A
Major Project Report On

IMAGE CAPTION GENERATOR

Submitted in partial fulfilment of the requirements for the award of degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING
(DATA SCIENCE)

Submitted By

S. SINDHUJA	**217Z1A6751**
A. SRAVANI	**217Z1A6705**
L. SANDEEP	**217Z1A6736**

Under the Guidance of
Mrs. K. Ashwini
Assistant Professor

SCHOOL OF ENGINEERING
Department of Computer Science and Engineering
(Data Science)

NALLA NARASIMHA REDDY
EDUCATION SOCIETY'S GROUP OF INSTITUTIONS
(AN AUTONOMOUS INSTITUTION)

Approved by AICTE, New Delhi, Chowdariguda (V) Korremula 'x' Roads,
Via Narapally, Ghatkesar (Mandal)Medchal (Dist), Telangana-500088

2024-2025

NALLA NARASIMHA REDDY
Education Society's Group of Institutions - Integrated Campus
(UGC AUTONOMOUS INSTITUTION)

SCHOOL OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

CERTIFICATE

This is to certify that the project report titled **"IMAGE CAPTION GENERATOR"** is being submitted by **S. Sindhuja (217Z1A6751), A. Sravani (217Z1A6705) and L. Sandeep (217Z1A6736)** in Partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering (Data Science)** is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide
(Mrs. K. Ashwini)

Head of Department
(Mrs. V. Indrani)

Submitted for Viva voce Examination held on……..…….

External Examiner

DECLARATION

We **S. Sindhuja, A. Sravani** and **L. Sandeep** are students of **Bachelor of Technology in Computer Science and Engineering (Data Science), Nalla Narasimha Reddy Education Society's Group of Institutions**, Hyderabad, Telangana, here by declare that the work presented in this project work entitled **IMAGE CAPTION GENERATOR** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of engineering ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning.

S. Sindhuja **217Z1A6751**

A. Sravani **217Z1A6705**

L. Sandeep **217Z1A6736**

Date:

Signature:

ACKNOWLEDGEMENT

We express our sincere gratitude to our guide **Mrs. K. Ashwini,** Assistant Professor in Computer Science and Engineering Department, NNRESGI, who motivated throughout the period of the project and also for her valuable and intellectual suggestions apart from her adequate guidance, constant encouragement right throughout our work.

We would like to express our profound gratitude to our Major Project In-charge **Mrs. T. Spoorthi Reddy,** Assistant Professor in Computer Science and Engineering (Data Science) Department, NNRESGI, for her support and guidance in completing our project and for giving us this opportunity to present the project work.

We profoundly express thanks to **Dr. K. Rameshwaraiah,** Professor and Head of Computer Science and Engineering Department, NNRESGI, for his cooperation and encouragement in completing the project successfully.

We profoundly express thanks to **Mrs. V. Indrani,** Head of Data Science Department, NNRESGI, for her cooperation and encouragement in completing the project successfully.

We wish to express our sincere thanks to **Dr. G. Janardhana Raju,** Dean School of Engineering, NNRESGI, for providing the facilities for completion of the project.

We wish to express our sincere thanks to **Dr**. **C. V. Krishna Reddy,** Director NNRESGI, for providing the facilities for completion of the project.

Finally, we would like to thank overall major-project coordinator, **members of Project Review Committee (PRC),** all the faculty members and supporting staff of the Department of Computer Science and Engineering, NNRESGI, for extending their help in all circumstances.

By :

S. Sindhuja **217Z1A6751**

A. Sravani **217Z1A6705**

L. Sandeep **217Z1A6736**

ABSTRACT

The Image Caption Generator creates detailed and contextually accurate descriptions from visual content like images, addressing the limitations of traditional models in capturing complex visual details and maintaining consistency. This project uses generative AI for feature extraction and description generation, leveraging its ability to process and interpret complex visual inputs. This approach captures both global and local visual relationships, resulting in richer and more detailed feature representations. Combined with a language model, the generator produces contextually relevant and coherent descriptions. Key advancements include an interactive user interface built with Stream lit for real-time captions and also extracts text from the images and gives it an advanced feature called text to speech. Trained on diverse datasets, the model generates high-quality descriptions across image domains, evaluated using metrics like BLEU to ensure grammatical and semantic accuracy. The image caption generator demonstrates significant advancements in descriptive depth and contextual accuracy, contributing to automated visual content interpretation. The project aims to democratize access to intelligent visual interpretation tools, promoting more intuitive and inclusive human-computer interaction through the convergence of vision and language.

Keywords: **Image Captioning, BLEU, Generative AI, Interactive User Interface (UI).**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

In the era of digital content explosion, the ability to understand and describe visual information is crucial. Manual captioning of images is time-consuming and often inconsistent. The need for automated solutions that generate accurate and relevant image descriptions has led to significant advancements in artificial intelligence. Image captioning combines computer vision and natural language processing to generate textual descriptions of images. This technology is especially beneficial for the visually impaired, enhancing accessibility and comprehension of visual content. Traditional captioning models often lack contextual depth and linguistic fluency. With the advent of deep learning and transformer-based architectures, automated image description systems have seen major improvements. The integration of Optical Character Recognition (OCR) and text-to-speech (TTS) technologies further enriches the interaction. This project aims to build a user-friendly and intelligent image caption generator that provides multilingual support and real-time interactions. The solution enhances digital communication, learning, and accessibility in various domains.

1.2 Context of the project

The project leverages AI technologies such as Google Gemini and GPT-4 Vision to automatically generate meaningful captions from image content. It uses advanced image processing for feature extraction and integrates Tesseract OCR for text recognition within images. A Streamlit-based frontend enhances usability through an interactive interface. Text-to-speech and emoji support make the output more engaging and accessible to a wider audience. This project is designed to assist not only content creators but also those with visual impairments, improving the inclusivity of digital platforms. By automating image captioning, it streamlines workflows in media, education, and social communication. Real-time processing ensures immediate feedback and supports live applications. The generated captions are evaluated for linguistic and contextual accuracy using BLEU metrics. The multilingual support helps in catering to users from different language backgrounds, expanding its global usability.

The project blends cutting-edge AI with practical usability to meet modern digital demands.

1.3 Objectives

The primary objective of this project is to design and develop a comprehensive, end-to-end AI-Powered Image Captioning System that automates the creation of semantically rich, contextually relevant image captions, while addressing critical challenges of accessibility, expressiveness, and interactivity. The system is tailored to cater to a diverse user base, ranging from visually impaired individuals to educators, researchers, social media managers, and software developers.

As society continues to evolve into a highly visual digital ecosystem, the volume of content consumed in image form has risen dramatically. However, not all users can interpret images equally. Particularly for those with visual impairments, understanding content on social platforms, in online classrooms, or in digital documentation often becomes a challenge without descriptive text or audio assistance. This system is developed with these core challenges in mind and is designed to provide a solution that not only generates accurate captions but also engages users through multimodal interactions.

1.3.1 Automatic Image Caption Generation

The cornerstone of the system lies in its ability to automatically generate meaningful, human-like descriptions for uploaded images using Google Gemini AI, one of the most advanced multimodal AI models available. Unlike traditional image processing systems that rely on object detection or simple classification, Gemini AI interprets the image holistically. It understands not just what is present (e.g., a dog, a tree, a person), but also what is happening (e.g., a child playing with a dog in a sunny park), delivering context-aware, coherent, and fluent natural language captions.

This feature is vital for visually impaired users who need detailed scene-level descriptions to comprehend visual information. It also supports educators who use visual aids in their teaching, enabling them to quickly generate and verify descriptive content. Additionally, social media users and marketers can benefit from automated captioning, saving time while ensuring their content remains accessible and engaging.

1.3.2 Text Extraction from Images (OCR Integration)

Visual content frequently includes embedded text—on signs, posters, books, digital screenshots, or forms. Accurately interpreting such images requires the system to go beyond captioning and extract textual data that appears visually. For this reason, the system integrates Tesseract OCR, a powerful open-source optical character recognition engine.

This module scans the image, identifies text regions, and extracts the textual information with high precision. The feature is crucial for multiple use cases:

- Visually impaired users can hear what is written in the image.
- Students can scan and convert visual learning material to text.
- Administrators or researchers can digitize and analyze visual documents efficiently.

In combining OCR with caption generation, the system delivers full-spectrum visual interpretation—handling both symbolic (text) and non-symbolic (scenes, objects) data.

1.3.3 Text-to-Speech Conversion (TTS)

Recognizing that not all users can read on-screen text, especially those with severe vision loss or cognitive processing challenges, the system incorporates a Text-to-Speech (TTS) module using the Python-based pyttsx3 engine. Once the image is processed—either for captions or OCR text—the user can choose to hear the output spoken aloud.

This auditory output ensures that visually impaired individuals can independently access and understand image content. It also benefits:

- Elderly users who prefer voice-based interactions.
- Users in hands-free environments or mobile contexts.
- Educational apps or smart learning tools requiring vocal feedback.

The integration of TTS makes the system a true accessibility tool, ensuring inclusivity through multisensory engagement.

1.3.4 BLEU Score Evaluation

To maintain high-quality captions, the system includes a BLEU (Bilingual Evaluation Understudy) score evaluation module. This feature allows users to enter a reference caption (usually human-written) and compare it to the AI-generated output. The BLEU algorithm then calculates a numerical score based on the similarity of n-grams (word sequences) between the two.

This feature serves multiple purposes:

- Developers can fine-tune model prompts or performance.
- Researchers can evaluate improvements over time or compare AI models.
- Users can verify the accuracy and relevance of generated content.

The addition of BLEU scoring introduces a quantitative metric to what is often a subjective task, making the system more robust, scientifically grounded, and self-assessable.

1.3.5 User-Friendly Streamlit Interface

All features of the system are brought together within a responsive and intuitive user interface, developed using Streamlit, a Python-based web application framework. The UI is minimal, fast-loading, and designed for both tech-savvy users and non-programmers.

Key interface capabilities include:

- Image upload through drag-and-drop or file browsing.
- Button-based interaction to trigger specific functions (caption, OCR, TTS, BLEU).
- Instant output visualization and download options.
- Accessible layout with clear fonts, spacing, and alerts for status updates.

This simple yet powerful UI ensures that no command-line knowledge or configuration is needed to operate the system, dramatically increasing its accessibility to educational institutions, nonprofits, or casual users.

