AI-POWERED IMAGE-TO-AUDIO STORY GENERATOR

Session (2021-2025)

Bachelor of Studies in Computer Science

Submitted By

Ahmer Kamal	**301-211005**
M. Hamza Javed	**301 -211008**

Supervised By

Dr. Amanullah Baloch

Assistant Professor,

Department of CS & IT

HAZARA UNIVERSITY, MANSEHRA

DECLARATION

We hereby declare that this software, neither as a whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied the report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be a reproduction of some other. We will stand by the consequences. No portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

____

Ahmer Kamal

____

M.Hamza Javed

CERTIFICATE OF APPROVAL

It is to certify that the final year project of BSCS **"AI-Powered Image to Story Generator"** was developed by **Ahmer Kamal (301-211005)**, **Muhammad Hamza Javed (301-211008)**, under the supervision of **"Assistant Professor Dr. Amanullah Baloch"** and that in (their/his/her) opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Studies in Computer Science.

Committee

Supervisor: -------------------------------------

External Examiner: -------------------------------------

Head of Department: -------------------------------------

ACKNOWLEDGMENT

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge under which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor **"Dr. Amanullah Baloch"**. Without his personal supervision, advice, and valuable guidance, completion of this project would have been doubtful. We are deeply indebted to him for his encouragement and continual help during this work.

We are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

Dedicated

To

Our beloved parents, respected teachers.

Whose support has given us the guidance and fortitude to accomplish the project.

They are assets of our life.

May Allah bless them with a very happy, successful and a healthy life!

(Ameen)

PREFACE

This project report concerns the development of a "**AI-Powered Image-to-Audio Story Generator**". This report covers the complete information about Development and Deployment.

This report is divided into six chapters, each focusing on a key aspect of the project. Chapter One introduces the background, objectives, motivation, and scope of the Image-to-Audio Story Generator. Chapter Two provides a review of existing systems and technologies, highlighting the limitations that the proposed system addresses. Chapter Three outlines the requirement specification, including both functional and non-functional requirements essential to system development. Chapter Four details the design and architecture of the system, including model workflows and system diagrams. Chapter Five focuses on the implementation, testing procedures, and evaluation of system performance. Chapter Six includes references to the tools, frameworks, and academic resources used throughout the development of the project.

Table of Contents

List Of Figures

List Of Acronym

Acronym	Full Form
AI	Artificial Intelligence
API	Application Programming Interface
BLIP	Bootstrapped Language-Image Pretraining
CS	Computer Science
CPU	Central Processing Unit
GPU	Graphics Processing Unit
gTTS	Google Text-to-Speech
HLD	High-Level Design
IT	Information Technology
LLaMA	Large Language Model Meta AI
LLD	Low-Level Design
SRS	Software Requirement Specification
TTS	Text-to-Speech
UI	User Interface
VLP	Vision-Language Pretraining
VQA	Vision-Language Question Answering / Visual Question Answering
ViT	Vision Transformer

Chapter 1

Introduction

1.1 Introduction

The quick advancement in artificial intelligence opened new ways for creative expression and human-computer interaction. One such area is the generation of creative stories from visual content using multimodal AI. This project, titled Image-to-Audio Story Generator, leverages state-of-the-art machine learning models to convert an input image into a narrated short story. By combining image captioning, text generation, and speech synthesis technologies, the system offers a seamless pipeline from visual input to audio output. This approach not only showcases AI's generative capabilities but also supports applications in storytelling, education, and accessibility.

1.2 Scope of the Project

This project's application domain includes education, digital media, and assistive technologies in Pakistan. In particular, it can serve as an engaging tool for children in rural schools, enhancing their learning through audio storytelling. Narrating stories based on images may also benefit visually impaired individuals.

1.3 Feasibility of the Project

1.3.1 Economic Feasibility

This project primarily utilizes open-source tools and free-tier APIs, making it cost-effective for development and testing.

1.3.2 Technical Feasibility

- **Hardware:** A system with a modern CPU and moderate GPU is sufficient.
- **Software:** The project uses Gradio for the frontend, Salesforce's BLIP model for image captioning, Together AI's LLaMA for story generation, and Google Text-to-Speech (gTTS) for audio output.
- **Expertise:** Tools used in this project are well-documented, and basic to intermediate knowledge of Python and machine learning is sufficient to implement the system.
- **Recurrent Costs:** Minimal recurrent costs are expected, mainly related to API usage and storage for audio files.

1.4 Tools

- **Gradio:** A Python library used for creating web-based interfaces for machine learning models.
- **BLIP (Bootstrapped Language Image Pretraining):** Used for generating descriptive captions from input images.
- **LLaMA (Large Language Model Meta AI) via Together API:** Generates realistic short stories based on the image caption.
- **gTTS (Google Text-to-Speech):** Converts the text story into spoken audio.
- **Together API:** Provides access to powerful LLMs including LLaMA 3.1 for natural language generation.

1.5 Report

This report presents a comprehensive overview of the Image-to-Audio Story Generator project. Chapter 1 introduces the project, its relevance, and the tools used. Chapter 2 outlines the requirement specifications, highlighting the limitations of any existing systems and the proposed solution. Chapter 3 details the system's architecture and design, including high and low-level designs. Chapter 4 covers the testing methodology and scenarios. Chapter 5 provides screenshots and results from the implemented system. Chapter 6 offers a reference and training manual for users. The report concludes with references to the tools and libraries used.

Chapter 2

Requirement Specification

2.1 Existing system

In the existing systems, the process of generating a story from an image is not fully automated or integrated. Typically, users are required to manually describe the image or manually enter a prompt that defines the genre, setting, and other storytelling elements. This process lacks automation in terms of interpreting visual content and transforming it into a structured narrative. Furthermore, users must rely on third-party websites or services to convert the text into speech after generating the textual story (often through AI writing tools or manual writing). These platforms are often separate from the story generation tools, requiring users to copy-paste the content and manually initiate the audio generation process.

This fragmented workflow is time-consuming and inconvenient, and can lead to loss of context or formatting during transitions between tools. Additionally, such systems do not offer customization options or unified outputs in the form of image, story, and audio within a single interface.

2.2 Limitations of The Existing System

2.2.1 Lack of Automation

Users must manually analyze images and create story prompts, which introduces subjectivity and inconsistency in storytelling. The absence of automated image understanding restricts scalability and user convenience.

2.2.2 Fragmented Workflow

Story generation and text-to-speech conversion are typically performed using separate tools. This disjointed process requires switching between multiple platforms, increasing the risk of data loss, formatting issues, and user frustration.

2.2.3 Limited Personalization

Existing systems rarely offer options to customize the tone, theme, conflict, or ending style of the story. This limits user engagement and the ability to produce diverse and creative outputs.

2.2.4 Inconsistent Output Quality

Manual prompt writing can lead to variable results depending on user input quality. Additionally, external TTS services may not support proper voice modulation, pacing, or natural-sounding speech.

