B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

OOMD Mini Project Report

Conversational Image Recognition Chatbot

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

Piyush Sharma--1BM23CS231
Aryan Kumar--1BM23CS362
Vansh Santoki--1BM23CS363
Pranav Easwar--1BM23CS365

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
2025-26

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, Piyush Sharma(1BM21CS231), Aryan Kumar(1BM23CS362),Vansh Santoki(1BM23CS363), Pranav Easwar(1BM23CS365)students of 5th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this OOMD Mini Project entitled "Conversational Image Recognition Chatbot" has been carried out in Department of CSE, B.M.S. College of Engineering, Bangalore during the academic semester August 2025- December 2025. I also declare that to the best of our knowledge and belief, the OOMD Mini Project report is not from part of any other report by any other students.

Signature of the Candidate

Piyush Sharma(1BM21CS231)

Aryan Kumar (1BM23CS362)

Vansh Santoki(1BM23CS363)

Pranav Easwar(1BM23CS365)

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the OOMD Mini Project titled "Conversational Image Recognition Chatbot" has been carried out by Piyush Sharma (1BM23CS231), Aryan Kumar (1BM23CS362),Vansh Santoki(1BM23CS363), Pranav Easwar(1BM23CS365) during the academic year 2025-2026.

Signature of the Faculty in Charge
Dr. Adarsha Sagar H V
Assistant Professor

Table of Contents

Chapter 1: Problem Statement

With the rapid growth of digital content, images have become one of the most common forms of communication, documentation, and information sharing. Users often capture photos for identifying objects, understanding scenes, reading text, or seeking assistance. However, existing image recognition systems are mostly limited to providing short, static outputs, such as labels or basic descriptions, without supporting interactive conversation. They cannot understand user intent, maintain context across queries, or answer follow-up questions related to the same image.

This creates a significant gap between user expectations and the capabilities of current tools. For example, a user may upload an image of a damaged electronic component and ask multiple questions like “What is this part?”, “Why might it be damaged?”, or “How can I fix it?”. Traditional systems fail to support such contextual dialogue. Similarly, students often need detailed explanations from textbook images, diagrams, or handwritten notes, but existing OCR or recognition tools only extract text without reasoning. People with visual impairments also face challenges because many systems provide incomplete or non-conversational descriptions.

The problem becomes more complex due to variations in image quality, lighting conditions, cluttered backgrounds, and diverse contexts that require not just recognition but interpretation and reasoning.

To overcome these limitations, there is a need for a **Conversational Image Recognition Chatbot** that can intelligently analyze images and communicate with users naturally. The chatbot should integrate computer vision, natural language understanding, and context management to deliver accurate answers, detailed explanations, and meaningful multi-turn conversations. This system aims to make image interaction more intuitive, accessible, and useful across domains such as education, daily life assistance, troubleshooting, accessibility support, and information retrieval.

The goal of this project is to develop a **Conversational Image Recognition Chatbot** that integrates image processing, machine learning, and natural language understanding. The system will accept images, analyze them, understand the context, and engage in meaningful conversation with the user. This chatbot aims to provide accurate image interpretation, continuous context-aware dialogue, and a seamless interactive experience that enhances accessibility, learning, and real-world problem-solving.

Chapter 2: Software Requirement Specification

"Conversational Image Recognition Chatbot"

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to define the complete requirements and specifications for the development of the **Conversational Image Recognition Chatbot**. It provides clear details about the system's objectives, scope, features, constraints, and performance expectations to guide the development and implementation process.

1.2 Scope of this Document

This document outlines the functionalities, architecture, and design constraints of the chatbot. It describes how the system will enable users to interact through text and images, how the chatbot will process these inputs using NLP and image recognition, and the cost and time required for development.

1.3 Overview

The Conversational Image Recognition Chatbot is an intelligent system capable of understanding text, analyzing uploaded images, and generating meaningful conversational responses. It combines Natural Language Processing (NLP) with advanced computer vision to provide users with interactive assistance, explanation, and information retrieval.

2. General Description

The chatbot system will support users in sending text or image queries and receiving intelligent responses. It will utilize external vision APIs to process images, NLP engines to interpret text, and a backend database to store chat history, image results, and user details. The system will be accessible through web and mobile interfaces and designed for both technical and non-technical users.

3. Functional Requirements

3.1 Text Processing

- Accept user text messages.
- Analyze text using NLP to determine intent and extract entities.
- Generate context-aware responses.

3.2 Image Processing

- Allow users to upload images in common formats (JPEG, PNG).
- Validate and store uploaded images.

- Use external vision APIs to detect objects, extract text, and classify scenes.

3.3 Response Generation

- Combine image analysis and NLP results for multi-modal responses.
- Provide descriptive, meaningful, and context-based replies.

3.4 Chat Session Management

- Create new chat sessions for each user.
- Track conversation history and maintain context.
- End chat sessions and store final logs.

3.5 User Account Management

- Allow users to register, log in, and view their history.
- Maintain user profiles and preferences.

3.6 Conversation History

- Store text messages, image results, and timestamps.
- Retrieve and display full chat history upon request.

4. Interface Requirements

4.1 User Interface

- Simple, intuitive, and interactive chat interface.
- Support for text input, image upload, and viewing history.
- Accessible via web browsers and mobile devices.

