Vision-Based Models for Crop Disease and Weed Detection with a Generative AI-Driven RAG System for Farmer Support

Rahat Bhatia

Yonas Seyoum

Cody Large

Eastern Washington University

Table Of Contents

Project Description

This project aims to develop an AI-powered system that combines advanced vision-based models, such as CLIP (Contrastive Language-Image Pretraining), with traditional computer vision techniques to accurately detect and diagnose crop diseases and identify invasive weeds. By integrating both deep learning and classical methods, the system will provide a robust solution to agricultural challenges, especially in regions with limited access to agricultural expertise.

In addition, the project will implement a Retrieval-Augmented Generation (RAG) system to assist farmers with contextually aware, intelligent responses regarding agricultural practices. This will include personalized guidance on crop disease remedies, weed control strategies, and relevant agricultural policies. The RAG system will leverage a vast database of agricultural knowledge, ensuring that responses are both up-to-date and relevant to local contexts.

The final output will be a fully functional, web-based platform, offering farmers state-of-the-art tools for real-time crop analysis and tailored agricultural support. This platform will empower farmers by providing easy-to-understand insights, improving productivity, sustainability, and decision-making in managing crop health and weed management.

User Stories

U1: As a farmer, I want to log into the website so that I can access the chatbot and view my past chat history.

Feature: Farmer Login

Scenario 1: Successful login with valid credentials

- Given I am on the login page
- When I enter my email "farmer@example.com" and password "password123"
- And I click the "Login" button
- Then I should be redirected to my personalized home screen
- And I should see a welcome message "Welcome, Farmer!"

Scenario 2: Unsuccessful login with invalid credentials

- Given I am on the login page
- When I enter my email "farmer@example.com" and an incorrect password "wrongpassword"
- And I click the "Login" button
- Then I should see an error message "Invalid email or password"
- And I should remain on the login page

U2: As a farmer, I want to sign up for the website so that I can access the chatbot.

Feature: Farmer Sign Up

Scenario 1: Successful sign up with email

- Given I am on the sign-up page

- When I enter my email "farmer@example.com" and password "password123"
- And I click the "Sign Up" button
- Then my account should be created
- And I should be redirected to my personalized home screen
- And I should see a welcome message "Welcome, Farmer!"

Scenario 2: Sign up using Google

- Given I am on the sign-up page
- When I click "Sign in with Google"
- And I authenticate via my Google account
- Then my account should be created
- And I should be redirected to my personalized home screen
- And I should see a welcome message "Welcome, Farmer!"

U3: As a farmer, I want to upload image for Disease and Weed Detection

Feature: Crop Disease/Weed Identification

Scenario 1: Successful image upload and disease detection

- Given I am on the crop disease detection page
- When I upload an image of my crops
- And I click the "Analyze" button
- Then the system should analyze the image
- And it should display a report showing any detected diseases or weeds
- And it should display recommended actions based on the detection

Scenario 2: Upload fails due to incorrect file type

- Given I am on the crop disease detection page
- When I upload a file that is not an image (e.g., "document.pdf")
- And I click the "Analyze" button

- Then I should see an error message "Please upload a valid image file"
- And the image upload should fail

U4: As a farmer, I want to get information about crop health based on text input.

Feature: Querying for Crop Health Information

Scenario 1: Query without an image

- Given I am on the crop health query page
- When I type a question about general crop health
- And I click the "Submit" button
- Then the system should display relevant information based on my query

U5: As a farmer, I want to get information about crop health based on image input.

Feature: Querying for Crop Health Information

Scenario 1: Query with an image

- Given I have uploaded an image for disease detection
- When I ask a question related to the detected disease
- And I click the "Submit" button
- Then the system should provide relevant information about the disease

U6: As a farmer, I want to view my previous chat sessions so that I can continue the conversation with updated crop information.

Feature: Viewing Previous Sessions

Scenario 1: Successful session retrieval

- Given I am logged in
- And I have queried for information in a previous session
- When I navigate to the "Chat History" tab
- Then I should see a list of my past chat sessions

Scenario 2: Continue a previous session with new data

- Given I am viewing a previous chat session
- When I upload a new image of my crops
- And I click the "Submit" button
- Then the system should analyze the new image and continue the conversation based on updated information

U7: As a farmer, I want to submit feedback on my experience with the system so that the system can be improved based on my experience.

Feature: Feedback Submission

Scenario 1: Successful feedback submission

- Given I have completed a session
- When I fill out the feedback form
- And I click the "Submit" button
- Then my feedback should be submitted
- And I should see a confirmation message "Thank you for your feedback!"

