



SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

“Mood2Caption - An AI-Powered Instagram Caption Generator using LLM and ML”

**AGILE SOFTWARE
ENGINEERING & DEVOPS**
(CS2004)
2024-2025



Submitted by

Misha N Devegowda	1RVU23CSE268
Nikhita K Nagavar	1RVU23CSE309
Spandana Sujay	1RVU23CSE464

Under the guidance of

Prof. C V Satyanarayana Reddy
Designation

School of Computer Science and Engineering
RV University-560059

RV UNIVERSITY, BENGALURU-59

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the project work titled **“Mood2Caption - An AI-Powered Instagram Caption Generator using LLM and ML”** is carried out by **Misha N Devegowda (1RVU3CSE268), Nikhita K Nagavar (1RVU23CSE309), Spandana Sujay (1RVU23CSE464)** in partial fulfillment of the completion of the course **Agile Software Engineering And DevOps with course code CS2004** of the IV Semester Computer Science Engineering programme, during the academic year 2024-2025. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the project report and duly approved by the faculty.

Signature of Faculty

Signature of Head of the Department

ABSTRACT

With social media platforms such as Instagram being the centre of digital articulation, their users themselves will struggle to articulate new and situation-specific captions. This project is proposing a Mood-Based Instagram Caption Generator that bridges the gap through Large Language Models (LLMs) and Machine Learning (ML) methods of creating and generating captions based on the user's mood or feeling. The system works to enhance social media engagement and individual expression through the provision of intelligent, mood-aware caption suggestions.

The project brings together two central modules: the LLM-based generation module and the ML-based recommendation module. The LLM module uses pre-trained Hugging Face transformer models to generate contextually appropriate and grammatically rich captions from user-input text or image descriptions. On the other hand, the ML module utilizes a mood-tagged structured caption dataset (e.g., inspirational, romantic, sarcastic, sad, happy) and carries out TF-IDF vectorization as well as cosine similarity to map user mood and caption input against the most relevant available captions.

The application is coded using an interactive Streamlit frontend with which users are able to upload an image, caption it, select their mood, and retrieve a list of generated or recommended captions. Preprocessing is done in some of the most critical steps that include tokenization, text cleaning, vector transformation, and mood classification to provide good input to the models. The model is coded using Python with wide usage of Pandas, NumPy, and NLTK in dealing and processing the data. The generator produces meaningful and expressive captions even if there's no best fit, exhibiting the strength of LLMs when generating natural language. In addition to that, the architecture of the model allows real-time capability and produces on-time responses appropriate for social media use speeds.

In terms of the use of ML and DL methods, Mood-Based Instagram Caption Generator offers a state-of-the-art approach to boosting user creativity and personalization. The system offers avenues for future work in the use of image-based mood detection, multilingual captioning, and reinforcement learning towards fostering contextual accuracy and variability in captions. In general, the project is a step in the right direction towards adopting smart algorithms in ubiquitous digital engagement.

Chapter 1: Introduction to the Project

With the advent of social media applications such as Instagram, users usually get stuck in trying to create a creative and suitable caption to match their mood or post. The wide range of moods, moments, and personal sentiments make it tough to come up with captions that appeal to the user's sentiments. The Mood-Based Instagram Caption Generator is built as a remedy to this problem by offering a savvy, user-friendly interface that provides mood-correlated caption suggestions. The platform enables users to upload or outline a picture, select a mood, and get suggested captions based on machine learning (ML) methods and large language models (LLMs). The platform reduces complexity with an easy and interactive Streamlit-based GUI that optimizes user engagement. The user input — textual description or mood choice — is computed against an existing dataset of captions and LLMs to produce or suggest descriptive captions based on the input.

The project comes under natural language processing and personalized content generation systems, which are essential to enhance digital communication in marketing and social contexts. An imaginative and mood-appropriate caption raises the post's visibility and engagement, whereas bad or cliché captions could lead to disengagement or lost interaction potential. Typing and mapping out of moods to captions tend to miss the delicacy and richness that is needed for expression. This framework fills the gap between the retrieval of context- and emotion-relevant pre-written captions by ML and the creative ability of LLM to produce new ones based on context and feeling. This two-method system provides an active and flexible solution for personalized captioning generation.

Goals:

- Implement an ML-based recommendation module with TF-IDF and cosine similarity to recommend captions that align with the user's chosen mood.
- Incorporate an LLM-based generation model to generate original captions based on mood and user input (text or image description).
- Implement a responsive Streamlit GUI to offer an interactive and engaging experience for users.
- Create an application that scales and can grow with bigger data sets or future image-based mood analysis.

Scope:

The application is a web-based, stand-alone interface that takes user mood or image/text descriptions as input and gives relevant caption suggestions. The application is designed for social media users, content creators, and marketers who are looking for creative captions for their posts, and can be further used to support other platforms apart from Instagram or even for other types of content like blogs, videos, or short-form media.

Chapter 2: Introduction to Agile and Execution Plan

Mood-Based Instagram Caption Generator was designed with Agile methods, which supported flexible, incremental development and strong collaboration throughout the project lifecycle. Agile was appropriately applied to this project because testing and tweaking repeatedly were required for both rule-based machine learning (ML) rules and big language models (LLMs), with a user-friendly Streamlit GUI. The system was developed incrementally, enabling steady feedback and optimizations rather than building towards a solitary complete version in the beginning. The project consisted of two distinct sprints with each leading towards a working copy of the system, motivated by pre-defined lists of moods and captions in place of an external dataset.

Agile Execution for the Caption Generator System

Agile practices like a product backlog, sprint planning, daily standups, sprint reviews, retrospectives, burndown charts, and velocity tracking were used to keep focus and momentum. The product backlog consisted of tasks such as building the mood-caption array, applying ML logic, GUI component design, LLM-based generation integration, and testing. Each task was allocated story points based on complexity — with the LLM integration getting the most because it was the most complex, and simple GUI features getting fewer.

Sprint 1: Rule-Based ML Logic and GUI (2 Weeks)

- Objective: Create a functional mood-based caption suggestion system with a basic ML logic over a fixed array, combined with a GUI.
- Tasks:
 1. Create an array of mood types (e.g., "happy", "sad", "romantic") and associate relevant Instagram-type captions for each mood.
 2. Create basic ML logic to fetch captions based on chosen mood using string matching or basic conditional logic.
 3. Implement a user interface in Streamlit that offers a dropdown menu for selecting mood and a place for caption display.
 4. Implement simple input validation so that a mood is chosen before displaying captions.
 5. Unit testing to ensure caption suggestions and GUI behavior.
- Backlog Items:
 1. Setup of mood-caption array (2 story points)
 2. ML retrieval logic (3 story points)
 3. GUI implementation (4 story points)
 4. Testing and verification (2 story points)

- Outcome:

By the end of Sprint 1, the team produced a working version where users would be able to choose a mood and immediately get a list of appropriate Instagram captions. In the sprint review, stakeholders liked the interface but recommended improvements like more customized caption output and fallback alternatives for unsupported moods.

Sprint 2: Dynamic Caption Generation Based on LLM (3 Weeks)

- Objective: Enhance the system with a transformer-based LLM to dynamically generate captions and improve the user interface with feedback from Sprint 1.

- Tasks:

1. Implement an open-source transformer model (e.g., GPT-2 or GPT-3.5 through API or Hugging Face) to dynamically generate captions according to the chosen mood.
2. Modify the GUI to enable users to switch between static (predefined) and dynamic (LLM-generated) caption modes.
3. Add logic to support unsupported or novel moods using the LLM or default caption suggestions.
4. Improve GUI responsiveness to minimize latency when generating the model.
5. Perform integration testing between the ML and LLM modules and gather usability feedback from colleagues.

- Backlog Items:

1. LLM integration and prompt development (6 story points)
2. GUI refresh with toggle functionality (3 story points)
3. Error checking and fallback logic (3 story points)
4. Final test and feedback (3 story points)

- Outcome:

Sprint 2 led to a refined system that presented both pre-determined captions and creatively generated ones based on the LLM. Users were able to choose a mood and pick between viewing a list of personally selected captions or receiving original AI-generated captions. Testers' feedback assured that the dual approach enhanced user experience, and the GUI was easy to use and responsive.

Agile Practices

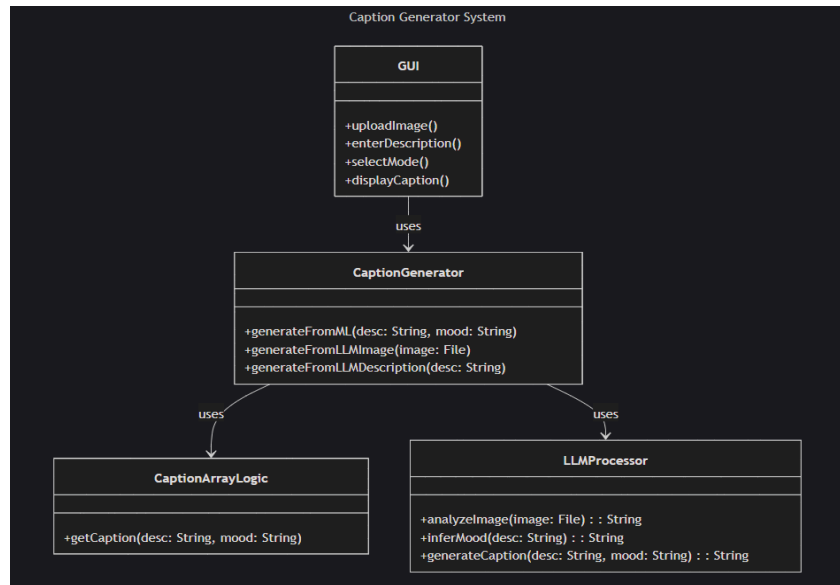
- Daily Standups: 15-minute stand-ups facilitated monitoring of work done such as "Finished ML array mapping," identifying blockers such as "LLM latency issues," and determining what to do next such as "Refactor mood selection handling."

