



International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.935

Volume 15, Issue 10, October 2026

AgriSense: AI-Powered Smart Farming Assistant

Hemanth T G, Jeyaseelan R, K. Usha Rani

U.G. Student, Department of Computer Science and Engineering, K.L.N. College of Engineering, Madurai,
Tamil Nadu, India

U.G. Student, Department of Computer Science and Engineering, K.L.N. College of Engineering, Madurai,
Tamil Nadu, India

Assistant Professor, Department of Computer Science and Engineering, K.L.N. College of Engineering, Madurai,
Tamil Nadu, India

ABSTRACT: Farmers must make several decisions during the cultivation cycle, such as selecting suitable crops, identifying plant diseases, planning field activities and obtaining reliable agricultural information, yet these services are usually offered through separate tools. This paper presents AgriSense, an AI-powered smart farming assistant developed as an integrated web application using machine learning, computer vision and full-stack web technologies. A TabM deep-learning model for tabular data recommends a crop from 22 classes using seven inputs: nitrogen, phosphorus, potassium, soil pH, temperature, humidity and rainfall. For disease detection, OpenCV preprocesses an uploaded leaf image and a custom PyTorch Convolutional Neural Network (CNN) predicts the disease class with a confidence score and management guidance. A rule-based Smart Crop Calendar organizes sowing, irrigation, fertilization, weeding, disease monitoring and harvesting activities, while an AI Chat Assistant based on Retrieval-Augmented Generation (RAG), embeddings, ChromaDB and a large language model (LLM) answers agricultural queries. The application uses a React.js frontend, a Node.js/Express.js backend, Python-based machine-learning services and MongoDB, with JWT and bcrypt providing secure authentication. The modules are demonstrated through working interfaces and a structured test plan; quantitative evaluation on field data remains future work. AgriSense is intended to support, not replace, expert agricultural advice, and its outputs depend on input quality, model performance and the knowledge base.

KEYWORDS: AgriSense, Smart Farming, Crop Recommendation, TabM, Plant Disease Detection, Convolutional Neural Network, OpenCV, Retrieval-Augmented Generation, ChromaDB, MERN Stack.

I. INTRODUCTION

Agriculture supports food production, rural livelihoods and economic development. Throughout the cultivation cycle, farmers must decide which crop to grow, how to recognize and treat plant diseases, when to carry out field operations and where to find dependable farming information. These decisions become difficult because of changing soil conditions, environmental factors, crop requirements and disease symptoms.

Recent progress in Artificial Intelligence (AI), Machine Learning (ML), computer vision and natural language processing offers new support for such decisions. Machine-learning models can analyze soil and climate parameters to recommend crops, computer vision can interpret visible disease symptoms on leaves, and conversational systems can deliver agricultural information in a user-friendly form. In practice, however, these capabilities are generally provided by separate applications, so a farmer may need several tools to obtain different kinds of assistance.

This paper presents AgriSense, an AI-powered smart farming assistant that brings these capabilities into one web-based platform. The system provides crop recommendation using the TabM model, plant disease detection using OpenCV and a custom PyTorch CNN, a Smart Crop Calendar, and an AI Chat Assistant built on RAG, ChromaDB, embeddings and an LLM. It also includes user authentication and user-specific history management. AgriSense is designed to complement farmers' knowledge and existing practices; its recommendations depend on input data quality, model performance and the relevance of the agricultural information available to it.

The paper is organized as follows. Section II reviews the related literature. Section III describes the proposed methodology, system architecture and module design. Section IV discusses the implementation results and testing. Section V presents the conclusion and future enhancements.

II. LITERATURE SURVEY

Moupojou et al. [1] introduced FieldPlant, a plant disease dataset collected directly from plantations rather than under controlled laboratory conditions. It contains 5,170 images with 8,629 individually annotated leaves across 27 disease classes, annotated under the supervision of plant pathologists. Field images contain complex backgrounds, overlapping leaves, shadows and varying illumination, which make classification harder than on laboratory-style datasets such as PlantVillage [5]. In their experiments, classification performance on FieldPlant was better than on PlantDoc. This work motivates the image-preprocessing stage of AgriSense, where OpenCV prepares each uploaded leaf image before it reaches the CNN.