1.3.6 Emoji Integration for Expressiveness

To make captions more expressive, engaging, and emotionally resonant, the system includes an emoji enhancement module. Using a keyword-to-emoji mapping dictionary, the system appends relevant emojis to captions based on detected themes or objects.

For example:

"A child playing with a dog in the park." → "A child playing with a dog in the park."
Though seemingly light-hearted, this feature serves important purposes:

- **Emotional communication:** Emojis help convey tone and mood.
- **Youth-friendly output:** Especially engaging for children or social media users.
- **Accessibility with style:** Enhances communication without sacrificing usability.

This module reinforces the idea that inclusivity can also be joyful, making the technology feel less clinical and more human-centered.

In essence, the project's objectives are not just functional checkboxes—they are carefully aligned to solve real-world challenges in accessibility, content management, and human-AI interaction. The system aspires to be more than an academic prototype; it is a practical, extensible, and empathetic solution that embodies the future of inclusive AI design.
Whether helping a blind student explore digital images, enabling a developer to benchmark AI captions, or assisting a content creator with instant image descriptions, this system is designed to make AI not only intelligent—but also meaningful and accessible to all.

1.4 Scope

The scope of this project is to develop a functional prototype of an AI-powered image caption generator system that can be extended and enhanced in the future. The project focuses on the following aspects:

- **Automated Caption Generation** – Using AI models like Google Gemini and GPT-4 Vision to generate meaningful and accurate image descriptions.

- **Text Extraction** – Implementing Tesseract OCR to recognize and extract any embedded text from images.
- **TTS Support** – Converting text to speech for accessibility.
- **Interactive User Interface** – Developing a user-friendly, real-time interface using Streamlit for smooth interaction.
- **Performance and Accuracy** – Ensuring fast processing, secure data handling, and high-quality outputs using evaluation metrics like BLEU.
- **Integration and Scalability** – Designing the system to allow seamless API integration and easy scaling for diverse use cases.

CHAPTER 2

LITERATURE SURVEY

Semantically-Oriented Image Captioning: A Deeper Understanding of Spatial and Geometrical Semantics.

Authors: Anwar Ul Haque , Sayeed Ghani.

Abstract: Most of the latest research work in the domain of image captioning uses the fundamental architecture known as encoder-decoder architecture or framework. The framework executes in the way that the image to feature extraction is done using the encoder module which can be constructed in several ways with various deep learning networks e.g., convolutional neural networks, auto-encoders, GAN, transformers, capsule networks, etc. The job of the decoder module is to map the features with the provided annotations during training to learn the conversion of features into human language. Once trained, the network can humanly annotate a randomly provided image with various details.

Basic encoder-decoder architecture is composed of CNN (as encoder) and RNN (Recurrent Neural Networks) (as a decoder). The image is fed to CNN for feature conversion while features are fed to RNN for mapping against the annotation words [12]–[15]. To make the network more innovative and efficient, various additions are done in the model, for example, incorporation of visual attention mechanisms [16], [17], region of interests, and attention behaviors [18], [19]. A significant group of researchers believes that attention and visual attention help in better understanding objects and their behaviors in the process of image captioning.

Visual attention is due to higher-order convolutional work, which reduces the spatial and localization information and the semantic impact on the output. Similarly, the region of interest application over images during the encoding phase is prevalent in the field of image captioning. The idea is to use multiple R-CNN-based object detectors and extract features from those regions for captioning. This helps in generating more verbose captions for each region separately but at the same time loses all the semantics and spatial relations among objects lying in the inter-region spaces of an image.

Advanced Semantic-Based Image Captioning Using Multimodal Transformers and Scene Graphs.

Authors: A. R. Sharma and K. L. Verma.

Abstract: Image captioning has emerged as a crucial area of research connecting computer vision and natural language processing. Traditional encoder-decoder architectures, which primarily rely on convolutional neural networks (CNN) for feature extraction and recurrent neural networks (RNN) for sequence generation, have shown limitations in handling complex semantics and spatial relationships between objects in images. To address these challenges, recent advancements propose the integration of multimodal transformers and scene graph generation techniques. In this work, a novel image captioning framework is proposed where the encoder not only extracts visual features but also generates a structured scene graph representing object relationships, spatial orientations, and contextual attributes. A multimodal transformer then fuses these representations with linguistic embeddings to generate coherent and semantically enriched captions. The use of scene graphs ensures that object interactions and positional contexts are preserved, enabling the model to describe scenes with greater accuracy and human-likeness. The proposed method contributes towards bridging the semantic gap in image captioning by effectively utilizing global scene context and local object behaviors.

Visual Image Caption Generator Using Deep Learning.

Authors: Grishma Sharma.

Abstract: Image Caption Generation has always been a study of great interest to the researchers in the Artificial Intelligence department. Being able to program a machine to accurately describe an image or an environment like an average human has major applications in the field of robotic vision, business and many more. This has been a challenging task in the field of artificial intelligence throughout the years. In this paper, we present different image caption generating models based on deep neural networks, focusing on the various RNN techniques and analyzing their influence on the sentence generation. We have also generated captions for sample images and compared the different feature extraction and encoder models to analyse which model gives better accuracy and generates the desired results.

2.1 EXISTING SYSTEM

Existing image captioning systems predominantly focus on generating simple, grammatically correct sentences by leveraging pre-trained deep learning models. While effective for basic descriptions, these systems exhibit significant limitations when applied to more complex or dynamic scenarios. They often struggle to capture the intricate details of an image's context, fail to recognize embedded textual information, and offer limited adaptability across diverse languages and environments. Additionally, most traditional systems are not designed with user engagement or accessibility in mind, reducing their usability for broader audiences.

Key Limitations of Existing Image Captioning Systems:

- **Lack of-Real-Time Processing:** Current systems typically require several seconds or longer to analyse an image and generate a caption. This latency renders them impractical for dynamic applications such as live video captioning, real-time augmented reality experiences, or assistive technologies for visually impaired users who require instantaneous feedback.
- **Absence of Text Recognition (OCR Integration):** Many images contain embedded text (e.g., street signs, menus, product labels, official documents), which carries critical contextual information. Most existing captioning models are not equipped with Optical Character Recognition (OCR) capabilities, making them unable to detect, interpret, and incorporate textual elements into the generated captions, thus missing essential aspects of the visual content.
- **Limited Accessibility Features:** Accessibility remains a major oversight. Features such as integrated text-to-speech (TTS) conversion, adjustable font sizes, high-contrast modes, and customizable voice outputs are either poorly implemented or entirely absent. As a result, users with visual, auditory, or cognitive impairments find it difficult to engage with these systems meaningfully.
- **Minimal User Interaction and Customization:** Traditional captioning tools offer static, one-way interactions where users passively receive the system's output without opportunities for feedback, adjustment, or personalization. There is often no option to refine captions based on user preferences, context, or intended usage (e.g., formal vs. casual tone). This lack of interactivity significantly limits the potential for user-driven optimization and adaptive learning.

- **Limited Multilingual Support:** Most systems are trained on datasets primarily in English and struggle with generating captions in other languages. In an increasingly globalized world, the inability to support multilingual users limits the adoption of these technologies in non-English speaking regions and multicultural environments.
- **Contextual Misinterpretation:** Without deeper semantic understanding, many systems generate captions that are either too generic or contextually inaccurate. They fail to recognize nuanced elements such as emotions, cultural symbols, or complex activities occurring within the scene, which could greatly enhance the richness and relevance of the generated descriptions.
- **Inability to Handle Complex Scenes:** When faced with images containing multiple objects, intricate backgrounds, or simultaneous activities, existing captioning systems often oversimplify the scene. They tend to focus only on the most prominent object while ignoring secondary but contextually important details, leading to incomplete or misleading captions.
- **Static Model Performance Without Continuous Learning:** Most captioning systems rely on pre-trained models that do not adapt over time. They lack mechanisms for incremental learning based on user feedback or exposure to new data, causing their performance to degrade or become outdated when faced with evolving visual trends, new objects, or emerging cultural references.

2.2 PROPOSED SYSTEM

In response to the limitations identified in current image captioning tools and the growing demand for inclusive, intelligent, and user-friendly solutions, we propose the development of an AI-Powered Image Captioning System. This system aims to unify modern advances in computer vision, natural language processing, optical character recognition (OCR), and text-to-speech (TTS) into a single, integrated, and accessible platform. The goal is to create a tool that can transform static images into context-aware, expressive, and multimodal outputs, catering to users of all abilities. The proposed system is designed to be modular, scalable, and intuitive, ensuring ease of use, quick interaction, and adaptability across different user groups, including students, educators, and the differently abled. By bringing powerful AI components like Google Gemini, Tesseract OCR, pyttsx3 TTS, and a web-based UI via Streamlit, this solution offers more than just captioning—it enables true visual understanding.

Core Components of the Proposed System

The system is composed of six main functional modules, each of which plays a distinct role in the image interpretation pipeline:

1. Image Upload and Pre-processing Module:

Users can upload images in common formats (e.g., JPG, PNG) using a simple browser-based interface. The image is previewed in real-time and preprocessed if necessary (e.g., resizing, converting to grayscale for OCR enhancement). This module ensures that the image is correctly formatted for downstream tasks and provides immediate visual feedback.

2. Caption Generation Module:

The heart of the system lies in its ability to generate natural, coherent, and semantically rich captions using Google Gemini AI. Once the image is uploaded, it is processed by Gemini's multimodal vision model to produce:

- A detailed scene description summarizing the primary elements and actions within the image.
- A concise one-line caption suitable for social media or alt-text usage.

Gemini's advanced contextual reasoning enables the system to go beyond mere object detection and generate captions that reflect relationships, emotions, and background details. This improves accuracy, relevance, and human-likeness in generated content.

3. Text Extraction Module (OCR):

Many images contain embedded text—signs, posters, books, labels—which is essential for full comprehension. Using Tesseract OCR, this module scans the image for any readable text and extracts it. The output can then be displayed, spoken aloud, or used to enrich the generated caption. This dual capability (caption + OCR) ensures comprehensive image interpretation, making the system suitable for educational, administrative, and assistive contexts.

4. Text-to-Speech Module (TTS):

To assist users with visual impairments or reading challenges, the system integrates pyttsx3, a lightweight and platform-independent text-to-speech engine. Captions and

extracted text can be read aloud on demand. This auditory interface makes the system more inclusive, enabling users to “hear” what is in the image. It also supports basic controls like play, pause, and adjust speed.

