Project on:

"AI Fan Fiction Assistant: A Chatbot for Generating Personalized Stories Based on User's Favorite Fandoms

Prepared By:

Name: Botta Sri Ram
Guthula Som Sunil

Reg no:12323134
12322522

Section:K23GF

Roll no:09,33

Submitted To:

Dipen Saini(23681)

Table of Contents

1. Project Overview
2. Module-Wise Breakdown
3. Functionalities
4. Technology Used
• Programming Languages:
• Libraries and Tools:Other Tools: [e.g., GitHub for version control]
5. Flow Diagram
6. Revision Tracking on GitHub
• Repository Name: [Insert Repository Name]
• GitHub Link: [Insert Link]
7. Conclusion and Future Scope
8. References
Appendix
A. AI-Generated Project Elaboration/Breakdown Report
[Paste the AI-generated breakdown of the project in detail]
B. Problem Statement: [Paste Problem Statement]
C. Solution/Code:
D. [Paste complete code of entire project here]

1. Project Overview

The **AI Fan Fiction Assistant** is a chatbot application designed to help users generate creative fanfiction stories based on their favourite fandoms, characters, and genres. Leveraging natural language processing and generative AI, the assistant takes user prompts—such as character names, settings, and plot ideas—and produces personalized, coherent, and imaginative stories.

This project aims to provide a fun, creative, and interactive experience for fans of books, movies, TV series, anime, and games, allowing them to bring their story ideas to life.

The system will simulate real-time memory access patterns and display how different algorithms handle memory requests, thus helping in understanding the internal working of an OS's memory management system.

2. Module-Wise Breakdown

The project is divided into the following core modules:

1. User Input Module

- Accepts user prompts including:
  - Fandom name (e.g., Harry Potter, Naruto)
  - Characters (e.g., Hermione, Sasuke)
  - Genre (e.g., Romance, Action, Comedy)

    - Plot idea or mood/theme
- Performs input validation and formatting.

2. Prompt Construction Module

- Dynamically creates a rich and context-aware prompt using the user's input.
- Ensures the prompt includes key elements like tone, story length, and character focus.

3. AI Story Generation Module

- Sends the constructed prompt to an AI API (e.g., OpenAI GPT or Gemini) for generating the story.
- Parses and formats the story for display.

4. API Integration Module

- Connects to generative AI models via APIs.
- Handles API authentication, error management, and response formatting.

5. Frontend Interface Module

- Provides a user-friendly interface to input data and view generated stories.
- Built using HTML, CSS (Tailwind), and JavaScript.

6. Response Display Module

- Displays generated fanfiction in a readable, aesthetic format.
- Includes options for copying, saving, or regenerating the story.

7. Story Customization Module (Optional/Future Enhancement)

- Allows the user to:
    - Specify length (short, long)
    - Choose narrative style (first-person, third-person)
    - Select tone (funny, serious, dark)

3. Functionalities

The **AI Fan Fiction Assistant** provides the following key functionalities:

1. Fanfiction Generation

- Automatically generates fanfiction based on user inputs (fandom, characters, genre, and plot).
- Supports multiple genres and crossovers.

2. Dynamic Prompt Handling

- Constructs intelligent prompts tailored to user preferences to ensure accurate story generation.

3. Real-Time Story Preview

- Displays the generated story immediately after API response.
- Allows users to read, copy, or modify the output.

4. Regenerate Story Option

- Provides a button for regenerating the story with the same or slightly varied prompt.

5. Multi-Fandom Support

- Supports input from a wide range of fandoms such as:
  - Harry Potter
  - Marvel/DC
  - Naruto
  - BTS, Blackpink
  - Stranger Things, etc.

6. Genre and Style Control

- Users can choose different tones (dark, romantic, comedic) and formats (dialogue-heavy, narrative, etc.).

7. API Integration

- Connects seamlessly with generative AI APIs (e.g., OpenAI, Gemini) to fetch creative content.

4. Technology Used

🖥️

Programming Languages:

- **HTML5** – Used for structuring the chatbot interface and input forms.
- **JavaScript** – Handles user interactions, prompt processing, and integration with APIs.

🎨

Libraries and Tools:

- **Tailwind CSS** – A utility-first CSS framework used to build a responsive, clean, and modern user interface for the chatbot.
- **Zapier AI** – Powers the AI story generation by sending user inputs to an AI model (like OpenAI or Gemini) and retrieving fanfiction responses. It eliminates the need to build a backend by automating the workflow using "Zaps."

🔐

Zapier AI Integration and API Key Usage:

Zapier AI is central to how the chatbot processes user prompts and returns AI-generated stories.

- **Zap Creation:** A "Zap" is created in Zapier to listen for incoming requests (via webhook) and send them to an AI model like OpenAI, returning the response back to the chatbot.
- **How it Works:**
    1. User submits their prompt (fandom, characters, genre).
    2. The frontend sends the input to a **Zapier Webhook URL**.
    3. Zapier triggers the AI to generate a story.
    4. The response is sent back and displayed in the chat interface.

