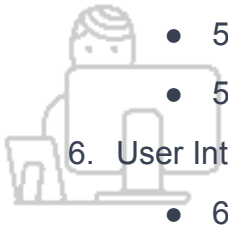
Personalized AI Storyteller Project

Table of Contents



1. Executive Summary

The Personalized AI Storyteller project is a web-based application designed to generate personalized stories based on user inputs regarding their feelings, character names, and character genders. Utilizing natural language processing (NLP) capabilities from the Cohere API and sentiment analysis through the TextBlob library, the application aims to create engaging narratives that resonate with users' emotions. The project also incorporates audio features for an enhanced user experience, including background music and sound effects corresponding to the generated story's mood.

2. Project Objectives

1. Story Generation: To develop an AI-driven story generation mechanism that creates personalized stories based on user input.
2. Sentiment Analysis: To analyze user feelings through sentiment analysis to influence the generated story's mood.
3. User Engagement: To enhance user experience through interactive elements, including audio feedback and a visually appealing interface.
4. Accessibility: To ensure the application is user-friendly and accessible to a wide range of users.

3. Introduction

3.1 Background

Storytelling has been an integral part of human culture, serving as a medium for sharing experiences, morals, and entertainment. With the advent of technology, the way stories are created and consumed has evolved. The Personalized AI Storyteller project aims to

harness the power of artificial intelligence to provide users with a unique storytelling experience tailored to their emotions.

3.2 Problem Statement

In an era where digital content is abundant, users often seek personalized experiences that resonate with their individual feelings and preferences. Traditional storytelling methods may not cater to this demand, leading to a gap in the market for interactive and personalized storytelling applications.

3.3 Significance of the Project

This project addresses the need for personalized storytelling by leveraging AI and sentiment analysis. By allowing users to input their feelings and preferences, the application creates a unique narrative that not only entertains but also connects with users on an emotional level.

4. Technical Overview

4.1 Technologies Used

- Backend:
    - Flask: A lightweight WSGI web application framework in Python, used to handle web requests and responses.
    - Cohere API: A natural language processing service used to generate text based on prompts.
    - TextBlob: A Python library for processing textual data, used for sentiment analysis.
- Frontend:

    - HTML/CSS: For structuring and styling the web application.
    - JavaScript: For handling user interactions and making asynchronous requests to the backend.
- Audio:
    - Various audio files (e.g., happy, sad, neutral sounds) are used to provide auditory feedback based on the mood of the generated story.

4.2 Architecture

The application follows a Model-View-Controller (MVC) architecture, where:

- Model: The backend logic for story generation and sentiment analysis.
- View: The HTML/CSS frontend that presents the user interface.
- Controller: The Flask routes that handle user input and responses.

5. Implementation Details

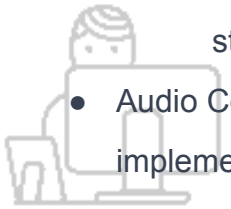
5.1 Backend Implementation

- Flask Application Structure: The application is structured with routes to handle different functionalities:
    - `/`: The main route that renders the index page.
    - `/generate_story`: A POST route that accepts user input, performs sentiment analysis, and generates a story.
- Story Generation Function: The `generate_story` function constructs a prompt based on user input and calls the Cohere API to generate a story.

- Sentiment Analysis Function: The `get_sentiment` function uses TextBlob to analyze the polarity of the user input and classify it into happy, sad, or neutral categories.

5.2 Frontend Implementation

- User Interface: The interface is designed to be intuitive, with clear prompts for user input and buttons for interaction.
- JavaScript for Interactivity: The `game.js` file manages user interactions, including:
    - Handling button clicks to generate stories.
    - Validating user input to ensure all fields are filled.
    - Fetching data from the backend and updating the UI with the generated story and mood.
- Audio Controls: Volume controls for background music and sound effects are implemented to enhance the user experience.



