# Report

## 1. Introduction

Storytelling is a fundamental part of human creativity. With the advancements in Artificial Intelligence (AI), it has become possible to generate dynamic and interactive stories. This project focuses on building an **AI-powered story generation system** that allows users to provide an initial plot, specify the genre and tone, and then interactively choose among different possible continuations. The system also integrates dictionary support and question-answering capabilities for a richer user experience.

## 2. Abstract

This project implements an **interactive story generation application** using **LangChain, Google Generative AI (Gemini), FAISS, and Streamlit**. The system generates multiple story continuations for a given plot, lets the user select the preferred continuation, and then continues the story iteratively. Additionally, the application supports dictionary lookups for unfamiliar words and allows users to ask context-based questions using a retrieval-based QA system.

By storing data in `st.session_state`, the application maintains continuity of the story across multiple user interactions.

## 3. Tools Used

**Python** – Core programming language for implementation.
**LangChain** – Framework for integrating LLMs, prompts, and retrieval mechanisms.
**Google Generative AI (Gemini 2.0)** – Large Language Model used for generating story continuations.
**Google Generative AI Embeddings** – Used to create embeddings for semantic search and question answering.
**FAISS (Facebook AI Similarity Search)** – Vector database for storing and retrieving story chunks for Q&A.
**Streamlit** – Web framework used to build an interactive front-end for story generation.
**Dictionary API** – External API to fetch meanings of words encountered in the story.
**dotenv** – To load environment variables (API keys).

## 4. Steps Involved in Building the Project

1. **Environment Setup**
   Installed required libraries: `langchain`, `streamlit`, `faiss-cpu`, `requests`, and `dotenv`.

2. **Session State Initialization**
   Used `st.session_state` to persist data across user interactions.

3. **Story Continuation Generation**
   Designed a `PromptTemplate` instructing the LLM to generate **3 possible continuations** in the specified genre and tone

4. **User Interaction**
   Displayed generated continuations to the user.
   **Vector Store Creation for Q&A**
   Embedded and stored these chunks in **FAISS**.
   Used `RetrievalQA` chain to allow the user to ask story-based questions.

5. **Additional Features**
   **Dictionary Lookup**: Integrated a dictionary API to fetch meanings of words.
   **Stopping Mechanism**: Allowed the user to end the story at any point.

6. **Final Story Assembly**
   Concatenated all chosen story continuations and display final story.

# 5. Conclusion

This project demonstrates the potential of **AI-driven interactive storytelling**. By combining **LLMs, embeddings, and vector databases**, it allows users to not only generate stories but also interact with them through **choices, Q&A, and dictionary support**.
This approach can be extended to **educational tools, creative writing assistance, and game-based storytelling applications**.