2.2.5 Time-Consuming Process

The user must perform multiple steps—prompt writing, story generation, copying the result, accessing another TTS tool, and downloading audio—making the process inefficient, especially for repeated use.

2.2.6 Accessibility Issues

Users with limited technical expertise may find it challenging to operate different tools or understand how to align the outputs correctly, making the system less inclusive.

2.3 Proposed System

The AI-Powered Image-to-Audio Story Converter is designed to revolutionize storytelling by integrating cutting-edge artificial intelligence technologies. The system will

2.3.1 Analyze Images with Computer Vision

Leverage Salesforce's BLIP image captioning model to generate descriptive captions from uploaded images, identifying key objects, scenes, and contexts.

2.3.2 Generate Creative and Contextually Rich Narratives

Employ Together AI's LLaMA-based NLP models to transform image captions into personalized, engaging short stories based on user-defined preferences such as tone, genre, and setting.

2.3.3 Provide Customization Options

To create narratives that align with their vision and imagination, offer users the ability to customize storytelling elements, including genre, setting, conflict, tone, theme, and ending.

2.3.4 Deliver Audio Narratives

Use gTTS (Google Text-to-Speech) to convert generated stories into audio files, making storytelling accessible and immersive.

2.3.5 Deliver a User-Friendly Experience

Implement a Gradio-based interface for seamless interaction, allowing users to:

- Upload images easily
- Choose storytelling preferences from dropdown menus.
- Generate and listen to audio stories in a straightforward and accessible workflow.

This system simplifies complex AI processes into an intuitive, engaging user experience, fostering creativity and accessibility.

2.4 Model Architecture BLIP (Bootstrapped Language-Image Pretraining)

BLIP is a **Vision-Language Pretraining (VLP)** framework designed to excel in both **image understanding** and **text generation** tasks. Unlike prior VLP methods that perform well only on specific tasks or rely on noisy web-collected data, BLIP introduces a robust **bootstrapping mechanism** for caption quality and a **flexible model architecture** that supports multiple downstream applications. (see figure 2.1)

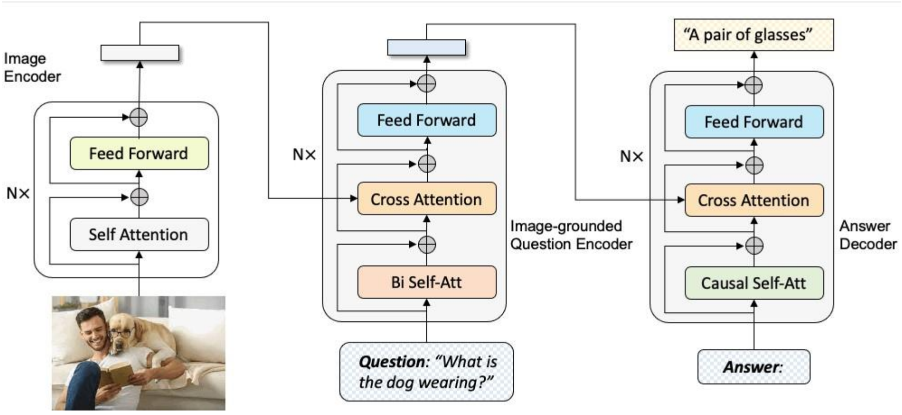


Figure 2.5 Model Architecture

The figure above illustrates a high-level architecture of a Vision-Language Question Answering (VQA) system. It integrates visual features from an image and textual information from a question to generate a relevant answer. The model is composed of three main components: an Image Encoder, an Image-grounded Question Encoder, and an Answer Decoder. Each module leverages attention mechanisms to capture contextual relationships within and across modalities. Below, each component is described in detail.

2.4.1 Key Components of BLIP Architecture

Vision Encoder (ViT - Vision Transformer)

- Processes the input image into feature representations.
- Typically based on Vision Transformer (ViT) pretrained with contrastive or masked

image modeling.

Text Encoder-Decoder (Transformer-based)

- Encodes captions and decodes them during generation tasks.
- Unified architecture for both understanding (e.g., retrieval, VQA) and generation (e.g., captioning).

Querying with a Captioner Filter Mechanism

- A **caption generator (captioner)** creates synthetic captions from noisy image-text web pairs.
- A **filter model** discards irrelevant or inaccurate captions.
- This self-bootstrapping allows for **better training supervision** with less noisy data.

Dual Training Objectives

- **Image-Text Contrastive Learning:** For understanding tasks (retrieval, classification).
- **Image Captioning Objective:** For generation tasks using teacher-forcing or autoregressive decoding.

2.5 Advantages of BLIP

- Excels at both **vision-language understanding** and **generation.**
- Adapts flexibly to tasks like image captioning, VQA (Visual Question Answering), and image-text retrieval.
- Enables **zero-shot** generalization to video-language tasks
- Avoids over-dependence on noisy web data by using **caption bootstrapping.**

2.6 Functional Requirements

The proposed system will consist of the following key functional components.

(see Figure 2.2)

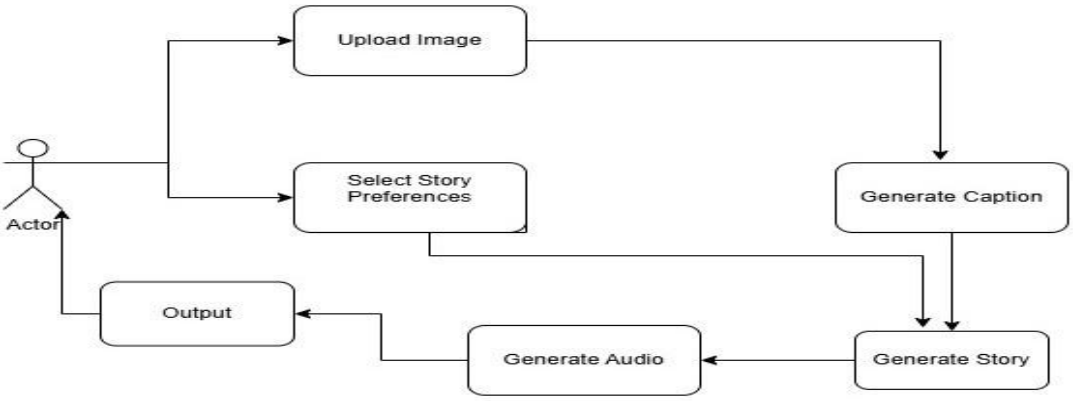


Figure 2.2 Functional Requirements

The diagram represents a system for automated story generation from images. An actor (user) uploads an image and selects story preferences. The system then generates a caption, which is used to create a story. Based on this story, audio is generated. Finally, all components (story and audio) are compiled and presented as output to the user.

2.6.1 Image Upload Interface

The system should allow users to upload an image (JPG, PNG, etc.).
The image will be processed as input for caption generation.