- **Sprint Reviews:** Sprints concluded with live demonstrations of the caption generator using both ML and LLM outputs. Feedback resulted in changes such as including default responses and altering UI layout for readability.
- **Sprint Retrospectives:** Team retrospections noted strengths (e.g., "Early adoption of ML logic") and areas of improvement (e.g., "Get more LLM test time"). The team committed to beginning earlier on model integration in subsequent iterations.
- **Burndown Charts:** Visually tracking showed a consistent completion of work, with slight slippages during LLM testing in Sprint 2. The time estimates were adjusted for future planning purposes.
- **Velocity:** The average sprint speed of the team was 12–14 story points. This measurement was used to determine realistic targets and ensure sustained progress without overwhelming team members.

Chapter 3: UML Diagrams

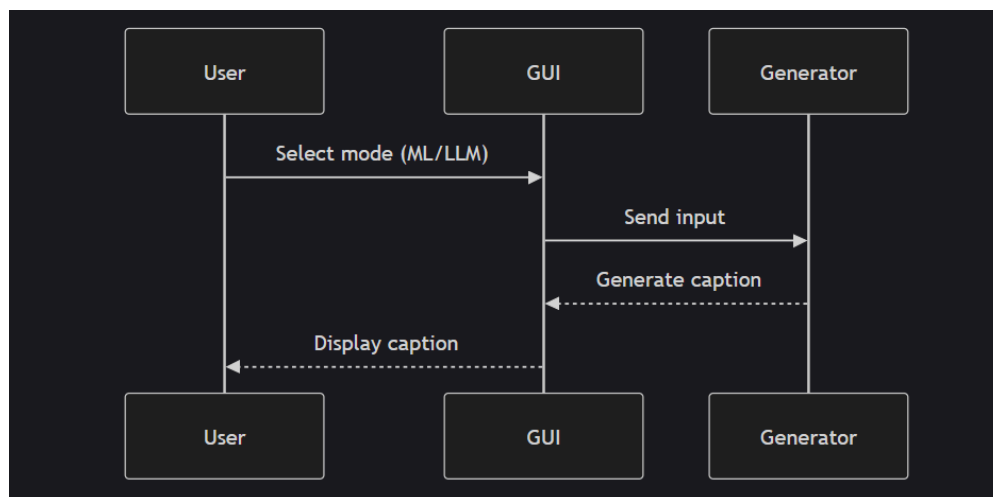
Class Diagram:

A class diagram is a type of static structure diagram in UML (Unified Modeling Language) that shows the classes in a system, their attributes, methods, and the relationships between them. It helps in visualizing the design and structure of object-oriented software.



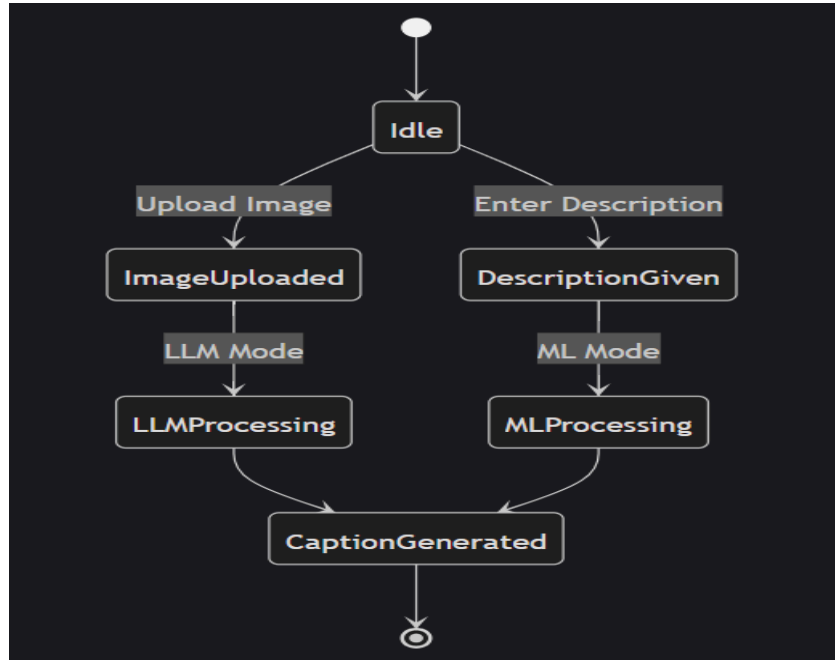
Sequence Diagram:

A sequence diagram is a UML diagram that shows the step-by-step interaction between objects or components over time to complete a specific process or function.



State Diagram:

A state diagram (also known as a state machine diagram) is a behavioral UML diagram that represents the different states an object can be in during its lifetime, along with the transitions between those states triggered by events. It is used to model the dynamic behavior of a system or an object in response to external stimuli.



Chapter 4: Introduction to ML, LLM, DL, QML, QNN

4.1 Machine Learning (ML)

Machine Learning (ML) is instrumental in the Instagram Caption Assistant in facilitating the system to suggest mood-congruent captions from user input. Contrary to hardcoded replies or filtering, the recommendation based on ML learns patterns from current caption data and uses similarity matching to offer corresponding alternatives.

The fundamental approach utilized relies on TF-IDF vectorization and cosine similarity. Initially, all available captions are turned into numerical vectors that capture the significance of words in the whole corpus.

Once a user inputs a caption and chooses a mood (e.g., "happy", "bold", "dreamy"), the system trims the dataset based on mood and calculates the cosine similarity with the input and available mood-matched captions.

This model-free method does not need training with labeled data sets, and so it is extremely efficient and good for real-time use. It performs especially well in a cold-start scenario in which past user interactions are not available.

With the use of textual similarity, the system recognizes thematic or stylistic intersections so that it can offer rapid and pertinent caption recommendations.

This ML module deepens user creativity by bringing to the surface alternative wording or phrasing that is in sync with a specified mood, thus personalizing the captioning experience without needing sophisticated infrastructure or massive training.

4.2 Large Language Model (LLM)

For this project, an LLM is utilized to create Instagram captions from input from the user. The LLM is accessed through the Hugging Face API, namely the Mixtral-8x7B-Instruct-v0.1 model, which is specifically fine-tuned for text generation tasks.

How It Works:

- **API Integration:** The LLM is invoked using a secure API via a Bearer Token.
- **Text Generation:** Users enter a description or upload an image. The app creates captions depending on the description and chosen mood (e.g., "chill," "motivational").

- User Interaction: Captions generated are shown within the Streamlit app so that users can view mood-based suggestions interactively.

Benefits:

- Personalized Captions: LLM creates innovative captions depending on the user's input and mood.
- Context-Aware: Captions are adapted to the given description and mood.
- Interactive: Captions are easily created by users using the app.

4.3 Deep Learning (DL)

Deep Learning (DL) is a branch of Machine Learning (ML) that uses artificial neural networks (ANNs) with many layers—commonly called deep neural networks—to represent complex and abstract patterns in data. These architectures are based on the structure and operation of the human brain, where each layer of neurons learns progressively more advanced features from raw input data.

In contrast to conventional ML models, which are frequently based on hand-crafted features, deep learning extracts features automatically using layer transformations. Deep neural networks use each layer to extract higher-level representations: e.g., in image processing, the early layers may detect edges, and the deeper layers can identify objects or textures.

- DL works particularly well with unstructured data like:
- Images (using Convolutional Neural Networks - CNNs),
- Text and natural language (through Recurrent Neural Networks - RNNs or Transformers),
- Audio and speech signals (through CNNs or spectrogram-based models).

DL has driven advances in computer vision, natural language processing, speech recognition, and others. Libraries such as TensorFlow, PyTorch, and Keras facilitate fast development and training of deep learning models, even on large data sets and complicated architectures.

4.4 Quantum Machine Learning (QML)

Quantum Machine Learning (QML) is an emerging field that integrates quantum computing principles with classical machine learning techniques to potentially improve the efficiency and scalability of data-driven models. QML explores how quantum mechanics—through properties like superposition, entanglement, and quantum interference—can accelerate certain computational tasks that are bottlenecks in classical ML.

Quantum algorithms are applied to represent and work with data in quantum bits or qubits, which are in more than one state at the same time. This makes the computation exponentially quicker in particular computations like:

- Calculating distance in clustering,
- Estimating the kernel for support vector machines,
- Probabilistic models' sampling,
- Dimension reduction of high-dimensional data.

Although most of the applied QML solutions are still in the experimental or research state—dependent on quantum hardware like IBM Q, Google Sycamore, or D-Wave systems—hybrid methods that integrate classical and quantum parts are currently being explored.

