SRI SIDDHARTHA ACADEMY OF HIGHER EDUCATION

(DEEMED TO BE UNIVERSITY)

Accredited A+ Grade by NAAC

SRI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(A Constituent College of SSAHE)

KUNIGAL ROAD, MARALURU, TUMAKURU-572105

Mini Project Report

on

"AI STORY GENERATOR"

Submitted in partial fulfillment of the requirement for the completion of VI semester of

BACHELOR OF ENGINEERING

Submitted

by

KARAN R JOSHI	21IS036
NITIN KUMAR T N	21IS058

Under the Guidance

of

CHAMPAKAMALA S B.E, M.Tech

Asst. Professor, Dept. of ISE

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

"**Accredited by NBA, NEW DELHI**"

2021-25

SRI SIDDHARTHA ACADEMY OF HIGHER EDUCATION

(DEEMED TO BE UNIVERSITY)

Accredited A^{+} Grade by NAAC

SRI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(A Constituent College of SSAHE)

KUNIGAL ROAD, MARALURU, TUMAKURU-572105

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certified that mini project entitled **"AI STORY GENERATOR",** is a bonafide work carried out by Mr, **KARAN R JOSHI (21IS036)** and Mr. **NITIN KUMAR T N (21IS058)** in partial fulfillment of the requirement for the completion of VI semester in **Information Science and Engineering** of Sri Siddhartha Institute of Technology-Tumakuru during the year 2023-24. It is certified that all corrections/suggestions indicated have been incorporated in the report. The report has been approved as it satisfies the academic requirements with respect to Mini Project work.

Signature of the Guide	**Signature of the HoD**
Champakamala S B.E, M.Tech	Dr. H S Annapurna
Asst. Professor ,	Professor and Head,
Dept. of Information Science & Engg.	Dept. of Information Science & Engg.

Name of the Examiners	**Signature**
1. ______________________________	____________________
2. ______________________________	____________________

DECLARATION

We, **KARAN R JOSHI (21IS036)** and **NITIN KUMAR T N (21IS058)** of VI semester, Department of Information Science and Engineering of Sri Siddhartha Institute of Technology, Tumakuru, hereby declare that this Mini project titled, **"AI STORY GENERATOR"**, has been carried out by us under the supervision of Champakamala S, Asst. Professor, Department of Information Science and Engineering, Sri Siddhartha Institute of Technology, Tumakuru in partial fulfillment of the requirement for the completion of VI semester in Information Science and Engineering.

Date:	**KARAN R JOSHI (21IS036)**
Place: Tumakuru	**NITIN KUMAR TN (21IS058)**

TABLE OF CONTENTS

LIST OF FIGURES		i
ABSTRACT		ii
Sl. No.	**Description**	**Page No.**
1.	Introduction	1
	1.1 Background	1
	1.2 Problem Statement	2
	1.3 Aim and Objective	4
2.	Literature Survey	5
3.	System Requirements	7
4.	System Design	10
	4.1 System Architecture	10
	4.2 Flow Diagram	13
5.	Implementation	14
6.	Results	20
7.	Conclusion	23
8.	Future Scope	24
	References	25

LIST OF FIGURES

Fig.No.	Figure Name	Page No.
Figure 4.1	System Overview	10
Figure 4.2	System Technologies	11
Figure 4.3	System Workflow	11
Figure 4.4	FlowChart	13
Figure 6.1	Result 01	20
Figure 6.2	Result 02	20
Figure 6.3	Result 03	20
Figure 6.4	Result 04	21
Figure 6.5	Result 05	21
Figure 6.6	Result 06	21
Figure 6.7	Result 07	22
Figure 6.8	Result 08	22
Figure 6.9	Result 09	22

ABSTRACT

AI Story Generator is a sophisticated platform that convergence of artificial intelligence with the art of storytelling. It is designed to democratize the creation of digital narratives, making it possible for anyone to craft and share stories that are rich in text, visuals, and audio. Here's a comprehensive explanation of the abstract for the AI Story Generator, presented in a detailed paragraph:

At its core, AI Story Generator that capabilities of advanced **Artificial Intelligence (AI)** models to generate text, synthesize images, and produce narration, creating a multimodal storytelling experience. The platform begins with a **Generative Pre-Trained Transformers (GPT)** GPT-based model that takes user prompts and weaves them into compelling narratives, each unique to the input provided. To complement the textual content, the Stable Diffusion technology is employed to produce images. Meanwhile, neural **Text-To-Speech (TTS)** technology brings the story to life with human-like narration, infusing each word with emotion and character. This trinity of AI components works in concert to transform simple prompts into rich, engaging stories that captivate the audience.

Ease of use is a cornerstone of the AI Story Generator. The platform is designed with an intuitive interface that invites users to input their story prompts and customize various elements of the narrative, such as style, mood, and pacing. This level of customization ensures that each story is not only generated by AI but also shaped by the user's creative vision. Furthermore, the project emphasizes scalability and community engagement, encouraging a wide range of users to explore the possibilities of AI-assisted storytelling.

In essence, AI Story Generator is more than just a technological innovation, it is a gateway to a new era of storytelling where barriers to creation are removed, and the power of narrative is placed in the hands of many. It is a part to the potential of AI but not only mimic human creativity but to amplify it, allowing us to tell our stories in ways that were once unimaginable.

CHAPTER 1
INTRODUCTION

1.1 Background

AI Storyteller: Crafting Multimodal Narratives

The **AI Storyteller** is an ambitious project that combines cutting-edge technologies to create captivating and immersive stories. Here's a detailed background:

Project Overview:

- **Name**: AI Storyteller
- **Purpose**: To generate multimodal narratives using AI techniques.

Components:

- **Text Generation**:
  - The core of the story generator is a text-based AI, similar to **GPT (Generative Pre-trained Transformer)**.
  - Given an opening prompt, the AI continues the story by weaving intricate plots.
  - The AI model is trained on a vast array of literature to understand and generate stories.
- **Image and Video Synthesis**:
  - For visual storytelling, AI models like **DALL-E** or **Stable Diffusion** generate images that correspond to the text.
  - Video synthesis involves combining these images or creating short video clips aligned with the story's progression.
- **Audio Generation**:
  - **Neural TTS (Text-to-Speech)** engines convert the generated text into spoken words, creating an MP3 audio file that serves as the story's narration.
  - Background music and sound effects enhance the auditory experience, often selected based on the mood and setting of the story.
- **Subtitles and Translation**:
  - Subtitles are generated by transcribing the TTS output or directly from the text.
  - AI translation models provide subtitles in multiple languages, making the story accessible to a global audience.

- **Integration and Synchronization**:
    - All these elements are integrated into a multimedia experience.
    - Synchronization ensures that visuals, audio, and subtitles match the timing and flow of the story.

Why It Matters:

- AI-generated storytelling opens up new creative possibilities.
- It can inspire novelists, entertain friends, or simply explore fictional realms.
- The AI Storyteller is your perfect companion for crafting unique narratives.

1.2 Problem Statement:Revolutionizing Digital Storytelling with AI

- **The Challenge**

The digital storytelling landscape is rapidly evolving, with audiences increasingly craving immersive experiences that blend text, visuals, and audio. Traditional storytelling, often limited to text, fails to meet this demand for a multisensory narrative journey. Moreover, the process of creating such stories involves a complex interplay of diverse talents—writers, illustrators, and voice actors—making it both costly and intricate.

The **Story Generator** project revolutionizes this paradigm by harnessing **advanced AI** to streamline the creation of **engaging multimodal stories**. Our platform empowers users to transform a simple text prompt into a **vibrant video story**, enriched with **AI-generated illustrations and narration**. This innovative approach not only **democratizes storytelling**, making it accessible to a wider audience, but also fosters **unprecedented creative freedom** in the digital realm.

By integrating **cutting-edge AI technologies** like **GPT** for text, **Stable Diffusion** for imagery, and **neural TTS** systems for voice, Story Generator offers a **single, unified platform** for crafting **cohesive and captivating stories.** We are committed to bridging the gap between **aspiring storytellers and professional content creation**, unlocking a new era of **digital expression**.

- **The Solution: Story Generator**

The proposed **AI Story Generator** allows users to input a **text prompt** as the opening line of a story. From there, the **GPT model** takes over, crafting a compelling narrative that unfolds with each sentence. Concurrently, **Stable Diffusion** generates vivid images that visually represent the story's progression, while the **Neural TTS** model brings the characters to life with engaging narration.