2.6.2 Image Captioning

The system must generate a caption based on the uploaded image using a pre-trained model (BLIP).
The caption will serve as the story prompt.
User Story Preferences Input.
Users should be able to select from predefined options for:
Genre (e.g., Mystery, Romance, Fantasy)
Setting (e.g., Modern day, Medieval)
Tone (e.g., Serious, Light-hearted)
Theme, Conflict Type, Twist, and Ending Style

2.6.3 Story Generation

The system will generate a short story (max 250 words) using a transformer-based language model (LLaMA) via Together API.
The story should align with the caption and user-selected preferences.

2.6.4 Audio Generation

The generated story should be converted into an audio file using the gTTS API.
The audio should be downloadable or playable within the application.

2.6.5 Display Outputs

The system should display the generated caption and story.
The audio player should allow playback of the narrated story.

2.7 Non-Functional Requirements

The proposed Image-to-Audio Story Generator must also satisfy several non-functional requirements to ensure quality, usability, and performance:

2.7.1 Performance

The system should be able to generate the caption, story, and audio output within a reasonable time

Story generation and audio synthesis should work efficiently, even on limited cloud resources

2.7.2 Scalability

The system architecture should be modular so that it can scale to include more customization options or deploy on cloud services with minimal changes.

Additional transformer models or APIs like ElevenLabs or Bark could be integrated in the future.

2.7.3 Availability

The application should maintain high availability during usage and should gracefully handle network/API

Proper exception handling should be in place to inform the user of any issues.

2.7.4 Usability

The interface should be simple, intuitive, and accessible to both technical and non-technical users.

Clear labels, dropdown menus, and media outputs should enhance the user experience.

2.7.5 Portability

The application should be platform-independent and runnable in environments like Google Colab, local Python environments, or simple web deployments.

It should not depend on proprietary software that limits installation or usage.

2.7.6 Maintainability

Code should be modular and well-documented, making it easy to update models, swap APIs, or improve the UI in future versions.

2.7.7 Security

The system should securely handle environment variables like API keys using .env files.

No sensitive user data is stored permanently

Chapter 3

Design of the Proposed System

3.1 System Architecture

The system architecture of the **AI-Powered Image-to-Audio Story Generator** consists of a modular pipeline that processes an input image and converts it into a narrated short story. This pipeline is divided into four main stages: **Image Upload**, **Image Captioning**, **Story Generation**, and **Text-to-Speech Conversion**, all orchestrated through a user-friendly web interface built with **Gradio**. (see Figure 3.1)

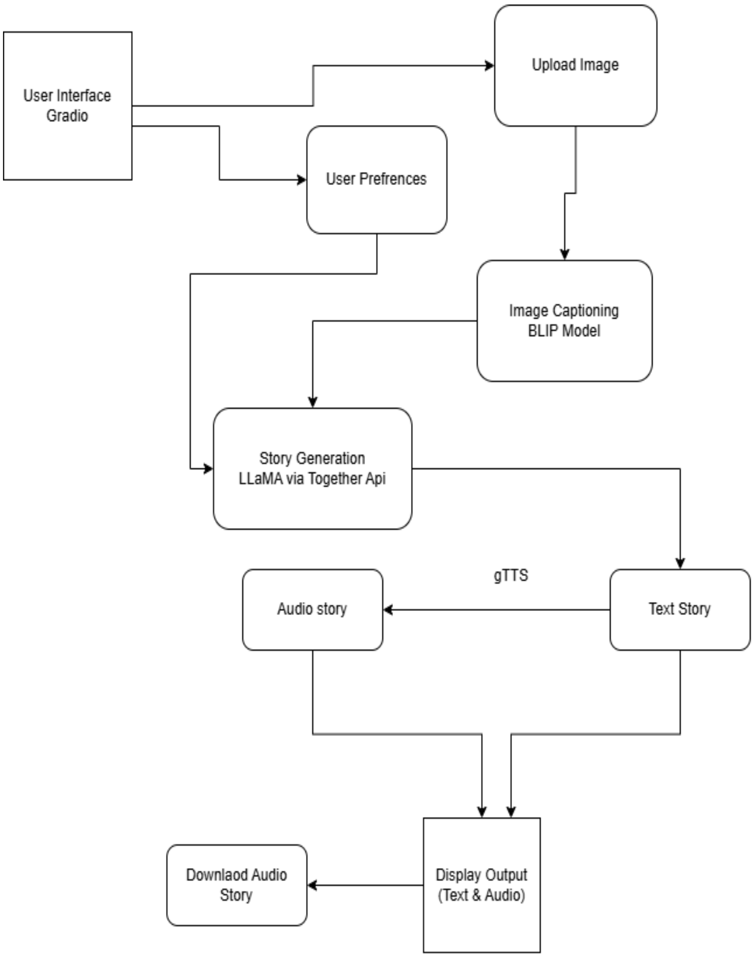


Figure 3-1: System Architecture

This diagram illustrates the complete functional flow of the system, from image upload through story generation and audio narration. It shows how user input is processed through BLIP for image captioning, LLaMA via the Together API for story generation, and gTTS for speech synthesis, with final output displayed and available for download.

1. User Interface (Gradio)

- User interacts with the system through a web-based UI built using Gradio.
- Users upload an image and optionally select preferences such as genre, tone, and story elements.

2. Image Upload

- The uploaded image is passed to the system as the primary input for processing.

3. Image Captioning (BLIP Model)

- The **BLIP (Bootstrapped Language-Image Pretraining)** model analyzes the image and generates a descriptive caption.
- This caption acts as the foundation or context for generating the story.

4. Story Generation (LLaMA via TogetherAI)

- The generated caption is passed to a **Large Language Model (LLaMA)** via the **Together AI API**.
- Based on the caption and user preferences, the model creates a short.
- The Generated Story will be up to 250 words.

5. Text-to-Speech (Google gTTS)

- Generated story text is converted into speech using **Google Text-to-Speech (gTTS)**.
- This produces a natural-sounding audio narration of the story.
- The story can be downloaded using the download button.

6. Output Display

- The final outputs include:

  - The **generated caption**
  - The **full story text**
  - The **narrated audio**

3.2 Design Constraints

While developing an **AI-Powered Image-to-Audio Story Generator**, several design constraints were taken into account to ensure the system remains efficient, accessible, and user-friendly. These constraints mainly relate to model hosting, performance considerations, and hardware requirements.

3.2.1 Model Hosting:

Story generation component of the system uses LLaMA (Large Language Model Meta AI), which is a highly capable but resource-intensive language model. Running the model locally would require significant hardware capabilities, including a powerful GPU and substantial memory. To address this challenge, the system uses the Together API, which provides access to LLaMA via the cloud.

This approach offers multiple advantages:

- Minimized local resource usage by offloading computation to remote servers.
- Improved scalability, allowing the system to handle more users concurrently.
- Simplified integration through the use of a ready-to-use API interface.

While this approach does rely on an external service, it necessitates a reliable internet connection. Additionally, it might encounter API usage restrictions and delays depending on the service agreement.

3.2.2 Performance Limitations:

Using a cloud-based API minimizes the reliance on high-performance local hardware, but it may result in slower response times because of network latency and the time it takes for model processing. Such delays become more apparent with longer or more intricate story prompts.