QML offers improved performance for finance, chemistry, logistics, and recommendation problems when data size or feature dimensionality hinders classical approaches.

4.5 Quantum Neural Networks (QNN)

Quantum Neural Networks (QNNs) are experimental and theoretical paradigms that are designed to take the learning capability of traditional neural networks and incorporate the computational efficiency of quantum computing. Quantum circuits in QNNs are applied to model neuron behavior, using qubits and quantum gates to substitute nodes and weights.

QNNs take advantage of quantum phenomena to execute parallel computation over superposition states that are complex. This allows them to search solution spaces that are large more efficiently, and thus possibly solve problems that are intractable classically for neural networks.

The main features of QNNs are:

- Entanglement: Enables qubits to exchange information instantly, boosting pattern recognition.
- State interference: Can be utilized to strengthen or eliminate specific outcomes, steering learning.
- Parameterization: Quantum gates are tuned during training (typically employing classical optimizers) to optimize loss functions, much like backpropagation.

Although practical usage is constrained by current quantum hardware limitations (e.g., decoherence, noise, small number of qubits), QNN research is expanding. Google, IBM, and Xanadu are among companies that are researching QNNs for applications including molecular modeling, cryptography, and complex recommendation systems.

In the long term, QNNs might be superior to regular deep learning through providing faster convergence, better generalization, and scalability to very large datasets.

Software Requirements Specification

For

Mood2Caption - An AI-Powered Instagram Caption Generator using LLM and ML

Prepared by

Misha N Devegowda	1RVU23CSE268	mishand.btech23@rvu.edu.in
Nikhita K Nagavar	1RVU23CSE309	nikhitak.btech23@rvu.edu.in
Spandana Sujay	1RVU23CSE464	spandanas.btech23@rvu.edu.in

Instructor: Prof. CVS N Reddy

Course: Agile Software Engineering and DevOps

Lab Section: B

Teaching Assistant: Prof. Harish KR

Contents

1	INTRODUCTION	1
1.1	DOCUMENT PURPOSE.....	1
1.2	PRODUCT SCOPE	1
1.3	INTENDED AUDIENCE AND DOCUMENT OVERVIEW	1
1.4	DEFINITIONS, ACRONYMS AND ABBREVIATIONS	1
1.5	DOCUMENT CONVENTIONS	1
1.6	REFERENCES AND ACKNOWLEDGMENTS.....	2
2	OVERALL DESCRIPTION	2
2.1	PRODUCT OVERVIEW	2
2.2	PRODUCT FUNCTIONALITY	3
2.3	DESIGN AND IMPLEMENTATION CONSTRAINTS	3
2.4	ASSUMPTIONS AND DEPENDENCIES.....	3
3	SPECIFIC REQUIREMENTS	4
3.1	EXTERNAL INTERFACE REQUIREMENTS	4
3.2	FUNCTIONAL REQUIREMENTS	4
3.3	USE CASE MODEL	5
4	OTHER NON-FUNCTIONAL REQUIREMENTS	6
4.1	PERFORMANCE REQUIREMENTS	6
4.2	SAFETY AND SECURITY REQUIREMENTS.....	6
4.3	SOFTWARE QUALITY ATTRIBUTES	6

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Misha N Devegowda, Nikhita K Nagavar, Spandana Sujay	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	14/05/2025

1 Introduction

The Caption Generation System has the purpose of generating customized captions according to user input through either an upload of an image or text description with mood. The system utilizes both machine learning (ML) and large language models (LLM) to create smart, context-based captions for users.

Layer 1: Web Client (Frontend)

The frontend is developed with Streamlit, providing a minimal, interactive interface for users to interact with the caption generation system.

- User Interaction:
 - a. Users can upload images for LLM-based caption generation.
 - b. Users can input a description and mood for ML-based caption generation.
- Features:
 - a. Image Upload: Users can upload an image, and the system will provide a caption based on the image using the LLM.
 - b. Text Input: Users can input a description and a mood (e.g., happy, sad, etc.), and the system will produce a caption based on ML-based models.
 - c. Real-time Feedback: Captions generated are shown in real time after submission.
 - d. Validation: Simple input validation checks for proper format of inputs, with error messages for incorrect inputs.

Layer 2: Web Server (Backend)

The backend receives user input, communicates with models, and returns the created captions.

- Handles User Requests:
 - a. Processes image upload or text input (with mood).
 - b. Preprocesses image data for LLM (if image upload is done).
 - c. Preprocesses text data for ML (if description and mood are given).
- Model Execution:
 - a. LLM Mode: Runs large language models (e.g., OpenAI's GPT or equivalent) to create captions from image data.
 - b. ML Mode: Employs a machine learning model to create captions from input text descriptions and moods.
- Processing:
 - a. Pre-processes model data (image processing for LLM, text encoding for ML).
 - b. Formats the input for model consumption and delivers processed captions back to the frontend.

Layer 3: Data & Processing (Backend Models)

This layer handles data processing, model training, and execution of all the data.

- **Data Management:**
 - a. **Dataset Input:** Contains image data for LLM processing as well as text data for ML-based captions.
 - b. Image files are preprocessed by pre-trained models to generate captions.
 - c. Text data (with mood) is processed by machine learning models trained on caption generation.
- **Model Training:**
 - a. **LLM:** Employs pre-trained large language models (e.g., GPT or equivalent) to train captions from images.
 - b. **ML:** Trained on a vast amount of text to produce context-sensitive captions from the given description and mood.
- **Scalability & Updates:**
 - a. The system is scalable and can be easily updated when new data is added.
 - b. The models get updated every now and then for more accuracy and relevance.

1.1 Document Purpose

The purpose of this document is to define the software requirements for the Caption Generation System. This system leverages Machine Learning (ML) and Large Language Models (LLM) to generate personalized captions based on user inputs. The system allows users to upload images or provide text descriptions with specific moods to receive context-aware captions. The goal is to provide users with an intelligent, personalized caption generation experience that enhances their interaction with visual and textual content.

1.2 Product Scope

The Caption Generation System aims to deliver an intuitive and intelligent application that generates meaningful captions based on user-provided image uploads or text descriptions. By integrating Machine Learning (ML) and Large Language Models (LLM), the system offers context-aware and mood-based caption suggestions. The focus is on simplicity, real-time interaction, and practical deployment for both text and image-based captioning tasks.

The scope of the product includes the following key capabilities:

➤ User-Centric Caption Interaction:

- Users can upload an image or provide a textual description along with the desired mood.

- The interface will display automatically generated captions based on input data.

➤ **AI-Powered Caption Generation:**

- A Machine Learning model is used to generate captions from text descriptions and mood tags.
- An LLM model is used for analyzing uploaded images and generating contextually appropriate captions.
- Results are dynamically displayed to the user for immediate feedback.

➤ **Architecture and Components:**

The Caption Generation System follows a three-tier architecture:

- **Layer 1: Presentation Layer (Web Client)**
 - Provides a web-based user interface built using Streamlit.
 - Allows users to upload images, input descriptions, and select moods to receive generated captions.
- **Layer 2: Application Layer (Web Server)**
 - Manages the logic for ML and LLM integration.
 - Processes inputs from the frontend and triggers appropriate models for caption generation.
- **Layer 3: Data & Processing Layer**
 - Handles any required pre-processing (e.g., text cleaning, image resizing).
 - Interfaces with trained ML/LLM models for inference and outputs formatted results.

1.3 Intended Audience and Document Overview

This Software Requirements Specification (SRS) document is intended to serve all stakeholders involved. This Software Requirements Specification (SRS) document outlines the design, functionality, and system architecture for the **Caption Generation System**, which leverages Machine Learning (ML) and Large Language Models (LLM) to generate context-aware image and text-based captions. The document serves as a detailed guide for all parties involved in the development and deployment of the system.

This document is intended for the following audiences:

- **Developers:** To understand how ML and LLM components integrate within the Streamlit frontend and backend logic, and to ensure smooth implementation and scalability.
- **Test Engineers:** To derive testing scenarios, create validation mechanisms, and ensure that the system performs accurately across multiple input types (images and text).

- **Project Managers:** To manage timelines, allocate resources effectively, and track progress against predefined milestones and deliverables.
- **Stakeholders and End-Users:** To assess whether the application meets user expectations for generating meaningful captions tailored to different contexts and moods.
- **Academic Evaluators:** To evaluate the technical complexity, real-world applicability, and AI integration in the context of modern caption generation systems.

1.4 Definitions, Acronyms and Abbreviations

S. No	Abbreviation	Expansion
1.	ML	Machine Learning
2.	LLM	Large Language Model
3.	GUI	Graphical User Interface
4.	API	Application Programming Interface
5.	NLP	Natural Language Processing

1.5 Document Conventions

This document uses the following conventions:

- **FR:** Functional Requirements
- **NFR:** Non-Functional Requirements
- **PR:** Performance Requirements
- Diagrams and tables are clearly labeled for easy understanding.

1.6 References and Acknowledgments

Agile Software Development Methodology

Scikit-learn Documentation – <https://scikit-learn.org>

Pandas Documentation – <https://pandas.pydata.org>

Streamlit GUI Library – <https://docs.streamlit.io>

We would like to express our gratitude to RV University and the faculty of the Agile Software Development course for their support and guidance throughout the development of this project. Special thanks to the open-source communities of Scikit-learn, Pandas, Flask, and Streamlit for providing reliable tools and documentation that enabled us to build our Caption Generation System efficiently.