- **Key Features:**

    - **Automated Story Generation**: Leveraging GPT's advanced language models to write coherent and imaginative stories based on user prompts.

- **Dynamic Visual Illustration**: Utilizing Stable Diffusion to create detailed and contextually relevant images for each segment of the story.

- **Narration Synthesis**: Employing cutting-edge neural TTS to produce clear and expressive voiceovers, enhancing the auditory dimension of the story.

- **Fully Animated Video Output**: Combining text, images, and audio to generate a fully animated video, delivering a **multisensory storytelling experience**.

➢ **User Experience:**

The platform is designed with a **user-friendly interface** that simplifies the creation process. Users can start with a default prompt or customize their story's beginning using the **--*writer_prompt*** argument. The final video, along with intermediate files such as images, audio files, and subtitles, is saved in a designated output directory, ready for sharing and viewing.

➢ **Accessibility and Creativity:**

StoryTeller democratizes content creation by making it accessible to individuals without the need for technical expertise or artistic skills. It opens up new avenues for **creative expression**, allowing users to tell their stories in a way that was previously only possible with a team of professionals.

1.3 Aim and Objective

Aim

The **AI Storyteller** project aims to transform traditional storytelling by harnessing advanced AI technologies. Our goal is to create captivating and immersive narratives that seamlessly blend text, visuals, and audio. Here's how we achieve this:

➢ **Multimodal Fusion**:

- We integrate cutting-edge AI models:
  - **GPT (Generative Pre-trained Transformer)** for text generation.
  - **Stable Diffusion** for creating vivid imagery.
  - Neural TTS systems for lifelike voice narration.
- These elements harmoniously converge to craft cohesive and captivating stories.

➢ **Democratizing Storytelling**:

- We believe that storytelling should be accessible to everyone.
- By removing barriers and complexities, we empower users to create their own narratives.

- **Creative Freedom Unleashed**:
  - The **AI Storyteller** fosters unprecedented creative freedom in the digital realm.
  - Users can explore imaginative scenarios, experiment with genres, and express their unique voices.

Objective

Our specific objectives are as follows:

- **User-Generated Content Platform**:
  - Create a platform for storytellers to create their stories.
  - Promote the art of storytelling.
- **Built-in Text-to-Speech (TTS)**:
  - Enable users to listen to stories in voice format.
  - Enhance the auditory experience through AI-generated narration.
- **Immersive Storytelling**:
  - Develop a system that seamlessly combines text, images, and audio.
  - Transport readers into captivating fictional realms.
- **Ethical and Responsible AI**:
  - Prioritize ethical guidelines in AI-generated conten

CHAPTER 2

LITERATURE SURVEY

GRANNY-GPT2-Based-Multiple_Genre-Story-Generator-Web-Application:

- This project is a web application that generates multiple genre stories based on OpenAI's GPT-1 model [1]. It is inspired by the stories of the developer's grandmother and aims to provide a platform for generating and sharing stories in various genres such as scary, funny, and sci-fi [1].
- The application includes features like text-to-speech narration, downloading stories as text files, generating globally shareable URLs and QR codes, suggesting grammatical and spelling corrections, and adding generated stories to a publicly accessible "Wall of Stories" [1].
- The backend of the application is built using Flask for fine-tuning the model and generating text, while the frontend is developed using HTML, CSS, and JavaScript [1].Use of LLMs for Narrative Generation

The AI Story Generator[2] offers the advantage of **enhanced creativity and idea generation**, providing creative suggestions and expanding on user-provided ideas, making it an invaluable resource for brainstorming and story development, whereas the GRANNY GPT-2 Based Story Generator focuses on generating stories inspired by the developer's grandmother's stories with additional features like text-to-speech narration and sharing options [1].

Image2Story:

- This project aims to generate stories from images in a specific style[3]. It is based on an image caption model that has been expanded to generate complete paragraphs instead of single sentences[3].
- The model consists of three parts: extracting sentence descriptions from images using an image caption network, converting these sentences into vectors using a skip-thought vectors model, and generating stories from these vectors using a story generator trained on romantic novels [3].
- The project uses a combination of pre-trained models, including an image caption model and a sentence-to-story model [3].Strategies to Handle Narrative Generation Using LLMs

The AI Story Generator[2] offers several **advantages over the Image2Story[3] project**, including enhanced creativity and idea generation, **flexibility and adaptability** across various genres, ease of use with no sign-up or login required, and scalability for large projects. In contrast, the Image2Story project focuses on generating stories from images using a combination of pre-trained models, which limits its flexibility and adaptability to various storytelling requirements.

Auto-Story-GPT:

- This project is a Flask web application that utilizes GPT-2.5 to generate AI responses for storytelling[4]. It allows users to input a story idea, select a character or narrator, and generate responses based on the provided inputs[4].
- Users can adjust for continuity or logic issues in a story and rewrite the story to improve it[4]. The project includes features like editing the created story at any time and continuing the story with as many generations as desired[4].
- The application is designed to maintain continuity by feeding the generated story back into ChatGPT on each call[4].

The AI Story Generator[2] offers the advantage of **ease of use**, with no sign-up or login required, and fast generation with no daily usage restrictions. In contrast, the Auto-Story-GPT[4] project requires user inputs for generating responses and maintaining continuity, which may involve more manual intervention and limit its user-friendliness.

Conclusion

- **Enhanced Creativity and Idea Generation**: The AI Story Generator offers creative suggestions and expands on user-provided ideas, making it an invaluable resource for brainstorming and story development.
- **Flexibility and Adaptability**: It is adaptable across various genres, allowing users to generate stories in multiple genres such as scary, funny, and sci-fi.
- **Ease of Use**: The AI Story Generator does not require sign-up or login, making it user-friendly and accessible to a wide audience.
- **Scalability**: It is scalable for large projects, making it suitable for extensive storytelling requirements.
- **Fast Generation**: The AI Story Generator provides fast generation with no daily usage restrictions, ensuring a smooth and efficient user experience.
- **Additional Features**: It includes features like text-to-speech narration, downloading stories as text files, generating globally shareable URLs and QR codes, suggesting grammatical and spelling corrections, and adding generated stories to a publicly accessible "Wall of Stories".

These advantages make the AI Story Generator a powerful and versatile tool for narrative generation.

CHAPTER 3
SYSTEM REQUIREMENTS

Hardware Requirements

➢ **Hardware Requirements for Digital Storytelling**

Digital storytelling is an immersive, interactive form of content published on the web. It allows creators to engage their audience, build brands, and achieve better results. Whether you’re a seasoned storyteller or just dipping your toes into this exciting realm, having the right hardware ensures a smooth and captivating experience.

➢ **The Essentials**

- **Processor**: A capable processor is the beating heart of your digital storytelling setup. Opt for at least a quad-core processor (such as Intel Core i5 or equivalent) to handle multimedia tasks seamlessly.
- **Memory (RAM)**: Memory is your creative workspace. Aim for 8 GB or higher to keep multiple applications running smoothly. More RAM means less frustration during editing and rendering.
- **Storage**: While digital stories don’t require massive storage, having a fast and reliable drive is essential. Consider an SSD (Solid State Drive) for quicker loading times and responsiveness.
- **Graphics**: Integrated graphics are fine for basic tasks, but if you’re diving deep into visual storytelling, a dedicated graphics card (like NVIDIA GTX 1050 or AMD Radeon RX 560) will enhance your experience.
- **Display**: Your canvas matters! A Full HD (1920 x 1080) resolution display ensures crisp visuals. If you’re investing in a larger monitor, consider higher resolutions for that wow factor.

➢ **Additional Considerations**

- **Internet Connection**: A stable internet connection is crucial. You’ll need it for research, cloud storage, and accessing external resources like APIs.
- **Input Devices**: A standard keyboard and mouse (or touchpad) are your trusty companions. If you’re into drawing or design, a stylus tablet can elevate your creativity.
- **Audio**: Built-in speakers or headphones are essential for audio narration playback. Clear sound enhances the storytelling experience.