Scenario 2: Feedback submission fails due to incomplete form

- Given I have completed a session
- When I fill out the feedback form but leave the comments field empty
- And I click the "Submit" button
- Then I should see an error message "Please provide feedback before submitting"
- And the feedback should not be submitted

Functional Requirements

User Management

Functional Requirements	Description/Specification
[FR-1 (U2.1)] User Registration and Authentication	The system must allow farmers to sign up and log in using their email or a Google account.
Priority	Level 0 (Essential)
[FR-2 (U2.2)] User Login	The farmer must have an easy way to login when re-visiting the website
Priority	Level 0 (Essential)

System use with image

Functional Requirements	Description/Specification
[FR-3 (U2.3)] Upload image of crops to be analyzed	The farmer should be able to easily upload/take live photos of crops to be analyzed by the system which will provide relevant information based on the photo.
Priority	Level 0 (Essential)
[FR-4 (U2.5)] Prompt the chatbot based on an image	The farmer should be able to prompt the LLM for any information relevant to the crops shown in the image.
Priority	Level 0 (Essential)

System use without image

Functional Requirements	Description/Specification
[FR-5 (U2.4)] Query Chatbot with text only input	The farmer should be able to prompt our chatbot about basic crop information without providing an image for context.
Priority	Level 0 (Essential)

User history

Functional Requirements	Description/Specification
[FR-6 (U2.6)] View chat history	Farmer wants to view a past chat and provide updated context on the situation to get more personalized information
Priority	Level 1 (High Priority)

System feedback

Functional Requirements	Description/Specification
[FR-7 (U2.7)] Submit Feedback	The farmer should have a way to provide feedback on the system's accuracy and their overall experience.
Priority	Level 1 (High Priority)

Use Case Diagram and Use Cases

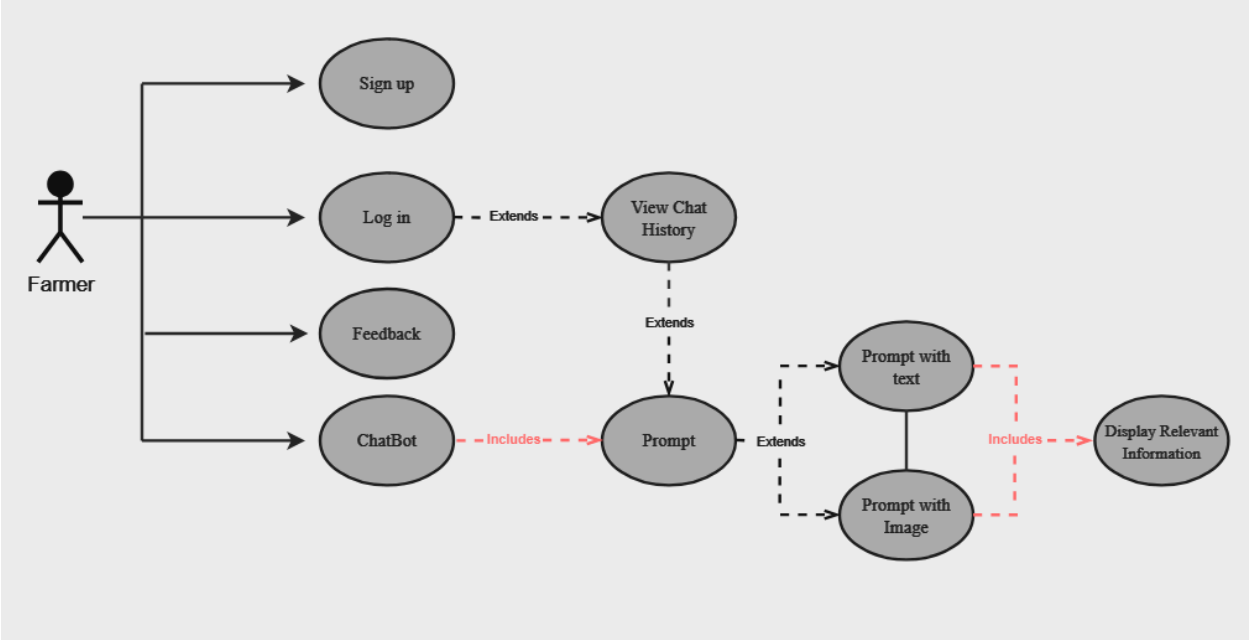

Figure 1: Use Case Diagram

Use Cases:

Use Case 1: Farmer Registration

Actors: Farmer
Preconditions: Farmer is on the sign-up page.
Postconditions: Farmer account is created.
Main Flow:

1. Farmer enters email and password (or uses Google account).
2. Clicks "Sign Up".
3. System creates the account and redirects to the home screen.
4. Displays welcome message.