5.3 Audio Integration

- Audio Elements: The application includes audio elements for background music and mood-based sound effects, enhancing the storytelling experience.
- Volume Control: Users can adjust the volume of background music and sound effects through sliders.

6. User Interface Design

6.1 User Experience Considerations

The user interface is designed with user experience in mind, ensuring that it is intuitive and easy to navigate. Key considerations include:

- Accessibility: The application is designed to be accessible to users with varying levels of technical expertise.
- Feedback Mechanisms: Users receive immediate feedback on their inputs, enhancing engagement.

6.2 Visual Design

The visual design incorporates a modern aesthetic with a focus on readability and user engagement. Key elements include:

- Color Scheme: A visually appealing color palette that enhances the storytelling experience.
- Responsive Design: The application is designed to be responsive, ensuring usability across different devices.

7. Testing and Quality Assurance

7.1 Testing Methodology

- Unit Testing: Individual functions, such as `generate_story` and `get_sentiment`, were tested for expected outputs based on sample inputs.
- Integration Testing: The interaction between the frontend and backend was tested to ensure data was correctly sent and received.
- User Acceptance Testing: Feedback was gathered from potential users to identify usability issues and areas for improvement.

7.2 Results and Findings

The application performed well during testing, with successful story generation and accurate sentiment analysis. User feedback indicated a positive experience, with suggestions for additional features such as more character options or story themes.

8. User Feedback and Iteration

User feedback played a crucial role in the development process. Key insights included:

- Users appreciated the personalized nature of the stories.
- Suggestions for additional features, such as saving stories and sharing options, were noted for future iterations.

9. Future Enhancements

1. Expanded Character Options: Allow users to select from predefined characters or story settings.
2. Story Themes: Introduce themes (e.g., adventure, mystery) that users can choose from to guide story generation.
3. User Profiles: Implement user accounts to save and retrieve past stories and preferences.
4. Mobile Optimization: Enhance the mobile experience to ensure the application is fully responsive and user-friendly on smaller screens.
5. Additional Languages: Support multiple languages for broader accessibility and user engagement.

10. Conclusion

The Personalized AI Storyteller project successfully combines advanced natural language processing with an engaging user interface to create a unique storytelling experience. By leveraging sentiment analysis and audio feedback, the application not only generates personalized stories but also enhances user interaction and emotional connection. With further enhancements and user feedback integration, the project has the potential to evolve into a comprehensive storytelling platform that appeals to a diverse audience.

11. References

- Cohere API Documentation
- TextBlob Documentation
- Flask Documentation
- User Experience Design Principles

12. Appendices

Appendix A: Code Snippets

```
1import cohere
2import os
3from flask import Flask, render_template, request, jsonify
4from textblob import TextBlob
5
```

```
app = Flask(__name__)

API_KEY = os.getenv('COHERE_API_KEY')  # Set your API key in the environment variable COHERE_API_KEY
co = cohere.Client(API_KEY)

def generate_story(mood, char_name, gender):
    prompt = f"Tell a complete story according to the mood '{mood}', with the character '{char_name}' and gender '{gender}' in 300 words."
    response = co.generate(
        model='command-xlarge-nightly',
        prompt=prompt,
        max_tokens=300,
        temperature=0.7
    )
    story = response.generations[0].text.strip()
    return story

def get_sentiment(text):
    blob = TextBlob(text)
    polarity = blob.sentiment.polarity
    if polarity > 0.1:
        return "happy"
    elif polarity < -0.1:
        return "sad"
    else:
        return "neutral"

@app.route('/')
```

```
def index():
    return render_template('index.html')

@app.route('/generate_story', methods=['POST'])
def generate_story_api():
    user_input = request.json.get('user_input')
    char_name = request.json.get('char_name')
    gender = request.json.get(' gender')
    mood = get_sentiment(user_input)
    story = generate_story(mood, char_name, gender)
    return jsonify({'story': story, 'mood': mood, 'gender': gender})

if __name__ == '__main__':
    app.run(debug=True)
```