Software Requirements

- **Operating Systems**:
  - **Windows 10 or later (64-bit)**: Widely used for general computing, productivity, and gaming.
  - **macOS High Sierra or later (64-bit)**: Preferred by Apple users for development and creative work.
  - **Linux (64-bit)** (e.g., Ubuntu, Debian, CentOS, Fedora): Ideal for server hosting, development, and customization.
- **Programming Languages**:
  - **Python 3.7 or later (64-bit)**: Versatile language for web development, data science, and automation.
  - **Python 3.8 or later (64-bit)** (recommended): Enhanced performance and features.
- **Libraries and Frameworks**:
  - **PyTorch**: Deep learning library for research and model development.
  - **Torchvision**: Provides tools for computer vision tasks within PyTorch.
  - **Transformers**: For natural language processing (NLP) using pre-trained models.
  - **PyYAML**: Handles YAML configuration files.
- **Other Libraries**:
  - **NumPy**: Essential for numerical computations in Python.
  - **SciPy**: Extends NumPy with additional scientific functions.
  - **Pandas**: Data manipulation and analysis.
  - **Matplotlib**: Plotting and visualization.
  - **Scikit-learn**: Machine learning algorithms.
- **Deep Learning Frameworks**:
  - **TensorFlow**: Widely used for neural network development and deployment.
- **NLP Libraries**:
  - **NLTK**: Natural language processing toolkit for text analysis.
  - **spaCy**: Efficient NLP library for tasks like named entity recognition and part-of-speech tagging.
- **Computer Vision Libraries**:
  - **OpenCV**: Image and video processing library.
  - **Pillow**: Python Imaging Library for image manipulation.
- **Databases**:
  - **MySQL**: Relational database management system.

- **PostgreSQL**: Open-source relational database.

- **Other Software**:
  - **Jupyter Notebook**: Interactive environment for data exploration and analysis.
  - **Git**: Version control system for collaborative development.

CHAPTER 4
SYSTEM DESIGN

4.1 System Architecture

Storyteller System Overview

The Storyteller system is a cutting-edge, AI-powered platform designed to generate fully animated video stories from user-inputted prompts. This innovative system leverages three core AI technologies to create engaging and immersive storytelling experiences.

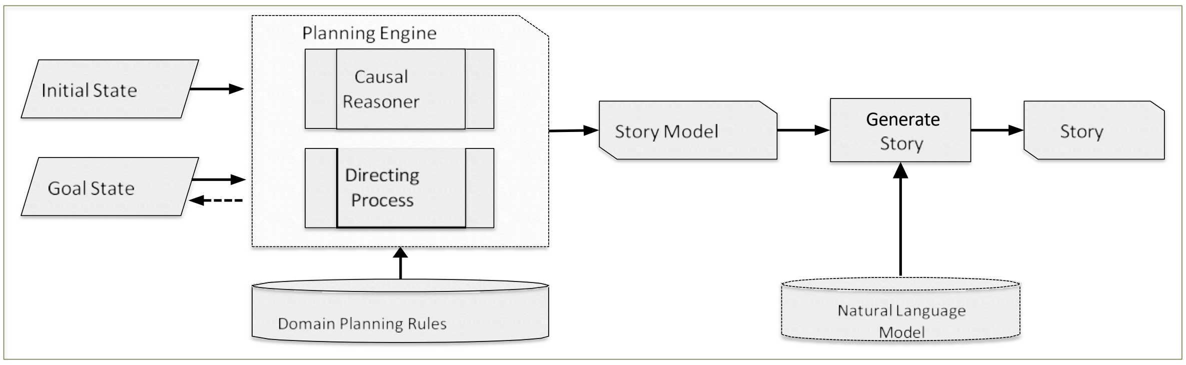

Figure 4.1:System Overview

Core AI Technologies

- **Generative Pre-trained Transformers (GPT):** GPT is a sophisticated language model that generates cohesive and contextually relevant narrative text based on user input.
- **Stable Diffusion:** Stable Diffusion is a state-of-the-art image synthesis model that translates text into stunning visual imagery, creating a unique and captivating visual representation of each scene.
- **Neural Text-to-Speech (TTS):** The neural TTS system converts the generated text into a lifelike voiceover, adding an extra layer of realism and emotional depth to the storytelling experience.

System Components

- **Input Interface:** Users provide a story prompt via a simple text input, initiating the storytelling process.
- **Story Generation Module:** The GPT model processes the input prompt, crafting a cohesive and engaging narrative, outputting text for each part of the story.
- **Image Synthesis Module:** Stable Diffusion translates the generated text into visual imagery for each segment of the narrative, creating a unique and captivating visual representation of each scene.
- **Narration Synthesis Module**: The neural TTS system converts the generated text into a lifelike voiceover, adding an extra layer of realism and emotional depth to the storytelling experience.
- **Output Assembly:** A video editor combines the generated text, images, and audio into a final animated video story, which is then rendered and saved in the output directory.

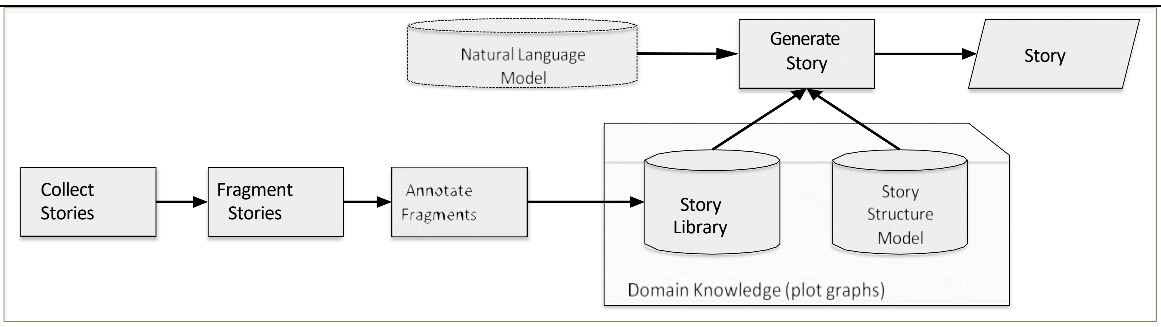

Figure 4.2:System Technologies

Workflow

The workflow proceeds as follows:

- ➢ The user inputs a prompt to initiate the story.
- ➢ The GPT model takes the prompt and crafts a narrative, outputting text for each part of the story.
- ➢ For each piece of text, the Stable Diffusion model generates an image that visually represents the scene.
- ➢ Simultaneously, the neural TTS model narrates the story, providing an audio file for each segment.
- ➢ Finally, the video editor compiles the text, images, and audio into a cohesive video story, which is then rendered and saved in the output directory.