2 Overall Description

2.1 Product Overview

The Caption Generation System is a web-based application designed to assist users in generating creative and contextually relevant Instagram captions using a combination of Hugging Face's large language model (LLM) and Machine Learning (ML) recommendation techniques. The system leverages deep learning models to analyze and generate captions either from user-uploaded images or text descriptions, matching the mood and style preferences set by the user. Additionally, it offers a recommendation feature to suggest similar captions based on mood and textual input.

The system operates with a user-friendly Streamlit interface, divided into two primary modules:

1. LLM-Based Caption Generator

- **Description:** This module utilizes the Hugging Face API to generate captions based on images or text descriptions provided by the user. The captions are generated by analyzing the context and mood described by the user, ensuring creativity and relevance to the input.
- **Key Features:**
 - Image Upload: Users can upload an image, and the system will attempt to infer a description from the photo, followed by generating a mood-appropriate caption.
 - Text Description Input: Users can describe the image in words, and the system will generate captions based on the given description.
 - Mood Selection: Users can choose the mood (e.g., "Humorous," "Romantic," "Motivational," etc.) to guide the caption generation.
 - Hugging Face Integration: The system communicates with the Hugging Face API to generate captions based on the chosen mood.

2. ML-Based Caption Recommender

- **Description:** This module recommends similar captions based on the user's input caption and selected mood. The system uses Machine Learning (ML) techniques such as TF-IDF Vectorization and Cosine Similarity to identify and suggest captions that align with the user's provided text.
- **Key Features:**
 - Mood Selection: Users can choose from various predefined moods such as "Chill," "Adventurous," "Motivational," etc.
 - Caption Input: Users can enter a caption, and the system will recommend similar captions that match the chosen mood.

- Top-N Recommendations: The user can select how many similar captions they wish to receive, with flexibility to choose between 1 to 5 recommendations.

System Workflow:

1. User Input: The user interacts with the Streamlit UI, uploading images or providing descriptions and selecting moods.
2. Caption Generation: Based on the inputs, the system either generates captions using Hugging Face's API or recommends similar captions from the predefined dataset using ML algorithms.
3. Real-Time Interaction: The system provides immediate feedback, allowing users to refine their captions or receive suggestions for enhancing their Instagram posts.

Key Functionalities of the System:

- Hugging Face Caption Generation: Generate creative Instagram captions based on user-uploaded images or descriptions, tailored to a chosen mood.
- ML-Based Caption Recommendation: Provide similar captions to a user's input caption, ensuring mood consistency through ML-based similarity measures.
- Mood-Based Customization: The system allows users to choose from various moods to ensure the generated captions fit their desired tone.
- Interactive UI: Developed with Streamlit, providing an intuitive and seamless experience for users to interact with the captioning system.

Performance Evaluation:

- Accuracy of Captioning: The generated captions are evaluated for relevance and creativity based on the mood and context.
- User Feedback Integration: The system collects user feedback on caption relevance, continuously improving recommendation accuracy.

User Authentication & Data Security:

- While the system focuses on generating captions and recommendations, it ensures that user input (such as captions) is processed securely.
- Data Privacy: Ensures the privacy of user-uploaded images and textual descriptions through secure data handling practices.

2.2 Product Functionality

The Instagram Caption Assistant is a web-based application designed to generate and recommend Instagram captions based on user inputs and moods. The system integrates both

large language models (LLMs) and machine learning (ML) techniques to provide creative, personalized, and relevant captions tailored to the user's preferences and the mood of the post.

Core Functionalities:

Web Interface for Caption Generation and Recommendations:

- **User Input:**
 - Allows users to upload a photo or provide a description of the image they wish to post.
 - Users can select a mood (e.g., humorous, romantic, motivational) that reflects the tone of their post.
- **Personalized Caption Generation:**
 - Based on the input, the system generates captions either using a Hugging Face-based model (LLM) or provides ML-based recommendations.
 - The LLM generates creative captions based on a text description or photo, tailored to the specified mood.
 - The ML system suggests captions based on mood and semantic similarity to previously popular or relevant captions.
- **Caption Suggestions and Comparison:**
 - After generating or recommending captions, the system presents a list of caption suggestions ranked by relevance and mood match.
 - Users can compare different caption options and choose the one that best fits their content.

Backend Functionality (Streamlit-based UI with Hugging Face API):

- **Hugging Face Integration:**
 - Uses a Hugging Face model to generate captions from a user-provided prompt or image description. It offers a text-to-text transformation to create captions.
 - Handles user requests and communicates with the Hugging Face API to generate captions based on the mood selected by the user.
- **ML-Based Caption Recommendation:**
 - The system uses a machine learning-based model to recommend captions based on mood and similarity to the user's provided caption.

- Uses TF-IDF (Term Frequency-Inverse Document Frequency) and cosine similarity to find captions that are most similar to the user's input, based on mood categorization.

Data Storage and User Preferences:

- **User Preferences:**

- Allows users to store preferred moods and caption preferences for future use, enabling quicker caption generation.
- Captions generated or recommended are logged for the user to reference, improving future recommendations.

ML and LLM Model Integration:

- **LLM-Based Captions (Hugging Face Model):**

- Generates creative captions using a pre-trained model from Hugging Face, allowing for diverse and engaging captions based on user inputs.
- The model adapts to different moods and provides a variety of outputs for better user engagement.

- **ML-Based Caption Recommendations:**

- Uses a dataset of pre-defined captions with mood classifications and employs machine learning algorithms like TF-IDF Vectorization and Cosine Similarity to suggest the most relevant captions.
- Allows for tailored recommendations based on user-provided captions, improving the accuracy of suggestions over time.

Performance Monitoring and Analysis:

- **Model Performance Evaluation:**

- Continuously evaluates the performance of both the Hugging Face model and the ML model based on user feedback and interaction.
- Keeps track of which mood-based captions are most successful for users, optimizing suggestions based on real-world performance.

- **Automatic Caption Mood Matching:**

- Automatically analyzes the user-provided caption and selects the best mood for the most accurate recommendations.

Security and User Experience:

- **Real-time Processing and UI Interactivity:**
 - The system provides an interactive user experience, processing user inputs in real-time and providing instant feedback on caption suggestions.

2.3 Design and Implementation Constraints

1. **Web Interface Design:** Must be responsive for compatibility across devices and browsers.
2. **Model Performance:** Balances accuracy and efficiency to ensure fast, real-time caption generation.
3. **Third-Party API Integration:** If external APIs are used, changes or disruptions in those APIs could affect functionality.
4. **Security Considerations:** User data must be securely stored and handled, with adherence to basic security practices.

2.4 Assumptions and Dependencies

Assumptions:

1. **Model Performance:** The caption generation and recommendation models (LLM-based and ML-based) are assumed to deliver accurate and contextually relevant captions.
2. **Streamlit:** Streamlit is assumed to efficiently handle user interactions and display results in a user-friendly interface.
3. **Image and Text Input:** Users are expected to provide relevant image descriptions or captions for accurate recommendations.
4. **Mood Data:** It is assumed that mood categories for captions are predefined and users will choose appropriate moods.

Dependencies:

1. **Hugging Face API:** The system depends on the Hugging Face API for caption generation via a pretrained model.
2. **ML Libraries:** Scikit-learn and PIL are used for caption recommendation and image processing.
3. **Web Hosting:** Cloud hosting services (e.g., AWS, Google Cloud) are required for deployment and scalability.
4. **Image Processing:** The system relies on PIL for image handling and caption description generation.

3 Specific Requirements

3.1 External Interface Requirements

User Interfaces:

1. Input Page:

- The user interface shall allow users to input information related to their desired Instagram caption, such as:
 - Photo Description: Users can describe the image they are uploading or input a text-based description.
 - Mood Selection: Users can select the mood that matches their caption preferences (e.g., "Adventurous", "Motivational", "Chill", etc.).
 - Image Upload: Option to upload an image to generate captions based on the photo.
- The input form should be intuitive, with clear labels for each section, easy-to-understand dropdowns, and tooltips to assist with mood selection and photo description.

2. Results Page:

- The system shall display the generated captions and recommendations based on the input provided by the user:
 - LLM-Based Captions: Generated captions based on the user's photo description and mood selection.
 - ML-Based Caption Recommendations: Suggested captions similar to the user's input caption, based on selected mood.
 - Mood and Caption Information: Display of selected mood and input caption.
 - Performance Metrics: Show feedback on the relevance of captions (e.g., how well they align with the selected mood, based on the ML model's evaluation).
- The results page shall be interactive, with the ability to allow users to generate multiple captions or suggestions with just one click, and the option to refine the input for better results.

Hardware Interfaces:

No Specific Hardware Dependencies:

- The system can be deployed on standard servers or cloud infrastructure.
- Caption generation models are lightweight and can run on conventional hardware or cloud-based servers.

Software Interfaces:

SQLite:

- Used to store user inputs, generated captions, and system logs securely and efficiently.