4.2 Integration Interfaces

- Integration with external vision APIs (Google Vision, AWS Rekognition, etc.).
- Connection to NLP engines (OpenAI, Dialogflow, spaCy).
- Database integration for persistent storage.

5. Performance Requirements

5.1 Response Time

- Text responses should be generated within 2 seconds.
- Image recognition responses should be generated within 5 seconds.

5.2 Scalability

- System should support at least 500 concurrent users.
- Efficient handling of growing image and chat data.

5.3 Data Integrity

- Ensure consistent, accurate storage of messages, images, and analysis results.

6. Design Constraints

6.1 Hardware Limitations

- Designed to run on standard web servers and user devices.
- Requires high-speed internet access for image processing.

6.2 Software Dependencies

- Requires a relational or NoSQL database (MySQL, MongoDB).
- Dependent on external APIs for NLP and image recognition.
- Developed using modern frameworks (Node.js, Python, Django, React, etc.).

7. Non-Functional Attributes

7.1 Security

- Implement user authentication and encrypted data transmission.
- Protect images and personal information using secure storage.

7.2 Reliability

- Ensure uptime above 99% with minimal service interruptions.

7.3 Scalability

- Support both system expansion and increased data load.

7.4 Portability

- Compatible with multiple browsers (Chrome, Firefox, Edge).
- Mobile device support through responsive UI.

7.5 Usability

- Provide a simple, clean chat interface understandable by all users.

7.6 Reusability

- Use modular components for NLP, image recognition, and UI to ease upgrades.

7.7 Compatibility

- Compatible with popular external APIs and standard web technologies.

7.8 Data Integrity

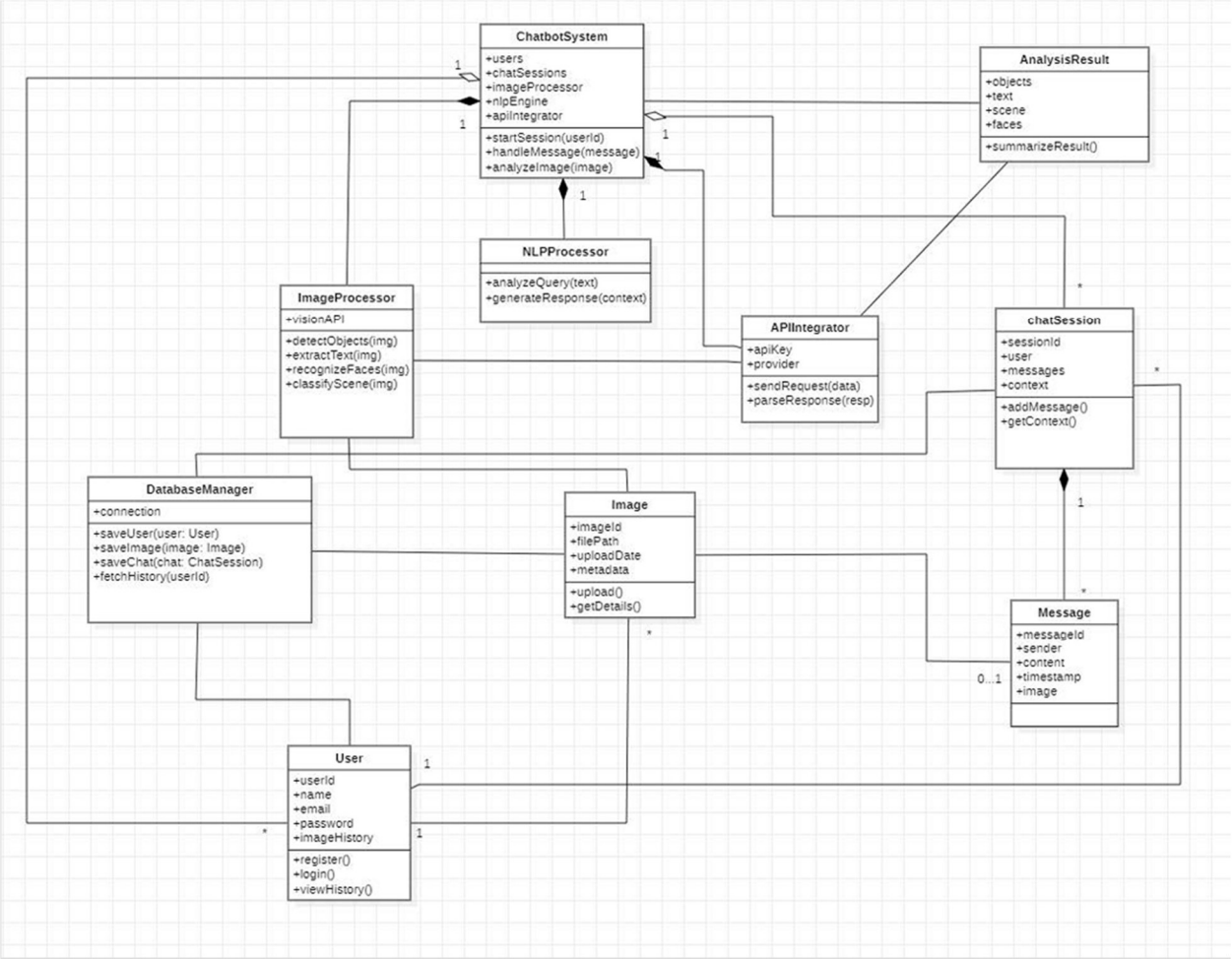
- Ensure consistent and accurate storage, retrieval, and backup of session data.