5. BLEU Score Evaluation Module:

One of the system’s innovative features is its ability to evaluate the quality of the generated caption. Users can input a reference caption (e.g., their own description of the image), and the system calculates a BLEU score comparing this reference with the AI-generated caption. This feature is especially useful for researchers, developers, and educators looking to measure performance, train models, or assess language quality. It adds an objective evaluation layer to what is otherwise a subjective task.

6. Emoji Integration Module:

To enhance the expressiveness and emotional tone of the caption, an optional emoji enhancer is included. Based on keywords detected in the caption, the system appends relevant emojis. For example:

- Caption: “A happy child playing with a dog.” → “A happy child playing with a dog.”

This not only makes the caption more engaging but also introduces an element of personalization and fun, which can be particularly useful in social media, storytelling, or education for younger audiences.

7. Real-Time Processing and Feedback Module:

The system is optimized for real-time or near-real-time processing, ensuring that users receive captions, OCR results, and TTS outputs within a few seconds of uploading an image. Efficient model integration and hardware acceleration (where available) allow for dynamic use cases such as live video frame captioning, classroom assistance, and accessibility-tools.

8. Rate-Limiting Handling and Retry Mechanism:

When the Google Gemini API hits a **rate limit (Resource Exhausted)**, the system doesn't crash immediately.

Instead, it **waits and retries automatically** (with exponential backoff like 2, 4, 8 seconds) up to 5 times before giving an error.

9. Interactive and Responsive UI with Streamlit:

We have built an interactive web UI using **Streamlit**, including:

- Real-time image preview after upload
- Buttons for different tasks (Describe Scene, Generate Caption, Extract Text)
- Dynamic display of outputs (captions, text areas, etc.)
- Sidebar with information and branding

CHAPTER 3

SYSTEM ANALYSIS

The Image Caption Generator is an AI-based system designed to automatically generate descriptive captions for images. It utilizes advanced technologies such as Google Gemini AI for caption generation and Tesseract OCR for extracting textual content from images. The system is built with accessibility in mind, offering features like text-to-speech and emoji integration for enhanced user engagement. A Streamlit-based UI enables seamless and interactive user experience. The proposed system addresses limitations of existing solutions by ensuring improved caption accuracy and real-time interaction. This approach helps users, including visually impaired individuals, to interpret image content more effectively. The combination of deep learning, NLP, and UI frameworks makes this solution both powerful and user-friendly. It also prioritizes data security, fast processing, and accurate output.

3.1 Detailed Explanation of the methods and techniques

METHODS :

1. **Image Caption Generation:**
   - This method uses AI to generate descriptive sentences based on image content.
   - The system processes image features and translates them into human- like language.
   - It combines both visual recognition and natural language generation.
   - The output is context-aware, ensuring relevance to the actual image scene.
   - This method is central to the system and is enhanced by Google Gemini AI.

2. **Optical Character Recognition (OCR):**
   - OCR is used to detect and extract any textual information from the image.
   - Tesseract OCR is implemented due to its support for multiple languages.
   - This step is crucial for images that contain text embedded within them.

- Extracted text is used to enrich the context of the final caption.
- The OCR method adds semantic depth to purely visual descriptions.

3. **Text-to-Speech Conversion:**
    - The system reads the generated captions aloud for accessibility.
    - A TTS engine is triggered once the caption is finalized.
    - It provides an audio version of the caption, helping visually impaired users.
    - Different voices and languages can be configured based on user preferences.
    - The method ensures that the application is inclusive and user-friendly.

4. **Interactive UI Workflow:**
    - The method involves user interaction through a web interface built on Streamlit.
    - It allows image upload, caption display, and voice playback all on one screen.
    - Button-based operations trigger backend processing in real time.
    - It's designed to be minimal, responsive, and easy to navigate.
    - This ensures a smooth and guided user experience throughout the system.

TECHNIQUES :

1. **Google Gemini AI:**
    - Google Gemini is a multimodal AI model capable of understanding both images and text.
    - It forms the backbone for caption generation with deep contextual understanding.
    - The model is cloud-based, making integration with web apps seamless via APIs.
    - Gemini produces natural and fluent language outputs that resemble human-written captions.
    - It significantly improves over older models in terms of coherence and flexibility.

2. **Tesseract OCR:**
    - Tesseract is an open-source optical character recognition engine by Google.
    - It is integrated into the backend using Python libraries for image text extraction.
    - It supports over 100 languages and performs well on varied text formats.
    - The engine works with pre-processing techniques like grayscale and binarization.
    - It plays a crucial role in combining visual and textual elements from images.

3. **Streamlit Framework:**
    - Streamlit is a Python library for creating interactive, real-time web applications.
    - It simplifies the development of dashboards and interfaces for AI projects.
    - The UI displays uploaded images, generated captions, and speech playback.
    - It allows backend logic to be linked directly with frontend widgets like buttons and sliders.
    - Streamlit makes rapid prototyping and testing easy with minimal code.

4. **Google API's:**
    - Google APIs allow secure and scalable interaction with AI and OCR services.
    - They help in sending images to cloud-based models like Gemini for captioning.
    - Authentication and API key management ensure safe usage.
    - These APIs offer high uptime and fast processing, enhancing system responsiveness.
    - Integration is done using Python packages that support RESTful communication.

3.2 Description of Tools, Software and Hardware used

TOOLS :

1. **Tesseract OCR**: For extracting printed text from image files.
2. **Google Gemini AI**: For understanding and generating image-based captions.
3. **Text-to-Speech Engines**: Tools like pyttsx3 or gTTS to convert text to speech.
4. **Streamlit**: Enables rapid web application development for interactive usage.
5. **Google APIs**: Support secure and robust connections to AI services.

SOFTWARE REQUIREMENTS :

1. **Operating System :** Windows 10+, Linux (Ubuntu/Debian), and macOS.
2. **Programming Language and Environment:** Python 3.8
3. **Platform:** VS Code, Jupyter Notebook, or PyCharm.
4. **Libraries and Frameworks**

    - **Streamlit**: For building the front-end UI with support for real-time updates and widgets.
    - **Google Generative AI SDK / LangChain**: For interacting with the Gemini Vision API to generate captions.
    - **Pytesseract**: A Python wrapper for the Tesseract OCR engine used to extract text from images.
    - **Pillow (PIL)**: Used for image processing and format conversions.
    - **pyttsx3**: Offline, cross-platform text-to-speech synthesis.
    - **nltk**: For computing BLEU scores and other NLP evaluations.
    - **NumPy, io, os, time, re**: Utility libraries for backend logic and file management.

HARDWARE REQUIREMENTS :

Hardware requirements define the minimum and recommended configurations needed to ensure smooth operation of the system during both development and deployment.

Minimum Requirements (For Local Testing)

- **Processor:** Intel Core i3 or equivalent (2.0 GHz+)
- **RAM:** 4 GB
- **Storage:** 5 GB of free disk space (for Python environment, packages, and temporary files)
- **Graphics:** Integrated GPU (no dedicated GPU required)
- **Internet:** Required for Gemini API interactions
- **TTS Compatibility:** Functional onboard sound card for text-to- speech playback

Recommended Requirements (For Development or Deployment)

- **Processor:** Intel Core i5 or higher (3.0 GHz+ quad-core) or AMD Ryzen 5 equivalent
- **RAM:** 8 GB or higher (to support real-time processing and multi-threading)
- **Storage:** SSD with 20 GB free space
- **Graphics:** Discrete GPU (optional; not required but beneficial for model visualization/testing)
- **Audio Hardware:** Headphones/speakers for TTS output; microphone (optional) for future voice-based interaction
- **Display:** 1080p monitor for full UI display
- **Network:** Stable high-speed internet connection (minimum 10 Mbps)

Server-Side Deployment (Optional)

- **Cloud Instance:** AWS EC2 (t2. medium or higher), DigitalOcean, or GCP Compute Engine
- **Docker Support:** For containerization and easy deployment
- **GPU (optional):** If future versions support model fine-tuning or inference.

CHAPTER 4

SYSTEM DESIGN

4.1 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general- purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing

object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of Object-Oriented tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.

4.1.1 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML)is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

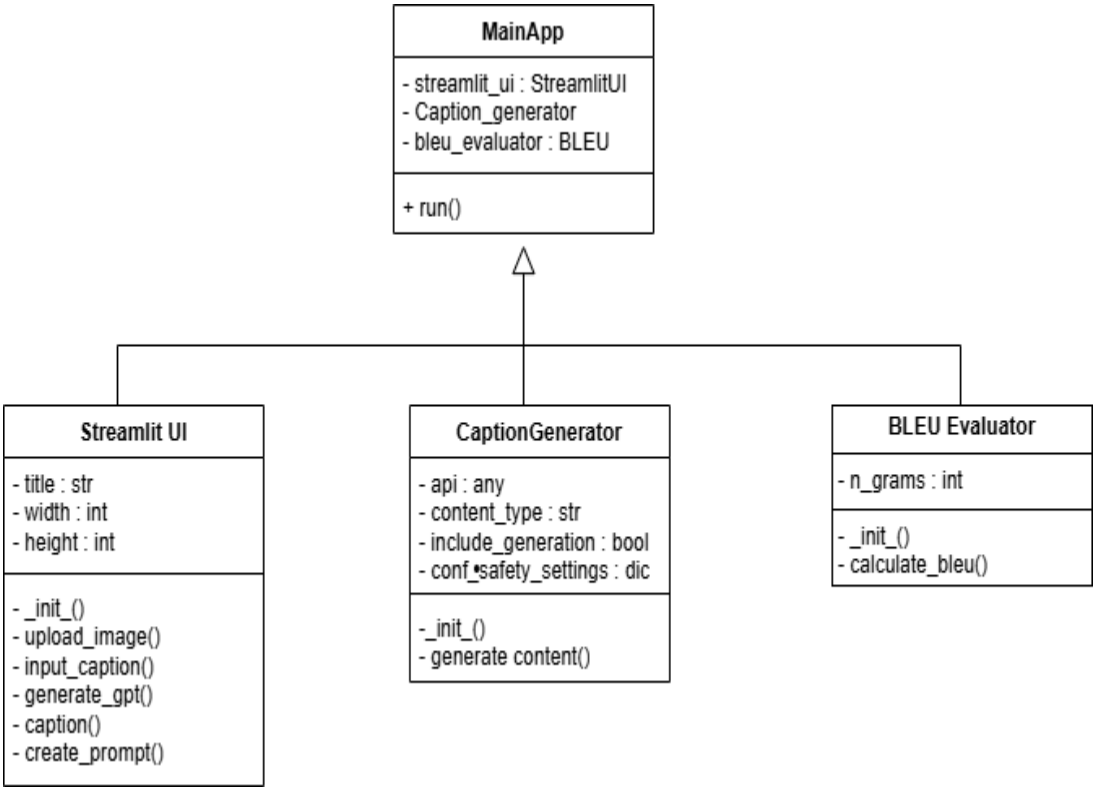

Fig: 4.1.1.1 Class Diagram for Main Application

4.1.2 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use case, and any dependencies between those usecases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