Backend Logic (Python):

- **Handles:**
 - Integration with machine learning libraries (e.g., TensorFlow, Hugging Face Transformers).
 - Preprocessing of input data.
 - Caption generation and mood-based classification.
 - Managing interactions and logic flow behind the Streamlit app.

Frontend (Streamlit in Python):

- Streamlit is used to build a simple, interactive, and responsive web-based user interface.
- Allows users to input data and view generated captions in real time without needing HTML/CSS/JavaScript.

ML Model Operations (Python):

- Caption generation models are trained and deployed using Python libraries.
- Includes natural language processing (NLP) and possibly computer vision models depending on your use case.

3.2 Functional Requirements

1. FR1: Data Input and Caption Generation

- Users can upload images or input text prompts through the Streamlit interface.
- The system processes the input and generates relevant captions using pre-trained NLP models.

2. FR2: Model Selection and Execution

- Multiple captioning models (e.g., Transformer, LSTM-based) are supported.
- If no model is selected, the system auto-selects the best-performing model based on prior evaluations.

3. FR3: Display of Results

- Generated captions are shown on the interface in real time.
- Additional details like caption confidence score or classification (e.g., mood/tone) may be displayed.

4. FR4: Data Storage

- User inputs, generated captions, and logs are stored securely in an SQLite database.

- Historical data can be used to analyze caption quality and user preferences.

5. FR5: User Authentication

- Basic login and registration features are implemented.
- Ensures only authenticated users can generate or view captions.

3.3 Non-Functional Requirements

1. Performance

- The system shall generate captions within 2–3 seconds after input submission.
- Streamlit interface should remain responsive during model inference.

2. NFR2: Scalability

- The system should support scaling to handle more users, models, and image/text data without significant delays.
- New captioning models and datasets should be integrable with minimal changes.

3. NFR3: Accuracy

- Caption models must maintain an accuracy above 85% (based on BLEU or other captioning metrics).
- The system shall log performance and alert if model quality degrades.

4. NFR4: Data Security

- Uploaded images/text inputs and generated captions must be handled securely.
- Basic encryption and secure storage (e.g., SQLite) should be implemented for user-related data.

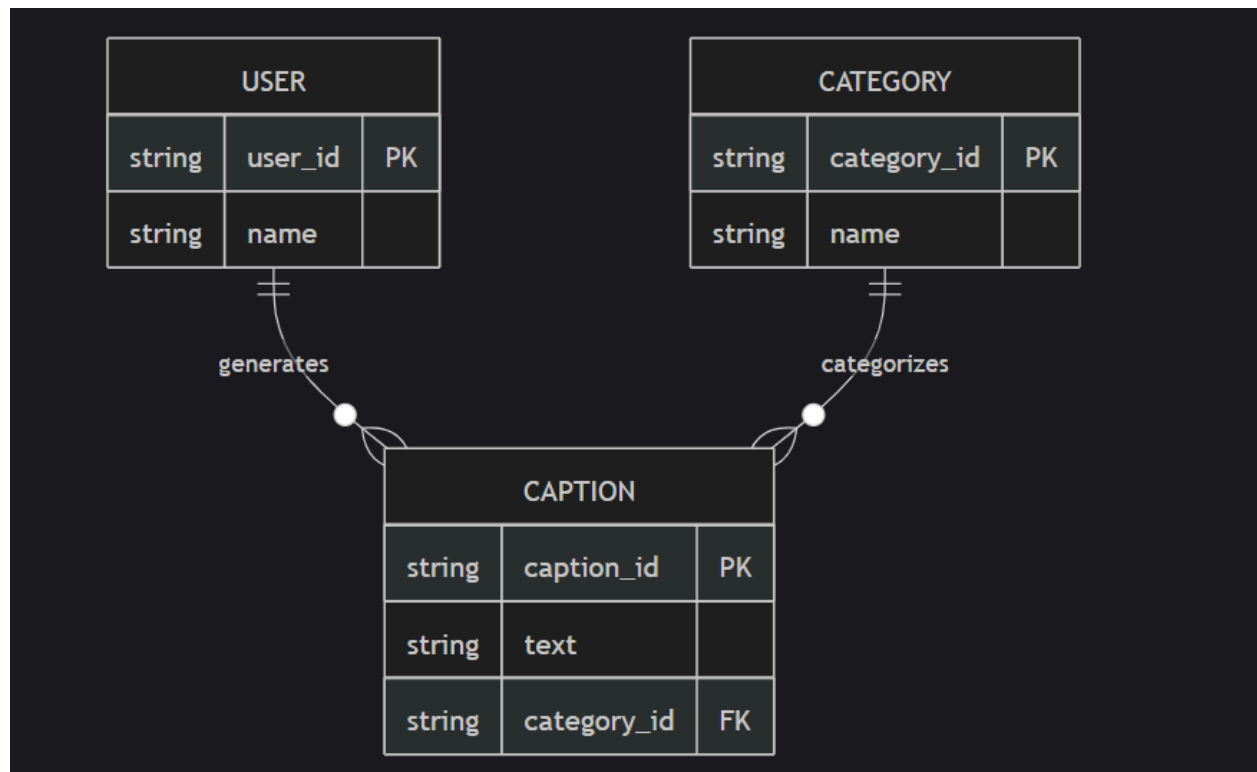
5. NFR5: Usability

- The Streamlit interface shall be intuitive with clear instructions and visual outputs.
- The design shall be responsive and accessible across devices.

6. NFR6: Maintainability

- The project shall use modular Python code and maintain organized folder structures.
- Proper documentation will ensure ease of model updates and debugging.

3.4 ER Diagram



4 Other Non-functional Requirements

4.1 Performance Requirements

- The system shall respond to user actions within 3 seconds.
- Caption generation (model inference) should complete in under 2 seconds after an image or text is submitted.
- Image upload and preprocessing shall take no more than 1–2 seconds for standard file sizes.

4.2 Safety and Security Requirements

- **SSR1: Data Confidentiality**
User-uploaded images and generated captions shall be stored securely and only accessible by the user.
- **SSR2: Input Validation and Attack Prevention**
The system shall validate all user inputs and uploads to prevent:

- SQL Injection
- Cross-Site Scripting (XSS)
- File Upload Vulnerabilities
- **SSR3: Data Backup and Recovery**
Regular backups of caption generation history and user data shall be maintained.
The system shall support recovery in case of failure or data loss.

4.3 Software Quality Attributes

- **Usability:**
The system shall offer a simple, intuitive, and visually clean interface for uploading images and viewing captions.
- **Reliability:**
The captioning engine shall consistently generate accurate and relevant image captions.
- **Performance:**
Image processing and caption generation shall be completed efficiently, ensuring minimal wait time.
- **Security:**
User-uploaded images and generated data shall be securely stored and transmitted with proper authentication.
- **Maintainability:**
The codebase shall be modular and well-documented to allow easy updates and debugging.
- **Portability:**
The application shall be accessible across various platforms, including desktop and mobile browsers.

5 Other Requirements

Not Applicable

Appendix A – Data Dictionary

Not Applicable

Appendix B - Group Log

Not Applicable

Chapter 6 : System Test Plan For ML and LLM

Project Title : Mood2Caption- An AI-Powered Instagram Caption Generator using LLM and ML

Prepared By: Misha N Devegowda(1RVU23CSE268)

Nikhita K Nagavar(1RVU23CSE309)

Spandana Sujay (1RVU23CSE464)

1) Test Plan Identifier

Test Plan Identifier: Instagram Caption Generator - System Test Plan

2) References

- IEEE 829:2008 Standard for Software and System Test Documentation.

Python Libraries Used:

- pandas, scikit-learn, pytest, requests, streamlit, PIL

Dataset:

- In-memory caption data set (embedded in the application) for mood-based caption recommendations.

3) Introduction

This document defines the system test plan for the Instagram Caption Generator application that utilizes both machine learning (ML) and large language model (LLM) techniques. The application generates Instagram captions based on user input using two methods:

- **ML-based recommendations** using a TF-IDF vectorizer and cosine similarity.
- **LLM-based generation** using the Hugging Face API to generate captions based on a description and mood.

The test plan will cover functional and integration testing, verifying both the LLM and ML caption recommendation features work seamlessly with the application.

4) Test Items

- ML-based caption recommendation using TF-IDF and cosine similarity.
- LLM-based caption generation via the Hugging Face API.
- Streamlit GUI, including handling of input data, caption display, and error management.

5) Software Risk Issues

- API failure or slow responses from Hugging Face API.
- Incorrect caption recommendations due to poorly tuned ML model or missing captions.
- GUI might not render properly across all platforms (Windows/Linux/Mac).

6) Features to be Tested

- ML-based caption recommendations:
 - Ensure captions are correctly recommended based on mood.
 - Check if cosine similarity correctly ranks captions.
- LLM-based caption generation:
 - Verify the Hugging Face API generates captions as expected.
 - Check for correct mood handling and relevance to input description.
- Streamlit GUI:
 - Input validation.
 - Correct display of captions and images.
 - Error handling for empty inputs, invalid moods, etc.

7) Features not to be Tested

- The functionality of external APIs beyond the caption generation.
- Advanced error handling for API timeouts or failures beyond standard exceptions.
- Detailed testing of image description extraction (only tested if an image is provided).

