Software Requirements Specification

Conversational Memory Bot – AI-Powered Photo Gallery Assistant

Submitted by:
Asrarul Hoque Eusha
ID: 30177

Date: March 9, 2025
Version ID: 1.0
Document ID: SWD/AI/30177

Revision History

Doc.	Date	Author	Reviewer	Description of Revision
1.0	2025-02-07	Asrarul Hoque Eusha		Initial draft
2.0	2025-02-07	Asrarul Hoque Eusha		Final draft

Contents

1 Introduction

1.1 Purpose

This document serves as a detailed description of the specifications for the "Conversational Memory Bot - AI-Powered Photo Gallery Assistant" web application. It will demonstrate the goal and full statement of the system's development. Additionally, it will describe the interface and system limitations. The main purpose of this document is to be presented to a client for approval and to serve as a guide for the development team while they create the initial version of the system.

1.2 Scope

The Conversational Memory Bot is an AI-powered chatbot, designed to revolutionize the way users interact with their personal photo galleries. By combining advanced Natural Language Processing (NLP) and other frameworks, this system enables users to query, retrieve, and explore their photos using natural language and visual features.

Users upload their images to the gallery at any time, they can upload single or multiple images to add to the gallery. Then users are allowed to ask queries to find a specific image by inserting a natural language query or an image similar to the searched image. This system will be implemented as a web-based application with an interactive gallery interface for better user engagement.

1.3 Intended Stakeholder

The intended stakeholders for this document are listed below:

1. **Clients/ Customers** – Uses SRS document to verify that system meets their business requirements and expectations.
2. **Project Managers** – Responsible for overseeing the development process, ensuring that the project adheres to the defined scope, timeline, and budget while meeting all requirements specified in this document.
3. **Software Developers** – Use the SRS document to understand the functional and technical requirements needed to build the system.
4. **Quality Assurance (QA) & Testers** – This document will guide the QA team in designing test cases to validate that the software meets all specified requirements and functions correctly.

1.4 Definitions, Acronyms, and Abbreviations

Table 1 shows the acronyms and abbreviations used in this document.

2 Overall Description

An overview of the entire system will be provided in this section. In order to introduce the system's fundamental operation and demonstrate how it interacts with users, it will

Table 1: Definitions of terms and acronyms used in this document.

Term/Acronym	Definition
NLP	Natural Language Processing
RAG	Retrieval-Augmented Generation
LLM	Large Language Model
SLM	Small Language Model
UI	User Interface
API	Application Programming Interface

be explained within its context. Additionally, it will outline the types of stakeholders that will utilize the system and the features that each can access. Finally, the system's assumptions and limitations will be discussed.

2.1 Objectives

- Implement an AI-powered chatbot for retrieving images using natural language queries.
- Generate detailed descriptions of images, including objects, activities, and settings.
- Enable visual similarity search to find images based on colors, textures, and objects.
- Develop a scalable and interactive gallery system for seamless user experience.

2.2 Overview

2.2.1 Product Perspective

The Conversational Memory Bot is an AI-powered Web Application system designed to enhance user experience in managing and retrieving images from personal photo galleries. It integrates Natural Language Processing (NLP), Image Feature Extraction, and Retrieval-Augmented Generation (RAG) to provide interactive image search capabilities. The system will interact with Large Language Vision Model for understanding the images in the gallery and RAG implementation. It acts as an intelligent assistant, enabling users to interact with their photo libraries in a natural and intuitive way.

2.2.2 Product Functions

Using the web application users can search for photos using text-based queries. The system matches text input with relevant images based on content and metadata. The system will automatically generate detailed descriptions for selected images. Also, it will allow users to find images with similar objects, colors, and backgrounds by uploading an image in the chat. For any text or visual query the system ranks retrieved images based on similarity and user intent. So, the users will see the most relevant images first and less relevant images last. It will provide a dynamic gallery interface for browsing images.

2.2.3 User Characteristics

The single type of user of the system is general users. General Users primarily use the system to organize, search, and retrieve their personal photo collections. They interact with the AI assistant through natural language queries to find specific images efficiently.