To improve responsiveness and ensure a seamless user experience, the following actions have been taken:

- Restricting the length of stories aids in minimizing the processing time for each request.

- The interface may feature loading indicators or utilize asynchronous handling to keep users updated during any delays.
- Whenever possible, caching can be implemented to save previously created stories and prevent redundant API requests.

3.2.3 Hardware Dependencies:

The image captioning process utilizes the BLIP model, which, although less resource-intensive than LLaMA, still gains from GPU acceleration. While it is entirely feasible to run BLIP on a CPU, doing so may result in reduced performance, particularly when handling larger or higher-resolution images.

To ensure versatility, the system is structured to:

- Accommodate CPU execution, guaranteeing it can function on typical hardware.
- Enhance performance by resizing or preprocessing images before captioning.
- Permit optional implementation in GPU-supported environments for quicker and more efficient processing.

3.3 High-Level Design

This High-Level Design (HLD) diagram illustrates the essential functional pathway of the Image to Story System, showcasing the primary system elements and their interactions in converting a user's image into a full audio storytelling experience. It depicts how data moves throughout the system without getting into the technical implementation specifics (see Figure 3.2).

Use Case Diagram

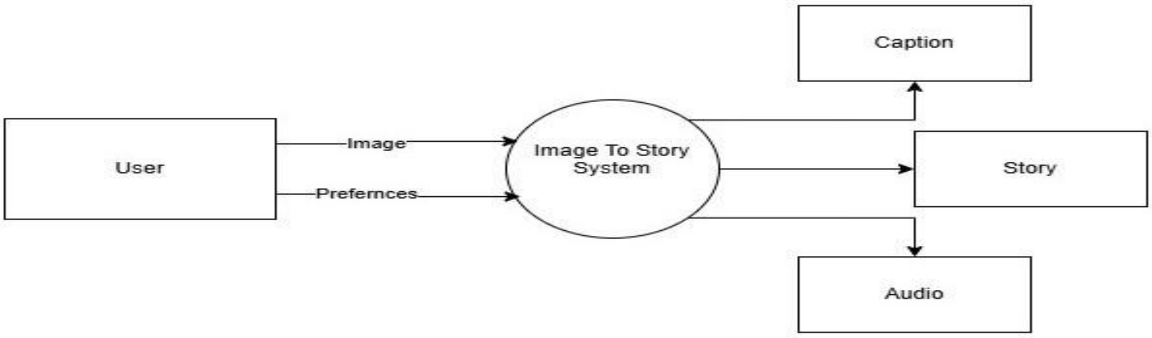


Figure 3-2: High-Level Design

This diagram outlines the user interaction with the system, including key processes such as image upload, caption generation, story creation, and audio narration, representing each use case as part of the system's logical flow.

3.3.1 User

a) The User is the primary actor interacting with the system.

b) The user submits two primary forms of input.

- **Image:** An image that the system will analyze and base the story on.
- **Preferences:** Additional user-defined settings such as intended story style, language, tone, or story duration.

3.3.2 Image to Story System

This is the main element where all the processing occurs. It consists of several internal components that collaborate to produce three different types of results. The system carries out the following important functions:

a) Caption Generation

- The system examines the uploaded image utilizing an image captioning model.
- It produces a concise and relevant caption that summarizes the visual details of the image.
- This caption lays the groundwork for the subsequent step: story creation.

b) Story Generation

- Based on the caption, the system formulates a more elaborate and imaginative story.
- This is generally achieved through a large language model.
- The story is customized according to the tone, theme, or language chosen by the user.

c) Audio Generation

- After the story is complete, it is sent to a text-to-speech engine.
- This component transforms the written story into an audio format, allowing users to listen to the narrative.

3.3.3 Results

The system generates three final results for the user:

a) **Caption:** A brief description of the image.

b) **Story:** A customized narrative created based on the caption and user preferences.

c) **Audio:** An audio rendition of the story, which can be listened to or downloaded.

3.3.4 Purpose of This HLD

This high-level design serves to:

- Clearly outline the boundaries of the system.
- Illustrate the input-output connections for each primary functional component.
- Offer a straightforward and comprehensible summary for non-technical stakeholders, such as clients or project managers.
- Establish the groundwork for more detailed design documents (e.g., low-level design, implementation flow, module architecture).

3.4 Low-Level Design

This Low-Level Design (LLD) diagram illustrates the detailed internal structure and operational mechanics of each component within the Image to Story System. It breaks down the high-level functional blocks into their constituent parts, specifying the technical implementation decisions and the precise interactions between sub-components, without delving into actual code. The diagram begins with the user interface, where inputs such as image uploads and story preferences are captured and routed into the system. These inputs undergo validation and preprocessing in the Input Handling module, ensuring that the data is both complete and formatted correctly for downstream tasks. The validated image is passed to the BLIP Captioner, where a machine learning model generates a descriptive caption, which is then refined and structured into a prompt tailored for story generation. This prompt is processed by a large language model (LLaMA) integrated via the TogetherAI API, which produces a creative, context-aware story based on both the image content and user preferences. The story text is cleaned and segmented before entering the gTTS Converter module, where it is transformed into speech using Google Text-to-Speech, then saved as an audio file. Finally, the Output Layer manages the delivery of both the generated audio and text back to the user, enabling playback, reading, and downloading options. This modular design not only enhances maintainability and scalability but also allows for individual component upgrades or replacements without disrupting the entire system. (see Figure 3.3)