Ashwitha and Latha [2] applied conventional machine-learning algorithms to crop recommendation and yield estimation using agricultural and atmospheric parameters such as temperature and rainfall, and exposed the results through a web interface. The study shows that data-driven models can reveal relationships between environmental conditions and crop suitability, while noting that reliability depends on the quality and representativeness of the training data. AgriSense follows the same idea of parameter-based recommendation but uses TabM, a deep-learning model for tabular data, with seven input parameters.

Pradhan et al. [3] proposed an integrated system that combines crop recommendation, plant disease detection and a conversational AI component. According to the paper, Random Forest was used for soil-based crop prediction, a CNN for disease classification and an XGBoost-based approach for yield forecasting, connected to a web interface. This confirms the feasibility of combining several AI functions in one agricultural platform. AgriSense differs in its implementation: it uses TabM for crop recommendation, OpenCV with a custom PyTorch CNN for disease detection, a rule-based Smart Crop Calendar, and a RAG chatbot with ChromaDB.

Liu and Wang [4] reviewed deep-learning methods for plant disease and pest detection, including image classification, object detection and segmentation. They emphasize the importance of datasets, preprocessing, model selection and evaluation, and identify practical challenges such as lighting variation, image quality and background complexity, noting that models trained on controlled images may not perform equally well in real environments. These findings support the use of image enhancement and preprocessing in AgriSense and the cautious presentation of its predictions.

From this literature, existing systems generally address crop recommendation, disease detection or information retrieval separately, and organized crop-activity scheduling and conversational access are less commonly integrated. AgriSense addresses this gap by combining crop recommendation, disease detection, crop scheduling, retrieval-based chat assistance and history management in a single application.

III. METHODOLOGY

A. System Analysis

Existing agricultural assistance systems typically provide separate solutions for crop recommendation, disease detection and information retrieval. Their main limitations are separate applications, limited integration between modules, general rather than query-specific information, limited crop scheduling, limited conversational assistance and fragmented history management. The proposed system removes these limitations by offering all functions through one web interface. Because the modules handle different data types, each uses a suitable technique: structured numerical data for crop recommendation, image data for disease detection, and text queries for the chat assistant.

B. System Architecture

AgriSense follows a modular layered architecture (Fig. 1). The User Interface Layer is a React.js [6] frontend built with Vite [7] and plain CSS. The Application Layer is a Node.js [8] and Express.js [9] backend that validates requests, exposes REST API routes and coordinates the other services. The Machine Learning Layer contains the TabM and CNN services implemented in Python, and the Image Processing Layer performs OpenCV-based

preprocessing of leaf images. The Database Layer uses MongoDB [10] for user information and prediction history. The AI Chat Assistant uses embeddings, ChromaDB and an LLM. After login or registration, the user selects crop recommendation, disease detection or chat, and the results are saved to MongoDB.

