DESIGNING AN AUTOMATED IMAGE CAPTIONING SYSTEM FOR DIVERSE IMAGE DATASETS

A dissertation submitted in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

In

Computer Science and Engineering (AI&ML)

By

NALLA ADITYA

(23U61A6610)

Under the guidance of

Mrs. Rayees Fatima

B. Tech., MTech.

Assistant Professor

A NAAC Accredited Institution

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

GLOBAL INSTITUTE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUH)

(Recognized under Section 2(f) of UGC Act 1956)

An ISO:9001-2015 Certified Institution

CHILKUR (V), MOINABAD (M), R.R. DIST. T.S - 501504

JUNE-2025

GLOBAL
Institute of Engineering & Technology
Approved By AICTE, Affiliated TO JNTUH EAMCET CODE: GLOB

(Approved by AICTE & Affiliated to JNTUH)
(Recognized under Section 2(f) of UGC Act 1956)
An ISO:9001-2015 Certified Institution
Survey No. 179, Chilkur (V), Moinabad (M), Ranga Reddy Dist. TS.
JNTUH Code (U6) CIVIL – CSE – CSM – CSD - MECH – ECE – MBA – M.Tech. EAMCET Code– GLOB

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

Mrs. M. Shirisha **Date :02/06/2025**
B. Tech., M. Tech.
Assistant Professor & Head

CERTIFICATE

This is to certify that the project work entitled **"Designing an Automated image Captioning system for diverse image datasets"**, is a bonafide work of **NALLA ADITYA (HT.No: 23U61A6610)** submitted in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science and Engineering (AI&ML)** during the academic year 2024-25. This is further certified that the work done under my guidance, and the results of this work have not been submitted elsewhere for the award of any other degree or diploma.

Internal Guide
Mrs. Rayees Fatima
Assistant Professor

Head of the Department
Mrs. M. Shirisha
Assistant Professor

DECLARATION

I hereby declare that the project work entitled **Designing an Automated image Captioning system for diverse image datasets**, submitted to **Department of Computer Science and Engineering (AI &ML), Global Institute of Engineering & Technology, Moinabad,** affiliated to **JNTUH, Hyderabad** in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering (AI&ML)** is the work done by me and has not been submitted elsewhere for the award of any degree or diploma.

NALLA ADITYA (23U61A6610)

ACKNOWLEDGEMENT

I am thankful to my guide **Mrs. Rayees Fatima**, Assistant Professor of CSE (AI&ML) Department for her valuable guidance for successful completion of this project.

I express my sincere thanks to **Mrs. Rayees Fatima**, Project Coordinator for giving me an opportunity to undertake the project **"Designing an Automated image Captioning system for diverse image datasets"** and for enlightening me on various aspects of my project work and assistance in the evaluation of material and facts. She not only encouraged me to take up this topic but also given her valuable guidance in assessing facts and arriving at conclusions.

I am also most obliged and grateful to **Mrs. M. Shirisha**, Assistant Professor and Head, Department of CSE (AI&ML) for giving me guidance in completing this project successfully.

I express my heart-felt gratitude to our Vice-Principal **Prof. Dr. G Ahmed Zeeshan**, Co-Ordinator Internal Quality Assurance Cell (IQAC) for his constant guidance, cooperation, motivation and support which have always kept me going ahead. I owe a lot of gratitude to him for always being there for me.

I also most obliged and grateful to our Principal **Dr. P. Raja Rao** for giving me guidance in completing this project successfully.

I also thank my parents for their constant encourage and support without which the project would have not come to an end.

Last but not the least, I would also like to thank all my classmates who have extended their cooperation during our project work.

NALLA ADITYA (23U61A6610)

VISION

The vision of the department is to produce professional computer science engineers who can meet the expectations of the globe and contribute to the advancement of engineering and technology which involves creativity and innovations by providing an excellent learning environment with the best quality facilities.

MISSION

1. To provide the students with a practical and qualitative education in a modern technical environment that will help to improve their abilities and skills in solving programming problems effectively with different ideas and knowledge.
2. To infuse the scientific temper in the students towards the research and development in Computer Science and Engineering trends.
3. To mould the graduates to assume leadership roles by possessing good communication skills, an appreciation for their social and ethical responsibility in a global setting, and the ability to work effectively as team members.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1: To provide graduates with a good foundation in mathematics, sciences and engineering fundamentals required to solve engineering problems that will facilitate them to find employment in MNC’s and / or to pursue post graduate studies with an appreciation for lifelong learning.

PEO2: To provide graduates with analytical and problem-solving skills to design algorithms, other hardware / software systems, and inculcate professional ethics, inter-personal skills to work in a multi-cultural team.

PEO3: To facilitate graduates to get familiarized with the art software / hardware tools, imbibing creativity and innovation that would enable them to develop cutting edge technologies of multi-disciplinary nature for societal development.

PROGRAMME OUTCOMES

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences.
PO3: Design/development of solutions: design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.
PO4: Conduct investigations of complex problems: use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5: Modern tool usage: create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6: The engineer and society: apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7: Environment sustainability: understand the impact of the professional engineering solutions in the societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8: Ethics: apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9: Individual and team work: function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10: Communication: communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11: Project management and finance: demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Lifelong learning: recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broader context of technological change.

PROGRAMME SPECIFIC OUTCOMES

PSO1: An Ability to Apply the fundamentals of mathematics, science, Computer Science and Engineering Knowledge to analyze and implement programs in the areas related to Algorithms, Data structures, Web Designing, Networking, Data science, Machine Learning for efficient Design of computer-based system to deal with Real time Problems.

PSO2: Ability to implement the solutions for problems using Artificial Intelligence and Machine Learning algorithms and techniques

.

ABSTRACT

The rapid proliferation of digital imagery across various platforms necessitates efficient systems for managing, indexing, and retrieving visual content. This project presents the design and implementation of an automated image captioning system tailored for diverse image datasets, aimed at improving the accessibility, discoverability, and usability of visual assets in large-scale digital environments. Leveraging advances in deep learning, particularly in convolutional neural networks (CNNs) for image feature extraction and transformer-based architectures for natural language generation, the system generates accurate, context-aware captions for images across domains such as e-commerce, media, healthcare, and social platforms.

The automated captions serve multiple functions: they act as metadata for intelligent tagging, support enhanced search capabilities through semantic indexing, and enable more personalized content recommendation engines. By integrating the captioning model with existing digital asset management (DAM) and content delivery systems, the platform achieves improved automation in organizing and retrieving media assets. The model is trained and evaluated on a diverse, annotated dataset to ensure generalizability and robustness in real-world scenarios.

A key innovation lies in the adaptability of the system to different domains, using fine-tuning strategies and domain-specific pre-processing to maintain high caption accuracy and relevance. Additionally, the system supports multilingual output, enhancing its utility in global applications. The result is a scalable, intelligent solution for image understanding that streamlines content workflows, enriches user experiences, and facilitates data-driven decisions in managing visual content. This documentation outlines the system architecture, model pipeline, training methodology, and integration strategies for deployment at scale.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The explosive growth of digital images across sectors such as e-commerce, social media, healthcare, and digital publishing has created an urgent need for intelligent systems that can effectively manage, describe, and retrieve visual content. Traditional methods of manually annotating or tagging images are inefficient, error-prone, and unable to scale with the volume and diversity of modern datasets. As a result, there is a growing demand for automated solutions that can not only interpret image content accurately but also translate that understanding into meaningful, human-readable descriptions.

Automated image captioning addresses this challenge by combining the capabilities of computer vision and natural language processing to generate textual descriptions of images. These captions provide semantic context, enabling intelligent tagging and indexing that enhance search functionality, support personalized content recommendations, and streamline digital asset management. The integration of deep learning models, including convolutional neural networks (CNNs) or vision transformers (ViTs) for image feature extraction and sequence models such as LSTMs or transformers for language generation, has significantly advanced the performance and reliability of such systems.

This documentation outlines the design and implementation of a scalable, domain-adaptable image captioning system capable of handling diverse image datasets. The system is intended to improve accessibility, automate metadata generation, and provide structured insights into unstructured visual data. With support for multilingual output and adaptability to specific domain requirements, the system ensures relevance and usability across global and enterprise-level platforms.