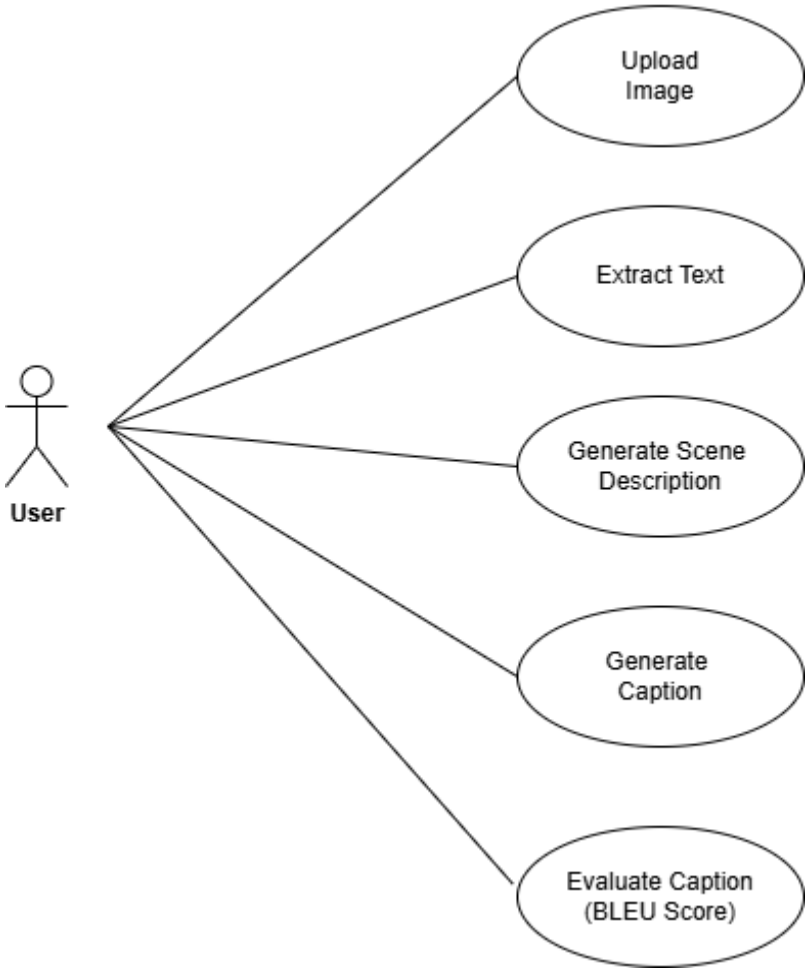

Fig: 4.1.2.1 Use case Diagram for User

4.1.3 Sequence Diagram

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

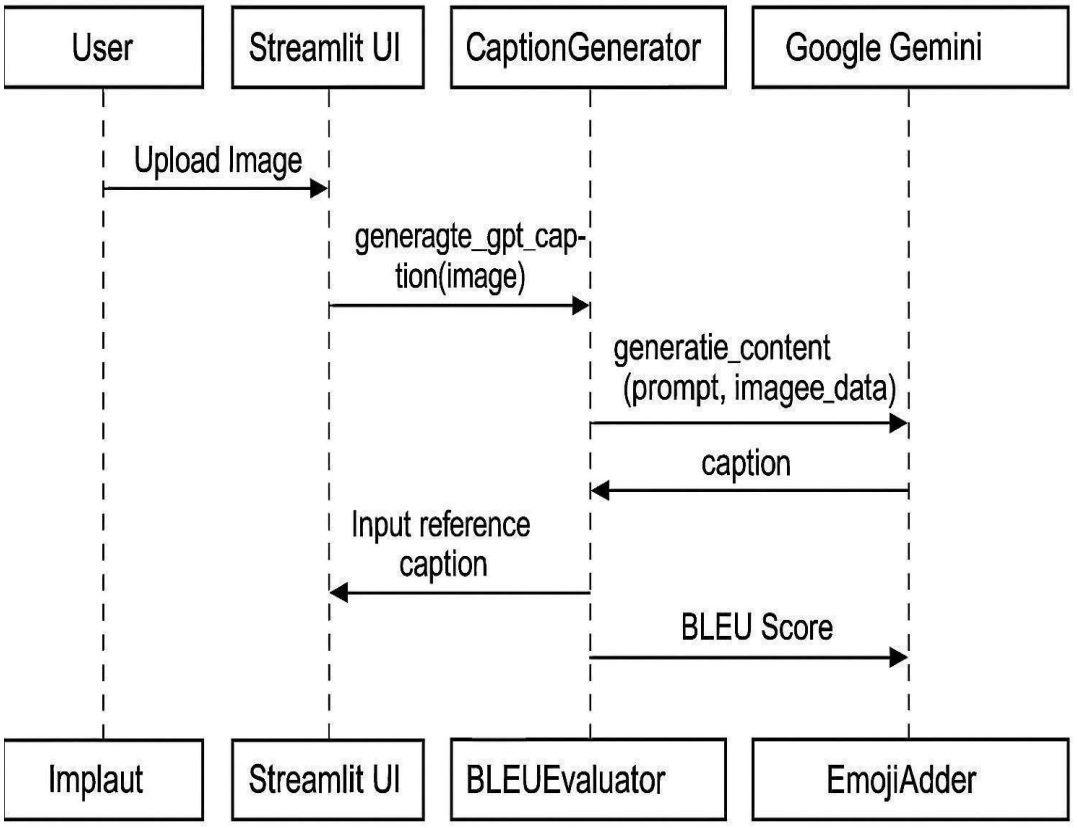

Fig: 4.1.3.1 Sequence Diagram for User

4.1.4 Data Flow Diagrams

A **Data Flow Diagram (DFD)** is a graphical representation that illustrates how data moves through a system or process. It helps analysts and developers visualize the flow of information, where it comes from, how it is processed, where it is stored, and how it exits the system. The DFD uses standardized symbols—rectangles for external entities, circles or rounded rectangles for processes, open-ended rectangles (or parallel lines) for data stores, and arrows to show the direction of data flow. Each of these elements is labelled with concise descriptions to clearly indicate the nature of the data and its movement.

At its core, a DFD breaks a system down into smaller, more manageable components. This is done through levels, with each level offering a different degree of detail. **Level 0**, also known as the **context diagram**, is the most basic form of a DFD. It shows the entire system as a single process and highlights its interaction with external entities—such as users, other systems, or departments.

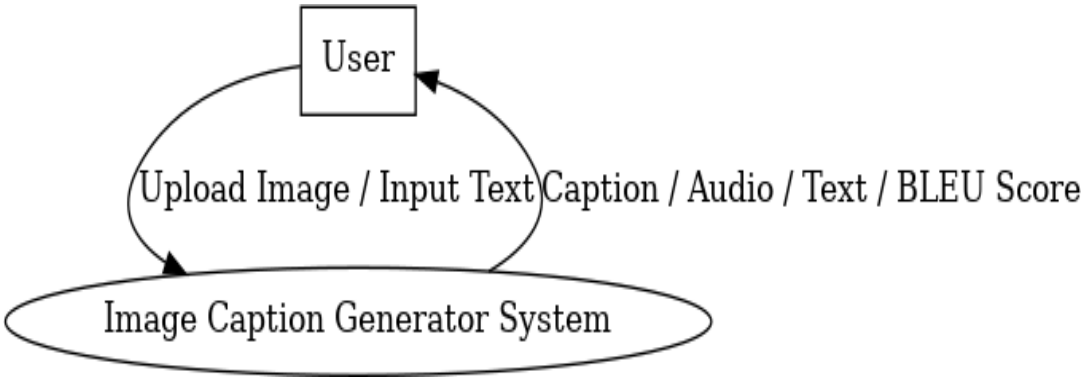

Fig: 4.1.4.1 Data Flow Diagram: level 0 for User

Level 1 of a DFD provides a deeper look into the internal workings of the system. It expands the single process shown in the context diagram into multiple subprocesses, each responsible for specific functions. This level also introduces data stores that reflect how and where data is saved or retrieved within the system. The interactions between processes, data flows, external entities, and storage points are all detailed, giving a more comprehensive understanding of how the system operates internally.