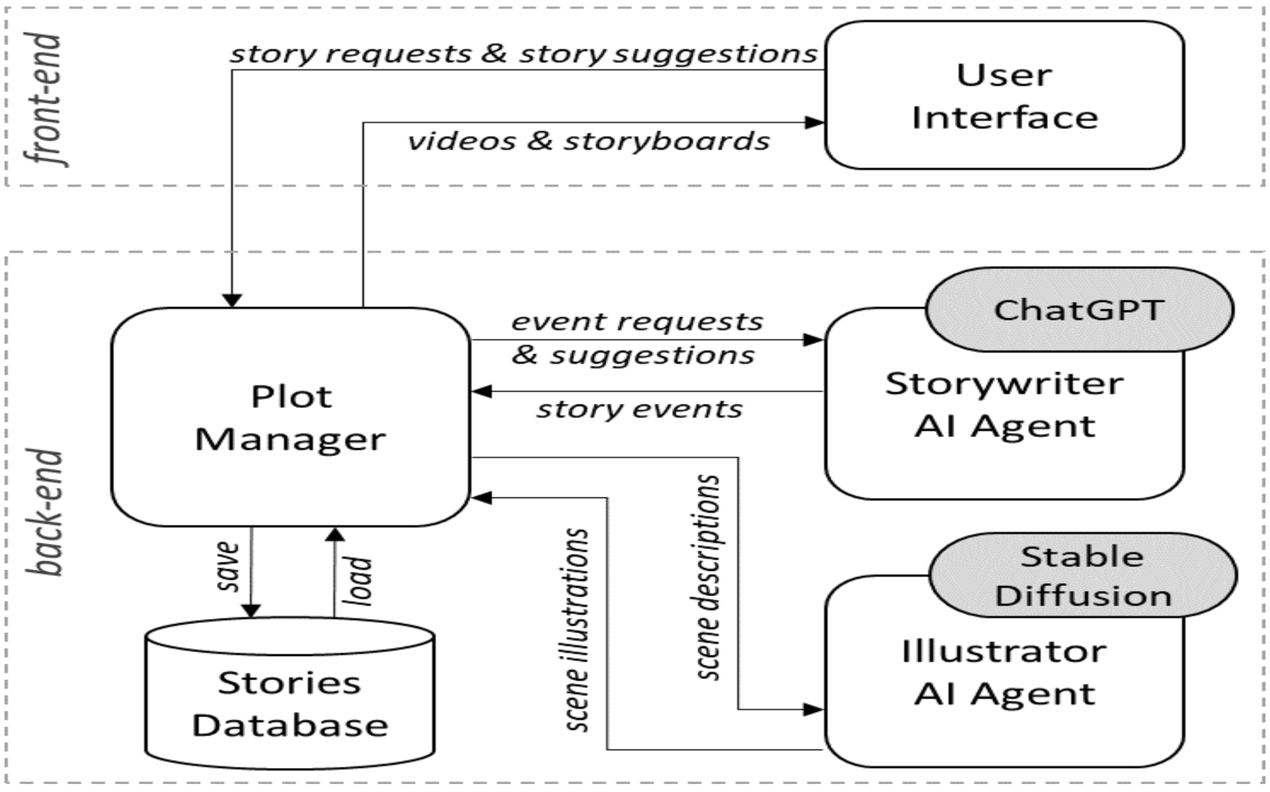

Figure 4.3:System WorkFlow

Output

The system produces a fully animated video story, complete with audio and visuals, ready for sharing and viewing. Intermediate files such as images, audio clips, and subtitles are also provided, allowing users to customize and refine their storytelling experience.

Advanced Features

- **Customizable Storytelling:** Users can adjust various parameters, such as tone, genre, and style, to tailor the storytelling experience to their preferences.
- **Real-time Feedback:** The system provides real-time feedback and suggestions to users, helping them refine their storytelling skills and create more engaging narratives.
- **Collaborative Storytelling:** Multiple users can collaborate on a single story, fostering a sense of community and creative exchange.

By leveraging the power of AI and machine learning, the Storyteller system revolutionizes the art of storytelling, making it more accessible, engaging, and immersive than ever before.

4.2 Flow Diagram

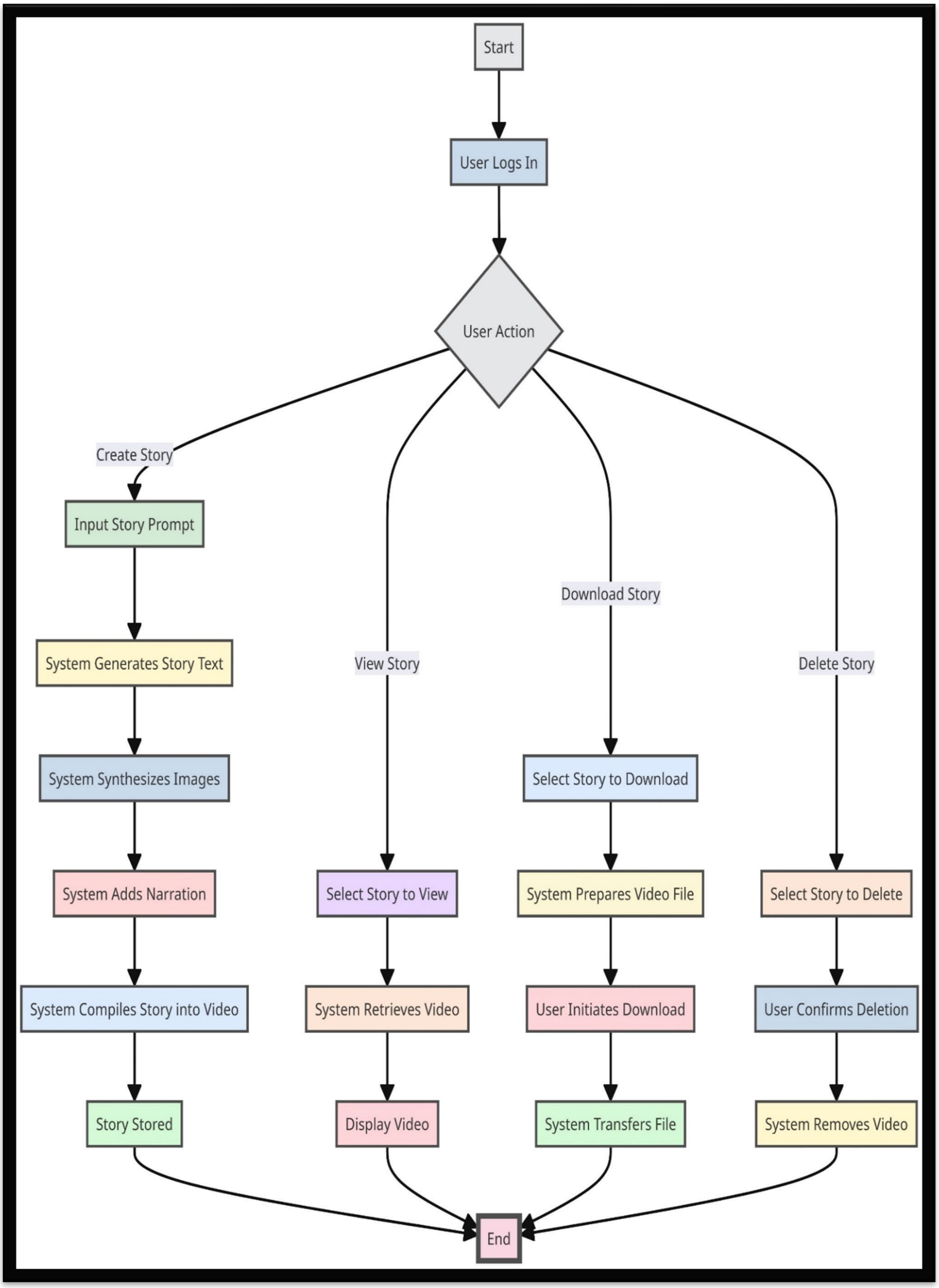

Figure 4.4:FlowChart

CHAPTER 5

IMPLEMENTATION

5.1 __init__.py

```
from importlib import metadata

from storyteller.config import StoryTellerConfig
from storyteller.model import StoryTeller

__version__ = metadata.version("storyteller-core")
__all__ = ["__version__", "StoryTellerConfig", "StoryTeller"]
```

This Python code defines the public interface for the "storyteller-core" package. It primarily does two things:

- **Imports necessary classes and defines the package version:**
  - It imports metadata from **importlib** to fetch the version of the "storyteller-core" package.
  - Imports **StoryTellerConfig** from ***storyteller.config*** and StoryTeller from storyteller. model.
  - It ***uses metadata.version("storyteller-core")*** to retrieve the version information and assigns it to the _version_ variable.
- **Defines the public API of the package:**
  - It creates a list named ***_all_*** containing the names of all public objects within the module.
  - This list includes the ***__version__ variable, StoryTellerConfig class,*** and ***StoryTeller class.***
  - **Ensures compatibility:** By importing specific classes and functions, the code ensures that the necessary components are available and compatible with the rest of the package.
  - **Centralized version management:** Using metadata.version("storyteller-core") allows for centralized management of the package version, making it easier to update and maintain.
  - **Modular design:** Importing classes from different modules (storyteller.config and storyteller.model) promotes a modular design, making the codebase more organized and maintainable.

5.2 config.py

- **StoryTellerConfigDefaults class**

StoryTellerConfigDefaults:

This dataclass defines default values for various configurations related to different models. These defaults include:

```
@dataclass(frozen=True)
class StoryTellerConfigDefaults:

    MAX_NEW_TOKENS: int = 50
    WRITER_MODEL: str = "gpt2"
    PAINTER_MODEL: str = "stabilityai/stable-diffusion-2"
    SPEAKER_MODEL: str = "tts_models/en/ljspeech/glow-tts"
    WRITER_DEVICE: str = "cpu"
    PAINTER_DEVICE: str = "cpu"
    SPEAKER_DEVICE: str = "cpu"
    WRITER_DTYPE: str = "float32"
    PAINTER_DTYPE: str = "float32"
    ENABLE_ATTENTION_SLICING: bool = True
    USE_DPM_SOLVER: bool = True
    NUM_PAINTER_STEPS: int = 20

    class ArgparseDefaults:

        WRITER_PROMPT: str = "unicorns
        once,roamed the earth"

        PAINTER_PROMPT: str = "Beautiful
        Painting"

        OUTPUT_DIR: str = "out"
        NUM_IMAGES: int = 2

        SEED: int = 42
```

- **MAX_NEW_TOKENS:** The maximum number of new tokens to be generated by the text model.
- **WRITER_MODEL, PAINTER_MODEL, SPEAKER_MODEL**: The names of the models to be used for text, image, and audio generation respectively.
- **WRITER_DEVICE, PAINTER_DEVICE, SPEAKER_DEVICE**: Devices to run the models (CPU or GPU).
- **WRITER_DTYPE, PAINTER_DTYPE**: Data types to use for the models.
- **ENABLE_ATTENTION_SLICING**: Whether to use attention slicing for image generation.
- **USE_DPM_SOLVER**: Whether to use a faster DPM solver for image generation.
- **NUM_PAINTER_STEPS**: Number of steps to run for image generation.