1.2 EXISTING SYSTEM

Current image captioning systems are primarily built on encoder-decoder architectures, where visual features are extracted using convolutional neural networks (CNNs) or vision transformers (ViTs), and captions are generated through sequence models such as LSTMs or transformers. Popular models like Show and Tell, Show, Attend and Tell, and transformer-based architectures such as BLIP and OFA have demonstrated notable performance on standard datasets like MS-COCO and Flickr30K.

These systems work effectively in constrained environments and on curated datasets. However, they are often trained on limited domains, resulting in reduced performance when applied to diverse or real-world images from multiple industries. Many existing solutions are standalone and not designed for seamless integration into enterprise-scale platforms, lacking features such as dynamic metadata tagging, intelligent indexing, or multilingual support.

DISADVANTAGES OF EXISTING SYSTEMS

- Limited generalization to diverse and domain-specific datasets.
- Poor integration with digital asset management (DAM) tools and recommendation systems.
- Tendency to produce generic or incomplete captions.
- Inability to scale effectively for large, unstructured image repositories.
- Lack of support for multilingual captioning or domain adaptability.

1.3 PROPOSED SYSTEM

The proposed image captioning system is a robust, scalable, and domain-adaptive solution designed to address the limitations of current captioning technologies. It integrates cutting-edge computer vision and natural language processing models to generate rich, context-aware captions for diverse image datasets. The system is architected to enhance intelligent tagging, semantic indexing, content recommendation, and digital asset management in large-scale environments.

At its core, the system uses a dual-stage architecture:

- **Visual Encoder**: Employs state-of-the-art models like Vision Transformers (ViTs) or pre-trained CNNs (e.g., ResNet,EfficientNet) for detailed image feature extraction.
- **Language Generator**: Utilizes transformer-based models (e.g., GPT, BERT, T5) for fluent and semantically accurate caption generation.

Domain-specific fine-tuning ensures that the model adapts to various industries such as retail, healthcare, education, and media. The architecture is modular and designed for seamless integration with existing platforms and enterprise systems.

Key features

- **Multilingual Caption Support**: Enables global accessibility and localization.
- **Domain Adaptability**: Custom fine-tuning for specific industries or image types.
- **Semantic Tagging & Indexing**: Automatically generates metadata for improved search and retrieval.
- **DAM & Recommendation Integration**: Compatible with digital asset management and content delivery systems.

- **Scalable Infrastructure**: Supports high-volume, real-time processing.
- **Continuous Learning**: Incremental updates with new data to improve performance over time.
- **Quality Assurance Pipeline**: Includes post-processing and validation modules to ensure caption relevance and accuracy.

1.4 ADVANTAGES OF THE PROPOSED SYSTEM

- Supports diverse and domain-specific datasets.
- Integrates seamlessly with DAM, search, and recommendation systems.
- Produces accurate, context-rich, and multilingual captions.
- Enables intelligent tagging and semantic indexing.
- Scalable and adaptable for large-scale, real-time deployment.
- Facilitates continuous learning and system improvement.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE SURVEY

The task of automated image captioning has gained significant attention in recent years due to its wide-ranging applications in information retrieval, accessibility, and content management. A variety of techniques have been proposed and refined, evolving from traditional template-based methods to modern deep learning architectures. This literature survey highlights five key research contributions that have shaped the field and are relevant to the design of scalable, domain-adaptive image captioning systems.

1. **Vinyals et al. (2015) – "Show and Tell: A Neural Image Caption Generator"**
   This pioneering work introduced an end-to-end neural network architecture combining a convolutional neural network (CNN) for image encoding with a long short-term memory (LSTM) network for sentence generation. Trained on the MS-COCO dataset, it demonstrated the effectiveness of using deep learning for generating human-like captions. However, it struggled with generalizing across domains and lacked contextual grounding for complex scenes.
2. **Xu et al. (2015) – "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention"**
   Building on the previous model, this paper introduced an attention mechanism that allowed the model to focus on different parts of an image when generating each word in a caption. This significantly improved performance and interpretability, especially for images with multiple objects, and laid the foundation for more adaptive and context-aware captioning approaches.
3. **Anderson et al. (2018) – "Bottom-Up and Top-Down Attention for Image Captioning and VQA"**
   This model introduced a novel combination of bottom-up attention (object detection-based feature extraction) and top-down attention (task-driven language modeling). The integration enabled the system to better capture salient visual regions and generate more relevant descriptions. It became a strong baseline for many subsequent captioning systems.
4. **Radford et al. (2021) – "Learning Transferable Visual Models From Natural Language Supervision (CLIP)"**
   CLIP proposed a multi-modal framework trained on large-scale image-text pairs from the internet. Although not explicitly an image captioning model, it demonstrated powerful zero-shot learning and cross-modal understanding. CLIP inspired newer captioning systems like BLIP and Flamingo that leverage pre-trained vision-language models to improve generalizability and performance on diverse datasets.

5. **Li et al. (2022) – "BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation"**
   BLIP advanced the field by integrating image-text contrastive learning with image-text matching and captioning in a unified framework. It performed strongly across tasks, including image captioning, and showed superior results on out-of-domain data. Its flexibility and strong performance make it suitable for scalable systems that must handle diverse, unstructured image inputs.

2.2 ABOUT PYTHON

2.2.1 FEATURES OF THE LANGUAGE USED

PYTHON

- Python Is an Object-Oriented, High-Level Language, Interpreted, Dynamic and Multipurpose Programming Language.
- Python Is Easy to Learn Yet Powerful and Versatile Scripting Language Which Makes It Attractive for Application Development.
- Python's Syntax and Dynamic Typing with Its Interpreted Nature, Make It an Ideal Language for Scripting and Rapid Application Development in Many Areas.
- Python Supports Multiple Programming Pattern, Including Object Oriented Programming, Imperative and Functional Programming or Procedural Styles.
- Python Is Not Intended to Work on Special Area Such as Web Programming. That Is Why It Is Known as Multipurpose Because It Can Be Used with Web, Enterprise, 3d Cad Etc.

PYTHON FEATURES

1) Easy To Use:

Python Is Easy to Very Easy to Use and High-Level Language. Thus, It Is Programmer-Friendly Language.

2) Expressive Language:

Python Language Is More Expressive. The Sense of Expressive Is the Code Is Easily Understandable.

3) Interpreted Language:

Python Is an Interpreted Language I.E. Interpreter Executes the Code Line by Line at a Time. This Makes Debugging Easy and Thus Suitable for Beginners.

4) Cross-Platform Language:

Python Can Run Equally on Different Platforms Such as Windows, Linux, Unix, Macintosh Etc. Thus, Python Is a Portable Language.

5) Free And Open Source:

Python Language Is Freely Available (Www.Python.Org). The Source-Code Is Also Available. Therefore, It Is Open Source.

6) Object-Oriented Language:

Python Supports Object Oriented Language. Concept Of Classes and Objects Comes into Existence.

7) Extensible:

It Implies That Other Languages Such As C/C++ Can Be Used to Compile the Code and Thus It Can Be Used Further in Your Python Code.

8) Large Standard Library:

Python Has a Large and Broad Library.

9) Gui Programming:

Graphical User Interfaces Can Be Developed Using Python.

10) Integrated:

It Can Be Easily Integrated with Languages Like C, C++, Java Etc.

PYTHON HISTORY

- Python Laid Its Foundation in The Late 1980s.
- The Implementation of Python Was Started in The December 1989 by Guido Van Rossum at Cwi in Netherland.
- Abc Programming Language Is Said to Be the Predecessor of Python Language Which Was Capable of Exception Handling and Interfacing with Amoeba Operating System.
- Python Is Influenced by Programming Languages Like:
- Modula-3

2.2.2 WHAT IS COMPUTER VISION?

Computer Vision Is the Broad Parent Name for Any Computations Involving Visual Content. That Means Images, Videos, Icons, And Anything Else with Pixels Involved. But Within This Parent Idea, There Are A Few Specific Tasks That Are Core Building Blocks:

- In Object Classification, You Train a Model on A Dataset of Specific Objects, And the Model Classifies New Objects as Belonging to One Or More of Your Training Categories.
- For Object Identification, Your Model Will Recognize a Specific Instance of an Object – For Example, Parsing Two Faces in an Image and Tagging One as Tom Cruise and One as Katie Holmes.
- A Classical Application of Computer Vision Is Handwriting Recognition for Digitizing Handwritten Content (We'll Explore More Use Cases Below). Outside Of Just Recognition, Other Methods of Analysis Include:
- Video Motion Analysis Uses Computer Vision to Estimate the Velocity of Objects in A Video, Or the Camera Itself.
- In Image Segmentation, Algorithms Partition Images into Multiple Sets of Views.
- Scene Reconstruction Creates A 3d Model of a Scene Inputted Through Images or Video (Check Out Selva).
- In Image Restoration, Noise Such as Blurring Is Removed from Photos Using Machine Learning Based Filters.
- Any Other Application That Involves Understanding Pixels Through Software Can Safely Be Labeled as Computer Vision.