Alternative Flow:

If the email is already registered:

1. Prompt the farmer to log in instead.

2. Redirect to the login page.

Related Requirements: FR-1

Use Case 2: Farmer Login

Actors: Farmer
Preconditions: Farmer has an existing account on the login page.
Postconditions: Farmer is logged in and redirected to the home screen.
Main Flow:

1. Farmer enters valid email and password.
2. Clicks "Login".
3. System authenticates and redirects to the home screen.
4. Displays welcome message.

Alternative Flow:

If the credentials are invalid:

1. Offer a "Forgot Password?" link.
2. Redirect to the password recovery page.

Related Requirements: FR-2

Use Case 3: Upload Image of Crops for Analysis

Actors: Farmer
Preconditions: Farmer is logged in on the disease detection page.
Postconditions: System analyzes the image and provides a report.
Main Flow:

1. Farmer uploads an image of crops.
2. Clicks "Analyze".
3. System analyzes the image.
4. Displays report with detected diseases and recommendations.

Alternative Flow:

If the uploaded file is not a valid image:

1. Give “File not supported” error
2. Prompt the farmer to choose a different file from specified formats

Related Requirements: FR-3

Use Case 4: Prompt chatbot based on an image input

Actors: Farmer
Preconditions: Farmer has uploaded an image for analysis.
Postconditions: Farmer receives information based on the image.
Main Flow:

1. Farmer uploads a relevant image.
2. Types a question about the detected issues.
3. Clicks "Submit".
4. System processes the image and question.
5. Displays relevant information and recommendations.

Alternative Flow:

If the image does not match any known issues:

1. Suggest uploading a different image or giving more information.
2. Provide links to online resources for reference on related crop diseases

Related Requirements: FR-4

Use Case 5: Query chatbot with text only input

Actors: Farmer
Preconditions: Farmer is logged in on the crop health query page.
Postconditions: Farmer receives relevant information based on the query.
Main Flow:

1. Farmer types a question about crop health.
2. Clicks "Submit".
3. System processes the query.
4. Displays relevant information.

Alternative Flow:

If the query cannot be processed,

- Display an error message saying “Error Processing Query”
- Prompt the farmer to enter a new question

Related Requirements: FR-5

Use Case 6: View Chat History

Actors: Farmer
Preconditions: Farmer is logged in and has previous chat sessions.
Postconditions: Farmer views a list of past chat sessions.
Main Flow:

1. Farmer navigates to the "Chat History" tab.
2. System retrieves past chat sessions.
3. Displays a list of previous sessions.

Alternative Flow:

If no previous sessions exist:

1. Suggest starting a new chat session.
2. Redirect to Prompt Page

Related Requirements: FR-6

Use Case 7: Submit Feedback

Actors: Farmer
Preconditions: Farmer has completed a session and is on the feedback form.
Postconditions: Feedback is submitted or an error message is shown.
Main Flow:

1. Farmer fills out the feedback form.
2. Clicks "Submit".
3. System submits the feedback.
4. Displays confirmation message.

Alternative Flow:

If the form is incomplete:

1. Highlight the missing fields.
2. Ask the farmer to complete those fields before submitting.

Related Requirements: FR-7

Non-functional Requirements

Non Functional requirements	Description
[NFR-1] Cross-Device Compatibility	The system must provide a responsive and intuitive user interface that functions seamlessly across all common devices, including desktops, smartphones, and tablets, ensuring consistent user experience.
[NFR-2] Performance	The system should be able to detect crop disease/weeds and respond to user queries within a time span of no more than 5 seconds.
[NFR-3] Accuracy of Detection	The system must achieve at least 90% accuracy in detecting crop diseases and weeds based on the images analyzed, ensuring that farmers receive reliable and actionable information.
[NFR-4] Camera Integration	The system must allow users to capture and upload images directly from their device's camera, ensuring a smooth workflow for disease and weed detection.
[NFR-4] Multilingual Accessibility	The application must support multiple languages, allowing users to select their preferred language for a more inclusive experience.
[NFR-6] System Reliability	The system must achieve a minimum uptime of 99%, ensuring consistent availability for users to access services without significant interruptions.
[NFR-7] Data Security	The system must implement industry-standard security measures, including data encryption and secure authentication processes, to protect user data and privacy.
[NFR-8] Maintainability	The system must allow easy maintenance and updates with minimal downtime, supported by comprehensive documentation.