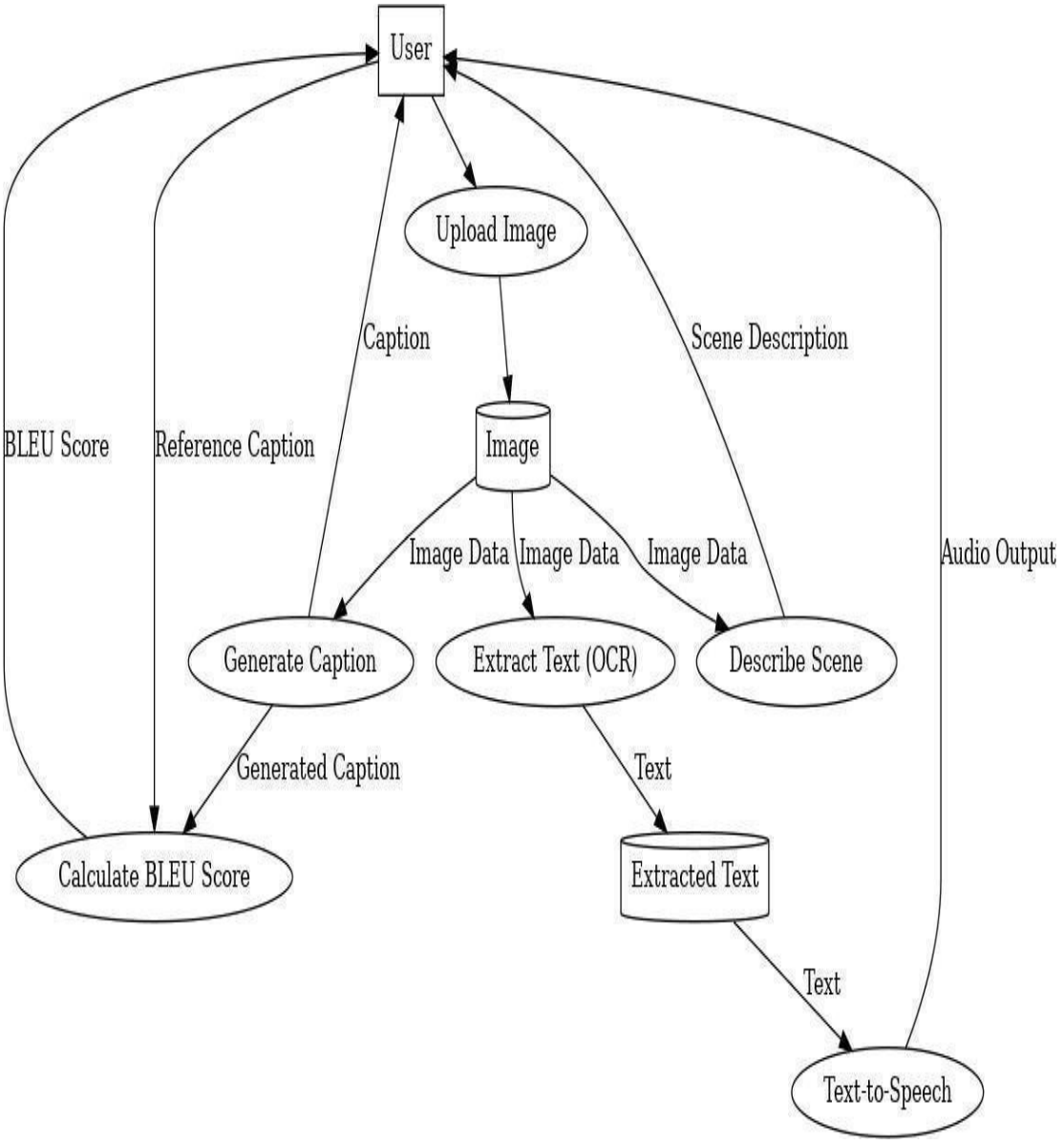

Fig: 4.1.4.2 Data Flow Diagram: Level 1 for User

4.1.5 Activity Diagram :

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step- by-step workflows of components in a system. An activity diagram shows the overall flow of control.

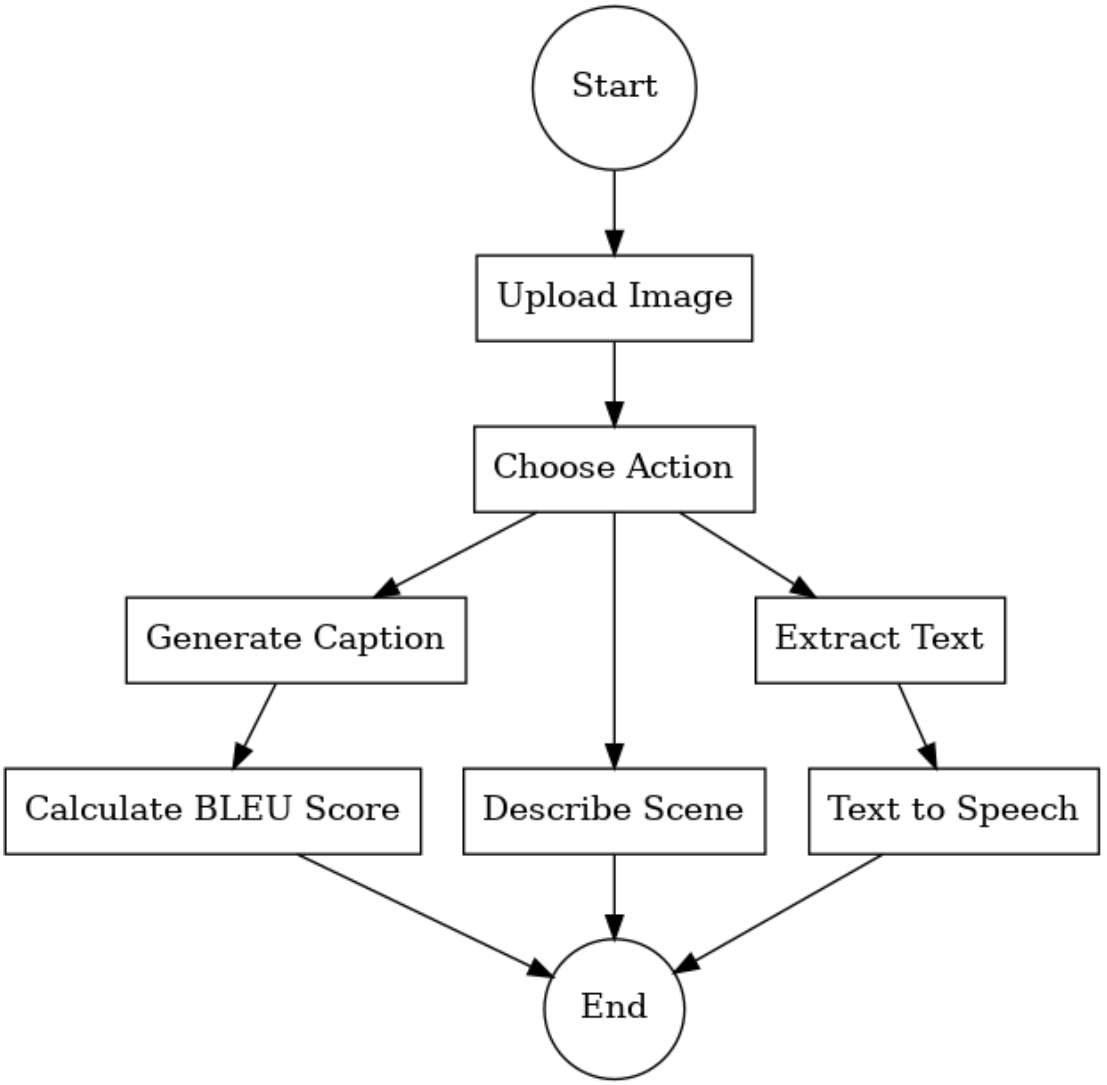

Fig :4.1.5.1 Activity Diagram for Image

4.2 MODULES

1. Streamlit

Streamlit is a Python-based open-source framework for building web applications for data science and machine learning. It is used to create a user-friendly interface for uploading images, viewing results, and interacting with the model in real-time.

2. OpenCV

OpenCV (Open Source Computer Vision Library) is used to handle image input and processing. In this project, it helps with visual tasks, although most visual analysis is offloaded to external APIs.

3. Pytesseract

Pytesseract is a wrapper for Google's Tesseract-OCR Engine. It extracts readable text from images, enabling OCR functionality in the app. It helps users get textual data embedded in images.

4. Google Generative AI (Gemini)

Google Gemini AI is a large-scale generative model used here to:
Generate scene descriptions.
Provide short captions for images. This model makes use of both textual and visual inputs, simulating human-like understanding of image content.

5. LangChain Integration

LangChain is used to interface the Gemini model for complex interactions and chaining AI responses, though it is used in a minimal form here for initialization.

6. PIL (Pillow)

Pillow is used for image manipulation tasks like reading, converting, or displaying uploaded images.

7. NLTK

The nltk library is used for evaluating the accuracy of captions using BLEU (Bilingual Evaluation Understudy) scores by comparing user-provided reference captions to generated ones.

8. pyttsx3

A Python text-to-speech library used to convert extracted text into speech, aiding accessibility for visually impaired users.

9. Regex and Time

Used internally to match emoji keywords and handle retry logic when the Gemini API is rate-limited.

CHAPTER 5

IMPLEMENTATION AND RESULTS

5.1 METHOD OF IMPLEMENTATION

5.1.1 What is Python:

Python is a high-level, interpreted, and general-purpose programming language known for its simplicity, readability, and vast ecosystem of libraries. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming, which makes it an ideal choice for both beginners and professionals. Python is widely adopted in the fields of **data science, machine learning, artificial intelligence, web development, and automation**.

One of the most compelling reasons for using Python in this project is its **extensive support for AI and image processing libraries**, such as TensorFlow, PyTorch, OpenCV, Tesseract (via pytesseract), Streamlit, and pyttsx3. Python also has a very active community, which means that most technical issues are well-documented and can be resolved quickly.

The AI-Powered Image Captioning System is developed entirely in Python due to the following advantages:

- **Simplicity**: Python's clean syntax allows for faster prototyping and reduced development time.
- **Library Support**: From machine learning to UIdevelopment, Python's ecosystem provides modules that cover every functionality needed in the project.
- **Integration**: Python seamlessly integrates APIs (such as Google Gemini), file systems, and UI frameworks.
- **Cross-Platform**: Python runs on Windows, Linux, and macOS, ensuring broader accessibility for both developers and end-users.

Python acts as the **bridge between AI models and user interaction**, enabling complex processes (captioning, OCR, speech synthesis) to be handled in just a few lines of high-level code. In essence, Python powers the entire logic layer of this intelligent system.

Python powers various domains, including:

- Machine Learning
- Web Development (via Django, Flask)

- Image Processing (OpenCV, PIL)
- Text Processing and OCR (Tesseract)
- GUI Applications (Tkinter, PyQt)
- Automation and Scripting

Python is highly extensible and supported by tech giants such as Google, Facebook, and Dropbox. One of its strongest features is the vast ecosystem of libraries and tools, making it ideal for rapid prototyping and scalable development.

5.1.2 What is Machine Learning:

Machine Learning (ML) is a subfield of artificial intelligence that focuses on building systems that can learn patterns from data and improve their performance over time without being explicitly programmed. Instead of writing rule-based code, machine learning allows systems to "learn" from examples and generalize to unseen scenarios. ML is broadly categorized into **supervised learning, unsupervised learning, and reinforcement learning**, depending on the type of feedback used during training.

In the context of **image captioning**, machine learning—specifically **deep learning**—plays a critical role. A typical image captioning model learns from large datasets of paired images and text (e.g., MS COCO) where it maps visual features to semantic language representations. Models such as **CNNs (for feature extraction)** and **RNNs/Transformers (for sequence generation)** are at the heart of this process.

