
            const response = await fetch(`${API_URL}?key=${API_KEY}`, {
              method: 'POST',
              headers: { 'Content-Type': 'application/json' },
              body: JSON.stringify(requestBody)
            });

            if (!response.ok) {
              throw new Error(`HTTP error! status: ${response.status}`);
            }

            const data = await response.json();
            const generatedText =
data?.candidates?.[0]?.content?.parts?.[0]?.text?.trim();
            if (!generatedText) {
              throw new Error('No valid response from Gemini API');
            }

            // Store the dialogue for continuation
            previousDialogue = generatedText;

            // Format the dialogue with character names and stage directions
            const formattedDialogue = generatedText.split('\n').map(line => {
              if (line.includes(':')) {
                const parts = line.split(':');
                const character = parts[0].trim();
                const characterClass = character === 'Character A' ?
'character-a' : character === 'Character B' ? 'character-b' : '';
                return `<div class="dialogue-line"><strong
class="${characterClass}">${parts[0]}:</strong>
<span>${parts.slice(1).join(':').trim()}</span></div>`;
              } else if (line.trim().startsWith('(') &&
line.trim().endsWith(')')) {
```

```
                return `<div class="dialogue-line stage-
direction">${line}</div>`;
              }
              return `<div class="dialogue-line">${line}</div>`;
            }).join('');

            // Append the dialogue as a new message
            const newMessage = document.createElement('div');
            newMessage.classList.add('ai-message');
            newMessage.innerHTML = `
              <div class="ai-avatar">AI</div>
              <div class="ai-message-content">${formattedDialogue}</div>
            `;
            messagesContainer.appendChild(newMessage);

            // Scroll to the bottom
            messagesContainer.scrollTop = messagesContainer.scrollHeight;
          } catch (error) {
            console.error('Error generating dialogue:', error);
            const errorMessage = document.createElement('div');
            errorMessage.classList.add('ai-message');
            errorMessage.innerHTML = `
              <div class="ai-avatar">AI</div>
              <div class="ai-message-content" style="color: #dc2626;">
                Error: Could not generate dialogue: ${error.message}. Please
try again.
              </div>
            `;
            messagesContainer.appendChild(errorMessage);
            messagesContainer.scrollTop = messagesContainer.scrollHeight;
          } finally {
            document.getElementById('loading').style.display = 'none';
            document.getElementById('situation').value = ''; // Clear input
          }
        }

        function continueDialogue() {
          if (!previousDialogue) {
            alert('No dialogue to continue! Generate some dialogue first.');
            return;
          }
          generateDialogue();
        }

        function newScene() {
          previousDialogue = '';
          const messagesContainer = document.querySelector('.ai-chat-
messages');
```

```
          messagesContainer.innerHTML = `
            <div class="ai-message">
              <div class="ai-avatar">AI</div>
              <div class="ai-message-content">
                Hello! Enter a movie scene description (e.g., "Romantic Drama,
Scene: Two lovers saying goodbye at a train station") and I'll generate a
realistic movie-style dialogue for you.
              </div>
            </div>
          `;
          document.getElementById('situation').value = '';
        }

        // Allow Enter key to trigger dialogue generation
        document.getElementById('situation').addEventListener('keydown', (e)
=> {
          if (e.key === 'Enter' && !e.shiftKey) {
            e.preventDefault();
            generateDialogue();
          }
        });
      </script>
</body>
</html>
```

-------THE END------