College Name: VIT BHOPAL UNIVERSITY
Student Name: SHREYA DUTTA
Registration No: 22BCG10107

GEN AI PROJECT PHASE 3 SUBMISSION DOCUMENT

Phase 3: Final Report and Submission

1. Project Title:

Generative AI Storyteller: Interactive Narrative Co-Creation

2. Summary of Work Done

Phase 1 – Proposal and Idea Submission (10 Marks):

- Activities: Defined the project scope, identified the problem of writer's block and the need for creative AI tools. Proposed a solution involving a web application integrated with an LLM API. Outlined clear objectives, expected outcomes, and the core technologies to be used.

- Deliverables: A comprehensive project proposal document detailing the project title, domain, problem statement, proposed solution, objectives, expected outcomes, tools & technologies, and references. This phase established a clear roadmap for the project.

- Key Technology Choice: Initial discussions explored various LLM options, with the final proposal leaning towards a readily accessible and powerful API like Google Gemini, due to its free tier and robust capabilities.

Phase 2 – Execution and Demonstration (15 Marks):

Activities:

- Backend Development:Developed a Python Flask application to serve as the backend. This involved setting up API endpoints, specifically `/generate_story`, to handle requests from the frontend.

- Google Gemini API Integration: Successfully integrated the Google Gemini API (using the `gemini-1.5-flash-latest` model via the `google-generativeai` Python SDK). Implemented logic to send user prompts to the Gemini API and process the generated text. Ensured secure API key management using `.env` files and the `python-dotenv` library.

- Frontend Development: Created an intuitive user interface using HTML, CSS, and JavaScript. The frontend allows users to input story prompts, displays a loading indicator during AI processing, and presents the AI-generated story or any error

messages. Implemented asynchronous communication with the backend using the Fetch API.

- Error Handling: Implemented error handling on both client and server sides to manage issues such as missing prompts, API errors (authentication, rate limits, content blocking), and general server errors, providing informative feedback to the user.

- Code Structuring and Documentation:** Organized the codebase into a logical structure with clear separation of frontend and backend components. Added comments and logging for better maintainability.

- Deliverables: A fully functional "Generative AI Storyteller" web application. A comprehensive Phase 2 report including system architecture, full code implementation details, and setup/run instructions. A live demonstration of the working application.

3. GitHub Repository Link

You can access the complete codebase, README instructions, and any related resources at the following GitHub link:

https://github.com/Reyd900/GEN_AI_Storyteller-

College Name: VIT BHOPAL UNIVERSITY
Student Name: SHREYA DUTTA
Registration No: 22BCG10107

4. Testing Phase

4.1 Testing Strategy

The testing strategy focused on ensuring the core functionality, reliability, and usability of the "Generative AI Storyteller." A multi-faceted approach was adopted, combining manual testing methods due to the project's scope and the qualitative nature of AI-generated content. The strategy prioritized:

- Core Functionality: Verifying that the story generation mechanism works as intended.
- API Integration: Ensuring seamless communication with the Google Gemini API.
- User Experience: Confirming the application is easy to use and provides clear feedback.
- Error Resilience: Testing how the application handles various error conditions.

4.2 Types of Testing Conducted

- **Unit Testing (Backend & Frontend):** Isolated testing of API endpoints and JavaScript functions with valid/invalid inputs, error handling (API key, prompt issues).
- **Integration Testing (Frontend-Backend-Gemini):** Manual testing of the complete request-response cycle, including potential Gemini API errors (rate limits, safety).
- **Functional Testing (End-to-End):** User-centric testing of all features: prompt entry, story generation, loading states, error display, button behavior.
- **Usability Testing (Ad-hoc):** Informal assessment of interface ease of use, clarity, and intuitiveness.
- **Content Quality Assessment:** Manual evaluation of generated story coherence, creativity, and relevance.

4.3 Results

Key Findings:

- **Core Functionality (Success):** Story generation worked well across various prompts.
- **API Integration (Success):** Gemini API integration (authentication, request/response) was successful.
- **Error Handling (Effective):** Client-side (empty prompt) and server-side (length, API key, rate limits, content blocks) error handling functioned as expected, with relevant messages displayed.
- **User Interface (Intuitive):** Simple and provided necessary feedback (loading, errors).

- **Content Quality (Good):** gemini-1.5-flash-latest produced generally creative and relevant continuations, influenced by prompt clarity.

Limitations:

- **Free-Tier Constraints:** Potential latency and stricter rate limits.
- **LLM Dependency:** Output quality varies based on prompt and LLM capabilities.
- **No Automated Testing:** Relied on manual verification.

5. Future Work

While the current application is fully functional, several enhancements could be considered for future iterations:

- **Advanced Prompt Engineering Options:** Allow users to select different "tones" (e.g., humorous, dramatic, mysterious) or genres for the story, which would translate to different system prompts sent to the LLM.
- **Iterative Storytelling:** Enable users to accept, reject, or request regeneration of the AI's contribution, and then add more of their own text, allowing for a more back-and-forth collaborative writing process.
- **Saving and Exporting Stories:** Implement functionality for users to save their co-created stories locally or export them as text files.
- **User Accounts:** Introduce user authentication to allow saving stories to an account and potentially track writing history.
- **Streaming Responses:** For longer story generations, implement streaming of the AI's response to the frontend so text appears word-by-word, improving perceived responsiveness.
- **Model Selection:** Allow users to choose from different available LLMs (if multiple compatible models are accessible via the API) to experiment with various generation styles.
- **Enhanced Frontend UI/UX:** Improve the visual design, potentially adding features like character counters, themes, or a richer text editor.
- **Deployment to a Public Cloud Platform:** Deploy the application to a service like Google Cloud Run, Vercel, or Heroku for wider accessibility.
- **Automated Testing:** Develop a basic suite of automated tests (e.g., using Selenium for UI tests or PyTest for backend API tests) to improve regression testing for future updates.
- **Fine-tuning (Ambitious):** For a highly specialized storytelling experience, explore the possibility of fine-tuning a smaller open-source LLM on a curated dataset of stories, though this significantly increases complexity and resource requirements.

College Name: VIT BHOPAL UNIVERSITY
Student Name: SHREYA DUTTA
Registration No: 22BCG10107

6. Conclusion

The "Generative AI Storyteller" project successfully demonstrates the practical application of large language models in fostering creativity and providing interactive entertainment. By integrating Google's Gemini API with a user-friendly web interface, the project has achieved its primary objective of creating a tool that can assist users in the narrative creation process.

Throughout its three phases, the project progressed from a conceptual idea to a fully realized and tested application. The use of Python Flask for the backend and standard web technologies for the frontend, combined with the power of the Gemini LLM, resulted in a robust and engaging proof-of-concept. The project highlights the potential of generative AI to act as a collaborative partner in creative endeavors and underscores the accessibility of powerful AI tools for developers.

While there are many avenues for future enhancements, the current implementation stands as a complete and functional system, effectively addressing the initial problem statement and meeting all outlined objectives.