8. Preliminary Schedule and Budget

The development of the Conversational Image Recognition Chatbot is estimated to take

4–5months with a projected budget of approximately **₹6,19,50,000** (6.195 crore rupees). . This includes requirement analysis, UI/UX design, integration with NLP and vision APIs, backend development, testing, and deployment.

Chapter 3: Class Modeling



1. ChatbotSystem (Main Controller Class)

Attributes:

- users
- chatSessions
- imageProcessor
- nlpEngine
- apiIntegrator

Methods:

- startSession(userId)
- handleMessage(message)
- analyzeImage(image)

Relevance:
This is the **central orchestrator** of the chatbot. It coordinates user interactions, delegates text and image processing tasks, and manages sessions. Every user request passes through this class.

2. User

Attributes:

- userId, name, email, password, imageHistory

Methods:

- register(), login(), viewHistory()

Relevance:
Represents individuals using the chatbot. Maintains login details and stores past images/messages for personalized interactions.

3. ChatSession

Attributes:

- sessionId, user, messages, context

Methods:

- addMessage()
- getContext()

Relevance:
Every conversation occurs inside a ChatSession. It stores messages, maintains context, and ensures multi-turn conversation works smoothly.

4. Message

Attributes:

- messageId, sender, content, timestamp, image

Relevance:
Represents each message sent by the user or chatbot. It may contain text or an image. Important for generating replies and saving chat logs.

5. Image

Attributes:

- imageId, filePath, uploadDate, metadata

Methods:

- upload(), getDetails()

Relevance:
Stores uploaded images and their metadata. Used by vision models to perform detection, OCR, or classification.

6. ImageProcessor

Attributes:

- visionAPI

Methods:

- detectObjects(img)
- extractText(img)
- recognizeFaces(img)
- classifyScene(img)

Relevance:
Handles all computer vision tasks. It converts the raw image into meaningful information (objects, text, scenes, faces) needed for generating a useful response.

7. NLPProcessor

Methods:

- analyzeQuery(text)
- generateResponse(context)

Relevance:
Responsible for understanding user queries and generating appropriate natural-language responses. It ensures conversational flow and context-awareness.

8. APIIntegrator

Attributes:

- apiKey, provider

Methods:

- sendRequest(data)
- parseResponse(resp)

Relevance:
Connects external services like Google Vision, OpenAI Vision, or other ML APIs. It acts as a communication bridge between the chatbot and external AI models.

9. AnalysisResult

Attributes:

- objects, text, scene, faces

Methods:

- summarizeResult()

Relevance:
Stores the processed output of an image. Used to generate meaningful conversational responses by summarizing image details.

10. DatabaseManager

Attributes:

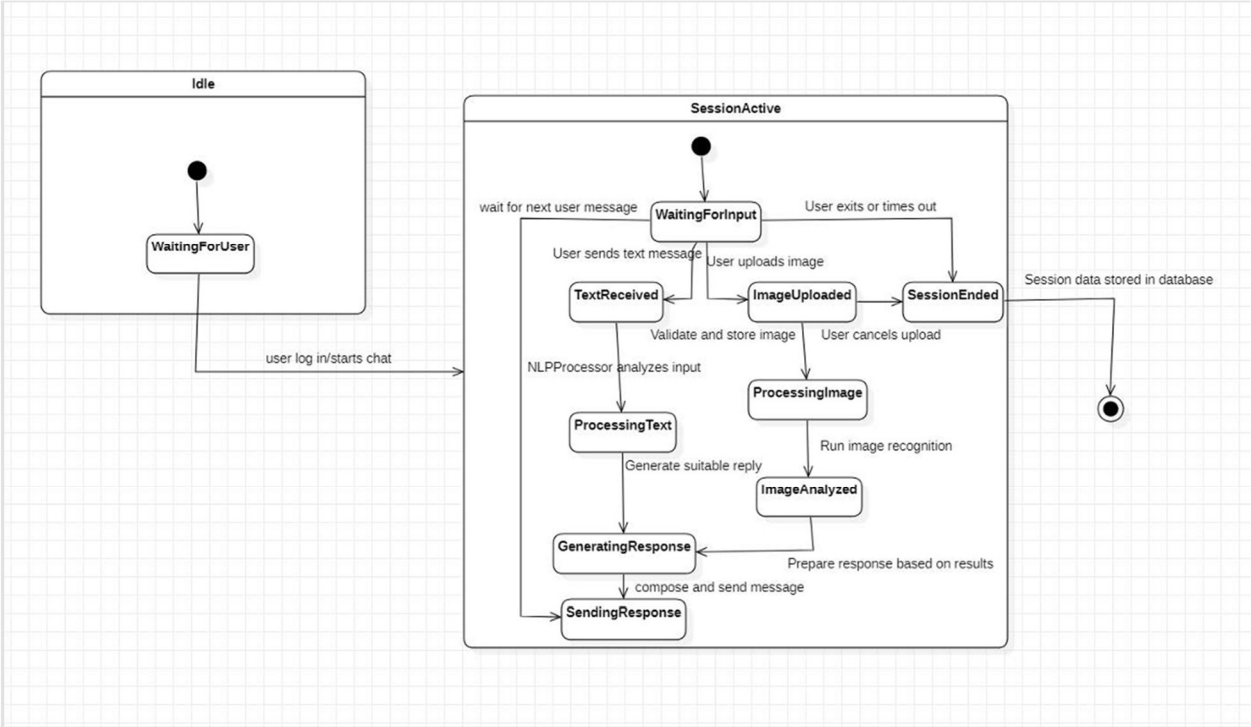
- connection

Methods:

- saveUser()
- saveImage()
- saveChat()
- fetchHistory()

Relevance:
Ensures persistent storage of user data, images, and chat sessions. Helps in generating past history, analytics, and continuity in chat.

Chapter 4: State Modeling



This state diagram describes the **entire interaction lifecycle** of the Conversational Image Recognition Chatbot.

1. The system begins in **Idle**, waiting for a user to start a chat.
2. When the user logs in, the system transitions to **SessionActive**.
3. Inside the active session, the system enters **WaitingForInput**, where it waits for:
    - **text message**, or
    - **image upload**
4. If the user sends **text**, states follow:
    - TextReceived → ProcessingText → GeneratingResponse → SendingResponse → back to WaitingForInput.
5. If the user **uploads an image**:
    - ImageUploaded → ProcessingImage → ImageAnalyzed → GeneratingResponse → SendingResponse → back to WaitingForInput.

6. If the user exits or remains inactive:
    - SessionEnded occurs.
    - The system saves chat history and returns to Idle.

This diagram effectively models a **real-time conversational system** that handles both text and image-based inputs while retaining context across multiple interactions.

“States”

1. Idle

System is inactive and waiting for a user to start the chat.

2. WaitingForUser

System is ready and waiting for a user login or chat request.

3. SessionActive

Represents an ongoing chat session with the user.

4. WaitingForInput

Chatbot is waiting for the user's next message (text or image).

5. TextReceived

User has sent a text message; ready to process it.

6. ProcessingText

NLP engine processes text to understand the user’s intent.

7. ImageUploaded

User uploads an image; system validates and stores it.

8. ProcessingImage

Image recognition engine analyzes the image contents.

9. ImageAnalyzed

Image analysis is complete; results are ready for response.

10. GeneratingResponse

Chatbot forms a reply based on text or image analysis.

11. SendingResponse

Final response is sent to the user and chat continues.

12. SessionEnded

User exits or times out; chat session is closed and saved.

"Events"

1. user login / start chat

Starts a new chat session (Idle → SessionActive).

2. User sends text message

Triggers text-processing flow (WaitingForInput → TextReceived).

3. NLP analyzes text

Generates understanding and moves to response generation (ProcessingText → GeneratingResponse).

4. User uploads image

Triggers image-processing workflow (WaitingForInput → ImageUploaded).

5. Validate & store image

Moves to detailed image analysis (ImageUploaded → ProcessingImage).

6. Image recognition completed

System has extracted objects/text/scenes (ProcessingImage → ImageAnalyzed).

7. Generate reply

Chatbot creates a meaningful response (ImageAnalyzed → GeneratingResponse).

8. Send message

Chatbot delivers the final output (GeneratingResponse → SendingResponse).

9. User exits / timeout

Ends the session (WaitingForInput → SessionEnded).

10. Save session data: Stores chat and image history in database before returning to Idle.

Chapter 5: Interaction Modeling

"USE CASE Diagram"-- ACTORS

1. User

The primary actor who interacts with the chatbot by sending text, uploading images, receiving responses, and viewing chat history.

2. External Vision API

A third-party service used to analyze uploaded images and return recognition results.

3. NLP Engine

Processes and understands the user's text messages to determine intent and generate meaningful responses.

4. System Administrator

Manages system settings, user accounts, AI model configurations, and database maintenance.

"USE CASE"

1. Start chat session

Allows the user to begin a new conversation with the chatbot.

2. Send text message

User sends text input for analysis.

3. Upload image

User provides an image for recognition.

4. Process text message

NLP engine analyzes the meaning and intent of the user's text.

5. Process image

Prepares and sends the uploaded image for recognition.

6. Run image recognition

External Vision API identifies objects, text, or scenes from the image.

7. Analyze text

NLP engine interprets the user's message.

8. Generate response

Creates a suitable reply using text or image analysis.

9. Receive response

User receives the chatbot's reply.

10. View chat history

Displays previous interactions saved in the system.

11. Store session data

Saves messages, images, and conversation history.

12. End chat session

Closes the ongoing conversation.

13. Manage user account (Admin)

Admin updates or controls user information.