Where can Tensor flow run?

TensorFlow can hardware, and software requirements can be classified into

Development Phase: This is when you train the mode. Training is usually done on your Desktop or laptop.

Run Phase or Inference Phase: Once training is done TensorFlow can be run on many different platforms. You can run it on

Desktop running Windows, macOS or Linux

Cloud as a web service

Mobile devices like iOS and Android

You can train it on multiple machines then you can run it on a different machine, once you have the trained model.

The model can be trained and used on GPUs as well as CPUs. GPUs were initially designed for video games. In late 2010, Stanford researchers found that GPU was also very good at matrix operations and algebra so that it makes them very fast for doing these kinds of calculations. Deep learning relies on a lot of matrix multiplication. TensorFlow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, TensorFlow can be accessed and controlled by other languages mainly, Python.

Finally, a significant feature of Tensor Flow is the Tensor Board. The Tensor Board enables to monitor graphically and visually what TensorFlow is doing.

List of Prominent Algorithms supported by TensorFlow

- Linear regression: tf. estimator. Linear Regressor
- Classification: tf. Estimator. Linear Classifier
- Deep learning classification: tf. estimator. DNN Classifier
- Python idle
- Anaconda navigator
- Opencv

2.3 ABOUT JAVA

Initially the language was called as "oak" but it was renamed as "Java" in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer's language.
- Java is cohesive and consistent.
- Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.
- Finally, Java is to Internet programming where C was to system programming.

IMPORTANCE OF JAVA TO THE INTERNET

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer.

They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

JAVA CAN BE USED TO CREATE TWO TYPES OF PROGRAMS

Applications and Applets: An application is a program that runs on our computer under the operating system of that computer. It is more or less like one creating using C or C++. Java's ability to create Applets makes it important.

An Applet is an application designed to be transmitted over the Internet and executed by a Java –compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

FEATURES OF JAVA

SECURITY

Every time you that you download a "normal" program, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scanned them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a "firewall" between a network application and your computer.

When you use a Java-compatible Web browser, you can safely download Java applets without fear of virus infection or malicious intent.

JAVA ARCHITECTURE

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

COMPILATION OF CODE

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine

JAVASCRIPT

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then updates the browser's display accordingly

Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client-side programming since most of the browsers supports it. JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags

Here are a few things we can do with JavaScript:

- ❖ Validate the contents of a form and make calculations.
- ❖ Add scrolling or changing messages to the Browser's status line.
- ❖ Detect the browser in use and display different content for different browsers.
- ❖ Detect installed plug-ins and notify the user if a plug-in is required.

We can do much more with JavaScript, including creating entire application.

JAVASCRIPT VS JAVA

- ❖ JavaScript and Java are entirely different languages. A few of the most glaring differences are: Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.
- ❖ While JavaScript is best suited to simple applications and adding interactive features to Web pages; Java can be used for incredibly complex applications.

There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact, they can be used together to combine their advantages.

ADVANTAGES

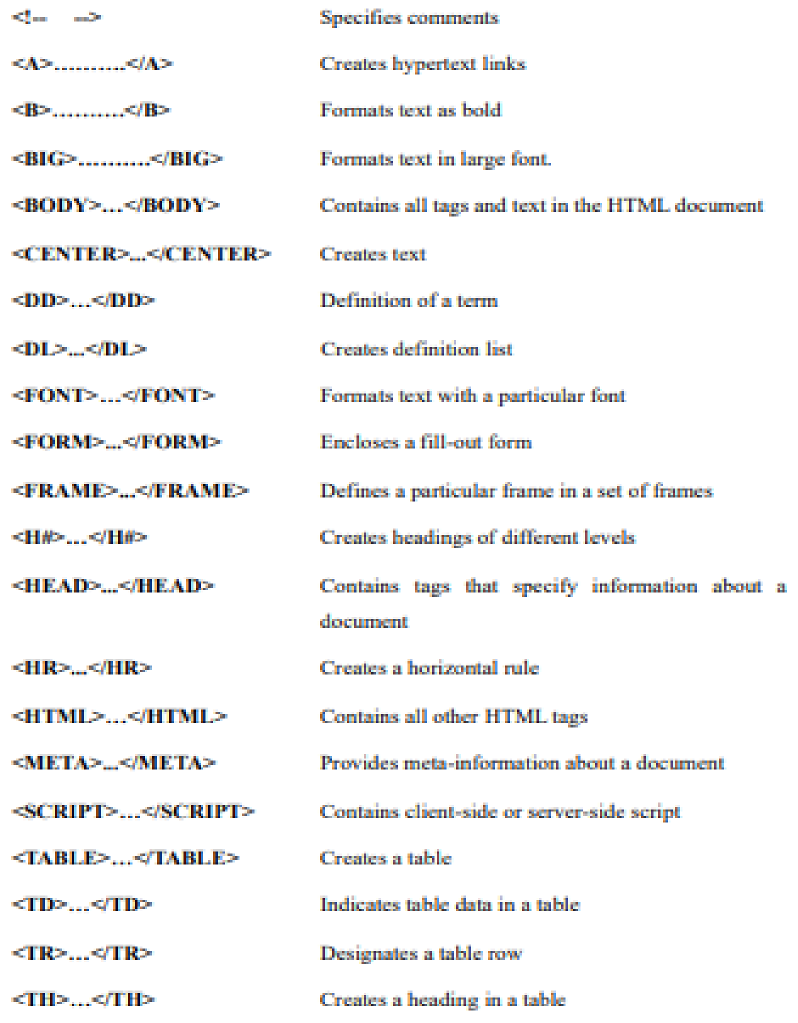
- ❖ JavaScript can be used for Sever-side and Client-side scripting.
- ❖ It is More flexible than VBScript.
- ❖ JavaScript is the default scripting languages at Client-side since all the browsers supports it.

2.4 HYPER TEXT MARKUP LANGUAGE (HTML)

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.



<!-- -->	Specifies comments
<A>..........</A>	Creates hypertext links
<B>..........</B>	Formats text as bold
<BIG>..........</BIG>	Formats text in large font.
<BODY>...</BODY>	Contains all tags and text in the HTML document
<CENTER>...</CENTER>	Creates text
<DD>...</DD>	Definition of a term
<DL>...</DL>	Creates definition list
<FONT>...</FONT>	Formats text with a particular font
<FORM>...</FORM>	Encloses a fill-out form
<FRAME>...</FRAME>	Defines a particular frame in a set of frames
<H#>...</H#>	Creates headings of different levels
<HEAD>...</HEAD>	Contains tags that specify information about a document
<HR>...</HR>	Creates a horizontal rule
<HTML>...</HTML>	Contains all other HTML tags
<META>...</META>	Provides meta-information about a document
<SCRIPT>...</SCRIPT>	Contains client-side or server-side script
<TABLE>...</TABLE>	Creates a table
<TD>...</TD>	Indicates table data in a table
<TR>...</TR>	Designates a table row
<TH>...</TH>	Creates a heading in a table

CHAPTER 3

3.1 SYSTEM REQUIREMENTS SPECIFICATIONS

3.1.1 HARDWARE REQUIREMENTS

PROCESSOR	Intel Core i5 or AMD Ryzen 5
HARD DISK	50 GB
MONITOR	15 VGA Colour.
MOUSE	Optical USB Mouse
RAM	8 GB and above