- **StoryTellerConfig:**

StoryTellerConfig:

This dataclass inherits from StoryTellerConfigDefaults and allows users to override the default values. It performs some validation on the configuration parameters.

```
@dataclass
class StoryTellerConfig:
    max_new_tokens: int = StoryTellerConfigDefaults.MAX_NEW_TOKENS
    writer: str = StoryTellerConfigDefaults.WRITER_MODEL
    painter: str = StoryTellerConfigDefaults.PAINTER_MODEL
    speaker: str = StoryTellerConfigDefaults.SPEAKER_MODEL
    writer_device: str = StoryTellerConfigDefaults.WRITER_DEVICE
    painter_device: str = StoryTellerConfigDefaults.PAINTER_DEVICE
    speaker_device: str = StoryTellerConfigDefaults.SPEAKER_DEVICE
    writer_dtype: str = StoryTellerConfigDefaults.WRITER_DTYPE
    painter_dtype: str = StoryTellerConfigDefaults.PAINTER_DTYPE
    enable_attention_slicing: bool =.ENABLE_ATTENTION_SLICING
    use_dpm_solver: bool = StoryTellerConfigDefaults.USE_DPM_SOLVER
    num_painter_steps: int = StoryTellerConfigDefaults.NUM_PAINTER_STEPS

    def __post_init__(self):
        if not hasattr(torch, self.writer_dtype):
            raise ValueError(f"Unsupported torch writer dtype {self.writer_dtype}")
        if not hasattr(torch, self.painter_dtype):
            raise ValueError(f"Unsupported torch paint dtype {self.painter_dtype}")
```

- **_post_init_** method checks if the specified data types (WRITER_DTYPE, PAINTER_DTYPE) are supported by PyTorch.

StoryTellerConfigArgparseHelpText:

This class provides helpful text descriptions for each configuration parameter, which can be used for command-line argument parsing.

```
class StoryTellerConfigArgparseHelpText:
    @staticmethod
    def _get_dataclass_var_name_from_f_string_eq(expression: str) -> str:
        return expression.split(".")[1].split("=")[0]

    # importing from typing for backwards compatibility
    _argparse_help_text: Dict[str, str] = {
        _get_dataclass_var_name_from_f_string_eq(
            f"{StoryTellerConfig.max_new_tokens=}"
        ): f"Maximum number of new tokens to generate in the writer model. Default:
{StoryTellerConfigDefaults.MAX_NEW_TOKENS}",
         _get_dataclass_var_name_from_f_string_eq(
            f"{StoryTellerConfig.writer=}"
        ): f"Text generation model to use. Default: '{StoryTellerConfigDefaults.WRITER_MODEL}'",
        _get_dataclass_var_name_from_f_string_eq(
            f"{StoryTellerConfig.painter=}"
        ): f"Image generation model to use. Default: '{StoryTellerConfigDefaults.PAINTER_MODEL}'",
        _get_dataclass_var_name_from_f_string_eq(
            f"{StoryTellerConfig.speaker=}"
```

- **_get_dataclass_var_name_from_f_string_eq** extracts the variable name from a string that assigns a value to it.
- **_argparse_help_text** dictionary stores the description for each configuration parameter.
- **get_help_text_for_var_name** retrieves the description for a given variable name.

➢ model.py

The StoryTeller class is a Python module that generates multimedia stories from text prompts. Here's a brief overview of how it works:

Initialization

The class is initialized with a StoryTellerConfig object, which sets up the necessary components:

```
def __init__(self, config: StoryTellerConfig):
    self.config = config
    writer_device = torch.device(config.writer_device)
    painter_device = torch.device(config.writer_device)
    speaker_device = torch.device(config.speaker_device)
    self.writer = pipeline("text-generation", model=config.writer, device=writer_device,
torch_dtype=getattr(torch, config.writer_dtype))
    self.painter = DiffusionPipeline.from_pretrained(config.painter, use_auth_token=False,
torch_dtype=getattr(torch, config.painter_dtype)).to(painter_device)
    if config.use_dpm_solver:
        self.painter.scheduler = DPMSolverMultistepScheduler.from_config(self.painter.scheduler.config)
    if config.enable_attention_slicing:
        self.painter.enable_attention_slicing()
    self.speaker = TTS(config.speaker).to(speaker_device)
    self.sample_rate = self.speaker.synthesizer.output_sample_rate
    self.output_dir = None
```

- **Text Generation**

The write method generates text based on a given prompt:

```
torch.inference_mode()
```

```
def write(self, prompt: str) -> str:
return self.writer(prompt, max_new_tokens=self.config.max_new_tokens)[0]["generated_text"]
```

- **Image Generation**

The paint method generates an image from a given prompt:

```
@torch.inference_mode()
def paint(self, prompt: str) -> Image:
return self.painter(prompt, num_inference_steps=self.config.num_painter_steps).images[0]
```

- **Speech Synthesis**

The speak method synthesizes audio from a given prompt:

```
@torch.inference_mode()
def speak(self, prompt: str) -> List[int]:
return self.speaker.tts(prompt)
```

- **Video Generation**

The generate method generates a multimedia story from a given writer prompt, image prompt prefix, and desired number of images:

```
def generate(self, writer_prompt: str, painter_prompt_prefix: str, num_images: int, output_dir: str) ->
None:
    video_paths = []
    self.output_dir = output_dir
    os.makedirs(output_dir, exist_ok=True)
    sentences = self.write_story(writer_prompt, num_images)
    for i, sentence in enumerate(sentences):
        video_path = self._generate(i, sentence, painter_prompt_prefix)
        video_paths.append(video_path)
    self.concat_videos(video_paths)
```

The _generate method creates a single video segment by generating an image, synthesizing audio, and combining them with subtitles:

```
def _generate(self, id_: int, sentence: str, painter_prompt_prefix: str) -> str:
    image_path = self.get_output_path(f"{id_}.png")
    audio_path = self.get_output_path(f"{id_}.wav")
    subtitle_path = self.get_output_path(f"{id_}.srt")
    video_path = self.get_output_path(f"{id_}.mp4")
    image = self.paint(f"{painter_prompt_prefix} {sentence}")
    image.save(image_path)
    audio = self.speak(sentence)
    duration, remainder = divmod(len(audio), self.sample_rate)
    if remainder:
        duration += 1
        audio.extend([0] * (self.sample_rate - remainder))
    sf.write(audio_path, audio, self.sample_rate)
    subtitle = f"0\n{make_timeline_string(0, duration)}\n{{\\fs5}}{sentence}"
    with open(subtitle_path, "w+") as f:
        f.write(subtitle)
    subtitle_path = subtitle_path.replace("\\", "/")
    subprocess_run(f"ffmpeg -loop 1 -i {image_path} -i {audio_path} -vf subtitles={subtitle_path} -tune stillimage -shortest {video_path}")
    return video_path
```

The **concat_videos** method concatenates the individual video segments into a final output video:

```
def concat_videos(self, video_paths: List[str]) -> None:
    files_path = self.get_output_path("files.txt")
    output_path = self.get_output_path("out.mp4")
    with open(files_path, "w+") as f:
        for video_path in video_paths:
            f.write(f"file   {os.path.split(video_path)[-1]}\n")
    subprocess_run(f"ffmpeg -f concat -i {files_path} -c copy {output_path}")
```