Appendix B: User Interface Code (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Personalized AI Storyteller</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}" />
```

```
8</head>
9<body>
10    <div class="area">
11        <ul class="circles">
12            <li></li>
13            <li></li>
14            <li></li>
15            <li></li>
16            <li></li>
17            <li></li>
18            <li></li>
19            <li></li>
20            <li></li>
21            <li></li>
22            <li></li>
23        </ul>
24        <div class="title"><h1>Personalized AI Storyteller</h1></div>
25        <div class="container">
26            <h2>How are you feeling today?</h2>
27            <textarea id="user-input" placeholder="Type your
    feelings..."></textarea>
28            <textarea id="char" placeholder="Say your character
    name"></textarea>
29            <textarea id="gender" placeholder="Say your character
    gender"></textarea>
30            <button id="generate-story-btn">Generate Story</button>
31
32            <h3 id="mood">Your Mood:</h3>
33            <div class="audio">
```

```
34                <div class="volume-control">
35                    <label for="background-volume">Background Music
    Volume:</label>
36                    <input type="range" id="background-volume" min="0"
    max="1" step="0.1" value="0.5" />
37                </div>
38                <div class="volume-control">
39                    <label for="sound-volume">Sound Effects
    Volume:</label>
40                    <input type="range" id="sound-volume" min="0" max="1"
    step="0.1" value="0.5" />
41                </div>
42            </div>
43            <p id="story"></p>
44
45            <audio id="background-music" loop>
46                <source src="{{ url_for('static',
    filename='background-music.mp3') }}" type="audio/mpeg" />
47                Your browser does not support the audio tag.
48            </audio>
49            <audio id="happy-sound">
50                <source src="{{ url_for('static', filename='happy.mp3')
    }}" type="audio/mpeg" />
51                Your browser does not support the audio tag.
52            </audio>
53            <audio id="sad-sound">
54                <source src="{{ url_for('static', filename='sad.mp3') }}"
    type="audio/mpeg" />
55                Your browser does not support the audio tag.
```

```
            </audio>
            <audio id="neutral-sound">
                <source src="{{ url_for('static', filename='neutral.mp3') }}" type="audio/mpeg" />
                Your browser does not support the audio tag.
            </audio>
        </div>
    </div>

    <script src="{{ url_for('static', filename='game.js') }}"></script>
</body>

</html>
```

Appendix C: JavaScript Code (game.js)

```
document.getElementById('generate-story-btn').addEventListener('click', async function () {
    const userInput = document.getElementById('user-input').value;
    const char_name = document.getElementById('char').value;
    const gender = document.getElementById('gender').value;

    if (!userInput) {
        alert("Please enter your feelings to generate a story.");
        return;
    }

```

```
11      if (!char_name) {
12          alert("Please enter a character name.");
13          return;
14      }
15
16      if (!gender) {
17          alert("Please enter a character gender.");
18          return;
19      }
20
21      const response = await fetch('/generate_story', {
22          method: 'POST',
23          headers: {
24              'Content-Type': 'application/json'
25          },
26          body: JSON.stringify({
27              user_input: userInput,
28              char_name: char_name,
29              gender: gender
30          })
31      });
32
33      const data = await response.json();
34
35      document.getElementById('mood').innerText = `Your Mood:
        ${data.mood.charAt(0).toUpperCase() + data.mood.slice(1)}`;
36      document.getElementById('story').innerText = data.story;
37
38      const backgroundMusic = document.get ElementById('background-music');
```