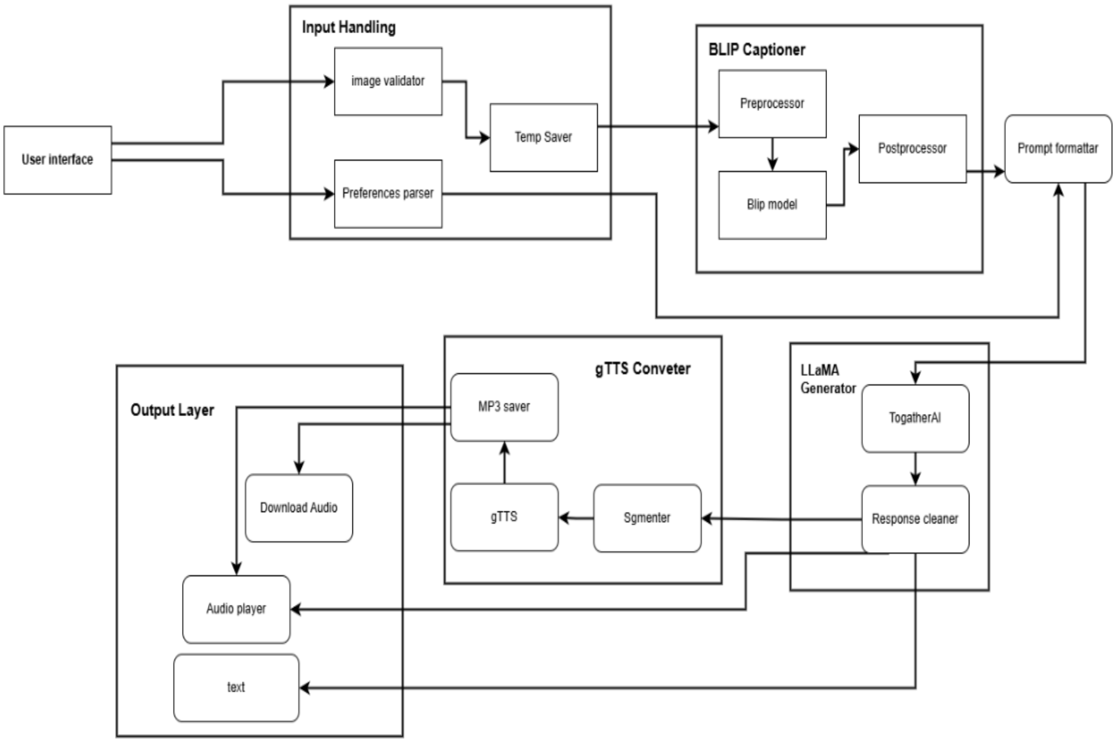


Figure 3.3: Low-Level Design

This Low-Level Design (LLD) diagram details the internal workflow of the Image to Story System. It shows how user inputs, like images and preferences, are processed through various modules, including captioning, story generation, and text-to-speech conversion. Each component interacts sequentially to transform an image into a narrated story. The output is delivered as both text and audio for user access.

3.4.1 User Interface

- **User Interface:** At this low level, this represents the specific interactive elements and visual layout provided by the chosen framework, Gradio components like image upload widgets, text display boxes, audio players, and submission buttons that enable the user to input images and receive multimodal output.

3.4.2 Input Handling

- **Image Validator:** This sub-component details the specific checks performed on the uploaded image. This includes algorithms for verifying file type, image dimensions, file size limits, and potentially basic content checks to ensure it's a valid image for processing.
- **Preferences Parser:** This outlines the precise method for extracting and interpreting user settings. This involves defining expected preference keys and their corresponding valid values (tone, genre, ending).
- **Temp Saver:** This specifies the exact temporary storage mechanism used, such as a designated temporary directory for files, an in-memory buffer, or a specific file system path where the raw image data and parsed preferences are held before being passed to the next stage.

3.4.3 BLIP Captioner

- **Preprocessor:** This details the exact image transformations applied to the raw image. This includes specific resizing algorithms (e.g., bilinear interpolation to 224x224 pixels), normalization values (mean and standard deviation for pixel values), and format conversions required by the BLIP model's input layer.
- **Blip model:** This represents the specific version and configuration of the BLIP Salesforce base model being loaded.
- **Postprocessor:** This component defines the specific text cleaning and formatting rules applied to the raw caption. This includes removal of specific tokens, stripping leading/trailing whitespace, capitalization rules, or basic grammar corrections to refine the caption.

3.4.4 LLaMA Generator

- **Prompt Formatter:** This specifies the exact template and concatenation logic used to construct the prompt for the LLaMA model. This includes defining placeholders for the BLIP caption and any fixed instructions or contextual phrases added to guide the LLaMA model's response.
- **TogetherAI:** At this low level, this refers to the precise API endpoint, authentication headers (e.g., API key usage), and JSON payload structure used to send the

formatted prompt to the Together AI service. It also covers the expected API response structure.

- **Response Cleaner:** This outlines the exact string manipulation techniques applied to the raw LLaMA response. This includes regular expressions for removing specific model-generated artifacts, boilerplate text, or instruction markers, and methods for ensuring the final text is free of unwanted characters or formatting.

3.4.5 gTTS Converter

- **Segmenter:** This component details the specific algorithm or library used to break down the text into manageable chunks for speech synthesis. This might involve splitting text by sentences, paragraphs, or using a specific phrase-detection logic to optimize for natural-sounding speech.
- **gTTS:** This specifies the exact parameters passed to the gTTS library or service for audio generation. This includes the language setting (e.g., 'en', 'es'), potentially voice options, and any other configuration parameters that influence the synthesized speech characteristics.
- **MP3 saver:** This details the exact file path, naming convention (e.g., using a timestamp or unique ID), and writing process to save the generated audio stream as an MP3 file to the local file system or a designated storage location.

3.4.6 Output Layer

- **Download Audio:** This specifies the mechanism that enables the user to retrieve the generated audio file.
- **Audio player:** This refers to the specific front-end audio player component, a Gradio gr.Audio widget is used to embed and control the playback of the generated MP3 file within the user interface.
- **Text:** This outlines the specific UI element and styling used to display the final, cleaned textual story to the user, ensuring readability and proper formatting.

Chapter 4

Testing

4.1 Test Case Scenarios of your SRS

4.1.1 Scenario 1: Complete Workflow

Objective: Ensure that the full pipeline from image input to audio output functions correctly.

Modules Tested: Image upload, caption generation, story generation, audio synthesis, UI display.

4.1.2 Scenario 2: Invalid or Empty Input

Objective: Verify that the system handles missing or invalid image input gracefully.

Modules Tested: Input validation, error handling.

4.1.3 Scenario 3: User Preference Selection

Objective: Ensure selected genre, tone, and other options influence story output as expected.

Modules Tested: UI controls, prompt formulation for LLaMA.

4.1.4 Scenario 4: API Downtime or Response Failure

Objective: Test system response when external APIs (Together API, gTTS) are unavailable.

Modules Tested: Error handling, user feedback.

4.2 Non-Functional Requirements Testing

4.2.1 Performance Testing

The average response time from image upload to story and audio output is under 12 seconds on Google Colab with GPU.

4.2.2 Usability Testing

Non-technical users were able to operate the system easily using the Gradio UI.

User feedback confirmed that dropdowns and interface layout were intuitive.

4.2.3 Scalability Testing

Tested with 10 different images and various combinations of preferences.

System handled input variety without crashing or lagging.

4.2.4 Error Handling

Missing inputs, API failures, and unexpected user actions were caught with informative messages.

4.3 Conclusions

The system was tested under multiple real-world conditions and edge cases to verify both functional and non-functional requirements. All core modules performed as expected, and proper exception handling ensured the application remained stable even during unexpected input or API issues.