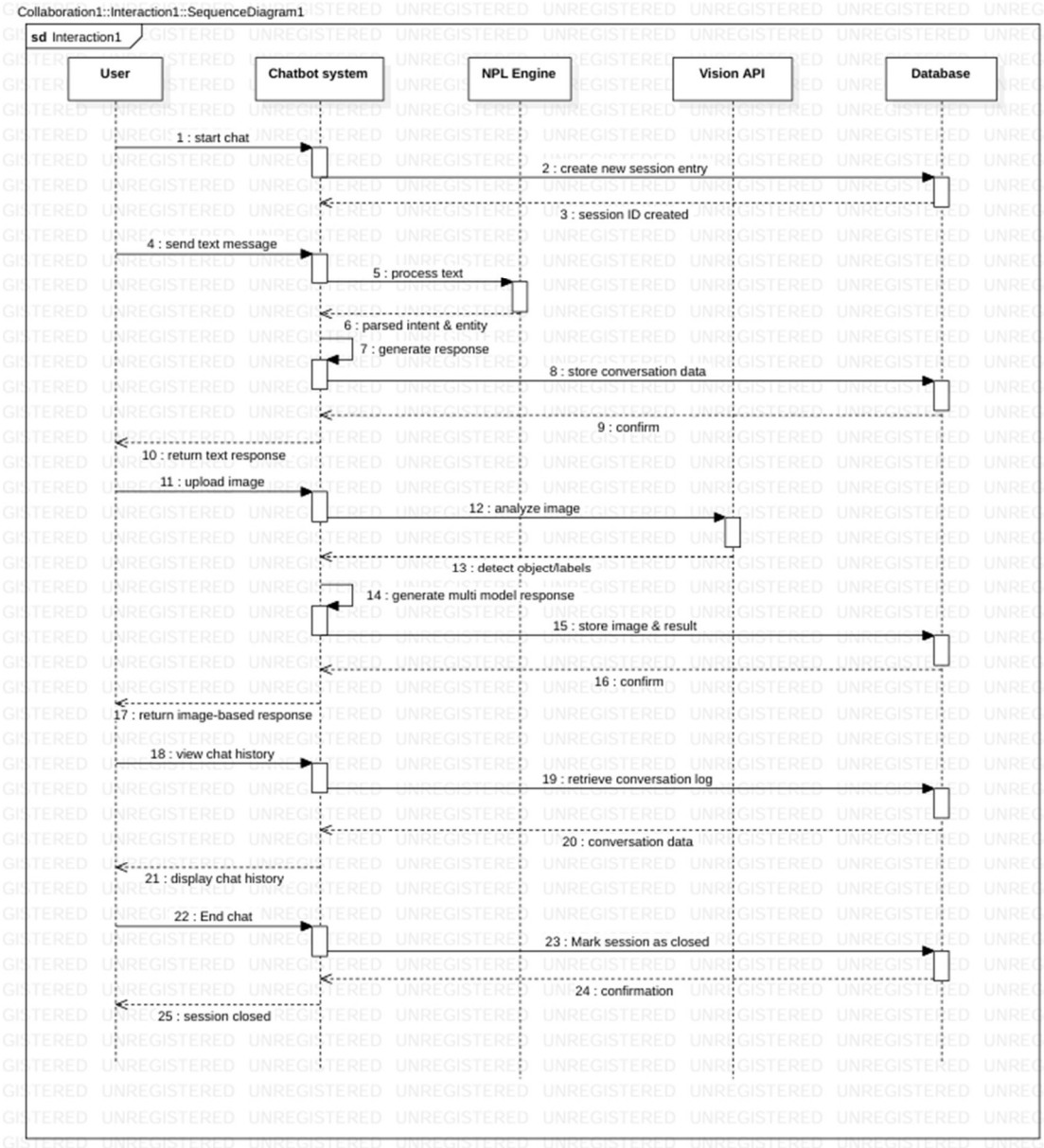
14. Backup database (Admin)

Admin performs data backups for safety.