3.1.2 SOFTWARE REQUIREMENTS

OPERATING SYSTEM	Windows 10/11
CODING LANGUAGE	PYTHON, JS, HTML, CSS

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

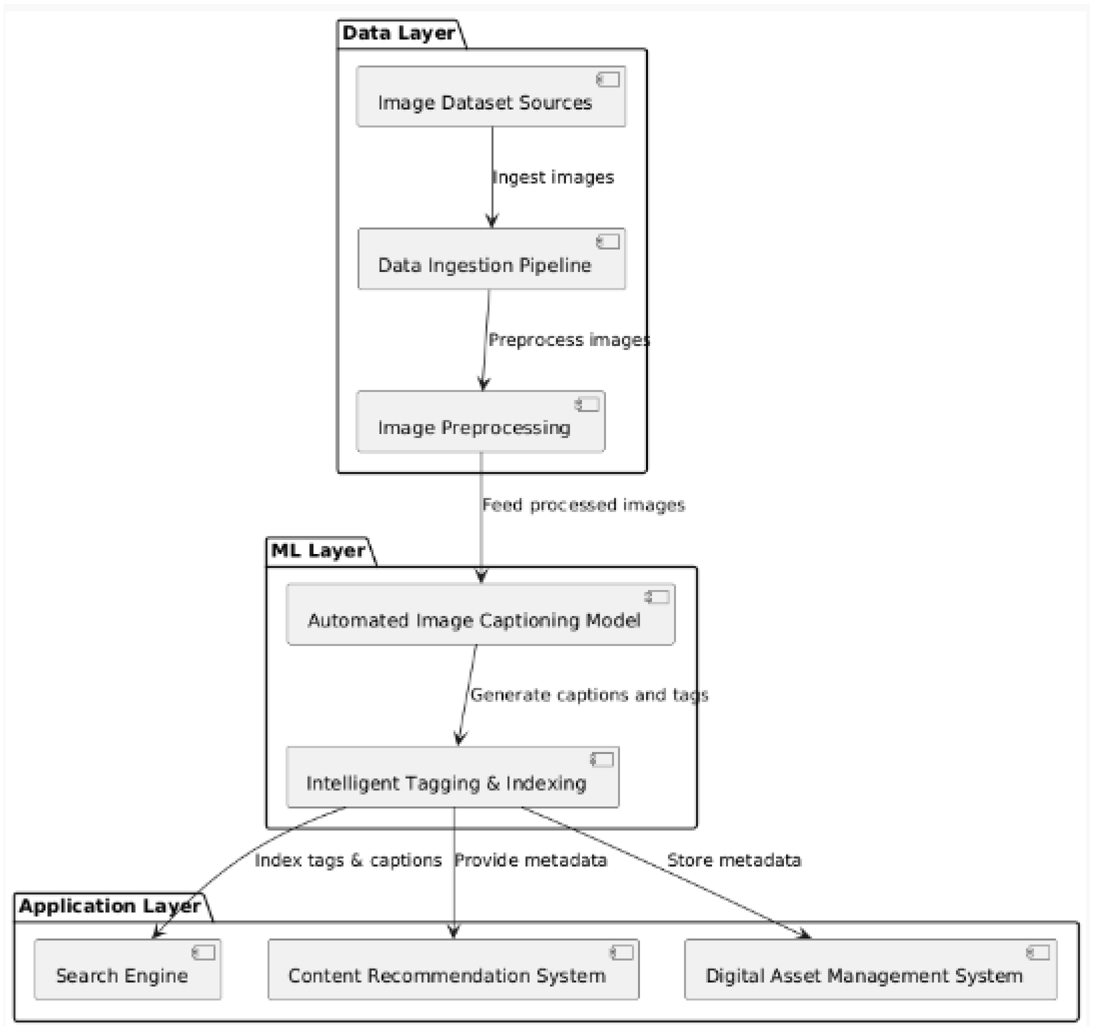


FIG 4.1 SYSTEM ARCHITECTURE

This system is designed to automatically generate accurate, context-aware captions and audio descriptions for visual content, improving accessibility on content management platforms.

1. **Image Dataset Sources**

    - Multiple origins of images, such as user uploads, public image repositories, social media feeds, or internal corporate databases.
    - The system is designed to handle **diverse image datasets**, including different formats, resolutions, and content types.

2. **Data Ingestion Pipeline**

    - Responsible for collecting images in real-time or batch mode.
    - Handles image validation, metadata extraction (e.g., timestamp, source), and queuing images for processing.
    - Scales to handle large volumes efficiently.

3. **Image Pre-processing**

    - Standardizes images by resizing, normalizing colours, and enhancing quality.
    - May include noise reduction or cropping to improve caption model accuracy.
    - Converts images into a format compatible with the captioning model input.

4. **Automated Image Captioning Model**

    - A deep learning model, typically based on CNN + RNN/Transformer architectures, trained to generate natural language descriptions (captions) for images.
    - It takes the pre-processed images and outputs textual captions describing their content.
    - Supports multi-lingual captions or domain-specific adaptation for better relevance.

5. **Intelligent Tagging & Indexing**

    - Parses generated captions to extract key entities, objects, scenes, and attributes.
    - Tags images with these keywords and indexes them in a metadata store optimized for fast querying.
    - Supports hierarchical and semantic tagging for enhanced search relevance.

6. **Search Engine**

    - Uses the indexed tags and captions to enable fast and relevant image search.
    - Supports keyword search, semantic search, and filters based on metadata.
    - Provides APIs or UI for end-users to retrieve images efficiently.

7. **Content Recommendation System**

- ➢ Uses tagging and caption metadata to find related or personalized image recommendations.
- ➢ Applies collaborative filtering, content-based filtering, or hybrid methods to suggest images based on user preferences or context.

8. **Digital Asset Management (DAM) System**

- ➢ Manages storage, retrieval, versioning, and access control of large-scale image assets.
- ➢ Uses captioning metadata for categorization and lifecycle management.
- ➢ Integrates with organizational workflows, enabling efficient asset utilization.

4.2 DATAFLOW DIAGRAM

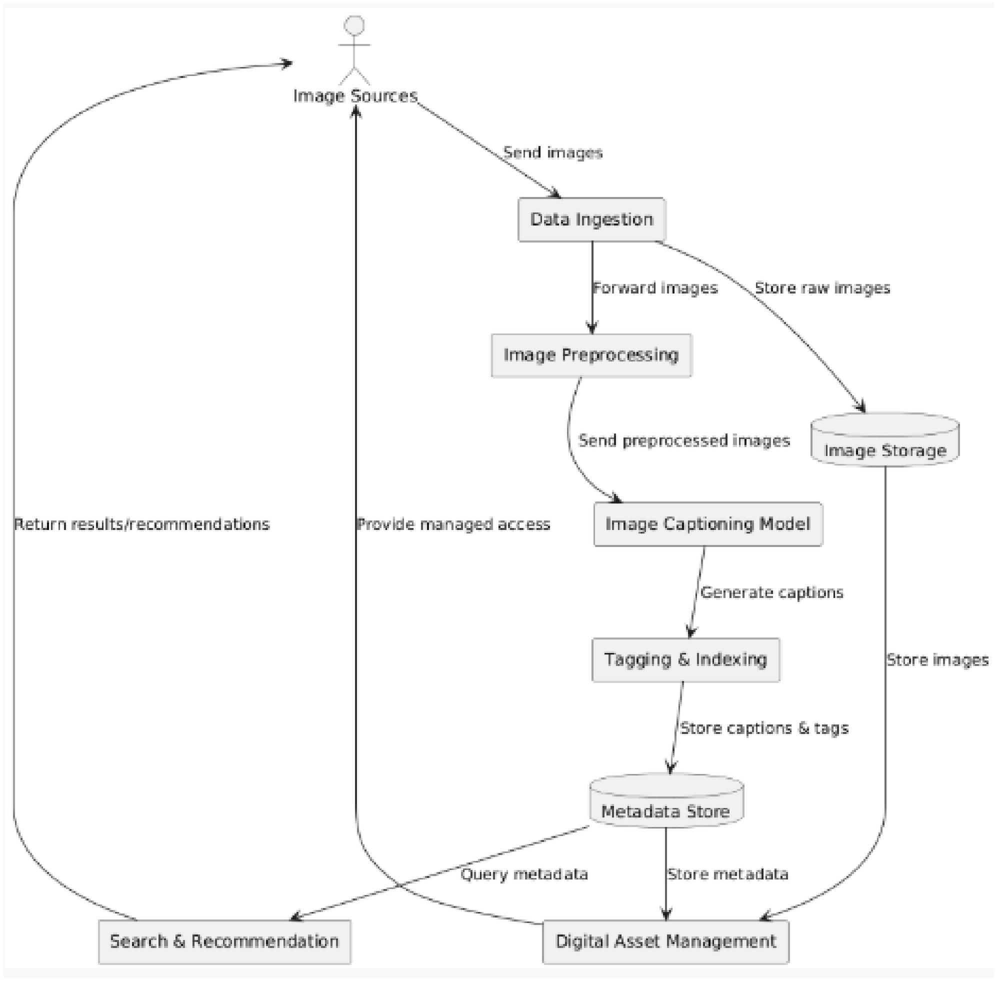


FIG 4.2 DATAFLOW DIAGRAM

EXTERNAL ENTITY

- **Image Sources** — Diverse input sources that provide raw images to the system.