This project leverages machine learning through:

- **Google Gemini AI**, a powerful generative multimodal model capable of interpreting images and generating coherent natural language captions.
- **BLEU scoring**, which is used to evaluate the accuracy of generated captions compared to human-written references.
- **Keyword-based emoji mapping**, where NLP techniques are used to understand the tone and emotion behind generated captions.

Though the ML models used in this project (e.g., Gemini) are accessed via API and not trained from scratch, they exemplify how ML is operationalized in real applications—**intelligent behavior without manual programming**. ML allows the system to adapt, interpret, and generate meaningful content from previously unseen inputs.

5.1.3 What is Image Captioning:

Image Captioning is the process of automatically generating a textual description of an image. It lies at the crossroads of Computer Vision (CV) and Natural Language Processing (NLP), two critical branches of AI. The goal of image captioning is not only to recognize objects within an image but also to understand their context, relationships, and actions, and describe them using grammatically correct and meaningful sentences. A complete image captioning system involves:

1. **Visual Feature Extraction** – Using models like CNNs or Transformers to convert raw images into numerical feature maps.
2. **Language Modeling** – Using sequence models like LSTMs or Transformers to decode those features into human-readable text.
3. **Context Awareness** – Recognizing the relationship between image elements (e.g., "a man riding a bike" vs. "a man standing next to a bike").

In this project, captioning is handled by the **Google Gemini Vision API**, which uses a large multimodal model pre-trained on vast datasets. The API returns both a detailed description and a concise caption, making it ideal for applications that require flexibility in tone and length. The system enhances these captions by:

- Integrating them with OCR-extracted text for more context.
- Adding emojis to increase relatability and emotional expression.
- Offering spoken output via TTS for improved accessibility.

Captioning is especially useful in assistive technologies, allowing visually impaired users to comprehend images, documents, and environments through auditory or tactile feedback. It also has widespread applications in digital marketing, e-learning, content automation, and dataset labelling.

5.2 EXPLANATION OF KEY FUNCTIONS

The AI-Powered Image Captioning System is comprised of several key functional modules, each designed to perform a specific task within the broader goal of converting visual content into readable and audible formats. The following is a detailed explanation of each key function used in the project, outlining its role, methodology, and significance.

1. Image Upload and Preprocessing

Purpose: To enable users to upload image files for captioning and text extraction.

Functionality:

- Allows upload of .jpg, .jpeg, .png formats.
- Displays image preview using Streamlit's st.image() function.
- Validates file type and size to prevent unsupported formats.

Importance: This module acts as the entry point for all downstream operations and ensures images are processed uniformly.

2. Automatic Caption Generation (Google Gemini API)

Purpose: To generate meaningful and context-aware image captions using a powerful AI model.

Functionality:

- Utilizes Google's Gemini Vision API, a state-of-the-art multimodal model.
- Sends the uploaded image to the API using the Python SDK.
- Receives two outputs:
- A short, social-media-friendly caption.
- A long-form scene description.

Technical Details:

- API interaction managed via google.generativeai.
- Requires API key authentication and JSON formatting.

Significance:

- The AI's understanding of spatial and contextual relationships enables natural and fluent captions that go beyond basic object detection.

3. Text Extraction via OCR (Tesseract Integration)

Purpose: To extract embedded or printed text present in the image (e.g., signs, posters, books).

Functionality:

- Uses **Tesseract OCR engine**, accessed through the pytesseract Python wrapper.
- Processes image pixels to detect character regions and convert them into editable strings.

Capabilities:

- Supports multi-line text extraction.
- Filters out non-printable and garbage characters.

Usage:

- Especially useful when images include written instructions, labels, or document scans.

Significance:

- Bridges the gap between **visual and linguistic content**, making the system suitable for document analysis and accessibility purposes.

4. Text-to-Speech (TTS) Conversion

Purpose: To provide auditory feedback for visually impaired users by reading aloud generated content.

Functionality:

- Converts both captions and OCR text to speech using **pyttsx3**, an offline TTS engine.
- Supports basic voice control:
- Start
- Stop
- Adjust speaking rate

Technical Insights:

- Unlike cloud-based TTS systems, pyttsx3 operates without internet, ensuring availability in low-connectivity environments.

Applications:

- Enhances accessibility, especially for blind or low-vision users, by allowing them to "listen to images."

5. BLEU Score Evaluation Module

Purpose: To measure the accuracy and quality of AI-generated captions compared to a human reference.

Functionality:

- Accepts a reference caption (manual description) as input.
- Computes the BLEU score using NLTK's sentence_bleu() method.
- Outputs a numerical value between 0 and 1 (or 0% to 100%).

Explanation:

- BLEU (Bilingual Evaluation Understudy) checks for n-gram overlap between the AI caption and reference.
- Higher BLEU scores indicate closer semantic similarity.

Importance:

- Provides a quantitative assessment for researchers or developers evaluating captioning models.
- Useful for A/B testing, model tuning, and demonstrating AI effectiveness.

6. Emoji Caption Enhancement Module

Purpose: To make the generated caption more expressive and engaging using emojis.

Functionality:

- Parses the generated caption for keywords (e.g., happy, dog, sun).
- Matches those keywords with an internal emoji dictionary.
- Appends one or more emojis to the end of the caption.

Example:

- Input: "A child is playing in the park."
- Output: "A child is playing in the park. ⚡ ⛺"

Significance:

- Adds emotional resonance and relatability to captions.
- Increases appeal for younger users, social media, and educational content.

7. Streamlit-Based User Interface

Purpose: To provide a real-time, browser-based graphical interface for all system functionalities.

Functionality:

- Displays image previews, outputs, captions, OCR results, and controls.
- Implements buttons to trigger:
- Captioning
- OCR
- TTS
- BLEU scoring

- Supports real-time updates and user feedback using Streamlit's reactive elements.

Benefits:

- Requires no technical expertise to use.
- Accessible on any device with a web browser.
- Lightweight and deployable via platforms like Streamlit Cloud.

8. Error Handling and System Feedback

Purpose: To provide robust error management and guide the user through every action.

Features:

- Invalid image formats prompt warnings.
- API failures trigger retry attempts and error messages.
- BLEU evaluation errors are caught and explained (e.g., if either caption is missing).

User Feedback Tools:

- st.warning(), st.error(), st.success() for intuitive alerts.
- Real-time spinners (st.spinner()) for loading feedback.

9. Modular Architecture and Code Reusability

Purpose: To ensure that each component can be developed, tested, and modified independently.

Benefits:

- Easy to maintain and scale.
- Enables future feature additions, such as:
- Real-time webcam capture
- Video captioning
- Multilingual TTS
- Saving and Cleanup:
- Users can exit the app by closing the browser tab. No explicit cleanup is required due to the stateless nature of Streamlit.

5.3 SOURCE CODE

```
import streamlit as st
import os
import pyttsx3
import pytesseract
import google.generativeai as genai
from langchain_google_genai import GoogleGenerativeAI
from nltk.translate.bleu_score import sentence_bleu
from PIL import Image
import io
import re
import time
from google.api_core.exceptions import ResourceExhausted
# --- SETUP ---
# Configure API Keys
GEMINI_API_KEY = "AIzaSyAiLIFpKGckyhEbM67kTg3cHC1fzQh7-yA" #
    Replace with your actual Gemini API Key
os.environ["GOOGLE_API_KEY"] = GEMINI_API_KEY
# Set Tesseract OCR Path
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-
    OCR\tesseract.exe'

# Initialize AI Models
llm = GoogleGenerativeAI(model="gemini-1.5-pro", api_key=GEMINI_API_KEY)
genai_model = genai.GenerativeModel("gemini-1.5-pro")

# Initialize Text-to-Speech
engine = pyttsx3.init()

# Emoji Mapping for Captions (kept for future use)
emoji_dict = {
   "dog": "cat": "car": "bike": ",
   "sun": "", "tree": "", "beach": "," "mountain":
```

```
    "ocean": "  "bird": "  "flower": "  ", "smile": "
}
# --- FUNCTIONS ---
def extract_text_from_image(image):
    """Extracts text from an image using OCR."""
    return pytesseract.image_to_string(image)
def text_to_speech(text):
    """Converts text to speech."""
    engine.say(text)
    engine.runAndWait()
def generate_scene_description(image_data):
    """Generates a scene description using Google Gemini with retry logic."""
    retries = 5
    for attempt in range(retries):
        try:
            response = genai_model.generate_content(["Describe this image.",
    image_data])
            return response.text
        except ResourceExhausted as e:
            if attempt < retries - 1:
                wait_time = 2 ** attempt
                st.warning(f"Rate limit exceeded. Retrying in {wait_time} seconds...")
                time.sleep(wait_time)
            else:
                st.error("Rate limit exceeded. Please try again later.")
                raise e

def generate_gpt_caption(image):
    """Generates a short one-line image caption using Google Gemini."""
    image_bytes = io.BytesIO()
    image.save(image_bytes, format="PNG")
    image_bytes.seek(0)
```

```
    image_data = {
        "mime_type": "image/png",
        "data": image_bytes.getvalue()
    }

    try:
        # Short prompt for fast, concise response
        response = genai_model.generate_content(["Give a one-line caption for this
    image.", image_data])
        caption = response.text.strip().split(".")[0] + "."
        return caption
    except Exception as e:
        st.error(f"Error generating caption: {e}")
        return "Could not generate caption."

def add_emojis(caption):
    """Adds emojis to the caption based on detected keywords (optional use)."""
    for word, emoji in emoji_dict.items():
        if re.search(rf'\b{word}\b', caption, re.IGNORECASE):
            caption += f" {emoji}"
    return caption
def calculate_bleu(reference, generated):
    """Calculates BLEU score to evaluate caption accuracy."""
    reference_tokens = reference.lower().split()
    generated_tokens = generated.lower().split()
    return sentence_bleu([reference_tokens], generated_tokens)
# --- STREAMLIT UI ---
st.set_page_config(page_title="Image Caption Generator", layout="wide",
    p a g e _ i c o n = " ")
st.markdown("<h1 style='text-align: center; color: #0662f6;'>Image Caption
    Generator</h1>", unsafe_allow_html=True)
```

```
st.markdown("<h4 style='text-align: center;'>Empowering the Partially Sighted with
    AI</h4>", unsafe_allow_html=True)

# Sidebar
st.sidebar.image("C:\\Users\\samudralasindhuja\\Downloads\\sidebar_image.jpg",
    width=250)
st.sidebar.title("About")
st.sidebar.markdown(
    """
    ''· □**Features**
    - ⁝ generated **scene descriptions**
    - ⁝_ *Image captions** with **emojis**
    - □·''. **Text extraction** from images
    - ''·□ **Multilingual translations**
    - ''·□ **Text-to-speech**
    """
)
# File Uploader
uploaded_file = ⁝ Upload an Image", type=["jpg", "jpeg", "png"])

 # Initialize session state for extracted text
 if "extracted_text" not in st.session_state:
     st.session_state.extracted_text = ""

if uploaded_file:
    image = Image.open(uploaded_file)
    st.image(image, cap ti o n = "  ⁝ Uploaded Image", use_column_width=True)
    col1, col2, col3 = st.columns(3)

    if col 1 . button(" ⁝_ Describe Scene"):
        with st.spinner("Analyzing scene..."):
```

```
            image_data = {"mime_type": uploaded_file.type, "data":
        uploaded_file.getvalue()}
            description = generate_scene_description(image_data)
            st.write(description)

        if col2.button("  Generate Caption"):
            with st.spinner("Generating caption..."):
                eng_caption = generate_gpt_caption(image)
                st.write(f"**Caption:** {eng_caption}")

        if col3.button("  Extract Text"):
            with st.spinner("Extracting text..."):
                st.session_state.extracted_text = extract_text_from_image(image)
                st.text_area("Extracted Text", st.session_state.extracted_text, height=150)

        if st.button("  Listen to Text") and st.session_state.extracted_text:
            text_to_speech(st.session_state.extracted_text)

        reference_caption = st.text_input("  Enter Reference Caption for BLEU Score:")
        if reference_caption:
            bleu_score = calculate_bleu(reference_caption, eng_caption)
            st.write(f"  BLEU Score: {bleu_score:.2f}")
    # Footer
    st.markdown("<hr><footer style='text-align:center;'>Built with £ˇ3 by Samudrala
        Sindhuja</footer>", unsafe_allow_html=True)
```