Traceability Matrix

Functional Requirement	Use Case	User Story	Priority
FR-1: User Registration and Authentication	UC-1: Farmer Registration	U2: As a farmer, I want to sign up for the website so that I can access the chatbot	Level 0 (Essential)
FR-2: User Login	UC-2: Farmer Login	U1: As a farmer, I want to log into the website so that I can access the chatbot and my chat history	Level 0 (Essential)
[FR-3] Upload image of crops to be analyzed	UC-3: Upload Image for Analysis	U3: As a farmer, I want to upload an image for disease and weed detection.	Level 0 (Essential)
[FR-4] Prompt the chatbot based on an image	UC-4: Prompt Chatbot Based on an Image Input	U5: As a farmer, I want to get information about crop health based on image input.	Level 0 (Essential)
[FR-5] Query Chatbot with text only input	UC-5: Query Chatbot with Text Only Input	U4: As a farmer, I want to get information about crop health based on text input.	Level 0 (Essential)
[FR-6] View chat history	UC-6: View Chat History	U6: As a farmer, I want to view my previous chat sessions to continue the conversation.	Level 1 (High Priority)
[FR-7] Submit Feedback	UC-7: Submit Feedback	U7: As a farmer, I want to submit feedback on my experience with the system so that the system can be improved based on my experience	Level 1 (High Priority)

Evaluation of Existing Solution

- Vision-Based Detection: Current implementations often rely on models like YOLO and Faster R-CNN, which are effective but may lack adaptability and may not generalize well to diverse crop conditions. Additionally, these models require substantial training on agricultural datasets, which can limit their applicability in real-world, data-scarce environments.

- RAG Systems for Agricultural Queries: Retrieval-Augmented Generation systems exist for general knowledge bases but are rarely tailored to agriculture-specific content. They may lack contextual accuracy in areas like disease management or policy-related responses, limiting their usefulness to farmers.

- Client Platform Usability: Many existing agricultural tech platforms are not optimized for low-resource settings, and some lack mobile-friendly interfaces. This limits accessibility for farmers who may rely primarily on mobile devices.

Individual Contributions

Rahat: I coordinated meetings with our sponsor, contributed to class assignments and presentations, completed tasks assigned by the sponsor, and documented all work to date, including records of sponsor meetings.

Yonas: I attended meetings, contributed to class assignments, and completed the required research provided by the sponsor.

Cody: Contributed to all required documents and presentations, attended meeting with sponsor, completed all research requested by sponsor