2.2.4 Constraints

The system must process and retrieve images as efficiently as possible to ensure a smooth user experience. Efficient processing and retrieval also depend on the computing power available in the cloud infrastructure. The system must support a large volume of images without degradation in performance.

2.2.5 Assumptions and Dependencies

Users will have a structured image collection.

2.3 Technical Platforms

2.3.1 Software and Technologies

- PyCharm
- Python 3.10
- FastAPI
- ReactJS

2.3.2 Model Dependencies

- Llama-3.2 90b vision preview
- Numpy/Pandas
- CLIP model for multi-modal embedding
- Chromadb as vector database

2.3.3 Version Control

- Git

3 Functional Requirements

3.1 Overview

3.1.1 Feature 1: Natural Language Querying

The requirements, completion criteria, and limitations for the "Natural Language Querying" feature are given in Table 3.

Table 2: Functional requirements of the system

Serial	Main Features	Description
1	Natural Language Querying	Allows users to search for images using conversational text.
2	Contextual Image Retrieval	Matches text queries with images using contextual understanding.
3	Image Descriptions	Generates descriptive captions for selected images.
4	Visual Similarity Search	Retrieves images visually similar to a given one.
5	Relevance Ranking	Ranks retrieved images based on similarity to the query.
6	Interactive Gallery	Allows users to browse images with added functionalities like zoom.

Table 3: Natural Language Query

Requirement ID	Requirement Description	Acceptability/ Completion Criteria	Limitations/ Constraints	Test case Identifier
NLQ_001	Users can interact with the system using natural language to ask questions about their images.	The system should accurately interpret user queries and return relevant images or information.	Accuracy depends on AI model performance.	**TC_NLQ_001**

3.1.2 Feature 2: Contextual Image Retrieval

The requirements, completion criteria, and limitations / constraints for the "Contextual Image Retrieval" feature are given in Table 4.

3.1.3 Feature 3: Image Description Generation

The requirements, completion criteria, and limitations/constraints for the "Image Description Generation" feature are given in Table 5.

3.1.4 Feature 4: Visual Similarity Search

The requirements, completion criteria, and limitations / constraints for the "Visual Similarity Search" feature are given in Table 6.

Table 4: Contextual Image Retrieval

Requirement ID	Requirement Description	Acceptability/ Completion Criteria	Limitations/ Constraints	Test case Identifier
CIR_002	Match user queries with images based on contextual and semantic understanding of the query.	The retrieved images should be contextually relevant to the user's query.	Requires well-labeled metadata and generated description for image. Performance depends on the VLM used.	**CIR_002**

Table 5: Image Description Generation

Requirement ID	Requirement Description	Acceptability/ Completion Criteria	Limitations/ Constraints	Test case Identifier
IDG_003	Generate detailed descriptions of selected images, including objects, activities, and settings.	The system should provide accurate and meaningful descriptions for at least 80% of the tested images.	AI-generated descriptions may not always be perfect; May not work well for abstract images.	**TC_IDG_003**

3.1.5 Feature 5: Relevance Ranking

The requirements, completion criteria, and limitations/constraints for the "Relevance Ranking" feature are given in Table 7.

3.1.6 Feature 6: Interactive Gallery

The requirements, completion criteria, and limitations/constraints for the "Interactive Gallery" feature are given in Table 8.

4 User Interface

The table 9 shows all user interfaces of the system.

5 Non-Functional Requirements

Table 6: Visual Similarity Search

Requirement ID	Requirement Description	Acceptability/ Completion Criteria	Limitations/ Constraints	Test case Identifier
VSS_004	Retrieve images visually similar to a selected one by comparing features like color, texture, and objects.	The retrieved images should have a similarity score of at least 70% with the selected image.	High computation cost for feature extraction; May struggle with abstract images.	**TC_VSS_004**

Table 7: Relevance Ranking

Requirement ID	Requirement Description	Acceptability/ Completion Criteria	Limitations/ Constraints	Test case Identifier
RR_005	Rank retrieved images based on how closely they match the user's query in context and meaning.	The system should provide ranked results where the most relevant images appear in the top 5 results.	Ranking depends on AI model performance; Some queries may return ambiguous results.	**TC_RR_005**

5.1 Performance Requirements

System should retrieve images as efficiently as possible for a standard query. It should also work in large image galleries without performance degradation.