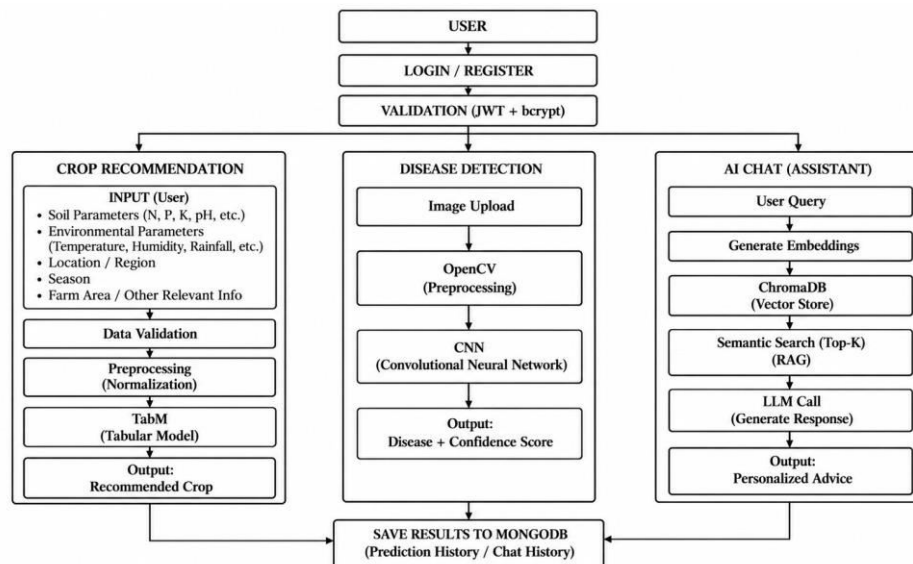


Fig.1.System architecture of AgriSense

The data flow (Fig. 2) follows a request–processing–response cycle. The farmer enters agricultural parameters, uploads a leaf image or submits a query; the frontend sends the input to the backend through API requests; the backend validates it and forwards it to the TabM service, the OpenCV–CNN pipeline or the RAG pipeline; and the result is returned to the frontend while relevant history is stored in MongoDB.

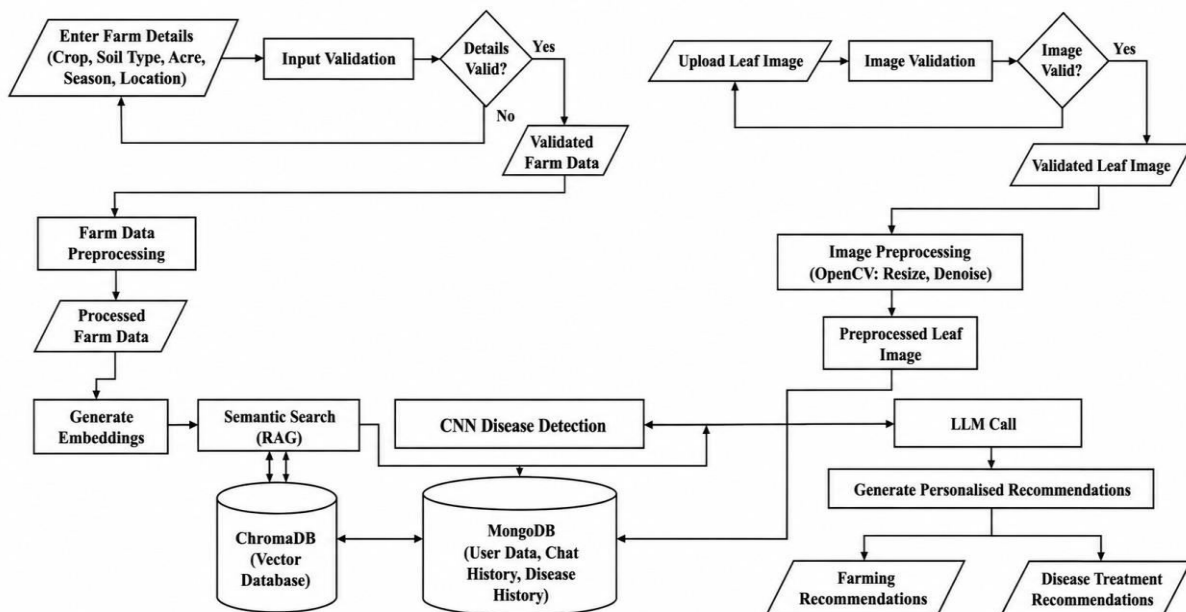


Fig.2. Data flow diagram of AgriSense.

Two actors interact with the platform (Fig. 3). The Farmer can register, log in, use crop recommendation, plant disease detection, the AI chatbot and the Smart Crop Calendar, and view prediction history. The Admin can log in, manage farmers and view system analytics.

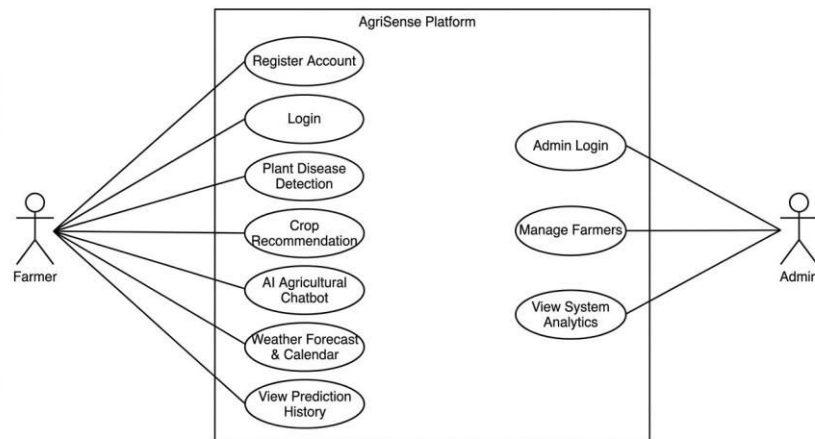


Fig.3. Use case diagram of AgriSense.