PROCESSES

- **Data Ingestion** — Collects and validates images from sources.
- **Image Preprocessing** — Normalizes images for the model.
- **Image Captioning Model** — Generates captions for images.
- **Tagging & Indexing** — Extracts tags from captions and indexes metadata.
- **Search & Recommendation** — Uses metadata to power search queries and content suggestions.
- **Digital Asset Management** — Stores images and metadata for long-term management.

DATA STORES:

- **Image Storage** — Repository of raw and processed images.
- **Metadata Store** — Storage for captions, tags, and indexed data.

4.3 UML DIAGRAMS

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as

- Actors
- Business Processes
- (Logical) Components
- Activities
- Programming Language Statements
- Database Schemas, And
- Reusable Software Components.

4.3.1 USE CASE DIAGRAM

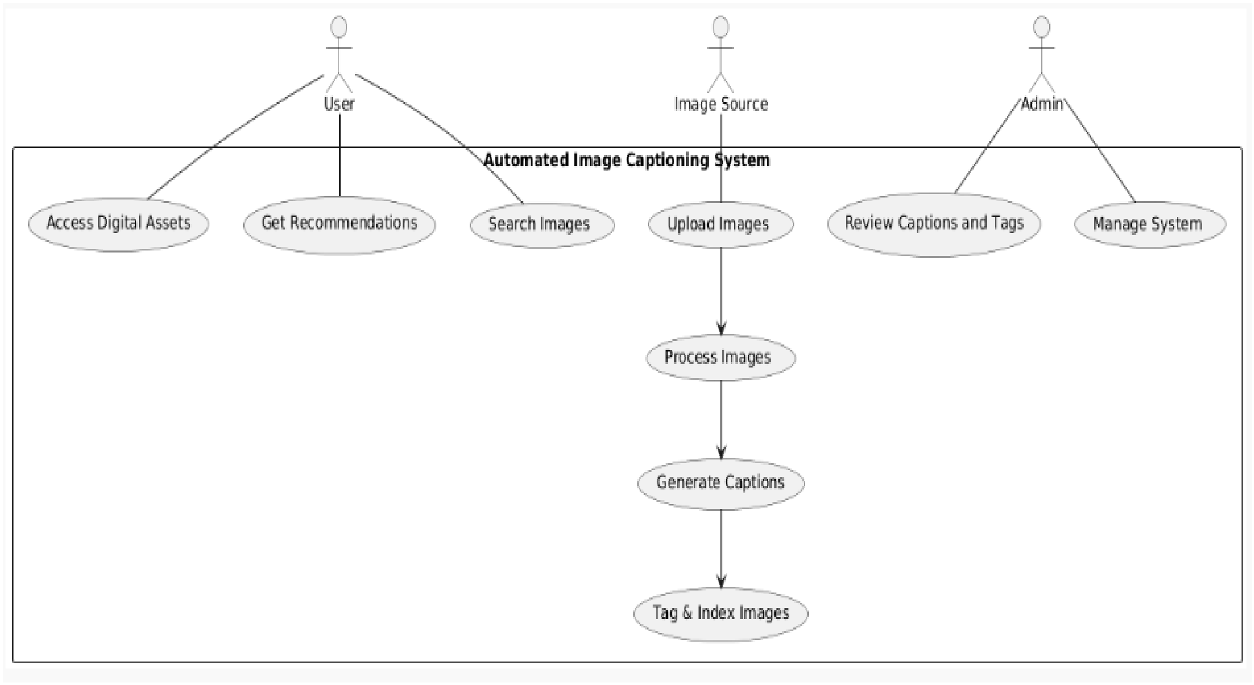


FIG 4.3.1 USE CASE DIAGRAM

- **Image Providers** (e.g., content creators, app integrations) upload images into the system.
- The system automatically:
  - **Generates captions** using deep learning models.
  - **Extracts relevant tags** from the captions.
  - **Indexes the image metadata** for future search and recommendations.
- **Users** can:
  - Search for images based on tags, descriptions, or content.
  - Receive personalized **content recommendations** based on image similarity or interests.
  - Access and download images as part of **digital asset management (DAM)**.
- **Admins** can:
  - Monitor system performance.
  - Manually review and edit captions/tags when automated results are not accurate.

4.3.2 SEQUENCE DIAGRAM

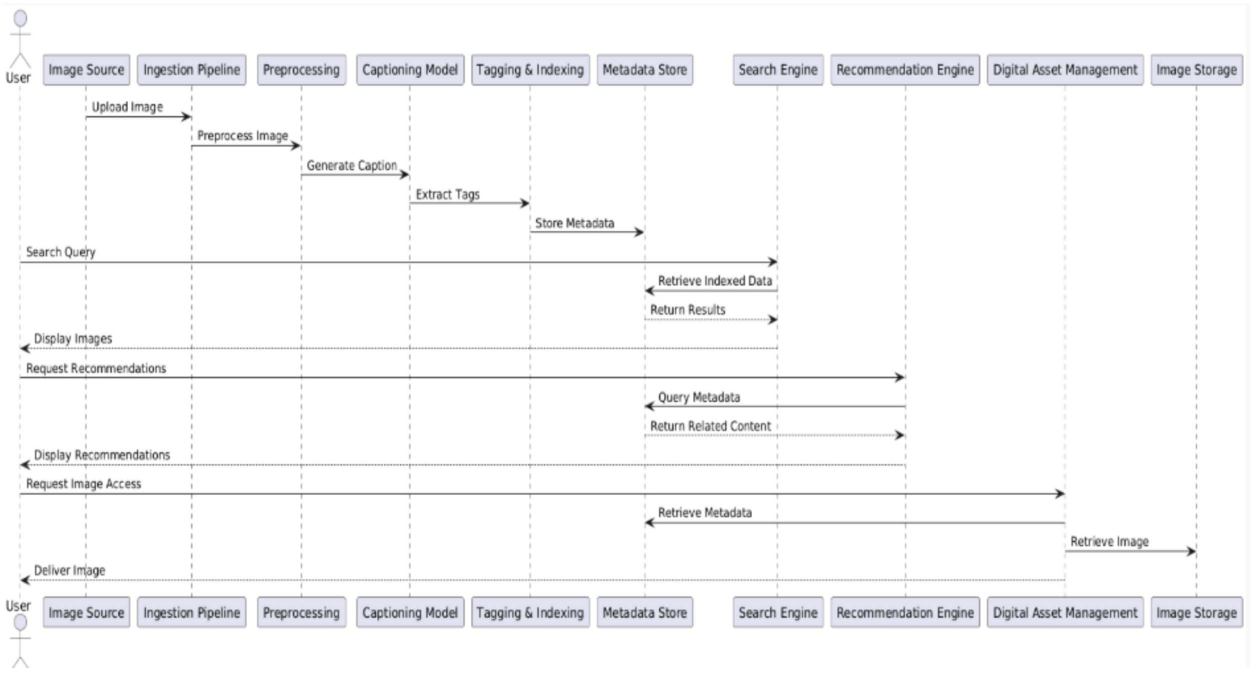


FIG 4.3.2 SEQUENCE DIAGRAM

This sequence diagram depicts the step-by-step flow of interactions between the user and the system components during the process of uploading visual media and obtaining captions with narration:

When an **image is uploaded**:

1. It's passed to a **preprocessing module** (resizing, normalizing).
2. A **captioning model** (e.g., a CNN + Transformer) analyzes the image and generates a descriptive caption.
3. The caption is passed to a **tagging engine** that extracts key concepts (tags).
4. All metadata (caption, tags, image info) is stored in a **metadata store** (e.g., Elasticsearch, PostgreSQL).