6 System Architecture

6.1 Front-End

A web-based user interface that allows users to interact with their photo gallery using a natural language or image similarity search. The front-end will be built with React.js and communicate with the back-end via API calls. Key Pages:

- Homepage: Users start their interaction with the application. All pages include a nav-bar on top to switch between pages.
- Gallery Page: Displays all images uploaded through the 'Batch Image Uploader' page to store in the gallery.
- Image Viewer Page: Provides a detailed view with descriptions.

Table 8: Interactive Gallery

Requirement ID	Requirement Description	Acceptability/ Completion Criteria	Limitations/ Constraints	Test case Identifier
IG_006	A gallery interface allowing users to browse images with advanced interactivity.	Users should be able to browse, modify image descriptions, and interact with the gallery seamlessly.	UI performance may depend on device capabilities; Requires dynamic UI updates.	**TC_IG_006**

Table 9: User interfaces

Serial	UI Name	Related Function
1	Homepage	Navigation and Overview
2	Gallery	Displays retrieved images
3	Batch Image Uploader	Allows users to upload multiple images
4	Image Viewer	Provides options like zoom and description
5	Chat Interface	Allows users to query images and interact with AI

- Batch Image Uploader Page: Allows users to upload multiple images for processing and storing.
- Chat Page: Users interact with the AI bot for image retrieval and exploration.

6.2 Backend

The backend is developed using FastAPI to handle user requests and manage image retrieval through API endpoints. Core responsibilities of the backend are:

- Processing User Queries: Extracts text or image input, validates requests, and routes them to the AI model.
- Handling Image Uploads: Generates tags and descriptions for uploaded images. And sends them to RAG pipeline to store image and text embeddings into vector database collection.
- Connecting to the RAG system: Receive user requests through the front-end and send and retrieve data from the Retrieval-Augmented Generation (RAG) pipeline.

6.3 RAG System

The RAG system is responsible for retrieving images and generating responses based on user queries. It enables contextually and semantically similar image retrieval by text or image query. The key components of the RAG system are the following:

- Image and Text Embeddings: Represents user queries and images in a shared vector space for similarity matching.