Example Call:

Javascript:-

```
fetch("https://hooks.zapier.com/hooks/catch/123456/abcde", {

method: "POST",

  headers: { "Content-Type": "application/json" },

  body: JSON.stringify({

    fandom: "Naruto",
```

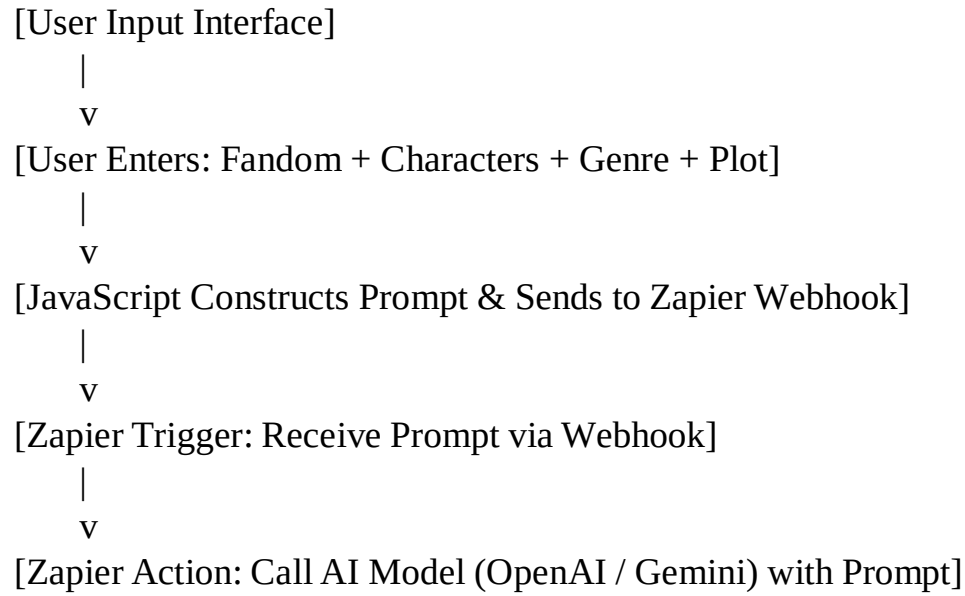
```
    characters: "Naruto and Hinata",
    genre: "Adventure",
    plot: "Exploring a mysterious new village",
  }),
})
.then(res => res.json())
.then(data => {
  console.log("Generated Story:", data.story);
});
```

Other Tools:

- **GitHub** – Used for version control and managing code revisions collaboratively.
- **VS Code** – The primary development environment for writing, testing, and debugging the chatbot code.
- **Zapier Dashboard** – Used to configure Zaps, manage API workflows, and test automation.

5. Flow Diagram

Here's a step-by-step **flow diagram (text-based)** representing how your **AI Fan Fiction Assistant** chatbot works:



|
v
[AI Generates Story Based on Prompt]
|
v
[Zapier Sends Response Back to Webhook Output]
|
v
[JavaScript Receives Story & Displays in Chat UI]
|
v
[User Can Read / Copy / Regenerate Story]

6. Revision Tracking on GitHub

To ensure transparency, collaboration, and version control during the development of this project, GitHub has been used extensively to manage all updates and contributions.

✅

Repository Name: ai-fanfiction-assistant-chatbot

🔗

GitHub Link:

https://github.com/ramsriram123/ai-fanfiction-assistant-chatbot

(Replace your-username with your actual GitHub username.)

📁

Tracking and Versioning Includes:

- Regular commits to document progress.
- Branches for experimenting with new features (e.g., UI update, genre filter, etc.).

- Pull requests (if working with collaborators).
- Issues to track bugs or improvement ideas.

7. Conclusion and Future Scope

Conclusion:

The **AI Fan Fiction Assistant Chatbot** successfully demonstrates how artificial intelligence can enhance creativity by generating personalized fanfiction stories based on user-selected fandoms, characters, and themes. By leveraging **Zapier AI**, **JavaScript**, and **Tailwind CSS**, the chatbot provides a seamless and interactive experience without requiring a dedicated backend server. The integration ensures ease of use, scalability, and modular development.

This project showcases the potential of combining **low-code tools with custom frontend development** to bring AI-driven creativity into the hands of users who love storytelling.

Code representation:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>AI Fan Fiction</title>
  <script src="https://cdn.tailwindcss.com"></script>
  <style>
    @keyframes glow {
      0% { text-shadow: 0 0 10px #00ffcc; }
      100% { text-shadow: 0 0 20px #ff007a; }
    }

    .glow-text {
      background: linear-gradient(45deg, #00ffcc, #ff007a);
      -webkit-background-clip: text;
      color: transparent;
      animation: glow 2s infinite alternate;
      text-transform: uppercase;
      letter-spacing: 5px;
    }
  </style>
</head>
<body class="min-h-screen bg-gray-900 text-white font-sans flex flex-col
items-center justify-start px-6 py-10 space-y-12">

  <!-- Header -->
  <header class="text-center space-y-2">
    <h1 class="text-5xl font-extrabold glow-text">AI Fan Fiction
Assistant</h1>
    <p class="text-gray-300 text-lg max-w-xl mx-auto">Unleash your
imagination with AI-powered storytelling. Enter your fandom and let the
magic unfold!</p>
  </header>

  <!-- Chatbot Section -->
  <section class="w-full max-w-4xl bg-gray-800 bg-opacity-70 backdrop-
blur-md rounded-xl p-6 shadow-lg">
    <zapier-interfaces-chatbot-embed
      is-popup='false'
      chatbot-id='cm98jbbra000oe1ypxax4ojmg'
```

```
      height='500px'
      width='100%'>
    </zapier-interfaces-chatbot-embed>
  </section>

  <script async type='module'
src='https://interfaces.zapier.com/assets/web-components/zapier-
interfaces/zapier-interfaces.esm.js'></script>
</body>
</html>
```

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