CHAPTER 6

RESULTS

Once upon a time, unicorns roamed the Earth

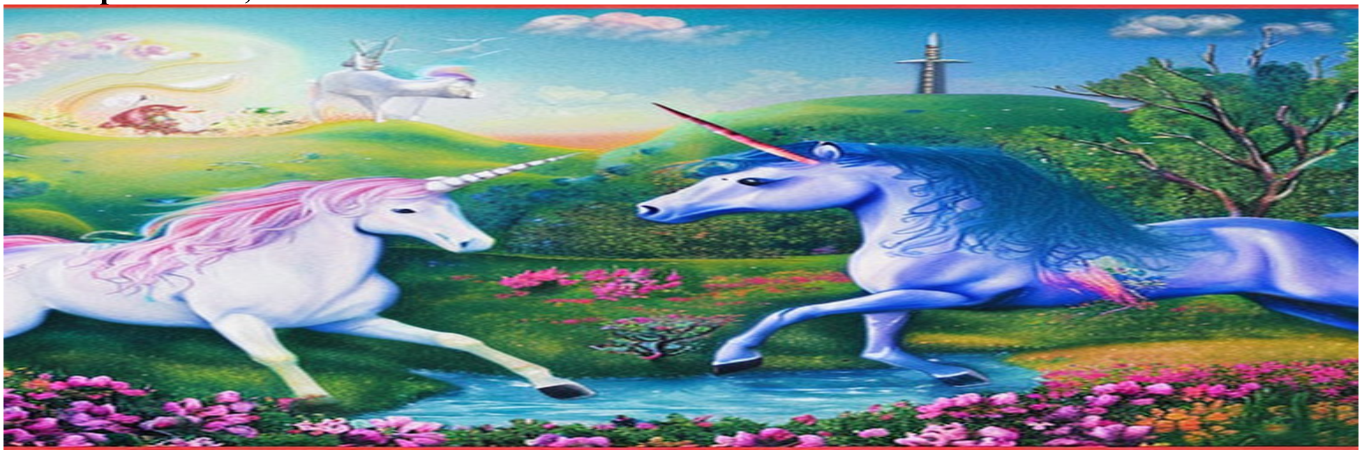

Figure 6.1:Result 01

On other species that were still alive during the 19th century, the inhabitants encountered these creatures in the sea where they would die just like a mortal human.

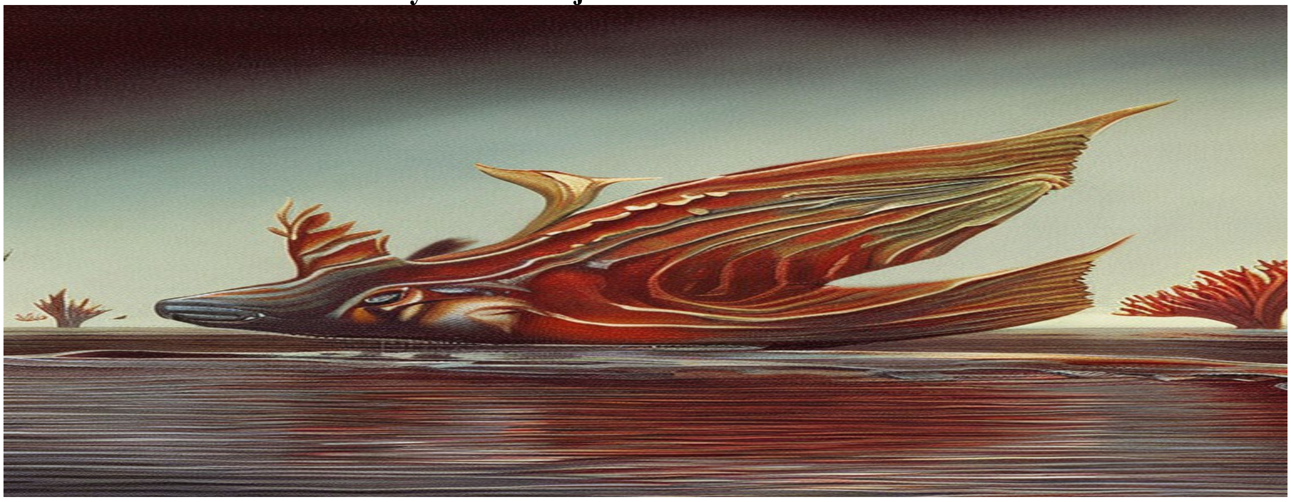

Figure 6.2:Result 02

The inhabitants would eat food such as turtles, crabs, seaweeds, and other living things such as insects and animals.

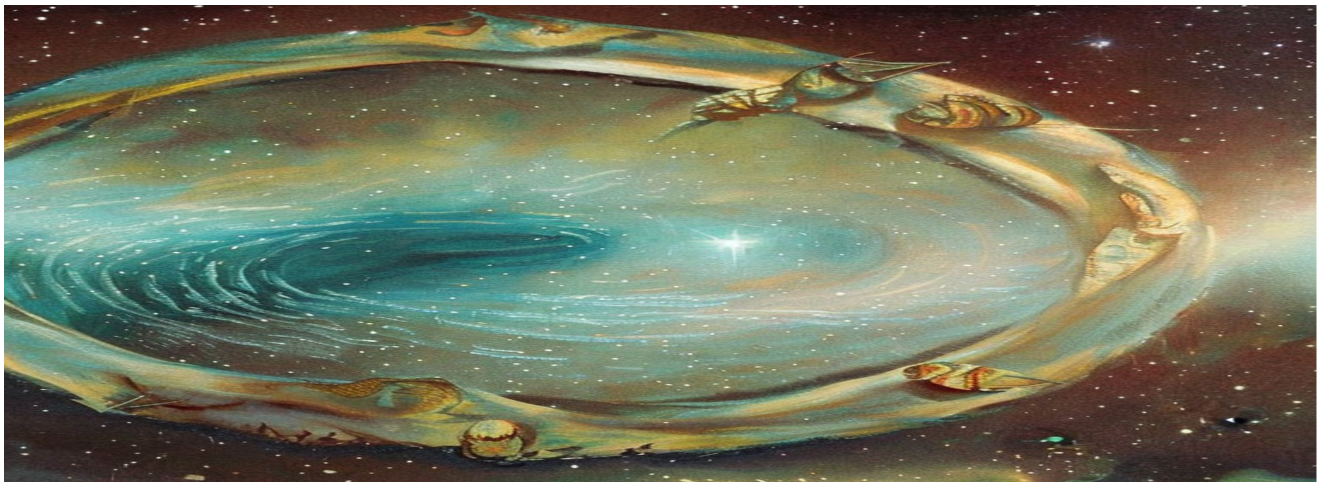

Figure 6.3:Result 03

The inhabitants would eat food such as turtles, crabs, seaweeds, and other living things such as insects and animals.

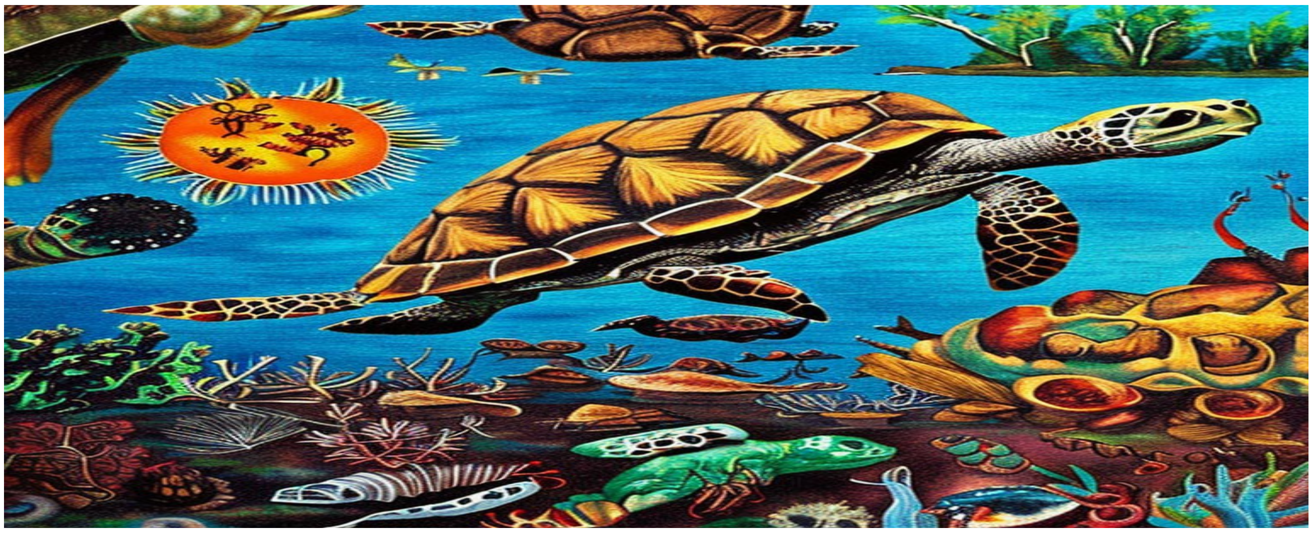

Figure 6.4:Result 04

Living things were not a source of sustenance for most of the inhabitants, as it had been only for some time.