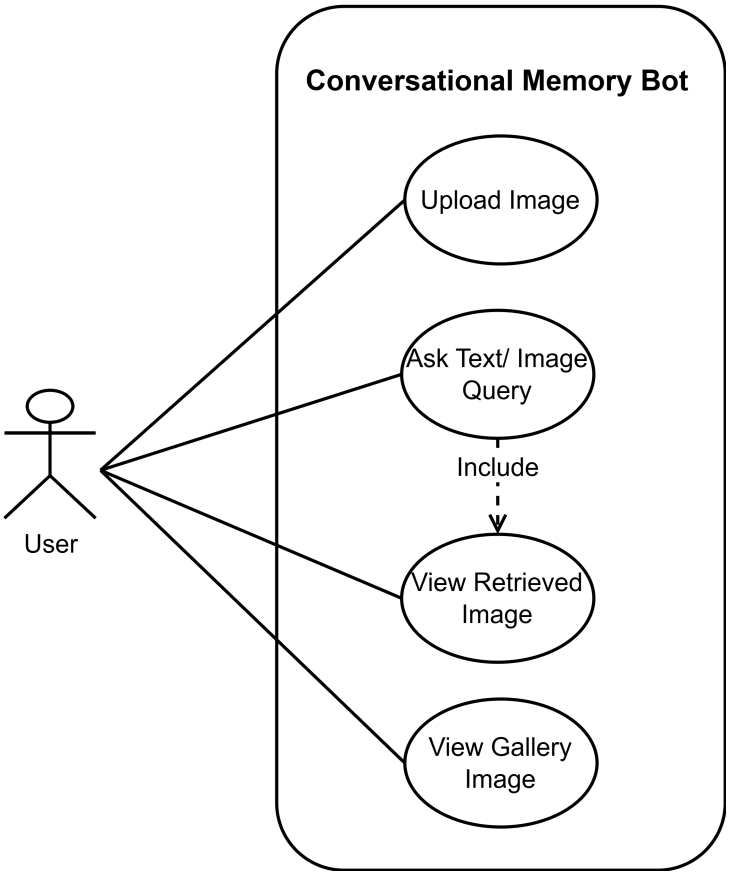

Figure 1: Overall Use Case Diagram of Conversational Memory Bot

- Semantic Search Engine: Retrieves the most relevant images according to the query.
- Vector Database Collection: Stores the multimodal embeddings along with tags for the images as metadata and description as a document for efficient retrieval.
- VLM Module: Generates meaningful descriptions and responses about retrieved images.

7 Components

7.1 User Interface

- The User Interface will be built with HTML, tailwind-CSS, and React.js.
- No particular framework or library other than the mentioned technologies will be used.
- The Chat page will feature a natural language or image input box where users can type queries. It will also display the retrieved images based on the user's query.
- A gallery will display all images uploaded to the time.
- Users can select an image from the gallery to view detailed descriptions.

7.2 Backend Server

- Python FastAPI will be used for building the Backend Server.
- The backend will handle HTTP requests/responses for user queries and image retrieval.
- It will preprocess user queries using NLP/Vision tools and send them to the RAG models for processing.
- The backend will post-process RAG output, such as image descriptions, similarity search results, and relevance rankings, in expected format before sending them to the frontend.
- It will interact with multimodal AI models (e.g., CLIP) for multimodal embeddings and Llama-3.2 90b for generating descriptions of images. It will also interact with RAG to retrieve contextually similar images.

7.3 Retrieval-Augmented Generation (RAG) System

- The RAG pipeline consists of Chromadb and Llama-3.2 90b. And Chromadb will be used as a vector database. Llama-3.2 90b from 'Groq' will be used as the VLM model to generate descriptions that align the user query with the understanding of the image.
- It will retrieve matching images from the vector database.
- This system will generate human-like responses about the retrieved images, providing detailed descriptions and contextual explanations.

8 Data Flow

In Figure 2, the flow diagram of the system is depicted. This flow diagram provides a clear visualization of how the system processes inputs, generates embeddings, stores data, and retrieves relevant responses in a retrieval-augmented system.

8.1 User Interaction with the Front-End

- The user interacts with the web interface through the different pages on the web screen for different tasks.
- The Front-End processes the user input and sends a request to the FastAPI Server.
-

8.2 FastAPI Server Processing

The FastAPI Server handles different types of requests sent from the frontend. It invokes the retrieval system for the retrieval of images based on the user query. Saves the uploaded image to a folder and invokes VLM to generate the description of the uploaded image. Then send the generated description and uploaded image to the RAG system to store in the vector database.