15. Configure AI models (Admin)

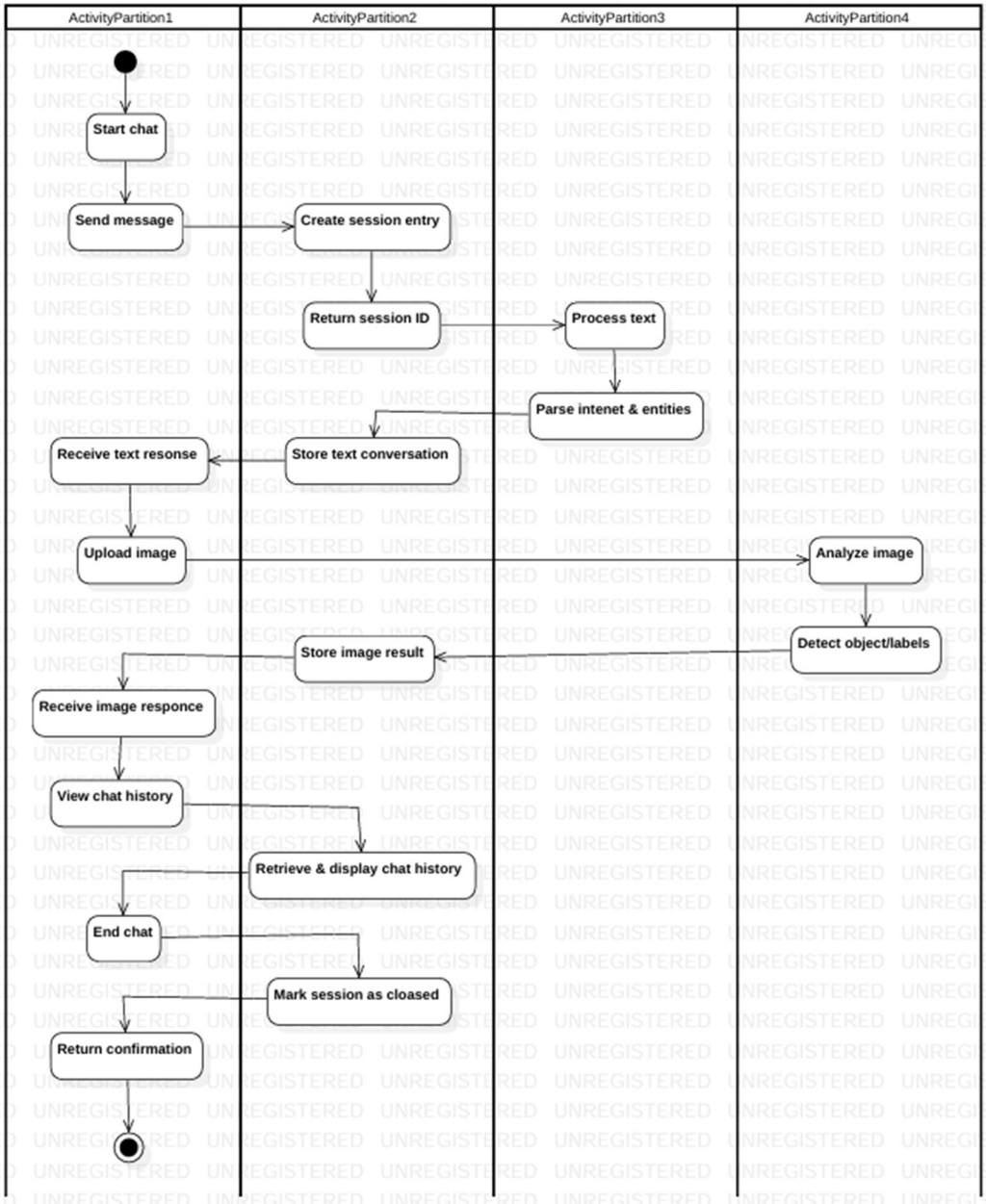
Admin updates AI settings and API configurations.

Sequence Diagram



The sequence diagram shows how the user interacts with the conversational image recognition chatbot. When a user starts a chat, the chatbot system creates a new conversation session by communicating with the database. During the conversation, if the user sends a text message, the chatbot forwards it to the NLP Engine, which analyzes the message, extracts intent, and generates a suitable response. The chatbot then stores this interaction in the database and sends the response back to the user.If the user uploads an image, the chatbot sends it to the Vision API, which performs image recognition by detecting objects, text, or scenes. The chatbot combines the vision results with NLP understanding to create a multi-modal response and saves the image analysis details in the database before replying to the user. Users can also request to view their chat history; in that case, the chatbot retrieves the stored conversation from the database and displays it back to them. When the user ends the chat, the chatbot updates the session status as closed in the database and confirms the closure to the user.

