The system design for this project has been carefully structured to ensure efficient and user-friendly detection of weed in images of crops. A modular approach is adapted where in the system is divided into multiple interacting modules, allowing for flexibility and scalability.

Architectural Overview

The architecture of the system can be broadly categorised into three main layers: the Frontend Interface, the Backend Processing and the Machine Learning Model. These layers are integrated together to function for a seamless workflow

Frontend Interface: This layer is the user facing side which allows the users to upload images of crops and view detection results. The users interact via a responsive and user-friendly interface. It is designed using HTML, CSS and JavaScript to enable interactivity, image preview and effective communication with the backend.

Backend Processing: This layer is implemented using Flask, a Python based web framework which acts as a bridge between the frontend and the machine learning model. It handles the requests and flow of data in the system. When an image is uploaded, the backend prepares it for model assessment by performing preprocessing and sends to the YOLOv5 model. It also handles the model’s output handling for displaying the result on the frontend.

Machine Learning Model: YOLOv5 is the core of the system, it receives the pre-processed images, performs tasks to identify and classify the weed and returns bounding boxes with labels and confidence scores.

Data Flow and Module Interactions

Image Uploading and Preprocessing: The UI allows the user to upload an image of a crop field. Once the image is submitted, it is sent to the backend where it undergoes preprocessing such as resizing and format conversion (if needed) ensuring to meet input requirements of the YOLOv5 model.

Model Interface: After preprocessing, the image is fed to YOLOv5 model for assessment. The model performs feature extraction, object detection and classification using. It predicts the bounding boxes around detected objects and assigns labels to each box, either “weed” or “crop” based on its training.

Data Structuring: The raw output from the model consists of coordinates of bounding boxes, confidence scores and class labels, the data is processed into a more structured format such as JSON to make it easier to interpret and transfer to the frontend.

Displaying Results: The results are displayed on the UI. The frontend uses JavaScript to render bounding boxes on the original image along with confidence scores.

Error Handling

Input Validation: The backend ensures that the uploaded file is a JPEG or PNG. If the input is invalid, the system displays an error message describing the incompatibility of the file.

Model Inference Errors: If errors occur during model inference such as unexpected input dimensions, the backend catches exception and returns an error message which prevents the system from crashing.