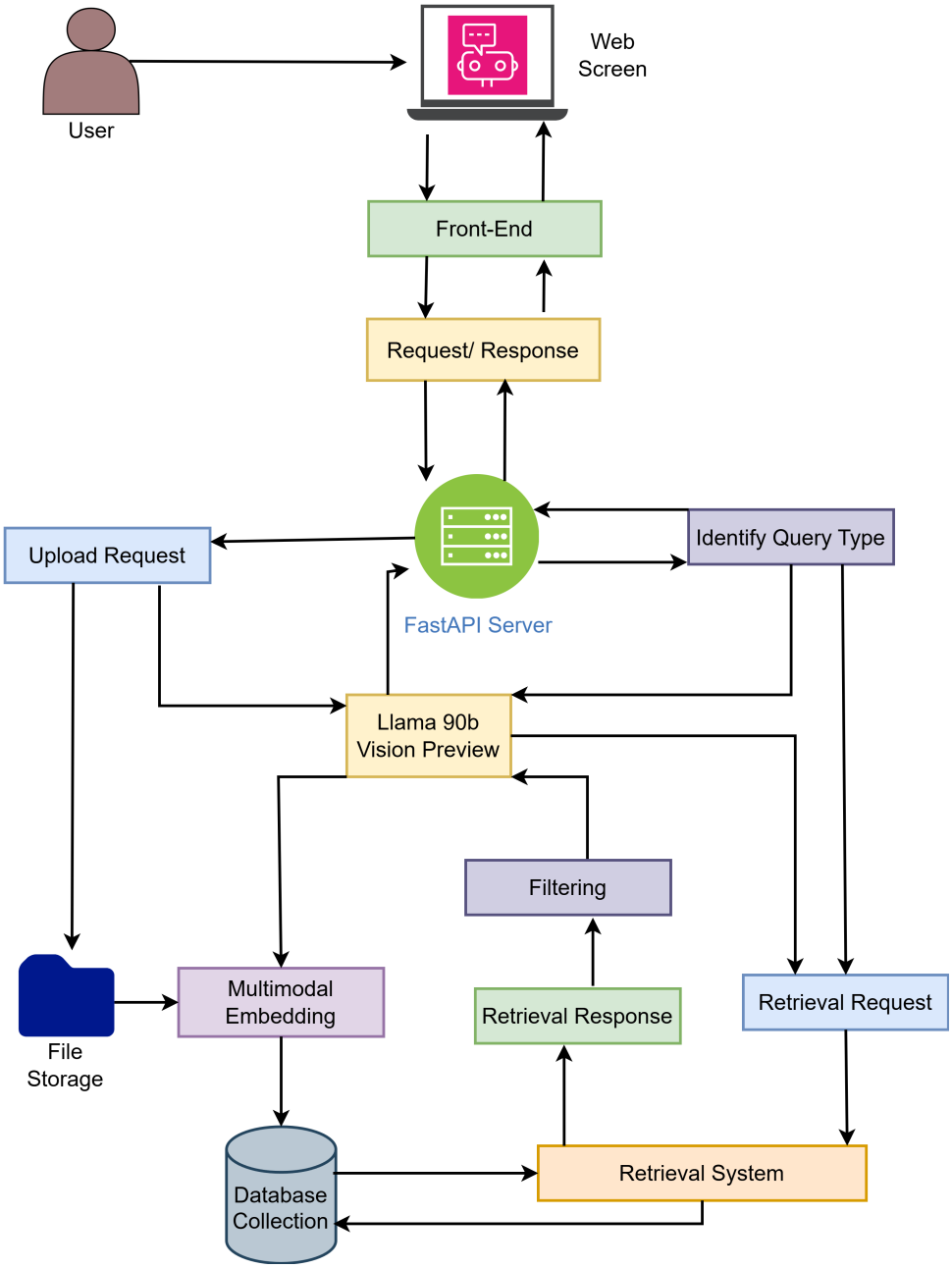

Figure 2: Flow Diagram of the System

8.3 Uploading New Images

- If the user uploads an image, the FastAPI Server processes the upload request and forwards the image to the VLM (Vision-Language Model).
- The VLM generates tags and descriptions for the given image.
- Then multimodal embeddings (representing both visual and textual features) were extracted for the image along with its description.
- The extracted embeddings with the tags of the image are stored in a Database Collection for future retrieval. The original image is stored in File Storage.

8.4 Retrieving Images Based on User Query

- If the user searches for an image (e.g., "Show me pictures of a dog in a park"), the FastAPI server processes the retrieval request.
- The request is forwarded to the Retrieval System, which queries the Database Collection to find matching embeddings.
- The Retrieval System fetches the most relevant images and sends the retrieval response back to the FastAPI Server through the VLM to generate description of image that aligns with the user query.

8.5 Response to User

The FastAPI Server processes the retrieved data and sends the results back to the Front-End. The user sees the retrieved images and related metadata on the Web Screen.