8) Approach

- **Unit Testing** for individual functions (caption recommendation, LLM generation).
- **Integration Testing** to verify the ML and LLM components work together with Streamlit.
- **GUI Testing** for usability, input handling, and display.
- **Negative Testing** for edge cases, such as empty inputs or invalid moods.

9) Item Pass/Fail Criteria

- **Pass:** If the system returns correct and relevant captions, handles errors gracefully, and the GUI functions without crashes.
- **Fail:** If the system fails to generate or recommend captions correctly, displays errors improperly, or crashes during interactions.

10) Suspension Criteria and Resumption Requirements

Testing will be suspended if:

- API failure prevents caption generation.
- Streamlit application fails to load or display properly.

11) Test Deliverables

- Test case documentation.

- Test execution results (pass/fail outcomes).
- Defect logs and reports.
- Final test summary report.

12) Remaining Test Tasks

- Set up test environment for Hugging Face API.
- Define test cases for each function and mood.
- Execute integration tests with the Streamlit app.

13) Environmental Needs

- Python 3.x environment with pytest, requests, streamlit, pandas, scikit-learn.
- Hugging Face API credentials for caption generation.
- Streamlit platform for GUI testing.

14) Staffing and Training Needs

- Testers with experience in Python, machine learning, and GUI testing.
- Familiarity with pytest and requests for API testing.

15) Responsibilities

- **Test Lead:** Ensure test coverage and execution.
- **Testers:** Execute and document test cases.
- **Developers:** Fix defects discovered during testing.
- **Project Manager:** Ensure timely testing completion.

16) Schedule

- Test Planning: 1 day.
- Test Case Development: 3 days.
- Test Execution: 4 days.
- Defect Fixes and Retesting: 2 days.
- Test Summary Report: 1 day.

17) Planning Risks and Contingencies

- **Risk:** Slow or unavailable Hugging Face API.
 - **Contingency:** Use mock responses for testing.
- **Risk:** Unresponsive GUI on different platforms.
 - **Contingency:** Focus testing on the most common platforms.

18) Traceability Matrix

<i>Requirement ID</i>	<i>Requirement Description</i>	<i>Test Case ID</i>	<i>Test Case Description</i>	<i>Test Type</i>
<i>REQ-LM-001</i>	<i>Generate captions based on the mood using Hugging Face API</i>	<i>TC-001</i>	<i>Test the Hugging Face caption generation when a description and mood are provided.</i>	<i>Functional Test</i>

<i>REQ-LM-002</i>	<i>Caption recommendations based on mood (ML-based)</i>	<i>TC-002</i>	<i>Verify that captions are recommended based on a given input caption and mood.</i>	<i>Functional Test</i>
<i>REQ-LM-003</i>	<i>Validate the recommendation logic for captions (KNN-based)</i>	<i>TC-003</i>	<i>Test that the ML-based recommendation system suggests the correct captions based on the input caption.</i>	<i>Functional Test</i>
<i>REQ-LM-004</i>	<i>Handle invalid or empty input captions</i>	<i>TC-004</i>	<i>Test that empty or invalid captions raise appropriate exceptions (ValueError).</i>	<i>Negative Test</i>
<i>REQ-LM-005</i>	<i>Handle invalid mood input</i>	<i>TC-005</i>	<i>Ensure that invalid mood inputs are properly handled with a ValueError, showing the error message.</i>	<i>Negative Test</i>
<i>REQ-LM-006</i>	<i>Return a limited number of results based on the top_n input</i>	<i>TC-006</i>	<i>Verify that the number of results returned does not exceed the specified top_n value.</i>	<i>Functional Test</i>
<i>REQ-LM-007</i>	<i>Display captions correctly for valid inputs</i>	<i>TC-007</i>	<i>Ensure that the captions are displayed correctly for the user in the output section.</i>	<i>GUI/Functional Test</i>

<i>REQ-LM-008</i>	<i>Ensure correct handling of image input</i>	<i>TC-008</i>	<i>Test the image upload and inference process for generating captions from an image description.</i>	<i>Functional Test</i>
<i>REQ-LM-009</i>	<i>Ensure the application handles caption generation with text</i>	<i>TC-009</i>	<i>Test if the caption generator can handle text input to generate a meaningful output.</i>	<i>Functional Test</i>
<i>REQ-LM-010</i>	<i>Display error message when there are issues with API requests</i>	<i>TC-010</i>	<i>Test the behavior when there is an issue with the Hugging Face API, and ensure proper error handling.</i>	<i>Integration Test</i>

Test Cases

```
import pytest

from recommender_LLM import recommend_captions_by_mood, generate_caption,
get_captions_by_mood

# ----- Positive Test Cases ----- #

def test_recommend_valid_caption():

    input_caption = "Feeling happy and vibrant"

    mood = "happy"

    results = recommend_captions_by_mood(input_caption, mood, top_n=3)
```

```

    assert len(results) == 3

    assert all(isinstance(c, str) for c in results)

def test_generate_caption_with_description():

    description = "A calm sunset by the beach with waves rolling in"

    mood = "Chill"

    prompt = (

        f"You are an AI Instagram caption generator. The user provided the  

following photo description:\n"

        f"{description}\n"

        f"The mood of the caption should be **{mood.lower()}**.\n"

        f"Generate 3 creative and mood-matching Instagram captions."

    )

    output = generate_caption(prompt)

    assert isinstance(output, str)

    assert len(output) > 0

def test_recommend_top_one():

    input_caption = "Lazy mornings are the best"

    mood = "lazy"

    result = recommend_captions_by_mood(input_caption, mood, top_n=1)

    assert len(result) == 1

# ----- Negative Test Cases ----- #

```

```
def test_recommend_empty_caption():  
    with pytest.raises(ValueError):  
        recommend_captions_by_mood("", "happy", top_n=3)  
  
def test_recommend_invalid_mood():  
    input_caption = "Sunset vibes"  
    mood = "nonexistent"  
    with pytest.raises(ValueError, match="No captions found for mood"):  
        recommend_captions_by_mood(input_caption, mood, top_n=3)  
  
def test_recommend_too_many_results():  
    input_caption = "Adventures in the mountains"  
    mood = "adventurous"  
    results = recommend_captions_by_mood(input_caption, mood, top_n=100)  
    assert len(results) <= len(get_captions_by_mood("adventurous"))
```


Chapter 7: Conclusion

The Mood-based Instagram Caption Generator and Recommendation System is a robust solution for personalized content generation in social media. By combining the power of Machine Learning (ML) and AI-driven text generation, the system provides users with the ability to generate contextually relevant and mood-tailored captions, as well as recommend similar captions based on user input. Here's a summary of the project:

Key Achievements:

1. AI-Powered Caption Generation:
 - Integrated Hugging Face's pre-trained language model for dynamic caption generation based on user-provided photo descriptions and mood selections.
 - The system offers the flexibility to create captions in various moods (e.g., motivational, chill, adventurous, sassy, etc.) based on the mood selected by the user.
2. ML-Based Caption Recommendation:
 - Utilized TF-IDF and Cosine Similarity algorithms to suggest captions based on the semantic similarity between user-inputted captions and a pre-existing database of mood-tagged captions.
 - The system ensures that users are provided with a personalized set of recommendations that align with their input, optimizing the relevance of suggestions.
3. User-Friendly Interface:
 - Built using Streamlit, the system offers a sleek and intuitive graphical user interface (GUI), allowing users to input captions, select moods, and receive results effortlessly.
 - The interface supports easy photo uploads and text-based descriptions for generating captions, offering flexibility in input methods.

4. Agile Development Methodology:

- Followed Agile principles throughout the development cycle, ensuring iterative improvements and the ability to quickly adapt to changes and user feedback.
- The use of sprints, backlog grooming, and retrospectives helped refine both the AI and ML components for better performance and user experience.

Limitations:

1. Lack of Real-Time User Interaction:

- The system does not yet incorporate real-time user feedback or preferences, which limits the ability to further personalize the recommendations based on individual user behavior and interactions.

2. Absence of Natural Language Processing (NLP):

- The current system does not utilize NLP for understanding deeper meanings in book descriptions or user reviews, limiting its ability to fully interpret nuanced inputs.

3. Scalability Issues:

- As the database grows, the performance may degrade, especially with large datasets. Optimizing the system for high scalability is a challenge that needs to be addressed for broader usage.

Future Scope:

1. Hybrid Recommendation Systems:

- Integrating Collaborative Filtering with the current content-based approach to improve the system's ability to recommend captions based on user preferences and previous behavior.
- Incorporating Natural Language Processing (NLP) for deeper understanding of the text, including the ability to parse and analyze book descriptions, reviews, or user-generated content more effectively.

2. Real-Time Data Integration:

- Integrating live data APIs from platforms like Instagram or Goodreads to pull real-time book data, user ratings, and trending topics, enriching the recommendation model.
- This would allow the system to offer up-to-date and highly relevant recommendations for users.

3. User Profile Management:

- Future versions could include features for user profile creation, where users can save their preferences, track recommended captions, and receive more personalized content recommendations over time.
- This would create a more interactive and engaging user experience.

4. Mobile and Web Deployment:

- Deployment on mobile and web platforms using frameworks like Flask, Django, or React Native, making the system more accessible and available for a wider audience.
- Creating a cross-platform experience would improve accessibility and usability for users on different devices.