C. Crop Recommendation Using TabM

The module accepts seven parameters: nitrogen (N), phosphorus (P), potassium (K), soil pH, temperature, humidity and rainfall. The backend checks that every value is present and numeric before forwarding it to the Python service, which standardizes the input using the preprocessing fitted during training. The standardized vector is passed to the TabM model [11], implemented in PyTorch [12], which produces probabilities for 22 crop classes. TabM uses a multi-head ensemble configuration with eight prediction heads; their probabilities are averaged to obtain the final class probabilities, and the crop with the highest probability is selected. The selected crop is then associated with additional information such as its scientific name, season, duration and water requirement. The output is decision-support information and not a guarantee of crop performance.

D. Plant Disease Detection

The user uploads a leaf image through the frontend, and the backend forwards it to the Python pipeline. OpenCV [13] decodes the image, converts the color space, applies masking and analyses leaf and lesion regions. The selected region is resized to 224×224 pixels, normalized and converted to a tensor for the custom PyTorch CNN, which extracts features through multiple convolutional layers and outputs class probabilities. The class with the highest probability is the predicted disease, reported with its confidence. Depending on the available disease information, the result also includes the pathogen, severity, organic remedies, chemical treatment guidance and preventive measures. The prediction depends on image quality, visible symptoms and the classes supported by the trained model, so it is an initial indication and not a substitute for expert diagnosis.

E. Smart Crop Calendar

The Smart Crop Calendar presents a structured schedule of sowing, irrigation, fertilization, weeding, disease monitoring and harvesting for the selected crop. It is a rule- and knowledge-based component that uses predefined crop-related and seasonal information and does not use a machine-learning model. The schedules are general guidance and may need adjustment for local climate, soil conditions, crop variety and farming practice.

F. AI Chat Assistant

The assistant uses Retrieval-Augmented Generation [14]. A user query is converted into an embedding; ChromaDB [15] performs semantic search over the vectorized agricultural knowledge base and returns the top matching passages; and these passages are supplied as context to the LLM through the Groq API [16], which generates a natural-language answer. The system prompt instructs the model to answer from the supplied context and to say so when the context does not contain the answer. Knowledge can be updated without retraining the TabM or CNN models. The reliability

of answers depends on the quality of the retrieved content, so critical decisions should be checked against local guidance.

G. Authentication and Data Management

Passwords are hashed with bcrypt [17] before storage, and a JSON Web Token (JWT) [18] issued after successful login identifies the user on protected endpoints, following standard password-storage guidance [19]. MongoDB stores user records together with crop recommendation and disease detection history for authenticated users, and all database operations are performed by the backend rather than the frontend. Storing history does not retrain the models.

H. Implementation Environment

The system was developed on Windows 10/11 using VS Code and tested in Chrome and Edge. The recommended minimum configuration is an Intel Core i3 processor, 8 GB RAM and 10 GB of free storage, with an internet connection for the application and AI services. The software stack comprises React.js, CSS, Node.js, Express.js, Python, PyTorch, OpenCV, MongoDB, ChromaDB and the Groq API.

IV. EXPERIMENTAL RESULTS

A. Crop Recommendation Result

Fig. 4 shows the crop recommendation interface. For the sample input $N = 133$, $P = 47$, $K = 24$, temperature = 25°C , humidity = 80 %, pH = 8 and rainfall = 100 mm, the TabM model recommends Cotton (*Gossypium hirsutum*) with a displayed fit of 99.96 %, together with its season (Kharif), duration (150–180 days) and water need (700–1100 mm). The interface also lists alternative candidate crops, soil nutrient indicators and quick soil-profile presets, and allows a Smart Crop Calendar to be created directly for the recommended crop.