When a **user performs a search**:

1. Their query is matched against indexed tags and captions.
2. Matching images are retrieved and presented.

When a **recommendation is requested**:

1. The system uses the user's past interactions and image similarity to suggest relevant assets.

When a **digital asset is accessed**:

1. The system fetches both the image and its metadata for viewing or download.

4.3.3 CLASS DIAGRAM

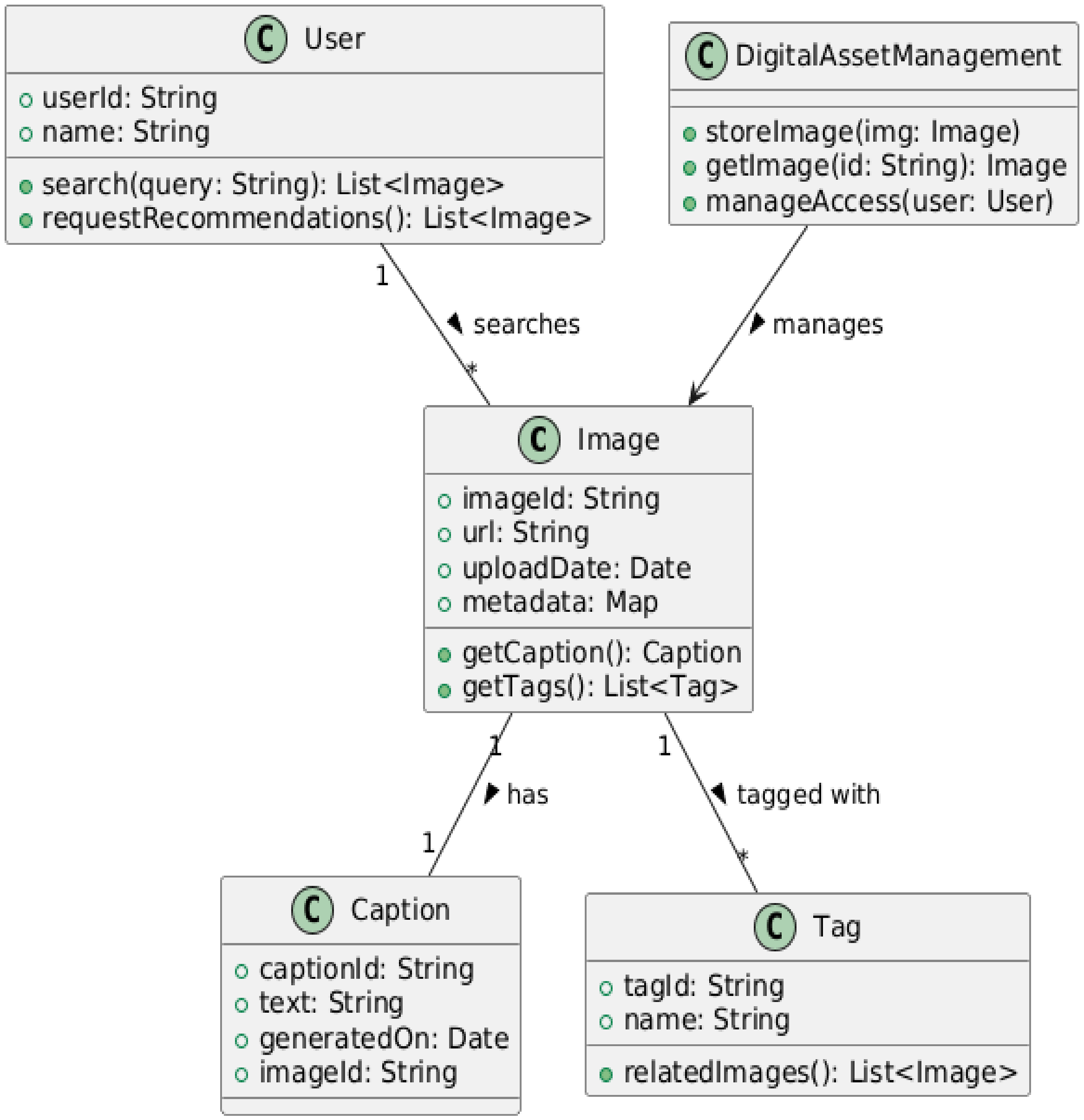
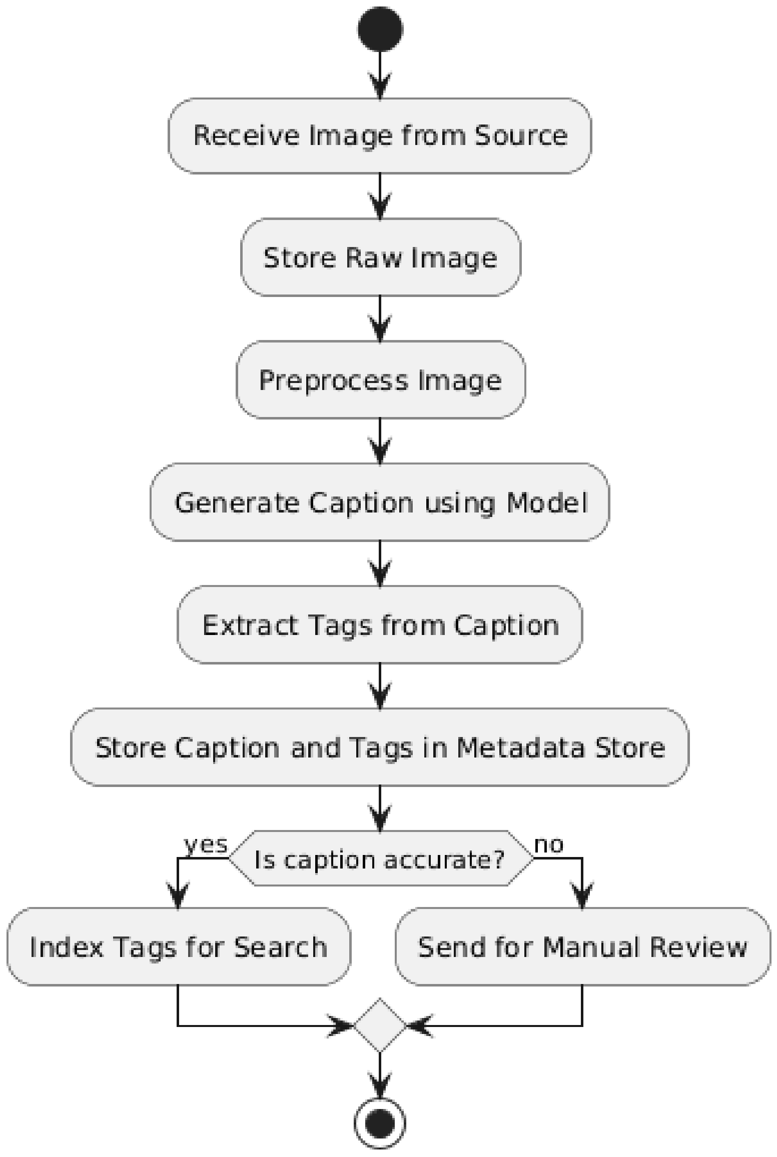


FIG 4.3.3 CLASS DIAGRAM

4.3.4 ACTIVITY DIAGRAM



Receive Image from Source
Store Raw Image
Preprocess Image
Generate Caption using Model
Extract Tags from Caption
Store Caption and Tags in Metadata Store
Is caption accurate?
yes
no
Index Tags for Search
Send for Manual Review

FIG 4.3.4 ACTIVITY DIAGRAM

CHAPTER 5
SYSTEM IMPLEMENTATION

5.1 MODULES

IMAGE CAPTIONING ARCHITECTURE: CNN + LSTM

The image captioning system is typically built using a **two-stage deep learning architecture**:

- CNN (Convolutional Neural Network) – for image feature extraction**.**
- LSTM (Long Short-Term Memory) – for sequential text generation**.**

WORKING PRINCIPLE

Image Feature Extraction (Encoder - CNN)

- ➢ A pre-trained CNN model (e.g., **ResNet**, **Inception**, or **VGG**) is used to extract high-level visual features from input images.
- ➢ The CNN acts as a fixed feature extractor. Its final classification layer is removed.
- ➢ The output is a high-dimensional **feature vector** representing the image's content.

Sequence Modeling (Decoder - LSTM)

- ➢ The extracted image features are passed to an LSTM-based RNN.
- ➢ The LSTM model takes these features and generates a **sequence of words** that form a human-readable caption.
- ➢ Each word is generated **one at a time**, conditioned on the previous words and the image features.

CAPTION GENERATION WORKFLOW

1. **Image Preprocessing**:

    - ➢ Resize and normalize the input image.
    - ➢ Feed it into the CNN to extract the feature vector.

2. **Text Preprocessing**:

    - ➢ Clean and tokenize captions.
    - ➢ Build a vocabulary of the most common words.
    - ➢ Add special tokens like <start> and <end> to each caption.