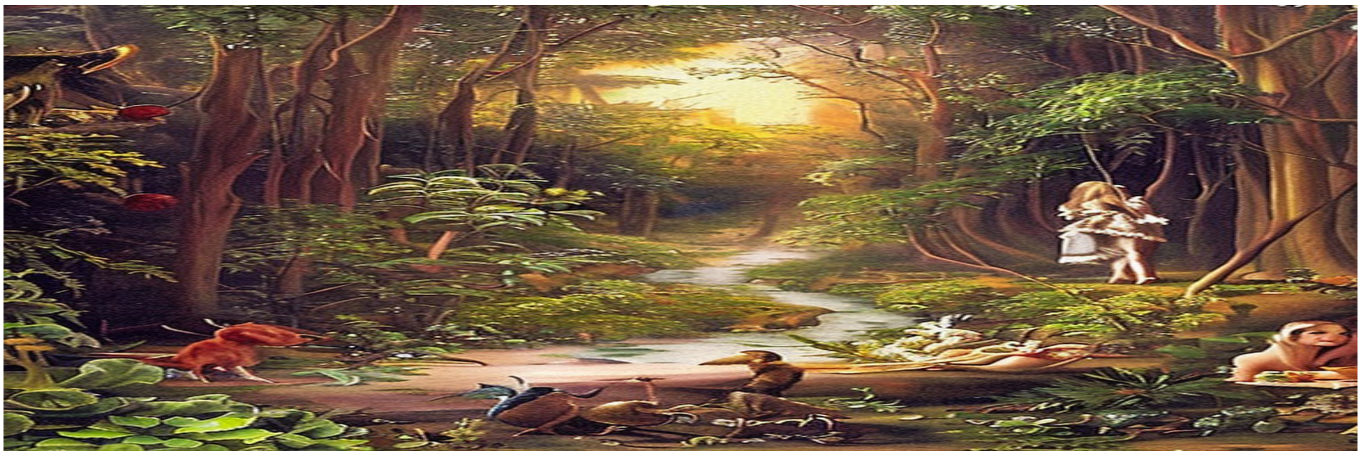

Figure 6.5:Result 05

When the colonists settled on a land of "living" worlds, they encountered the remnants of their native planet but were defeated.

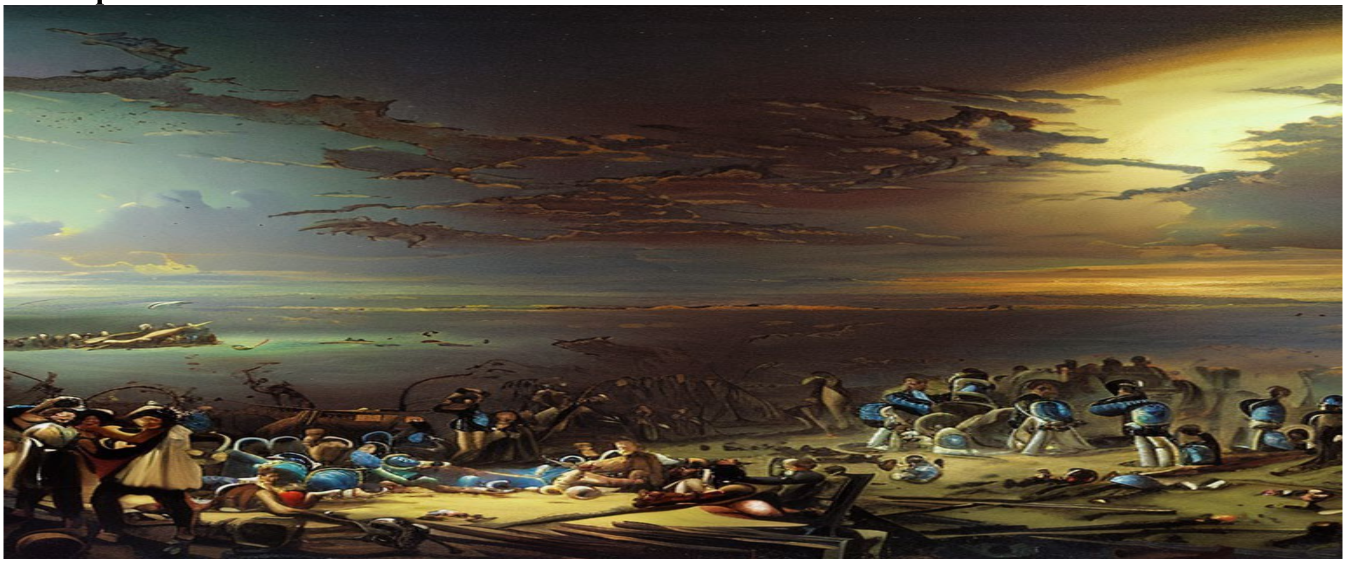

Figure 6.6:Result 06

At some point after the colonizers left Earth and returned to the planet they left behind, the inhabitants of a newly discovered colony colony from the 22nd century and the last time they fought was a centuries later, when they captured a few of their former slaves.

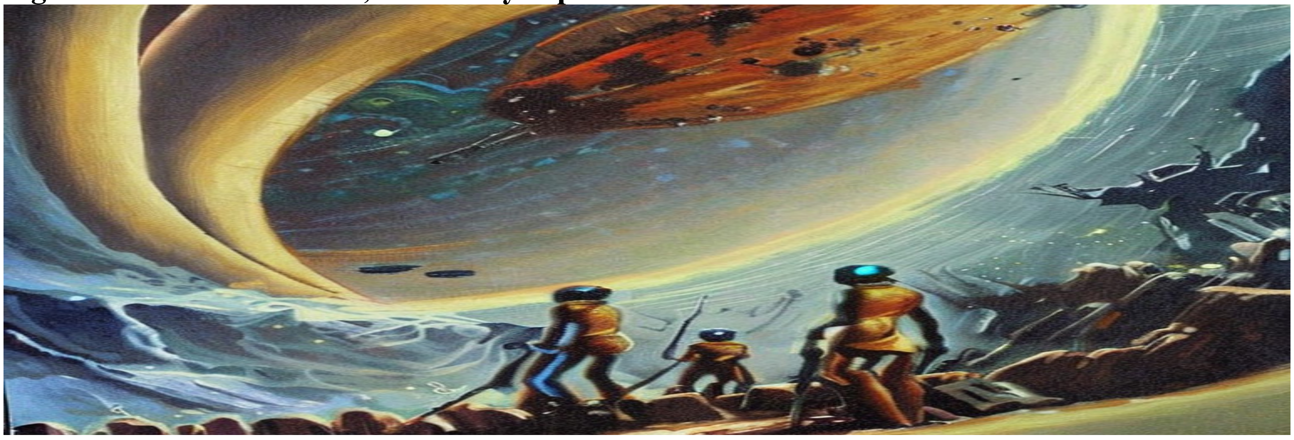

Figure 6.7:Result 07

Some time before the 21st century, the inhabitants of the Earth were captured on a space mission to find a planet where humans were going extinct.