5.4 OUTPUT SCREENS

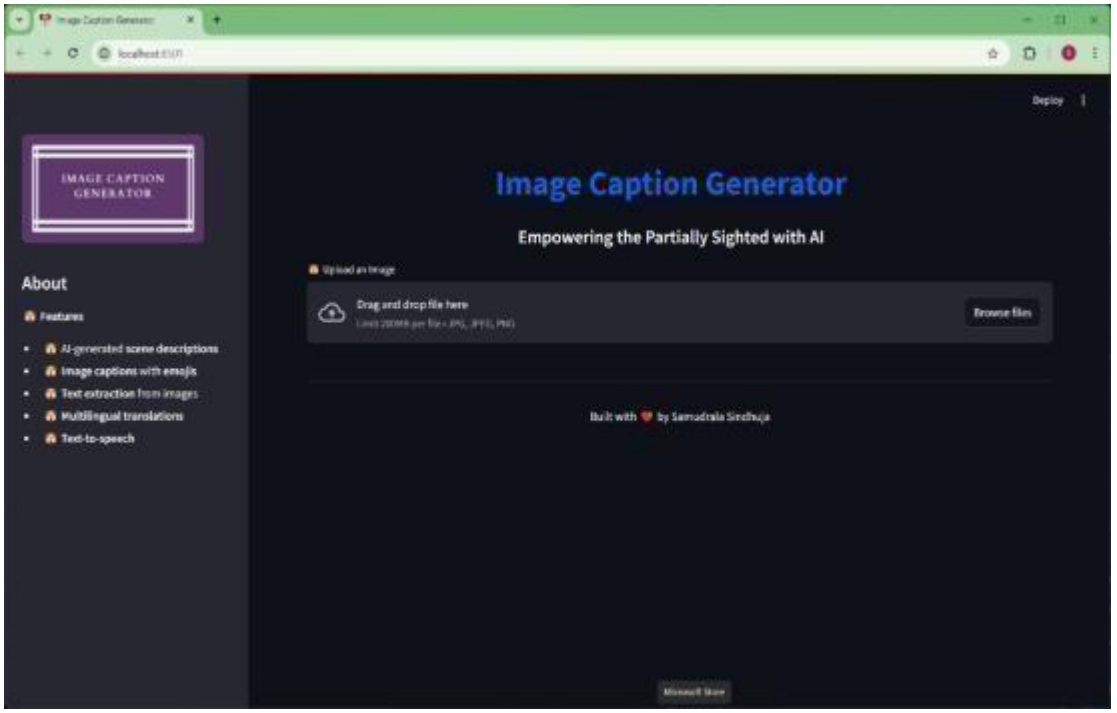

Fig 5.4.1: Initial display

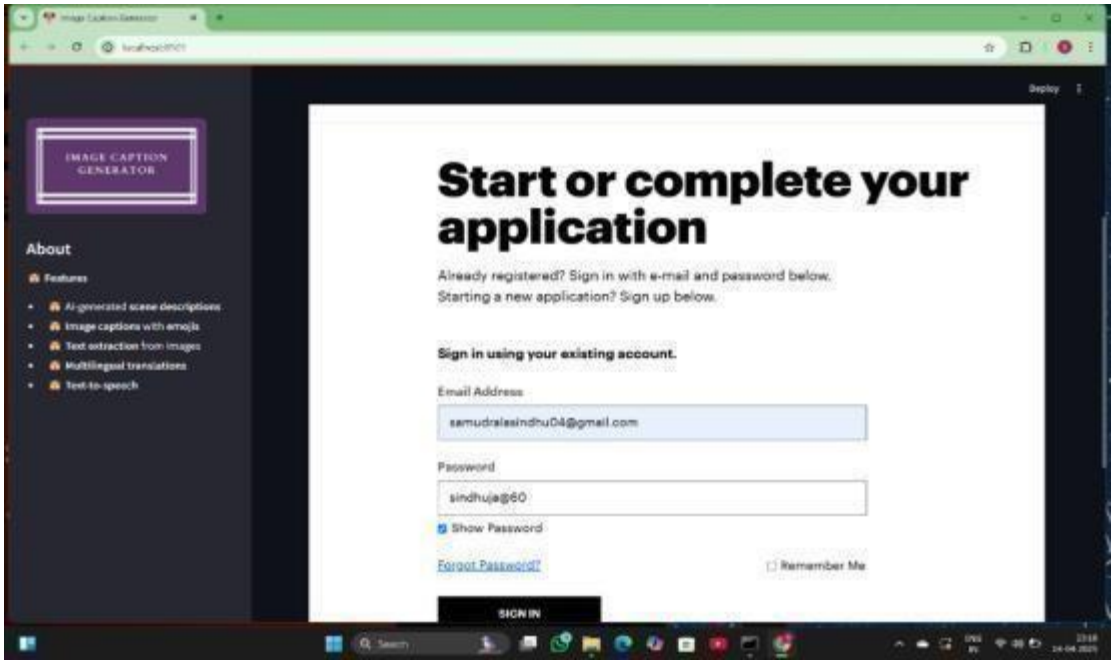

Fig 5.4.2: Application Overview

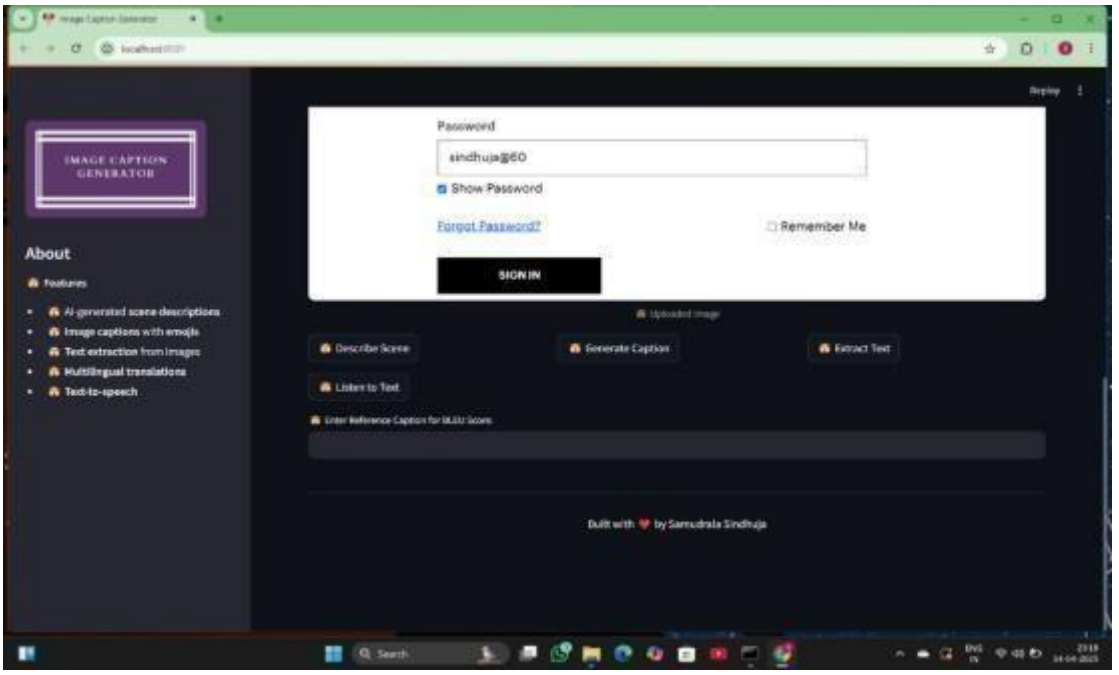

Fig 5.4.3: Options available

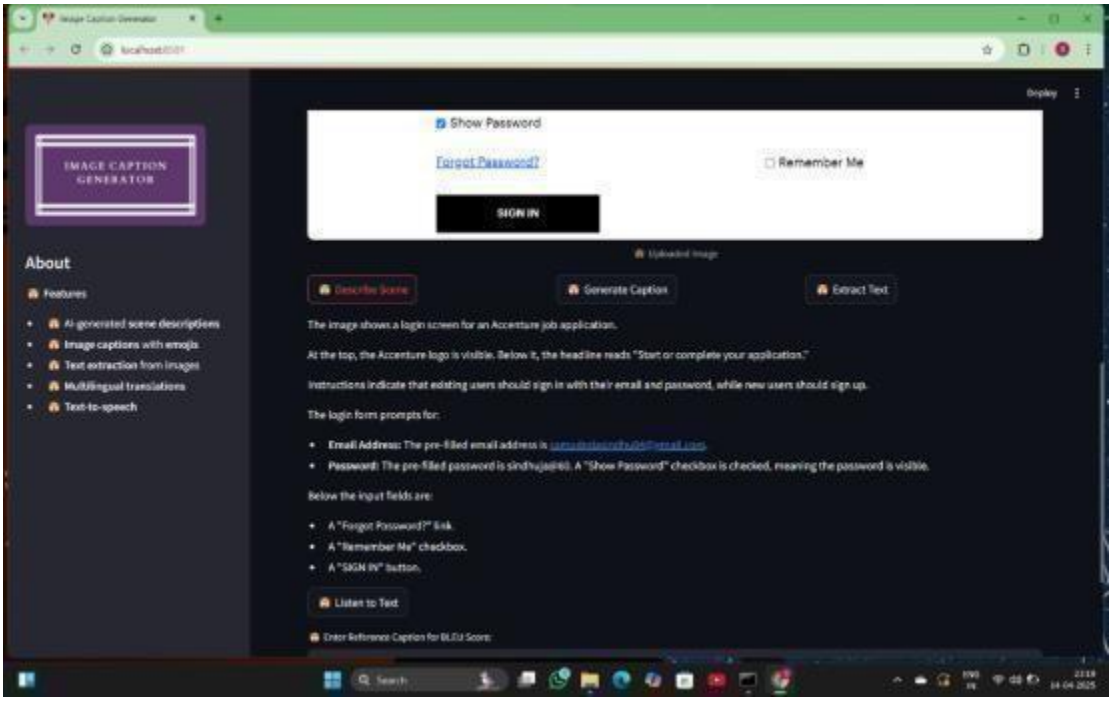

Fig 5.4.4: Describing the scene

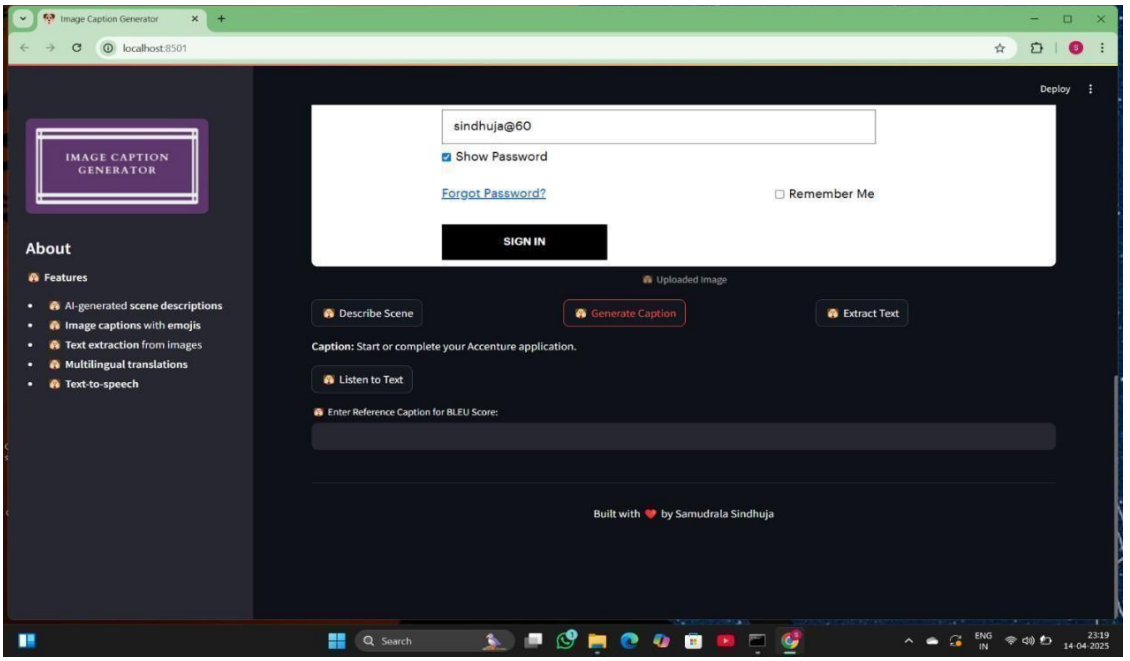

Fig 5.4.5: Caption generation

CHAPTER 6

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.1 TYPES OF TESTS

Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/ Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, it is purpose. It is used to test areas that cannot be reached from a black box level

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

It is a testing in which the software under test is treated, as a black box you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.2 VARIOUS TESTCASE SCENARIOS

Test CaseID	Description	Pre-Condition	Test Case Steps	Expected Result	Actual Result	Status
TC01	Upload imageand generate caption	Useris on the web app with internet	1. Upload valid .jpg/.png image 2. Click "Generate Caption"”	One-line captionis generated and displayed	As expected	Pass
TC02	Extract text from image	User uploads an imagewith visible text	1. Upload image 2. Click "Extract Text"	Extracted text appears in a text area	As expected	Pass
TC03	Describe sceneusing Gemini	Gemini APIkey is active	1. Upload image. 2. Describe Scene.	Scene description is returned from Gemini	As expected	Pass
TC04	Play extracted textusing TTS	Text has been extracted	1. Extract text from image 2. Click "Listento Text"	Speech outputreads text aloud	As expected	Pass
TC05	Calculate BLEU score	Caption has been generated	Captionhas been generated	BLEU score displayed on UI	As expected	Pass

TC06	Add emojis based on caption	Caption contains known keywords	1. Generate caption 2. Emoji added to captiontext	Emoji appearsin line with caption	As expected	Pass
TC07	Handle empty image upload	User does not upload any image	Click any action button without uploading image	Warningor no action taken.	As expected	Pass

Table 6.2.1 : Testcases

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

In this project, we successfully developed an AI-powered image captioning system aimed at empowering the partially sighted. The system utilizes Google Gemini AI for intelligent scene and caption generation, Tesseract for text extraction from images, and pyttsx3 for text-to-speech conversion, all within a Streamlit-based web application.
The solution provides a seamless user experience, enabling users to interact with images in real time by uploading them and receiving accurate textual content. The integration of accessibility features like voice feedback and intuitive captioning makes it ideal for visually impaired users.

Key Achievements:

- **AI-Powered Image Understanding:** Generated meaningful captions and scene descriptions using Google Gemini AI.
- **Text Extraction with OCR:** Incorporated Tesseract OCR to read text from images.
- **BLEU Score Evaluation:** Implemented caption evaluation through BLEU scoring.
- **Accessible User Interface:** Created an interactive web interface using Streamlit for live interactions.
- **Text-to-Speech Integration:** Enabled auditory output of extracted text to enhance usability.

This project stands out as a cost-effective, scalable, and accessible tool built using open-source technologies and APIs, making it highly adaptable for a wide range of real-world applications.

7.2 FUTURE ENHANCEMENT

While the current system is functional and effective, there are several opportunities for improvement and expansion:

- Multilingual Captioning: Extend the system to support automatic caption translations in regional languages.
- Voice Command Integration: Introduce voice-based interactions to enhance accessibility.
- Mobile App Deployment: Port the solution into a mobile-friendly app for greater reach and convenience.
- Cloud Storage and Sharing: Allow users to save and retrieve generated content via cloud integration.
- Extended Media Support: Add support for video frame extraction and captioning.
- AI Model Optimization: Optimize and fine-tune the captioning model for specific use cases like educational or medical images.
- User Profile Customization: Enable users to set preferences for languages, voice types, or caption formats.
- Offline Functionality: Introduce offline support by integrating lightweight models for use in environments without internet.
- Real-time Collaboration: Allow multiple users to collaborate on caption editing and review in real time.
- Accessibility Analytics: Provide insights and analytics on user interaction patterns to improve the accessibility experience. Add support for video frame extraction and captioning.

These enhancements would further solidify the system's role in aiding accessibility and broaden its user base.

CHAPTER 8

REFERENCES

8.1 WEBSITES

- www.google.com
- www.tutorialspoint.com
- www.developers.generativeai.google
- ieeexplore.ieee.org
- streamlit.io
- www.pypi.org/project/PyPDF2

8.2 PAPER REFERENCES

[1] Sonal Gupta, Dr. Satish Kumar, "Air-Writing Recognition Using Machine Learning Techniques," International Journal of Advanced Research in Computer Science and Software Engineering, Volume 8, Issue 4, April 2018.

[2] T. Zhang, S. Y. Huang, "Gesture-Based Human-Computer Interaction Using Convolutional Neural Networks," International Conference on Machine Vision and Information Technology (MVIT), October 2019.

[3] Y. Huang, X. Liu, X. Zhang, and L. Jin, "A Pointing Gesture Based Egocentric Interaction System: Dataset, Approach, and Application," 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Las Vegas, NV, pp. 370-377, 2016.

[4] P. Ramasamy, G. Prabhu, and R. Srinivasan, "An economical air writing system converting finger movements to text using a web camera," 2016 International Conference on Recent Trends in Information Technology (ICRTIT), Chennai, pp. 1-6, 2016.

[5] Saira Beg, M. Fahad Khan and Faisal Baig, "Text Writing in Air," Journal of Information Display Volume 14, Issue 4, 2013.

[6] Alper Yilmaz, Omar Javed, Mubarak Shah, "Object Tracking: A Survey", ACM Computer Survey. Vol. 38, Issue. 4, Article 13, Pp. 1-45, 2006.

[7] The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions by Sepp Hochreiter.

[8] Where to put the Image in an Image Caption Generator by Marc Tanti, Albert Gatt, Kenneth P. Camilleri.

[9] Sequence to sequence -video to text by Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, Kate Saenko.

[10] Learning phrase representations using RNN encoder-decoder for statistical machine translation by K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares,H. Schwenk, and Y. Bengi.

[11] TVPRNN for image caption generation. Liang Yang and Haifeng Hu.

[12] Image Captioning in the Wild: How People Caption Images Flickr Philipp Blandfort, Tushar Karayil Damian Borth, Andreas Dengel,German Institute for Articial Intelligence, Kaiserslautern, Germany.

[13 Image Caption Generator Based On Deep Neural Networks Jianhui Chen ,Wenqiang Dong, Minchen Li ,CS Department. ACM 2014.

[14] BLEU:A method for automatic evaluation of machine translation.In ACL, 2002 by K. Papineni, S. Roukos, T. Ward, and W. J. Zhu.