Activity Diagram

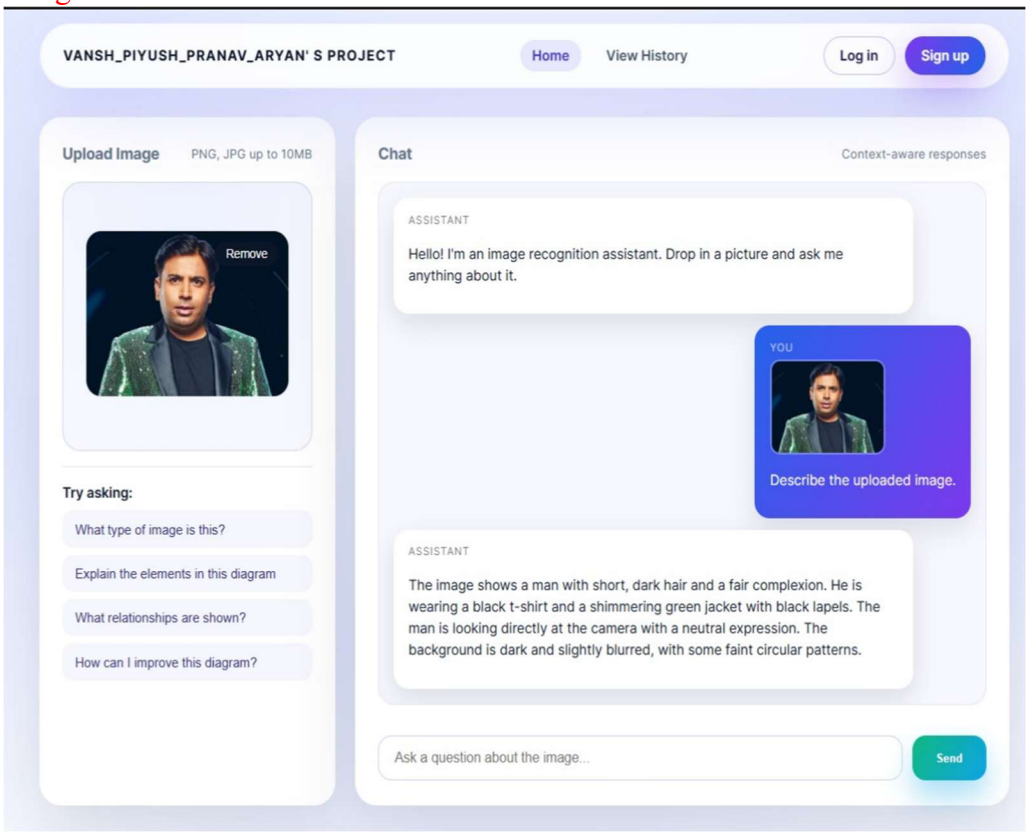


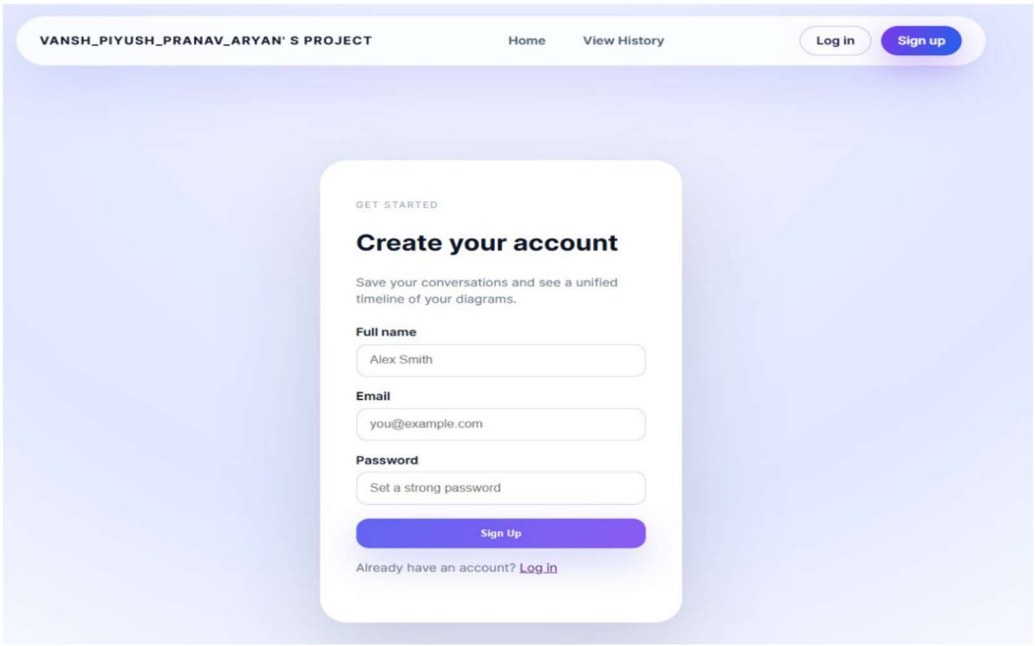
- Swimlanes: The activity diagram illustrates the complete workflow of the *Conversational Image Recognition Chatbot System* and is divided into **four swimlanes**, each representing a different component: the **User**, the **Chatbot System**, the **NLP Engine**, and the **Vision API**. These swimlanes help separate responsibilities clearly. The User swimlane handles actions such as starting the chat, sending messages, uploading images, viewing history, and ending the session. The Chatbot System swimlane manages tasks like creating the session, storing text/image results, retrieving history, and marking the session as closed.

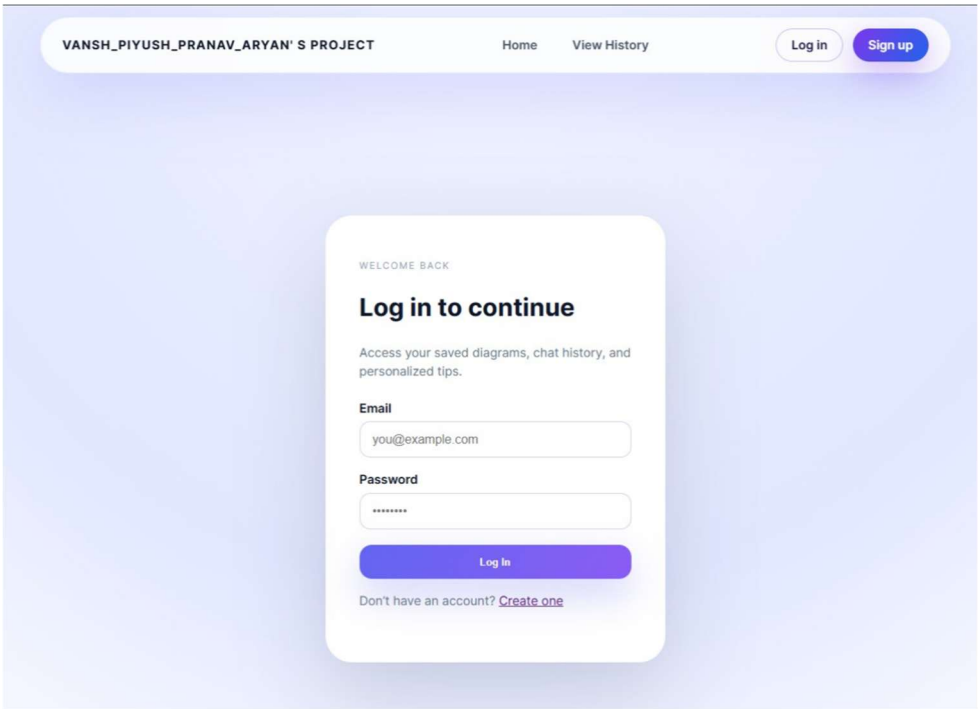
The NLP Engine swimlane is responsible only for processing text and extracting intent, while the Vision API swimlane handles image analysis and object detection.