The testing phase demonstrated that the Image-to-Audio Story Generator is:

- Reliable
- User-friendly
- Functionally accurate
- Resilient to external failures

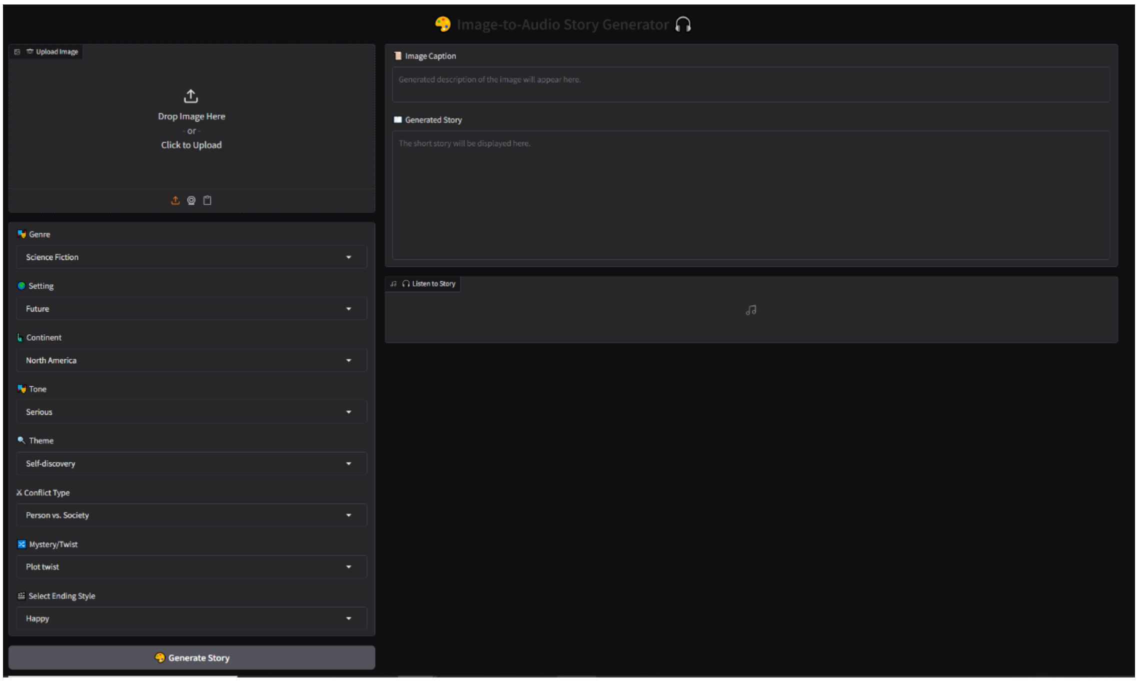
This establishes a strong foundation for further enhancements such as multilingual support, voice customization, or real-time deployment.

Chapter 5

Results & Screenshot

5.1 Main Interface

This is a screenshot of an "Image-to-Audio Story Generator" interface. It includes options to upload an image and generate a story. The left sidebar offers genre selections like Science Fiction, Fantasy, and Romance. A "Listen to Story" section and a "Generate Story" button are also visible.



Generated description of the image will appear here.

The short story will be displayed here.

Genre
Science Fiction

Setting
Future

Continent
North America

Tone
Serious

Theme
Self-discovery

Conflict Type
Person vs. Society

Mystery/Twist
Plot twist

Select Ending Style
Happy

Figure 5-1-1: Main Interface

5.2 Upload Image

This image shows an "Upload Image" section of a user interface. It prompts users to either "Drop Image Here" or "Click to Upload" with a central upload icon. A button labeled "Upload Image" is at the top left. Icons for additional actions like settings and clipboard are visible at the bottom.

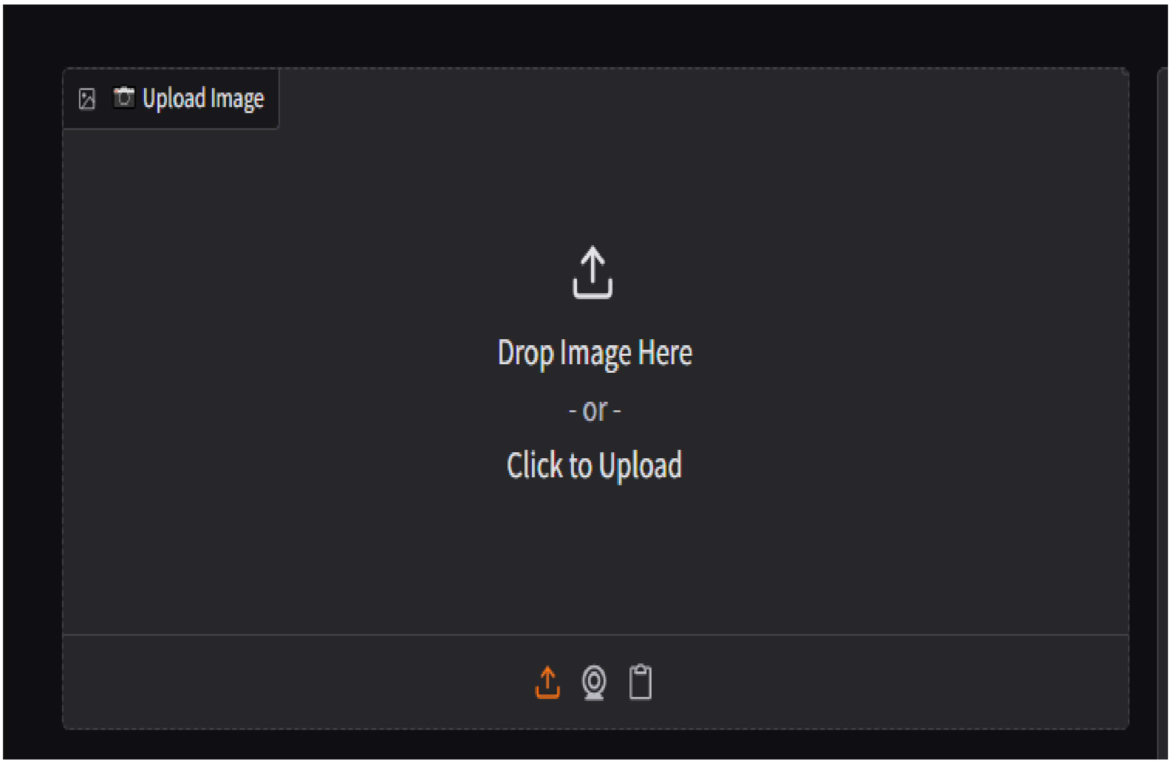
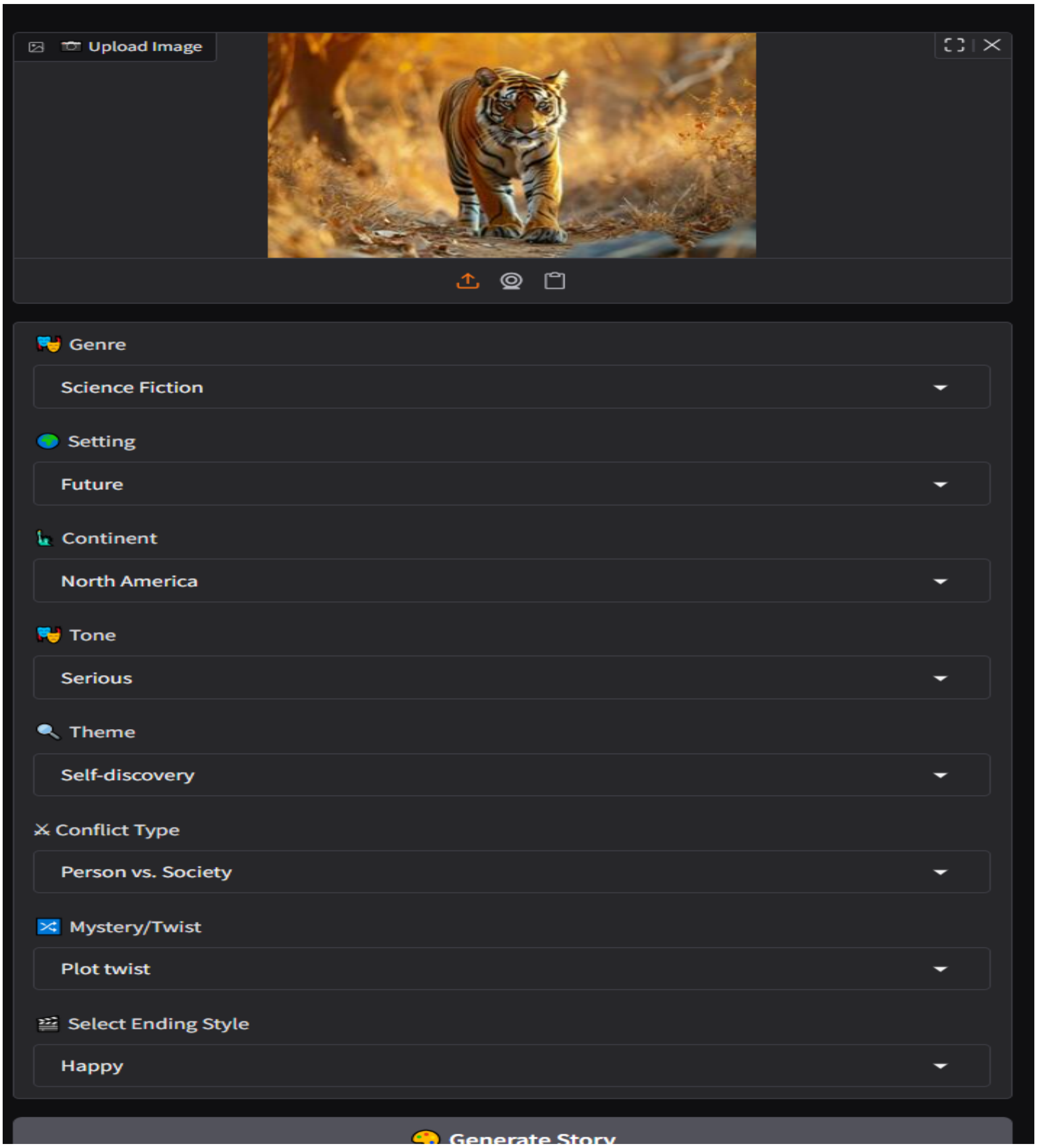