5. Exploration of Quantum Machine Learning (QML):

- Investigating the integration of Quantum Neural Networks (QNN) and Quantum Machine Learning techniques to explore potential improvements in data processing and recommendation accuracy, especially for large datasets.
- QML holds promise for exponentially faster processing times, which could drastically enhance the system's scalability and efficiency.

Final Thoughts:

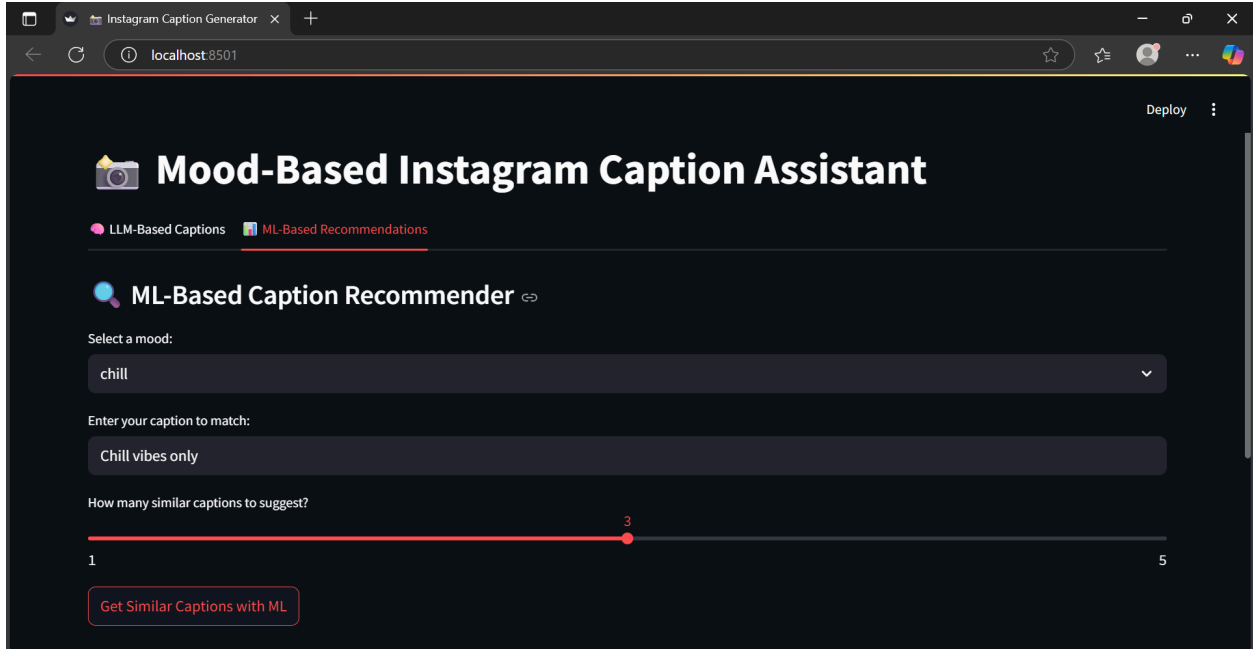
- The Mood-based Instagram Caption Generator is a perfect example of how AI and ML can revolutionize content creation on social media. It allows users to generate creative

captions aligned with their mood and preferences in real-time.

- By combining machine learning techniques for content-based recommendation and AI-powered text generation for unique and tailored outputs, the system enhances the user experience and ensures that the content is both engaging and relevant.
- The project is not only a valuable tool for casual users but also holds potential for future integration into marketing platforms, social media influencers, and other content creators looking for a seamless way to create appealing and mood-appropriate captions for their audiences.

Chapter 8: Output

ML-Recommendation:



The screenshot shows a web browser window with the title "Instagram Caption Generator". The address bar shows "localhost:8501". The page has a dark theme and a "Deploy" button in the top right corner. The main heading is "Mood-Based Instagram Caption Assistant" with a camera icon. Below the heading are two tabs: "LLM-Based Captions" and "ML-Based Recommendations", with the latter being selected. The section is titled "ML-Based Caption Recommender" with a link icon. It contains a "Select a mood:" dropdown menu with "chill" selected. Below that is a text input field with "Chill vibes only". A slider for "How many similar captions to suggest?" is set to 3, with a range from 1 to 5. A button labeled "Get Similar Captions with ML" is at the bottom.

Instagram Caption Generator x +

localhost:8501

Deploy

Mood-Based Instagram Caption Assistant

LLM-Based Captions ML-Based Recommendations

ML-Based Caption Recommender

Select a mood:

chill

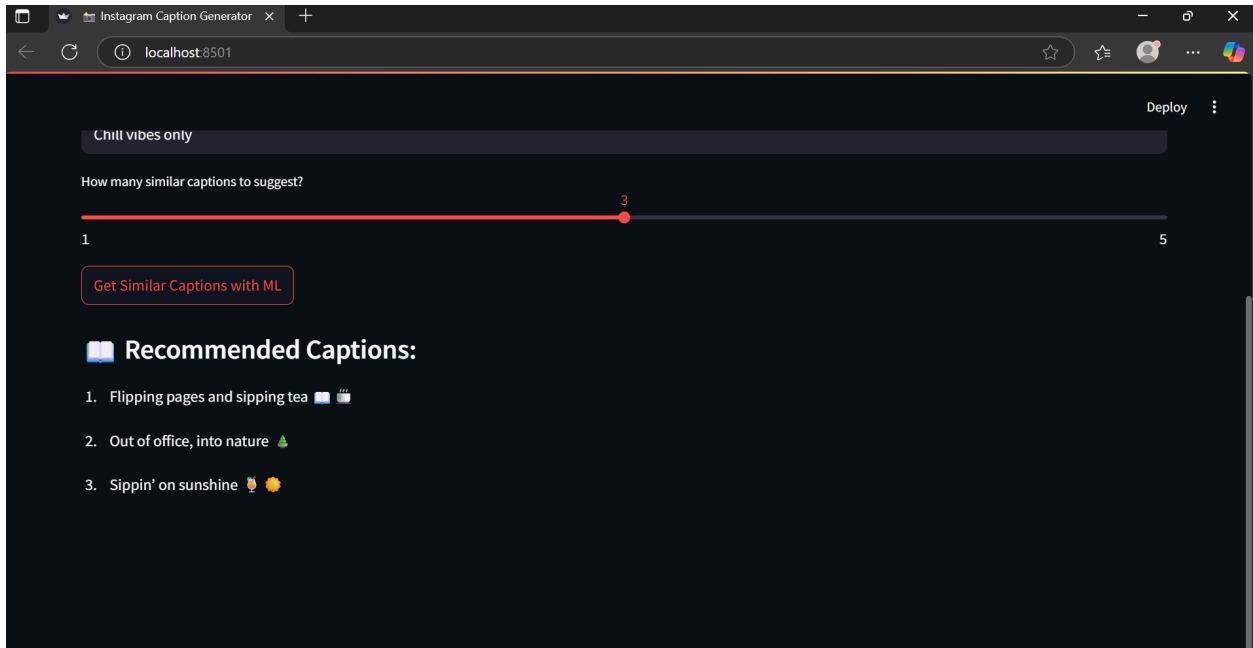
Enter your caption to match:

Chill vibes only

How many similar captions to suggest?

1 3 5

Get Similar Captions with ML



The screenshot shows the same web browser window as the previous one, but the "Recommended Captions" section is now visible. It contains a list of three recommended captions, each with a small icon representing the mood.

Instagram Caption Generator x +

localhost:8501

Deploy

Chill vibes only

How many similar captions to suggest?

1 3 5

Get Similar Captions with ML

Recommended Captions:

1. Flipping pages and sipping tea 📖 ☕
2. Out of office, into nature 🌿
3. Sippin' on sunshine ☀️ 🥤

LLM-Recommendation:


Instagram Caption Generator

localhost:8501


Deploy

🌟 Hugging Face Caption Generator

Upload a photo (optional)

 Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Browse files

 images.jpeg 6.3KB

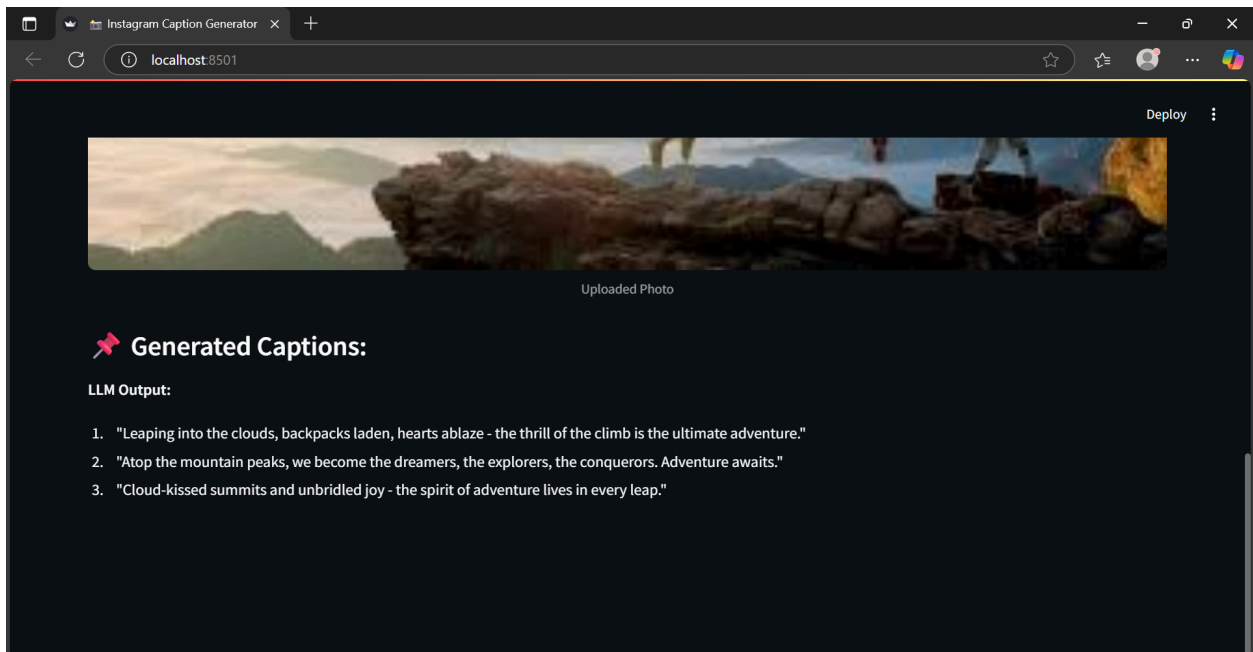
Describe the photo in words:

Silhouetted against a dreamy, cloud-kissed mountain scape, four figures leap in shared triumph, their backpacks hinting at an adventurous climb. The vast sky amplifies their joyous moment at the summit.

Choose the mood for your caption

Adventurous

Generate Captions with LLM



Chapter 9: Code

```
import streamlit as st

import requests

from PIL import Image

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity

import os


# ----- Hugging Face Caption Generator ----- #

API_URL =
"https://api-inference.huggingface.co/models/mistralai/Mixtral-8x7B-Instru
ct-v0.1"

API_TOKEN = "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" # Store your token
securely in environment variables

headers = {"Authorization": f"Bearer {API_TOKEN}" }


def generate_caption(prompt, max_tokens=150):

    payload = {

        "inputs": prompt,

        "parameters": {

            "max_new_tokens": max_tokens,

            "temperature": 0.75,

            "top_p": 0.9,

            "do_sample": True
```

```

    }

}

try:

    response = requests.post(API_URL, headers=headers, json=payload)

    response.raise_for_status()

    result = response.json()

    if isinstance(result, list) and "generated_text" in result[0]:

        return result[0]["generated_text"].replace(prompt, "").strip()

    else:

        return "⚠️ Could not generate caption."

except requests.exceptions.RequestException as e:

    return f"❌ Request failed: {e}"

# ----- ML Caption Recommendation ----- #

caption_data = [

    ("Sunsets and palm trees 🌴🌅", "chill"),

    ("Coffee first, adulting second ☕", "relatable"),

    ("Chasing dreams, not people ✨", "motivational"),

    ("Beach days are the best days 🏖️", "happy"),

    ("Just vibing and thriving 😎", "chill"),

    ("Adventures fill your soul 🌍", "adventurous"),

    ("Life's too short for bad vibes 🍷", "motivational"),

    ("Happiness looks good on me 😊", "happy"),

```


("Weekend mode: ON 🏠📺", "lazy"),
("Smiles, sunshine, and selfies ☀️📷", "happy"),
("Wander often, wonder always 🧑🌌", "adventurous"),
("Stay wild, moon child 🌙", "dreamy"),
("City lights and late nights 🏙️", "urban"),
("Peace, love, and sandy feet 🌊", "chill"),
("No filter needed when you're glowing ✨", "confident"),
("Living my best life 🦋✨", "confident"),
("Good vibes only 🌈", "happy"),
("Lost in the moment 🌀", "dreamy"),
("Sippin' on sunshine 🍹☀️", "chill"),
("Catch flights, not feelings ✈️💔", "sassy"),
("Making memories all day long 📷", "happy"),
("Born to stand out ✨", "confident"),
("Take only pictures, leave only footprints 🦶", "adventurous"),
("Keepin' it real since day one 🔥", "bold"),
("Too glam to give a damn 🍷💅", "sassy"),
("On cloud nine ☁️", "happy"),
("Netflix, snacks, repeat 🍿", "lazy"),
("Elegance never goes out of style 👠", "confident"),
("Heart full of wanderlust ✨🌍", "adventurous"),
("Doing nothing, and proud of it 🛌", "lazy"),
("Don't quit your daydream ☁️", "dreamy"),

```

("Slaying in silence 🦋", "bold"),

("Can't hear the haters over my playlist 🎧", "bold"),

("Less perfection, more authenticity 🌱", "relatable"),

("Fuelled by caffeine and ambition ☕🚀", "motivational"),

("Throwing sass like confetti 🎉", "sassy"),

("Not lost, just exploring 🗺️", "adventurous"),

("Creating my own sunshine ☀️", "motivational"),

("Pillow talks and pastel skies 🌸", "dreamy"),

("Keep calm and scroll on 📱", "relatable"),

("Catch me under the stars ✨", "dreamy"),

("Soft heart, strong mind ❤️🧠", "confident"),

("Out of office, into nature 🌲", "chill"),

("Flipping pages and sipping tea 📖🍵", "chill"),

("Sparkle every step of the way ✨👟", "motivational"),

("Brunch is always a good idea 🥞", "relatable"),

("Sweet as honey, stings when needed 🐝", "sassy"),

("Midnight thoughts, morning coffee ☕🌙", "dreamy"),

("Just a girl building her empire 👑", "motivational"),

("Unbothered and moisturized 💅", "bold"),

]

def get_captions_by_mood(mood):

    return [caption for caption, m in caption_data if m == mood]

```

```

def recommend_captions_by_mood(input_caption, mood, top_n=3):

    if not input_caption.strip():

        raise ValueError("Input caption cannot be empty.")

    captions = get_captions_by_mood(mood)

    if not captions:

        raise ValueError(f"No captions found for mood: '{mood}'")

    captions_with_input = captions + [input_caption]

    vectorizer = TfidfVectorizer(stop_words='english')

    tfidf_matrix = vectorizer.fit_transform(captions_with_input)

    cosine_similarities = cosine_similarity(tfidf_matrix[-1],
tfidf_matrix[:-1]).flatten()

    similar_indices = cosine_similarities.argsort()[-top_n:]::-1

    return [captions[i] for i in similar_indices]

# ----- Streamlit UI ----- #

st.set_page_config(page_title="🌟 Instagram Caption Generator",
layout="wide")

st.title("🌟 Mood-Based Instagram Caption Assistant")

```

```

tab1, tab2 = st.tabs(["🧠 LLM-Based Captions", "📺 ML-Based
Recommendations"])

# --- LLM-Based Caption Generator --- #

with tab1:

    st.subheader("✨ Hugging Face Caption Generator")

    image = st.file_uploader("Upload a photo (optional)", type=["jpg",
"jpeg", "png"])

    description = st.text_area("Describe the photo in words:")

    hf_mood = st.selectbox("Choose the mood for your caption", [

        "Humorous", "Romantic", "Introspective",

        "Adventurous", "Chill", "Empowering",

        "Motivational", "Dreamy", "Sassy",

        "Bold", "Confident", "Relatable", "Lazy", "Urban"

    ])

    if st.button("Generate Captions with LLM"):

        with st.spinner("Crafting your perfect caption..."):

            if not image and not description:

                st.warning("Please upload a photo and/or provide a
description.")

                st.stop()

            if image:

```

```

        image_data = Image.open(image)

        st.image(image_data, caption="Uploaded Photo",
use_container_width=True)

    # Compose a detailed description using both inputs

    if image and description:

        photo_desc = (

            f"The user uploaded a photo and described it as
follows:\n"

            f"'{description}'."

        )

    elif image:

        photo_desc = "The user uploaded a photo. You can infer a
possible description from it."

    else:

        photo_desc = f"The user provided the following description
of the photo:\n'{description}'."

    prompt = (

        f"You are an AI Instagram caption generator. Below is the
image context:\n\n"

        f"{photo_desc}\n\n"

        f"The desired mood for the caption is
**{hf_mood.lower()}**.\n"

        f"Generate 3 creative and mood-matching Instagram
captions."

```

```

    )

    hf_output = generate_caption(prompt)

    st.markdown("### 📌 Generated Captions:")

    st.markdown("***LLM Output***")

    st.write(hf_output)

# --- ML-Based Caption Recommender --- #

with tab2:

    st.subheader("🔍 ML-Based Caption Recommender")

    moods = sorted(set(mood for _, mood in caption_data))

    selected_mood = st.selectbox("Select a mood:", moods)

    input_caption = st.text_input("Enter your caption to match:")

    top_n = st.slider("How many similar captions to suggest?", 1, 5, 3)

    if st.button("Get Similar Captions with ML"):

        if input_caption:

            try:

                recommendations =
recommend_captions_by_mood(input_caption, selected_mood, top_n)

                st.markdown("### 📖 Recommended Captions:")

                for i, cap in enumerate(recommendations, 1):

                    st.write(f"{i}. {cap}")

            except ValueError as ve:

```

```
st.error(str(ve))
```

```
else:
```

```
st.warning("Please enter a caption to get recommendations.")
```