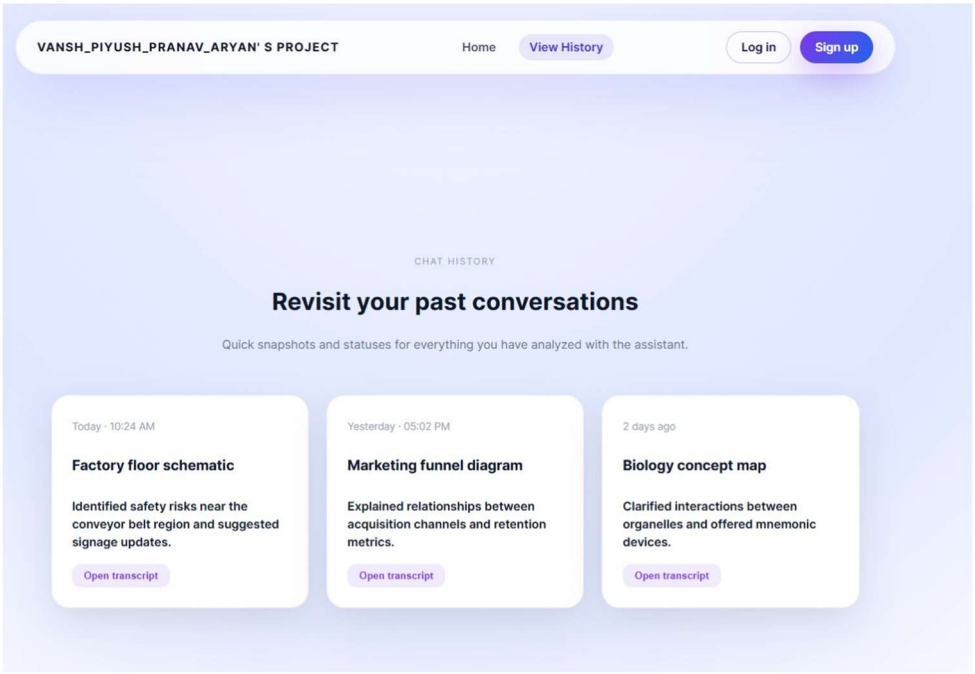
- Splitting and merging of control: The diagram also shows **splitting and merging of control flow**. When the user sends a text message, the control flows from the Chatbot System to the NLP Engine for processing, and then merges back into the Chatbot System for storing data and sending a response. Similarly, when the user uploads an image, the flow branches to the Vision API for image analysis and detection before merging back to the Chatbot System to save the result and respond to the user. These splits represent delegation of tasks to specialized components, while the merges represent the return of control to continue the main workflow. The diagram ends with merging all branches into the final activity where confirmation is returned to the user
- Explain your diagram: The activity diagram is divided into four swimlanes—User, Chatbot System, NLP Engine, and Vision API—each representing the responsibilities of a specific component in the workflow. The User swimlane includes actions such as starting the chat, sending messages, uploading images, viewing history, and ending the session. The Chatbot System swimlane handles session creation, storing text and image results, retrieving conversation history, and generating responses. The NLP Engine swimlane is responsible for processing text and extracting user intent, while the Vision API swimlane performs image analysis and object detection. The diagram also shows splitting of control, where the chatbot delegates text processing to the NLP Engine or image analysis to the Vision API. After these tasks are completed, the control flow merges back into the Chatbot System to store the results and send the appropriate response. This splitting and merging illustrate how different system components work together while maintaining a clear and coordinated flow of activities.

UI Design:


VANSH_PIYUSH_PRANAV_ARYAN' S PROJECT
Home
View History
Log in
Sign up
Upload Image
PNG, JPG up to 10MB
Remove
Try asking:
What type of image is this?
Explain the elements in this diagram
What relationships are shown?
How can I improve this diagram?
Chat
Context-aware responses
ASSISTANT
Hello! I'm an image recognition assistant. Drop in a picture and ask me anything about it.
YOU
Describe the uploaded image.
ASSISTANT
The image shows a man with short, dark hair and a fair complexion. He is wearing a black t-shirt and a shimmering green jacket with black lapels. The man is looking directly at the camera with a neutral expression. The background is dark and slightly blurred, with some faint circular patterns.
Ask a question about the image...
Send


VANSH_PIYUSH_PRANAV_ARYAN' S PROJECT
Home
View History
Log in
Sign up
GET STARTED
Create your account
Save your conversations and see a unified timeline of your diagrams.
Full name
Alex Smith
Email
you@example.com
Password
Set a strong password
Sign Up
Already have an account? Log in





This is our image-recognition chatbot interface, where users can upload an image and ask questions about it. The system analyses the picture and generates context-aware responses in real time. It highlights our clean UI, smooth workflow, and accurate image description capability.