Figure 5-1-2: Upload Image

5.3 Image Uploaded

The image displays a user interface for a story generation tool. It features an uploaded image of a tiger walking through tall, golden grass and includes various dropdown menus for defining story elements such as Genre (Science Fiction), Setting (Future), Continent (North America), Tone (Serious), Theme (Self-discovery), Conflict Type (Person vs. Society), Mystery/Twist (Plot twist), and Ending Style (Happy). At the bottom, there's a button labeled "Generate Story."



Genre
Science Fiction

Setting
Future

Continent
North America

Tone
Serious

Theme
Self-discovery

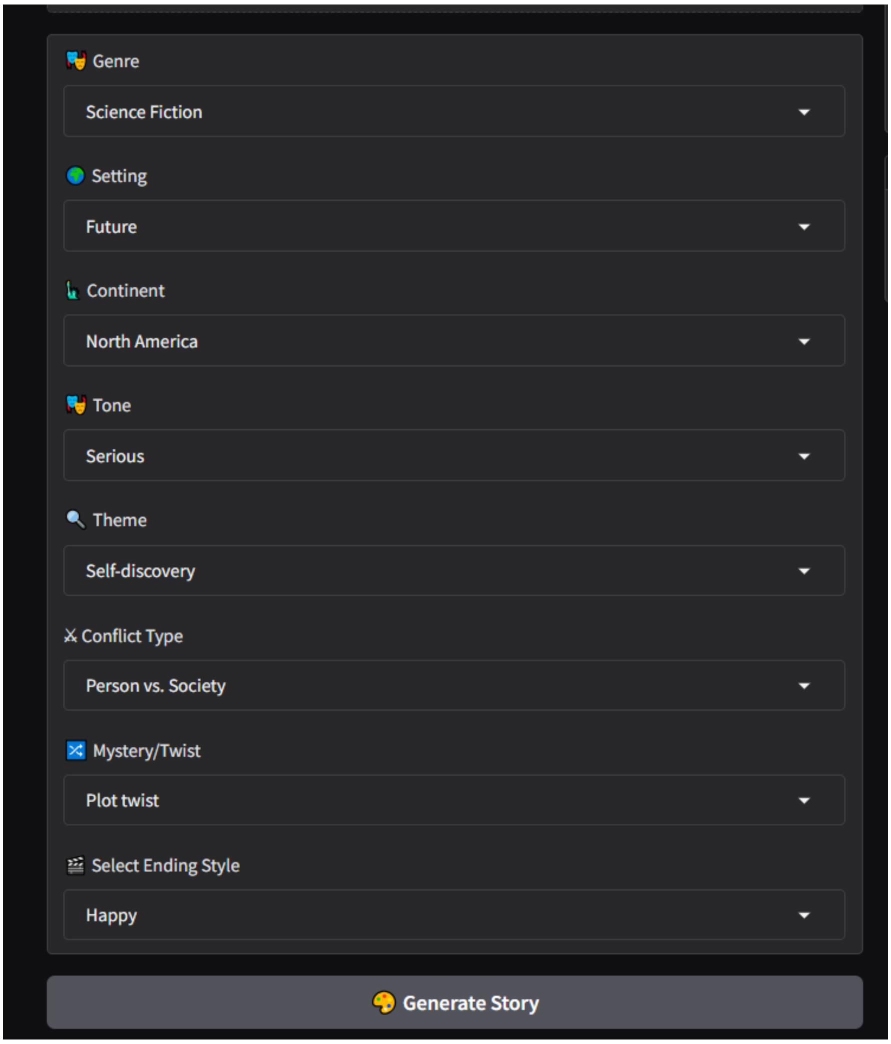
Conflict Type
Person vs. Society

Mystery/Twist
Plot twist

Select Ending Style
Happy

5.4 Select Preferences

This image showcases a user-friendly interface for an AI story generation tool. It allows users to precisely define their desired narrative by selecting from various dropdown menus. Options include a serious Science Fiction tale set in a future North America, focusing on themes of self-discovery. The chosen conflict type, "Person vs. Society," coupled with a "Plot twist" and a "Happy" ending, indicates a desire for a complex yet ultimately uplifting story.