```
    backgroundMusic.volume =
document.getElementById('background-volume').value; // Set volume from
slider
    backgroundMusic.play();

    let moodSound;
    switch (data.mood) {
        case 'happy':
            moodSound = document.getElementById('happy-sound');
            break;
        case 'sad':
            moodSound = document.getElementById('sad-sound');
            break;
        case 'neutral':
            moodSound = document.getElementById('neutral-sound');
            break;
    }
    if (moodSound) {
        moodSound.volume = document.getElementById('sound-volume').value;
// Set volume from slider
        moodSound.currentTime = 0; // Reset sound to start
        moodSound.play();
    }

    const utterance = new SpeechSynthesisUtterance(data.story);
    speechSynthesis.speak(utterance);
});

// Event listeners for volume controls
```

```
65document.getElementById('background-volume').addEventListener('input',
function () {
66    const backgroundMusic = document.getElementById('background-music');
67    backgroundMusic.volume = this.value; // Adjust background music
volume
68});
69
70document.getElementById('sound-volume').addEventListener('input',
function () {
71    const happySound = document.getElementById('happy-sound');
72    const sadSound = document.getElementById('sad-sound');
73    const neutralSound = document.getElementById('neutral-sound');
74
75    happySound.volume = this.value;
76    sadSound.volume = this.value;
77    neutralSound.volume = this.value;
78});
```


Appendix D: Audio Files

- happy.mp3: Contains sound effects for happy mood.
- sad.mp3: Contains sound effects for sad mood.
- neutral.mp3: Contains sound effects for neutral mood.
- background-music.mp3: Background music for the storytelling experience.

Appendix E: CSS Styles (style.css)

```
1/* Global Styles */
2* {
3    box-sizing: border-box;
4    margin: 0;
5    padding: 0;
6    font-family: 'Arial', sans-serif;
7}
8
9body {
10    color: #E0E0E0;
11    justify-content: center;
12    align-items: center;
13}
14
15.container {
16    width: 90%;
17    max-width: 600px;
18    text-align: center;
19    position: relative;
20    top: 15vh;
21    left: 5vw;
22}
23
24.title {
25    font-size: 2.5rem;
```

```
    color: #FFD700;
    animation: fadeIn 1.5s ease-out;
    margin-bottom: 10px;
    text-align: center;
    position: relative;
    top: 5vh;
}

textarea {
    width: 100%;
    height: 50px;
    border-radius: 8px;
    border: none;
    padding: 15px;
    font-size: 1rem;
    resize: none;
    background-color: #1e1e1e;
    color: #FFFFFF;
}

#generate-story-btn {
    padding: 12px;
    border: none;
    border-radius: 8px;
    background-color: #FFD700;
    color: #121212;
    font-weight: bold;
    cursor: pointer;
    transition: background-color 0.3s;
```

```
    font-size: 1rem;
    position: relative;
    top: 2vh;
}

#story {
    margin-top: 30px;
    padding: 20px;
    border-radius: 8px;
    opacity: 0;
    transform: scale(0.95);
    animation: fadeInUp 0.7s ease-out forwards;
    position: relative;
    left: 50vw;
    top: -70vh;
}

#mood {
    margin-top: 15px;
    font-size: 2rem;
    color: #FFD700;
    font-style: italic;
}

/* Animations */
@keyframes fadeIn {
    0% { opacity: 0; }
    100% { opacity: 1; }
}
```

```
84
85@keyframes fadeInUp {
86    0% { opacity: 0; transform: translateY(20px); }
87    100% { opacity: 1; transform: translateY(0); }
88}
```

This detailed report provides a comprehensive overview of the Personalized AI Storyteller project, covering all aspects from technical implementation to user experience and future enhancements. The project demonstrates the potential of AI in creating personalized content and engaging user experiences.

Project Metadata

- Version: 1.0
- Development Period: November , 2024
- Primary Technologies: Python, Flask, Cohere AI, JavaScript

Prepared by: **TechWizards**

Teammates :

1. **T cherish chandra reddy**
2. **Thirumula thanmayee roy**
3. **K Roshmitha reddy**
4. **Uday kumar reddy m**
5. **Khushi M**

Date: 09-11-2024