9 Design Constraints

Primarily the system will not implement optional requirements such as image tagging and interactive description update. They might be implemented after completing other important features successfully. The system cannot be heavier, as it will be used in personal devices. The system must use VLM which are free to use, and the model should be compatible with the RAM size of working PC. In addition, response time or complexity should be minimized as much as possible.

10 Implementation Plan

- Create web interface for uploading and displaying results and other pages
- Develop FastAPI backend
- Implement RAG pipeline
- Integrate components and perform testing
- Optimize performance and scalability

11 Folder Structure

```
30177_memovision/

    memovision_backend/
        app/
            main.py
            constants.py
            models.py
            question.csv
```

```
            .env
            api/
                gallery.py
                query.py
                upload.py
                viewer.py
            chroma_db_storage/
            images/
                uploaded_images/
                new_folder/
            services/
                ask_llama.py
                delete_image.py
                filtering_query_results.py
                handling_images.py
                ques_category.py
                vectordb.py
        project_venv/ (Virtual environment)
        multimodal_rag.ipynb (Notebook for R&D)
        requirements.txt (Backend dependencies)

    memovision_frontend/
        src/
            components/
                BatchUploader.jsx
                ChatBot.jsx
                Gallery.jsx
                Homepage.jsx
                ImageViewer.jsx
                Navbar.jsx
            assets/
            api.js
            App.jsx
            index.css
            main.jsx
        public/
        node_modules/
        .idea/
        package.json
        index.html
        postcss.config.js
        tailwind.config.js
        vite.config.js
    README.md
```

12 User Interfaces

12.1 Home

The "Home" page describes the system and its features. It also shows the technology used and the reason behind choosing this technology.

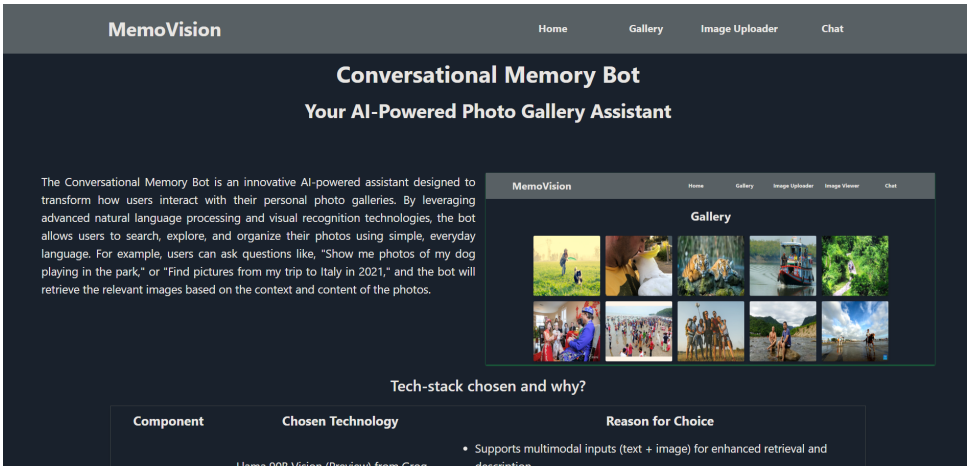

Figure 3: Home page.

12.2 Gallery

The "Gallery" page shows all the uploaded images in smaller form without description generated by the Llama.

Figure 4: Gallery page.

12.3 Image Uploader

Allows the user to upload multiple images at the same time to add to the "Gallery" page and the vector database. For all of the uploaded images, the Llama generates descriptions and tags. Then both text and images are stored in the vector database for retrieval while querying through the "Chat" page.