Genre
Science Fiction

Setting
Future

Continent
North America

Tone
Serious

Theme
Self-discovery

Conflict Type
Person vs. Society

Mystery/Twist
Plot twist

Select Ending Style
Happy

Figure 5-1-3: User Preference

5.5 Story Generated

This image presents the result of a story generation process, showcasing a narrative titled "The Tiger's Quest." The story itself details the journey of Raja, a magnificent tiger, as he discovers and settles into a perfect, peaceful glade in the heart of the forest. Below the text, an integrated audio player with a visual waveform suggests that the generated story can also be experienced in an audible format, highlighting the tool's versatility. The overall display indicates a successful creation of a short, complete story based on user-defined inputs.

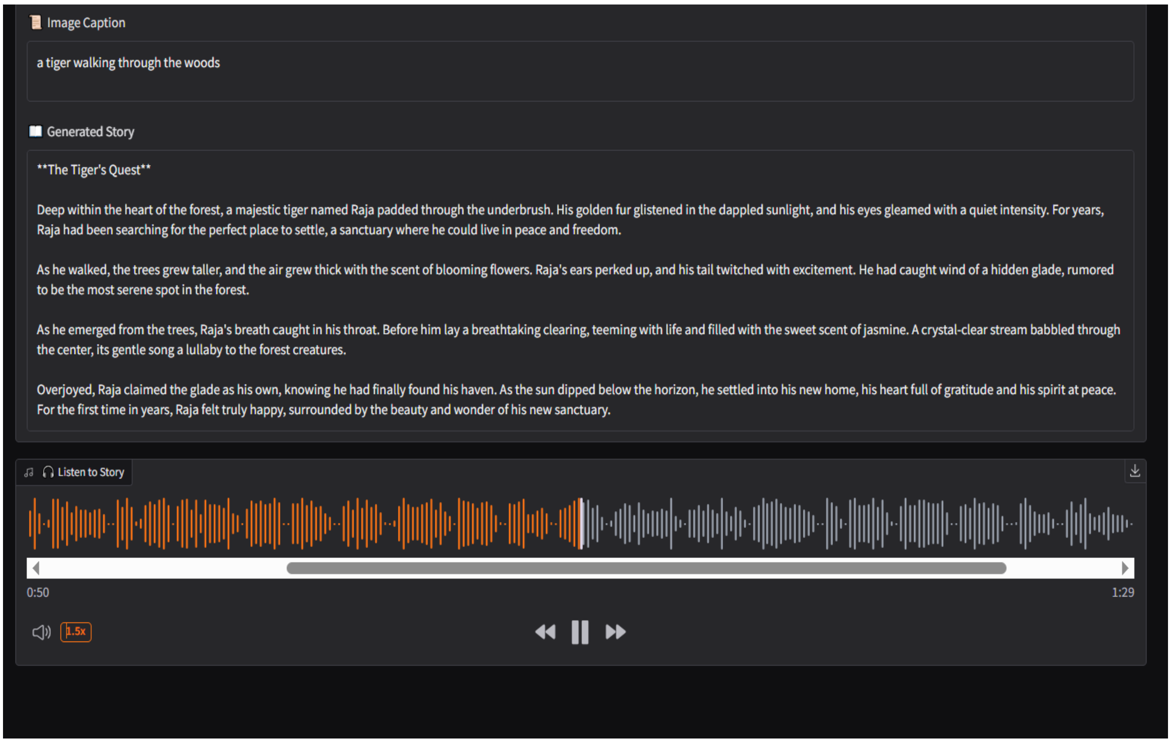


Image Caption

a tiger walking through the woods

Generated Story

The Tiger's Quest

Deep within the heart of the forest, a majestic tiger named Raja padded through the underbrush. His golden fur glistened in the dappled sunlight, and his eyes gleamed with a quiet intensity. For years, Raja had been searching for the perfect place to settle, a sanctuary where he could live in peace and freedom.

As he walked, the trees grew taller, and the air grew thick with the scent of blooming flowers. Raja's ears perked up, and his tail twitched with excitement. He had caught wind of a hidden glade, rumored to be the most serene spot in the forest.

As he emerged from the trees, Raja's breath caught in his throat. Before him lay a breathtaking clearing, teeming with life and filled with the sweet scent of jasmine. A crystal-clear stream babbled through the center, its gentle song a lullaby to the forest creatures.

Overjoyed, Raja claimed the glade as his own, knowing he had finally found his haven. As the sun dipped below the horizon, he settled into his new home, his heart full of gratitude and his spirit at peace. For the first time in years, Raja felt truly happy, surrounded by the beauty and wonder of his new sanctuary.

Figure 5-1-4: Story Generated

Chapter 6

Reference Manual or Training Manua

6.1 Training Manual

How to Use the Application

1) Launching the App
    - the Python file app.py.
    - Gradio will launch a local or public URL in your browser.
2) Uploading an Image
    - Click on the Upload Image box.
    - Select a clear, meaningful image related to the story you want generated.
3) Selecting Preferences
    - Choose the following options from the dropdown menus:
        - Genre (e.g., Romance, Mystery)
        - Setting (e.g., Future, Medieval times)
        - Continent (e.g., Europe, Asia)
        - Tone (e.g., Light-hearted, Dark)
        - Theme (e.g., Redemption, Justice)
        - Conflict Type (e.g., Internal struggle)
        - Twist (e.g., Time paradox)
        - Ending (e.g., Happy, Tragic)
4) Generating Story
    - Click the Generate Story button.
    - Wait for the system to:
        - Extract a caption from the image.
        - Generate a realistic short story.
        - Convert the story into an audio file.
5) Viewing and Listening
    - The caption will be displayed in the caption box.
    - The story will be shown in the story text area.
    - The audio will appear in a player—click "Play" to listen or download it.

References

[1] Li, J., Li, D., Xiong, C., & Hoi, S. (2022). BLIP: Bootstrapped Language-Image Pretraining for Unified Vision-Language Understanding and Generation. Retrieved from https://github.com/salesforce/BLIP

[2] Meta AI. (2024). LLaMA 3.1: Open Foundation Language Models. Retrieved from https://www.together.ai

[3] Google. (n.d.). gTTS – Google Text-to-Speech Python Library. Retrieved from https://pypi.org/project/gTTS/

[4] Abid, A., Chaudhary, S., Zhang, M., & Kundaje, A. (2021). Gradio: Create UIs for your Machine Learning models in Python. Retrieved from https://www.gradio.app.

[5] Hugging Face. (n.d.). Transformers: State-of-the-art Machine Learning for Pytorch, TensorFlow, and JAX. Retrieved from https://huggingface.co/transformers/.

[6] Together Computer Inc. (2024). Together API Documentation. Retrieved from https://www.together.ai/docs.

[7] Kumar, S. (n.d.). python-dotenv: Reads key-value pairs from a .env file and adds them to environment variables. Retrieved from https://pypi.org/project/python-dotenv/

[8] Clark, A., & Contributors. (n.d.). Pillow (PIL Fork) for image processing in Python. Retrieved from https://pypi.org/project/Pillow/