System Architecture Design

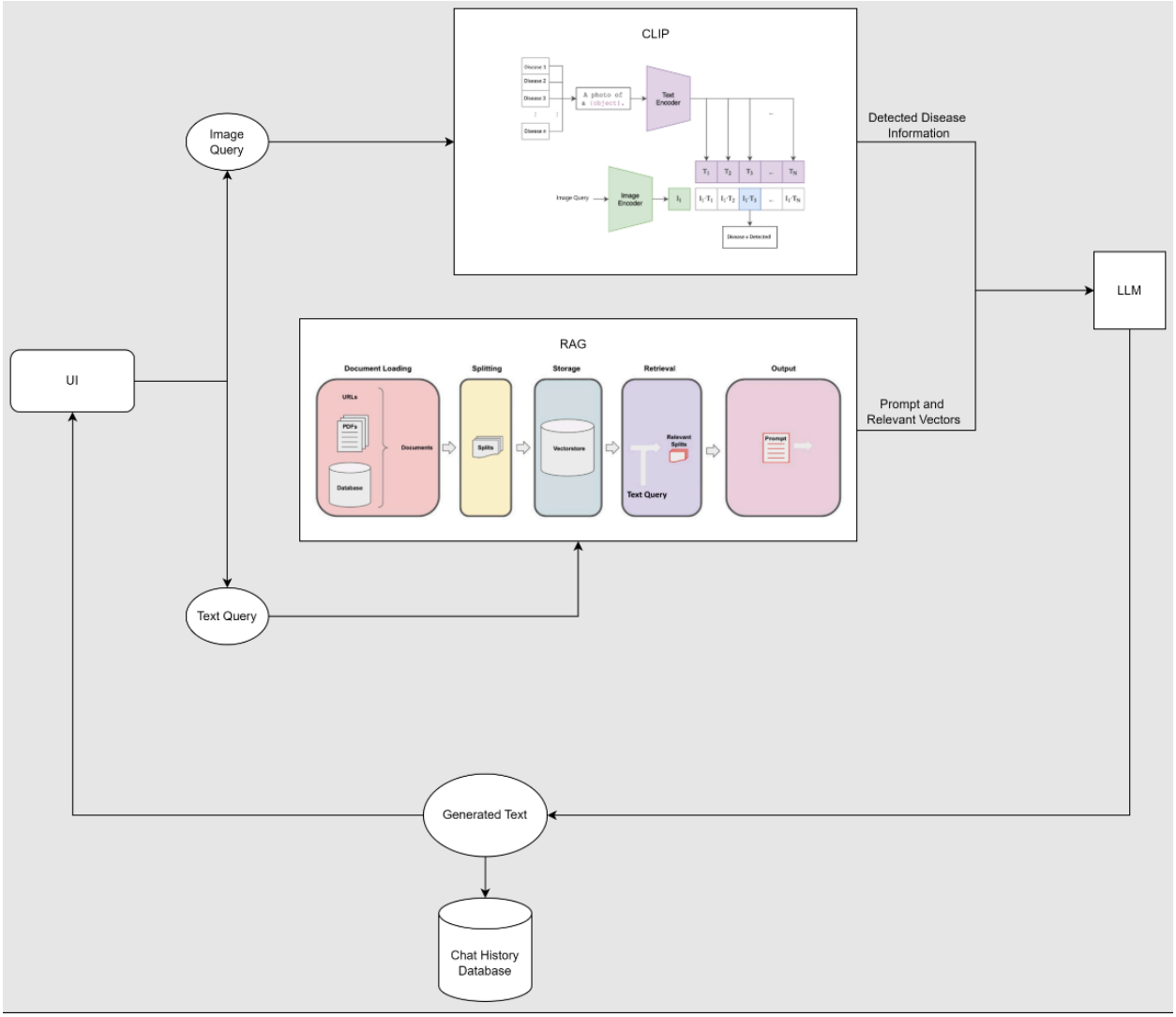

Figure 2: System Architecture

This architecture diagram represents an AI platform for handling image and text queries, focusing on agricultural disease detection and information retrieval.

1. **User Interface (UI)**: Allows users to submit Image Queries for plant disease identification or Text Queries for information retrieval.

2. **CLIP Module**: Processes image queries by encoding them and matching with disease descriptions in the database. This zero-shot classification identifies diseases visually without specific training. The detected disease information is sent to the LLM.

3. **RAG (Retrieval-Augmented Generation) Module**: Manages text queries through:

    - **Document Loading**: Pulls relevant documents (e.g., URLs, PDFs, databases).
    - **Splitting**: Segments large documents.
    - **Storage**: Converts segments into embeddings for efficient retrieval.
    - **Retrieval**: Fetches relevant segments based on the text query.
    - **Output**: Sends retrieved text segments and a prompt to the LLM.

4. **LLM (Large Language Model)**: Uses disease info from CLIP or text vectors from RAG to generate detailed responses.

5. **Generated Text & Chat History Database**: Stores generated responses and maintains chat history for context in future interactions.

Flow Summary:

- **Image Query Path**: UI ➔ CLIP ➔ LLM ➔ Generated Text ➔ Chat History Database ➔ UI
- **Text Query Path**: UI ➔ RAG ➔ LLM ➔ Generated Text ➔ Chat History Database ➔ UI

Activity Diagram

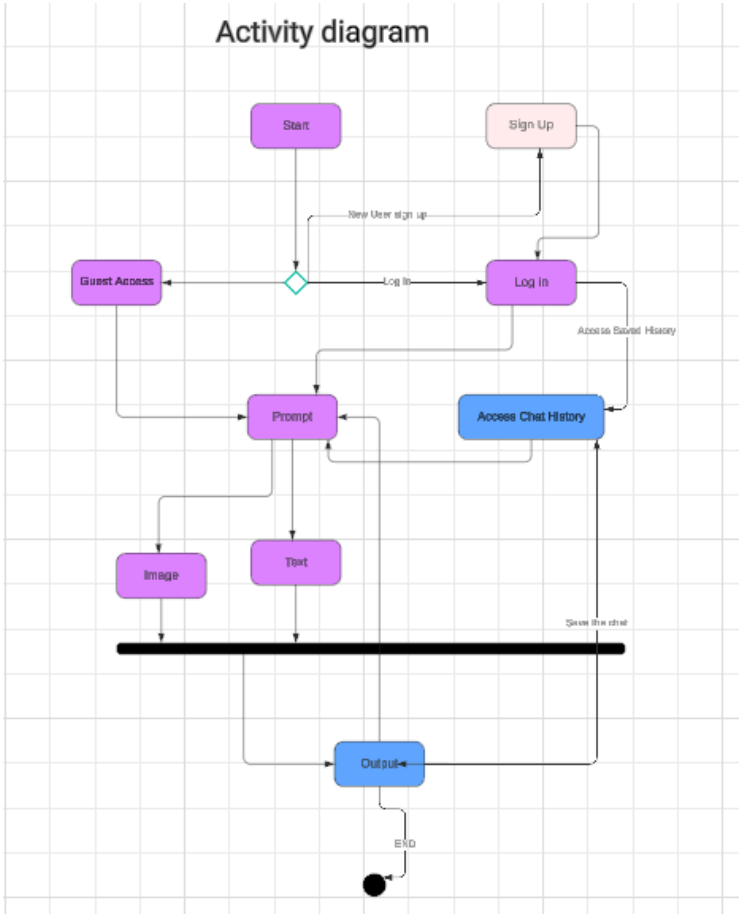

Figure 3: Activity Diagram

Start:

- The process begins when a user engages with the system.

Guest Access / Login / Sign Up:

- Users choose between:
  - Guest Access.
  - Logging in for personalized features.
  - Signing up if they are new.