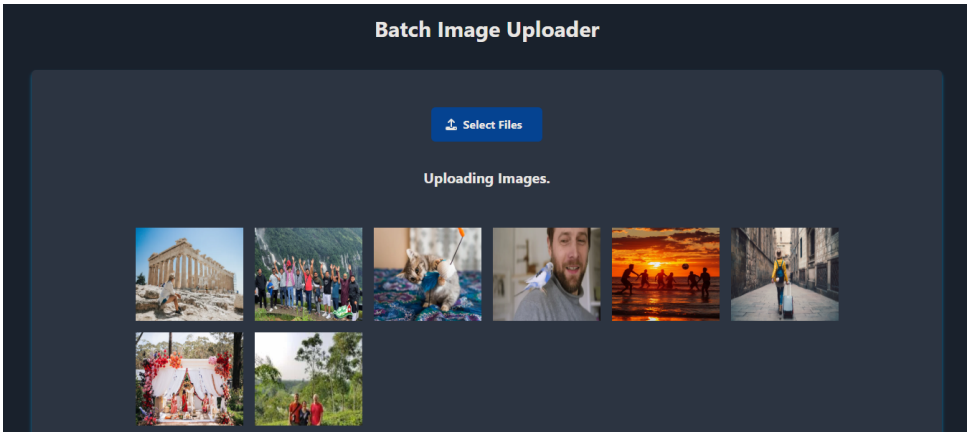

Figure 5: Batch Image Uploader page.

12.4 Chat Page

The "Chat" page allows users to ask for the retrieval of the contexutal image based on natural language query, image query, and multimodal query. It also allows the user to request to describe an image.

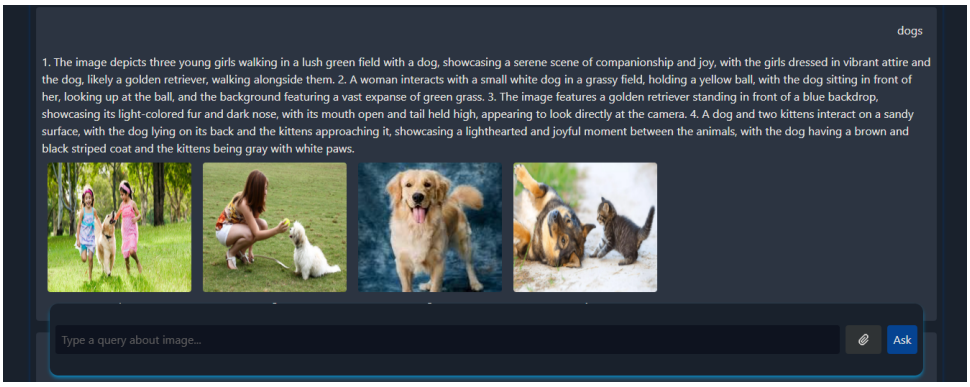

Figure 6: Chat page.

12.5 Image Viewer

View the clicked image with the details description and tags if the user clicked on any image from the "Gallery" page or the retrieved image in the "Chat" page.

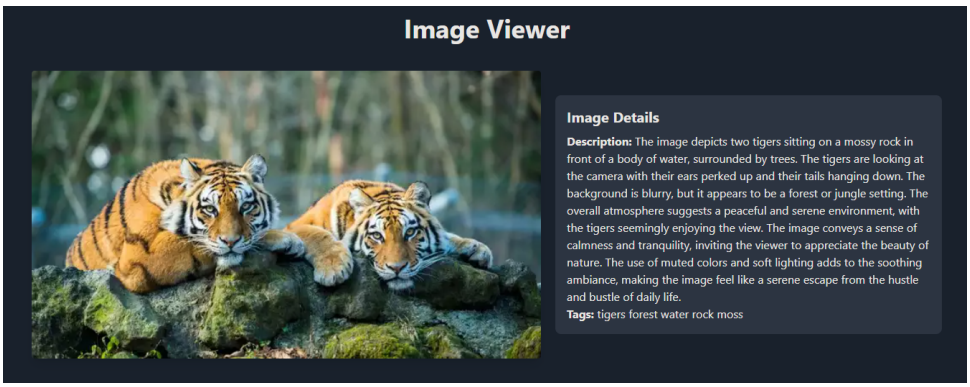

Figure 7: Image Viewer page.