3. **Training**:

    - ➢ The model is trained on (image, partial caption) pairs to predict the next word in the caption.
    - ➢ Uses **teacher forcing** to guide the LSTM during training.

4. **Inference**:

    - ➢ At test time, only the image and a <start> token are input.
    - ➢ The model generates one word at a time until the <end> token is produced or the max length is reached.

5. **Tagging and Indexing**

    - ➢ **Natural Language Processing (NLP)** techniques (e.g., POS tagging, named entity recognition) are used to extract **key tags or keywords** from the caption.
    - ➢ These tags are used to **index** the image in a metadata store (e.g., Elasticsearch or a database).
    - ➢ This enables **fast retrieval**, **semantic search**, and **tag-based browsing**.

6. **Searchability and Content Recommendation**

    - ➢ Users can search using keywords or natural language.
    - ➢ The system retrieves images by matching the query with captions and tags.

Content recommendation is enabled using:

- **Caption/text similarity** (e.g., cosine similarity of embeddings).
- **Visual similarity** (image feature vectors).
- **User interaction history**.

7. **Digital Asset Management (DAM)**

The final system is integrated into a **Digital Asset Management platform** which includes:

- Organized storage of images and metadata.
- Easy retrieval via intelligent search.
- Caption and tag editing (manual override by admins).
- Integration with other enterprise tools (e.g., CMS, analytics).

5.2 SOURCE CODE

Main.py

```
import os
import pickle
import numpy as np
from flask import Flask, request, render_template
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.applications.inception_v3 import InceptionV3, preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array
app = Flask(__name__)
UPLOAD_FOLDER = os.path.join('static', 'uploads')
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
# Load trained model and tokenizer
model = load_model("saved_model/model_saved.keras")
with open("saved_model/tokenizer.pkl", "rb") as f:
    tokenizer = pickle.load(f)
with open("saved_model/config.pkl", "rb") as f:
    config = pickle.load(f)
max_length = config['max_caption_length']
# Load CNN model for feature extraction
cnn_model = InceptionV3(weights='imagenet', include_top=False, pooling='avg')
def extract_features(image_path):
    image = load_img(image_path, target_size=(299, 299))
    image = img_to_array(image)
    image = np.expand_dims(image, axis=0)
    image = preprocess_input(image)
    return cnn_model.predict(image)
def generate_caption(image_path):
    features = extract_features(image_path)
    in_text = ''
    generated_words = set()
    for _ in range(max_length):
        sequence = tokenizer.texts_to_sequences([in_text])[0] if in_text else []
        sequence = pad_sequences([sequence], maxlen=max_length)
        yhat = model.predict([features, sequence], verbose=0)
        yhat = np.argmax(yhat)
        word = tokenizer.index_word.get(yhat)
        if word is None:
            break
        if word in generated_words:
            break
        if word.lower() == 'end':
            in_text += '.'
            break
        generated_words.add(word)
        in_text += ' ' + word
        if word in ['.', '!', '?']:
            break
    return in_text.strip().capitalize()
@app.route('/', methods=['GET', 'POST'])
def index():
    caption = None
    image_path = None
    if request.method == 'POST':
        file = request.files.get('image')
        if file:
            os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
            image_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
            file.save(image_path)
            caption = generate_caption(image_path)
    return render_template('index.html', caption=caption, image_path=image_path)
if __name__ == '__main__':
    app.run(debug=True)
```

Index.Html

```
<!-- templates/index.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Image Captioning</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: #f4f4f4;
            display: flex;
            flex-direction: column;
            align-items: center;
            padding: 40px;
        }
        h1 {
            color: #333;
        }
        .caption {
            font-size: 20px;
            color: #444;
            margin: 20px 0;
        }
        .image-preview {
            margin: 20px 0;
        }
        img {
            max-width: 500px;
            border: 1px solid #ccc;
            box-shadow: 2px 2px 10px rgba(0,0,0,0.1);
        }
        form {
            margin-top: 20px;
        }
        input[type="file"] {
            margin-bottom: 10px;
        }
        input[type="submit"] {
            padding: 8px 16px;
            font-size: 16px;
```

```
            cursor: pointer;
        }
    </style>
</head>
<body>

    <h1>Generate Image Caption</h1>

    {% if caption %}
        <div class="caption"><strong>Caption:</strong> {{ caption }}</div>
    {% endif %}

    {% if image_path %}
        <div class="image-preview">
            <img src="{{ image_path }}" alt="Uploaded Image">
        </div>
    {% endif %}

    <form method="post" enctype="multipart/form-data">
        <input type="file" name="image" accept="image/*" required><br>
        <input type="submit" value="Generate Caption">
    </form>

</body>
</html>
```

CHAPTER 6

SYSTEM TESTING

6.1. 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

6.2. TYPES OF TESTS

6.2.1. 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.

6.2.2. 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.

6.2.3. FUNCTIONAL TEST

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

6.3. TEST CASES

6.3.1. UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

6.3.2. INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.3.3. Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 7