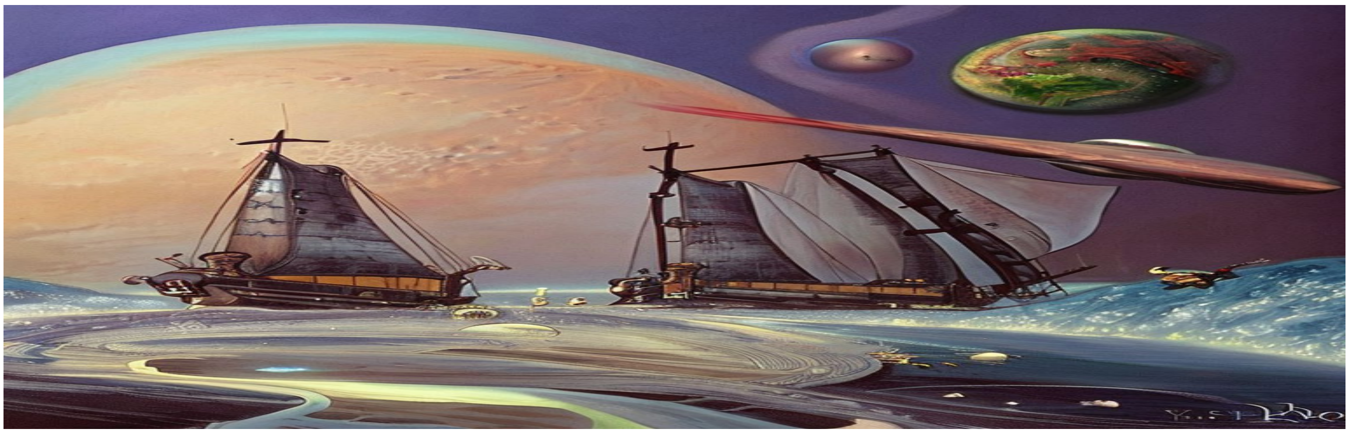

Figure 6.8:Result 08

Following the return from Earth, all of the inhabitants of the Earth would die while the planet remained alive.

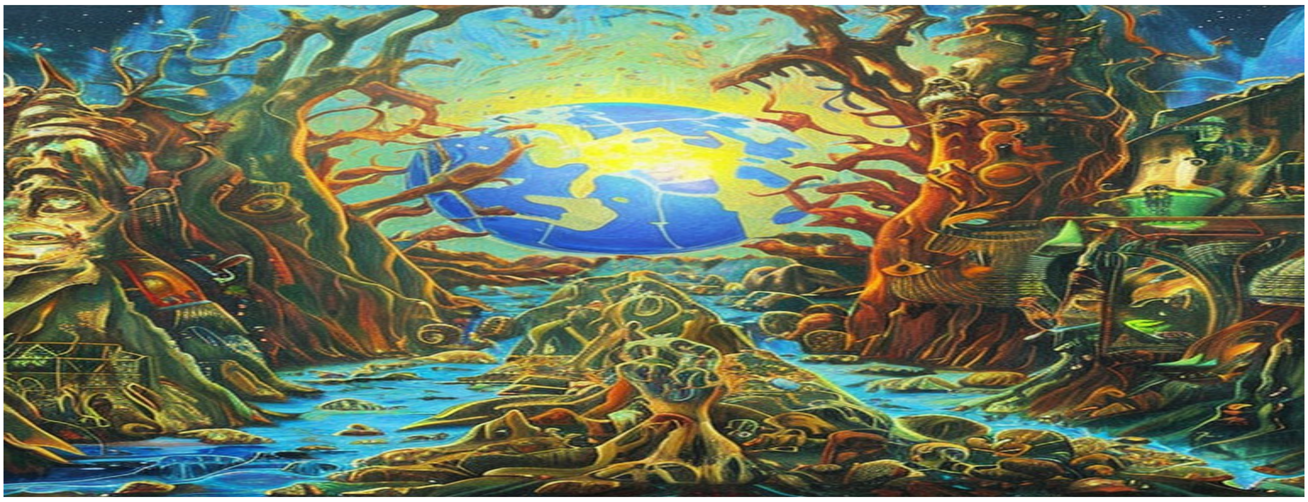

Figure 6.9:Result 09

CHAPTER 7
CONCLUSION

The AI-STORY-GENERATOR project is a remarkable endeavor that has made significant strides in the realm of digital storytelling. Here’s an enhanced conclusion that ties directly to the project:

➢ Achievements:

- The AI-STORY-GENERATOR project has successfully harnessed the synergy of text, image, and audio modalities to craft immersive and cohesive narratives.
- This initiative exemplifies the transformative power of AI in the creative domain, offering a vivid demonstration of how technology can serve as a catalyst for human ingenuity.

➢ Challenges and Learnings:

- The journey of the project was marked by challenges in maintaining the coherence and contextual relevance of content generated across various modalities.
- The experience has imparted crucial insights into the intricacies of fine-tuning AI models and devising user-centric interfaces that facilitate interaction and personalization.

➢ Future Directions:

- Looking ahead, the project paves the way for continued exploration and enhancement, focusing on elevating the caliber of generated content and diversifying the spectrum of storytelling styles and genres.
- The platform holds promise for application in educational contexts, cultural heritage preservation, and as an invaluable resource for professional storytellers and content creators.

➢ Community Impact:

- Embracing the spirit of open-source collaboration, the project extends an invitation to developers, writers, and artists across the globe to contribute and co-create.
- It has the potential to cultivate a vibrant community centered on AI-assisted storytelling, propelling a wave of innovation and collective learning.

In its essence, the AI-STORY-GENERATOR stands as a testament to the dynamic interplay between AI and human creativity, offering a window into the future of digital narratives. It represents a leap forward in the storytelling tradition, where tales are not merely recounted but also visualized and heard, thus enriching the tapestry of storytelling with the prowess of contemporary technology.

CHAPTER 8

FUTURE SCOPE

➢ **Future Scope:**

- **Improving Story Coherence and Contextual Understanding**: The project can focus on enhancing the story generator's ability to understand context, maintain coherence, and create more engaging narratives.

- **Expanding Domain Knowledge and Vocabulary**: The AI Story Generator can be trained on a broader range of domains, genres, and styles to increase its vocabulary and generate more diverse stories.

- **User Interaction and Personalization**: The project can explore integrating user input, preferences, and feedback to create personalized stories, making the experience more immersive and enjoyable.

- **Multimodal Storytelling**: The AI Story Generator can be extended to incorporate multimedia elements, such as images, videos, or audio, to create a more immersive storytelling experience.

- **Real-time Story Generation and Collaboration**: The project can focus on developing real-time story generation capabilities, enabling multiple users to collaborate on storytelling and creating a more dynamic experience.

- **Evaluation Metrics and Story Quality Assessment**: The project can develop and refine evaluation metrics to assess the quality and coherence of generated stories, ensuring the AI Story Generator produces high-quality narratives.

- **Deployment and Integration with Other Platforms**: The AI Story Generator can be deployed as a web application, mobile app, or integrated with other platforms, such as writing tools, educational software, or entertainment platforms.

REFERENCES

[1] Ekagrashukla. (n.d.). *GitHub - ekagrashukla/GRANNY-GPT2-Based-Multiple_Genre-Story-Generator-Web-Application:* A Web Application to generate multiple genre stories based on Open AI's GPT-2 Model. GitHub. *https://github.com/ekagrashukla/GRANNY-GPT2-Based-Multiple_Genre-Story-Generator-Web-Application*

[2] Karan. (n.d.). AI-STORY-GENERATOR · karan2003/AI-STORY-GENERATOR. GitHub *https://github.com/karan2003/AI-STORY-GENERATOR/tree/main/storyteller-master/storyteller*

[3] Seaweiqing. (n.d.). GitHub - seaweiqing/image2story: A project for telling stories according to images in some particular style. GitHub. *https://github.com/seaweiqing/image2story*

[4] JeremiahPetersen. (n.d.). *GitHub - JeremiahPetersen/Auto-Story-GPT:* Generate AI responses and create a cohesive story based on characters and themes you create. GitHub. *https://github.com/JeremiahPetersen/Auto-Story-GPT*

[5] ChatGPT - Open A i GPT 3.5. (n.d.). ChatGPT. *https://chatgpt.com/g/g-F00faAwkE-open-a-i-gpt-3-5*

[6] Stability AI - Developer Platform. (n.d.). *https://platform.stability.ai/docs/api-reference*

[7] Text to Speech – Realistic AI Voice Generator | Microsoft Azure. (n.d.). *https://azure.microsoft.com/en-in/products/ai-services/text-to-speech*

[8] GeeksforGeeks. (2024, May 28). Convert text to speech in Python. *GeeksforGeeks. https://www.geeksforgeeks.org/convert-text-speech-python*

[9] GeeksforGeeks. (2024a, March 28). Generate Images from Text in Python Stable Diffusion. GeeksforGeeks. *https://www.geeksforgeeks.org/generate-images-from-text-in-python-stable-diffusion*

[10] GeeksforGeeks. (2023c, June 6).Diffusion Models GeeksforGeeks. *https://www.geeksforgeeks.org/what-are-diffusion-models*