Prompt:

- Users input either an image or text.

Access Chat History (for logged-in users):

- Users can retrieve previous chat history for context.

Output:

- The system provides a response based on the input.

Save the Chat:

- Saves the session for logged-in users.

End:

- The workflow concludes or loops for further input.

Class Diagram

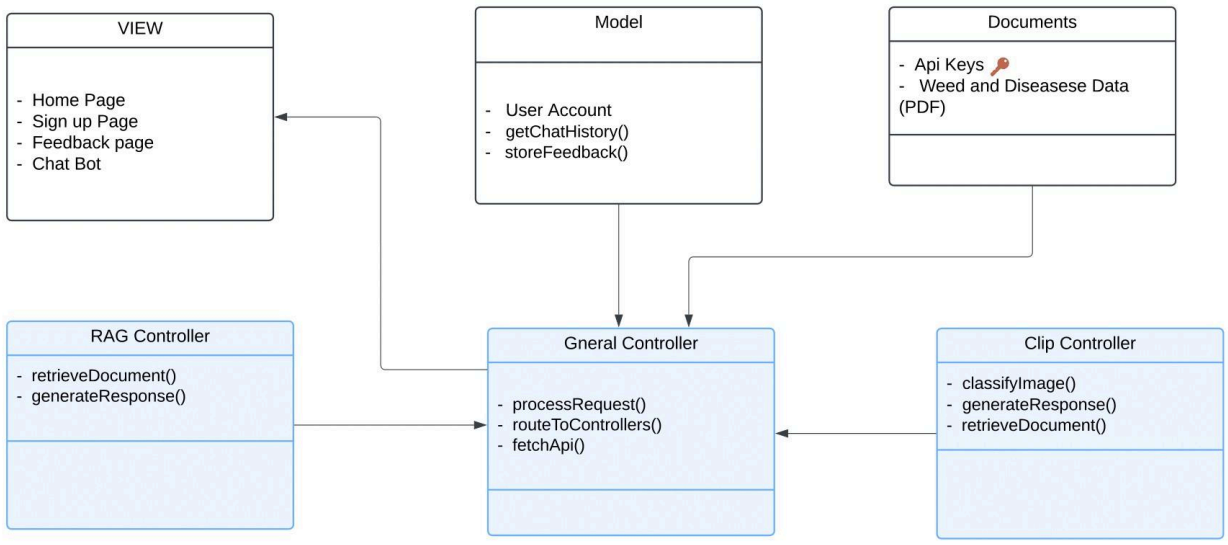

Figure 4: Class Diagram

View:

- Represents the user interface components.
- Includes pages like Home Page, Sign Up Page, Feedback Page, and Chat Bot.
- Allows users to interact with the system by providing input or retrieving information.

Model:

- Represents the user-related data and operations.
- User Account: Manages user data, chat history, and feedback.

Documents:

- Stores external resources and data.

RAG Controller:

- Handles the retrieval and generation of responses using the RAG (Retrieval-Augmented Generation) system.

General Controller:

- Acts as the central coordinator.

Clip Controller:

- Responsible for image classification and document retrieval using CLIP

UI Design

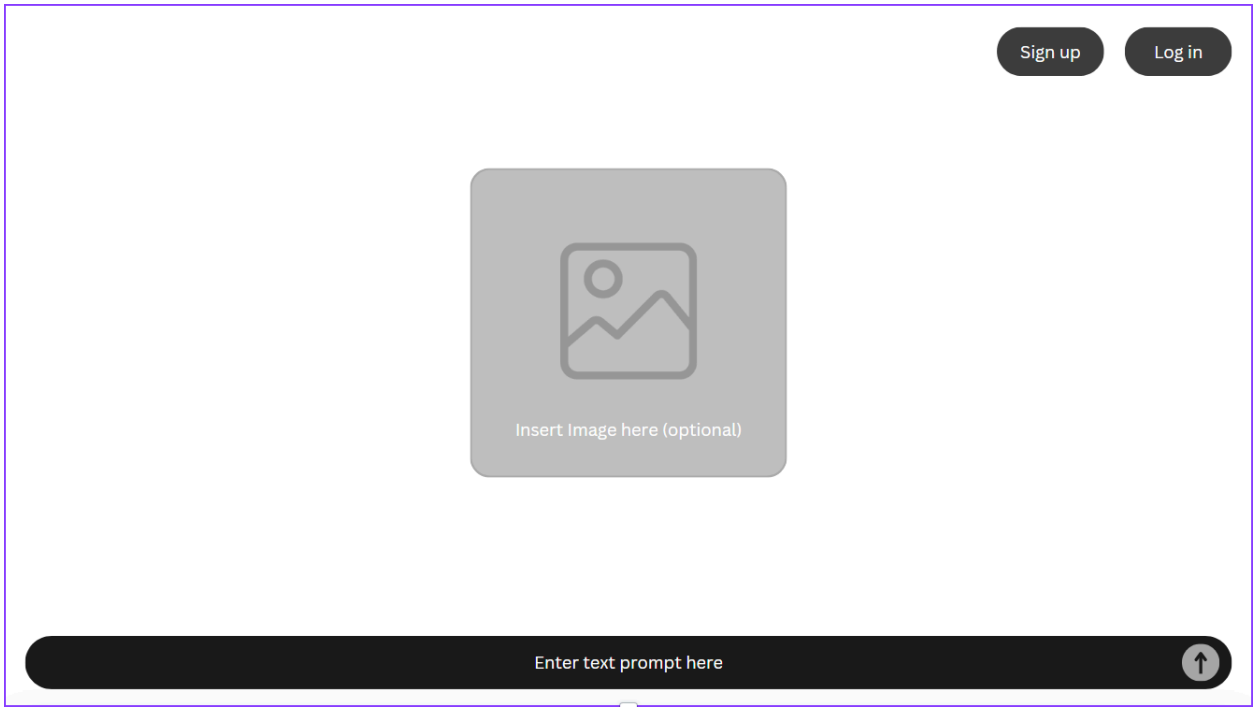

Figure 5.1: Landing page UI

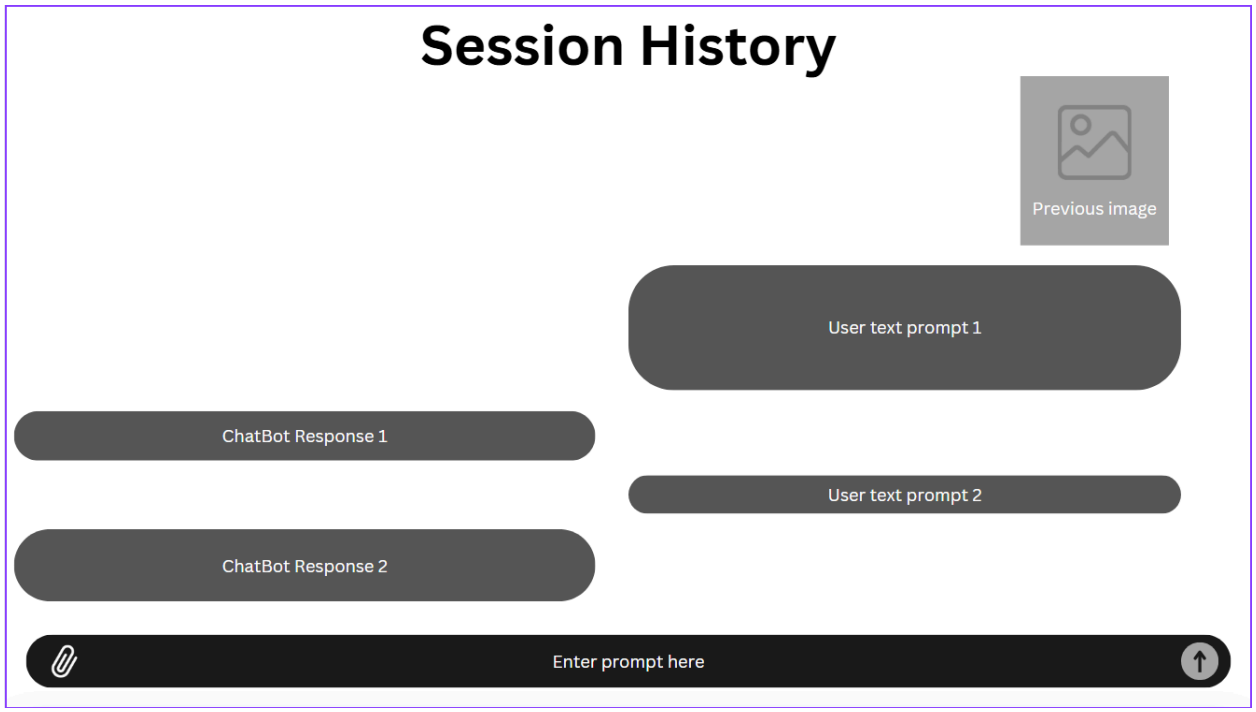

Figure 5.2: Session History UI

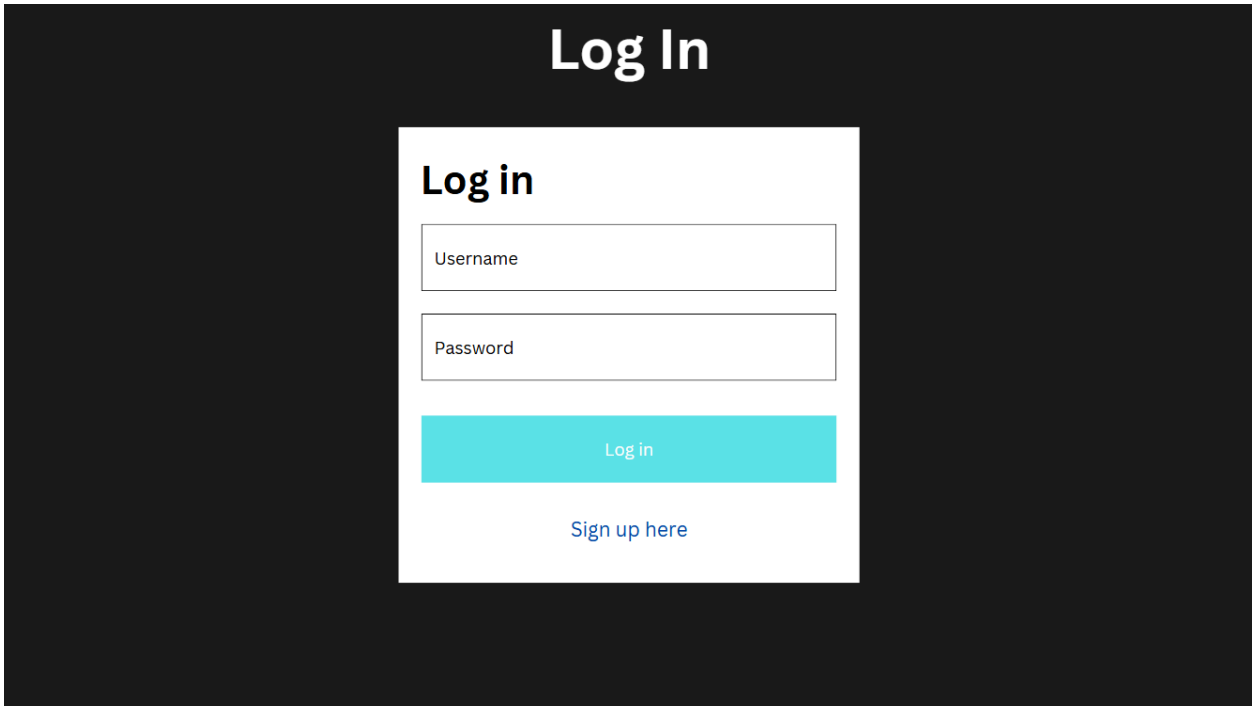

Figure 5.3: Login UI

The user interface is designed to be intuitive, allowing users to enter text or image inputs seamlessly (**Figure 5.1**). Additionally, users have the option to log in or sign up at any point while interacting with the chatbot.

In **Figure 5.2**, the chat session is displayed. Users can send either a text prompt or an image prompt using the attachment icon. User prompts appear on the right side of the chat window, while the chatbot's responses are shown on the left.

Figure 5.3 illustrates the login page, where users can enter their credentials to access the platform. New users can easily sign up by clicking the "sign up here" link.

Data Design

Data Model:

Key Entities and Relationships:

- **Farmer**: Represents the users of the system. Farmers are associated with multiple Chat Sessions
- **Chat Session**: Stores entire sessions with ChatBot including images sent, and vector embeddings.
- **VectorStorage**: An entity to store embeddings for semantic similarity searches. Each embedding represents an entity such as a disease, symptom description, or image analysis output.

Users Database

user_id	Unique ID for each user
email	Email used to log in
created_at	Date and time of account creation

Chat History Database

session_id	ID for each chat session
user_id	ID for user
question	Question given by user
answer	Answer given by RAG system
time_stamp	Date and time of question/answer pair

Environment Setup

Environment: Google Colab

Core Libraries & Frameworks:

Transformers (by Hugging Face) makes it easy to load CLIP models.

torch and torchvision - Essential for running CLIP and other vision models.

Pandas - For data manipulation and organization, particularly if you're working with large datasets of questions, images, or results.

CLIP: Enables zero-shot image classification by linking images with text descriptions, making it ideal for tasks without labeled datasets.

PyTorch: Supports deep learning, particularly for tasks requiring efficient model training and deployment.

LangChain:

- **ConversationalRetrievalChain & RetrievalQA**: Enable building of conversational AI systems that retain context and retrieve information based on queries.
- **OpenAIEmbeddings & ChatOpenAI**: Create text embeddings and generate conversational responses, powering the AI's understanding and dialogue abilities.

Roboflow: Manages datasets and models, essential for organizing image data used in training and classification tasks.

OpenAI: Direct API interaction for generating responses and text processing using OpenAI models, crucial for language-based tasks.

PIL: Handles image processing, such as loading, transforming, and displaying images within the application.