RESULT

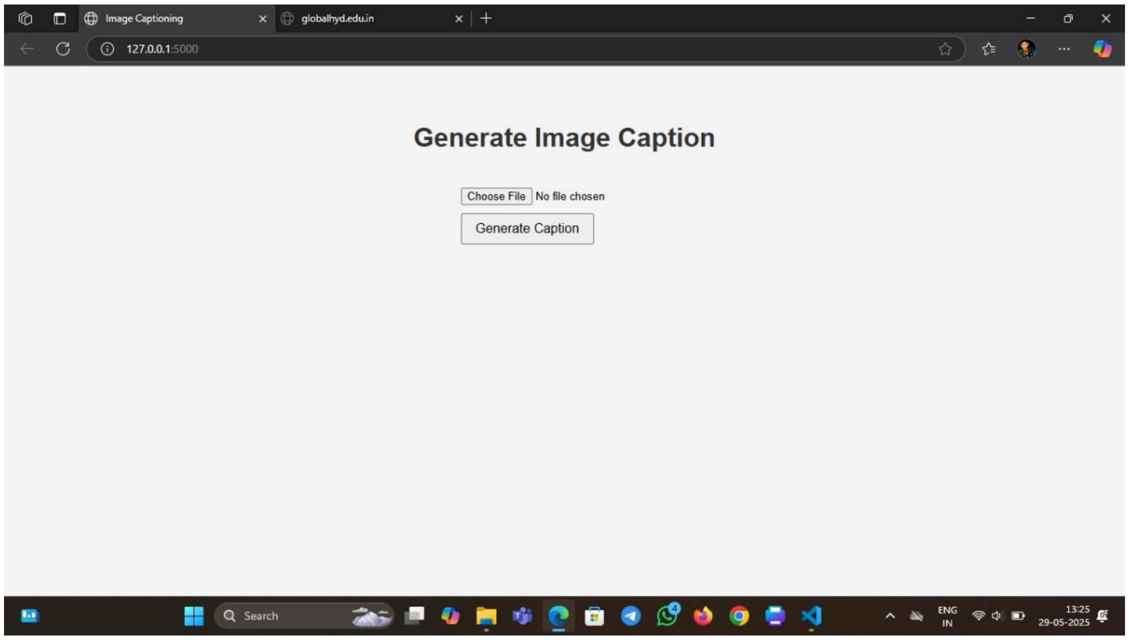


FIG 7.1 WEB PAGE

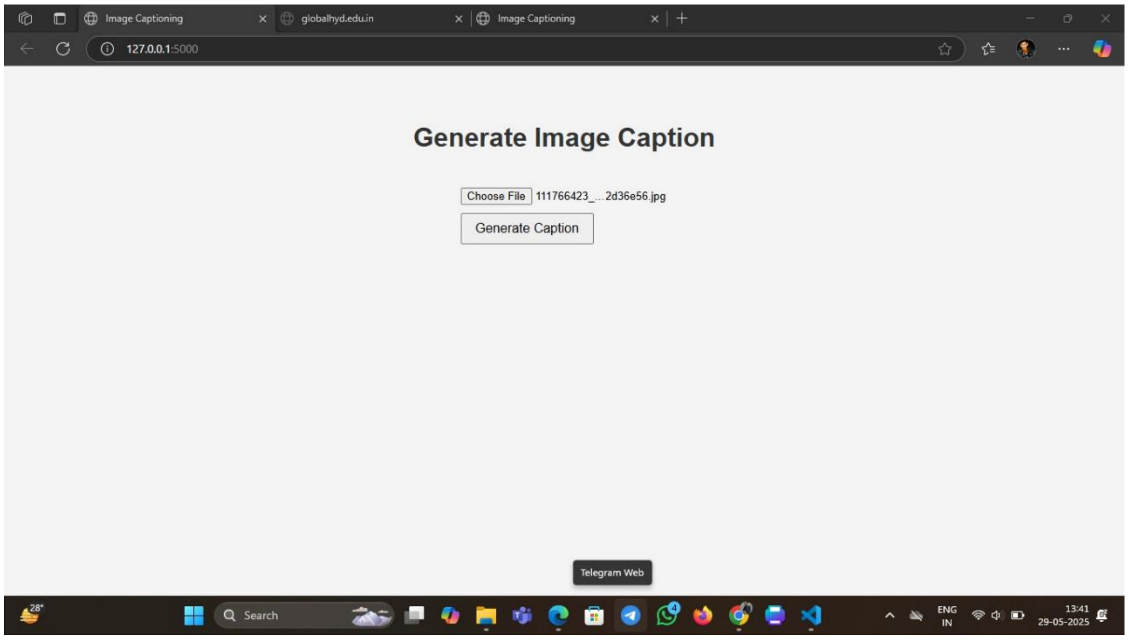
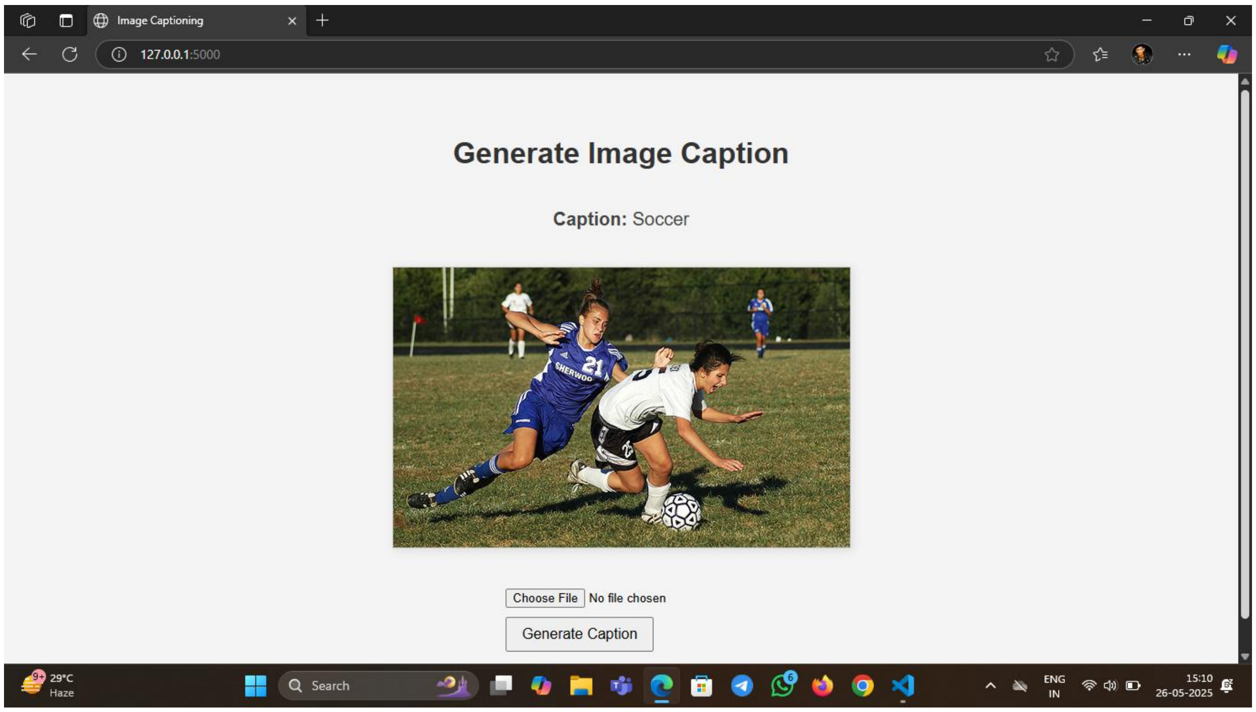


FIG 7.2 INPUT



Caption: Soccer

FIG 7.3 OUTPUT

CHAPTER 8

CONCLUSION

The development of an **automated image captioning system** for diverse image datasets represents a significant advancement in intelligent multimedia understanding. By leveraging deep learning models such as **CNN-LSTM**, the system can generate meaningful and context-aware captions for images. These captions not only describe visual content in human-readable language but also enable the **automatic extraction of tags**, which serve as metadata for efficient **indexing and retrieval**.

This approach enhances the **searchability** of visual assets, facilitates **personalized content recommendations**, and strengthens **digital asset management** on large-scale platforms. The system's ability to process varied and diverse image data makes it highly scalable and adaptable to real-world applications across domains such as e-commerce, media libraries, social platforms, and enterprise content systems.

Ultimately, this solution bridges the gap between visual data and textual understanding, transforming unstructured image content into structured, searchable, and actionable information—thereby delivering smarter user experiences and more efficient data management.

FUTURE SCOPE

The future of automated image captioning systems holds immense potential, particularly as deep learning and artificial intelligence technologies continue to evolve. One significant area of growth lies in the integration of **multimodal learning**, where image captioning can be combined with audio, video, and contextual metadata to produce more descriptive and context-aware outputs. This will be particularly beneficial in domains like surveillance, healthcare, and digital journalism.

Additionally, **domain-specific captioning models** trained on specialized datasets can vastly improve the relevance and accuracy of captions in fields like medical diagnostics, autonomous driving, and e-commerce. The incorporation of **multilingual capabilities** will make these systems more inclusive, allowing captions to be generated in multiple languages, improving accessibility and global usability.

Emerging techniques such as **self-supervised learning** and **transformer-based models** (e.g., BLIP, GIT, and Flamingo) promise to reduce dependency on large labeled datasets while improving generalization across diverse image types. Real-time image captioning for **assistive technologies** and **augmented reality** is also a promising frontier.

Furthermore, with the integration of **semantic tagging** and **knowledge graphs**, future systems will not only describe images but also reason about their content—enhancing **searchability, recommendation systems**, and **automated digital asset management** on an unprecedented scale.

REFERENCES

1. **Vinyals, O., Toshev, A., Bengio, S., & Erhan, D.** (2015).
   Show and Tell: A Neural Image Caption Generator.
   In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
   https://doi.org/10.1109/CVPR.2015.16

   Introduced the CNN-LSTM architecture for end-to-end image captioning.

2. **Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., ... & Bengio, Y.** (2015).
   Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.
   In ICML 2015.
   https://arxiv.org/abs/1502.03044

   Introduced attention mechanisms in image captioning.

3. **Hossain, M. Z., Sohel, F., Shiratuddin, M. F., & Laga, H.** (2019).
   A Comprehensive Survey of Deep Learning for Image Captioning.
   ACM Computing Surveys (CSUR), 51(6), 118.
   https://doi.org/10.1145/3295748

   An extensive survey on architectures, datasets, and challenges in image captioning.

4. **Chen, X., & Lawrence Zitnick, C.** (2015).
   Mind's Eye: A Recurrent Visual Representation for Image Caption Generation.
   In CVPR 2015.
   https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Chen_Mind's_Eye_A_2015_CVPR_paper.html

   Presents techniques on visual-semantic embedding models for caption generation.

5. **Microsoft COCO (Common Objects in Context)**
   https://cocodataset.org/

   Widely used dataset for training and evaluating image captioning systems.

6. **Flickr8k and Flickr30k Datasets**
   https://forms.illinois.edu/sec/1713398

   Contains images and multiple human-written captions; useful for training models on diverse data.

7. **TensorFlow Image Captioning Tutorial**
https://www.tensorflow.org/tutorials/text/image_captioning

   Official tutorial using CNN and RNNs (LSTM) for generating captions.

8. **PyTorch Image Captioning Implementation (GitHub)**
https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning

   A detailed PyTorch-based tutorial for implementing an image captioning pipeline.

9. **Google Cloud Vision API**
https://cloud.google.com/vision

   Offers automated tagging and object detection — used in enterprise DAM and content management.

10. **Amazon Rekognition**
https://aws.amazon.com/rekognition/

    Provides image and video analysis, including label detection and moderation.