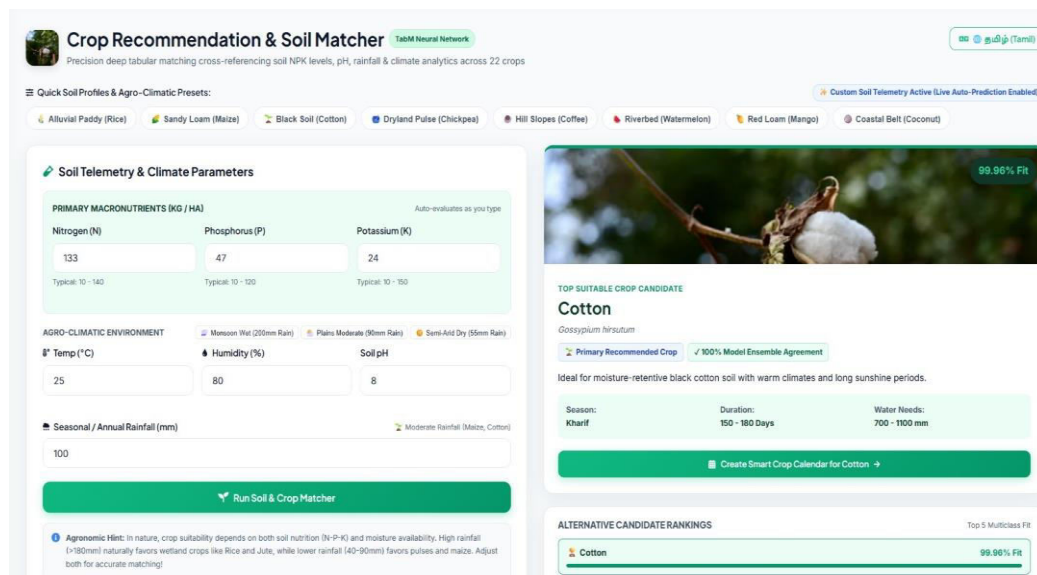


Fig.4.Crop recommendation result

B. Plant Disease Detection Result

In Fig. 5 the uploaded tomato leaf is identified as Tomato Late Blight (*Phytophthora infestans*) with a displayed confidence of 93.5 %. The result page shows the pre-processing summary, observed symptoms, separate organic and chemical treatment sections, and field prevention and sanitation practices. Sample images are provided for quick testing, and users are advised to capture clear images in natural light. The confidence value is the model's prediction confidence and not a guarantee of diagnostic accuracy.

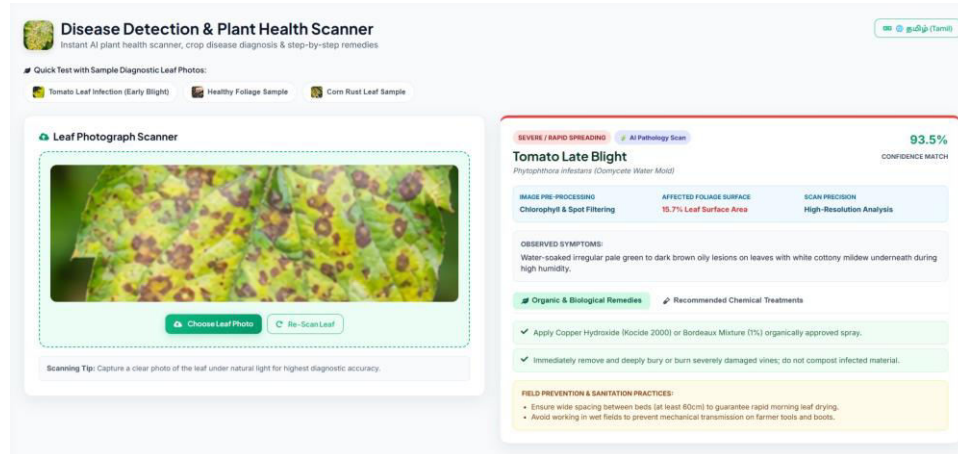


Fig.5.Plant disease detection result

C. Smart Crop Calendar Result

Fig. 6 shows the calendar for Rice (Paddy). The user provides the crop, soil parameters, sowing date, season, irrigation system and farm size, and the module generates a stage-wise production roadmap with a sowing date, estimated harvest date and total duration (for the sample entry, 125 days from 13 October 2026). This presents the sequence of cultivation tasks in one place and links crop selection with cultivation planning.

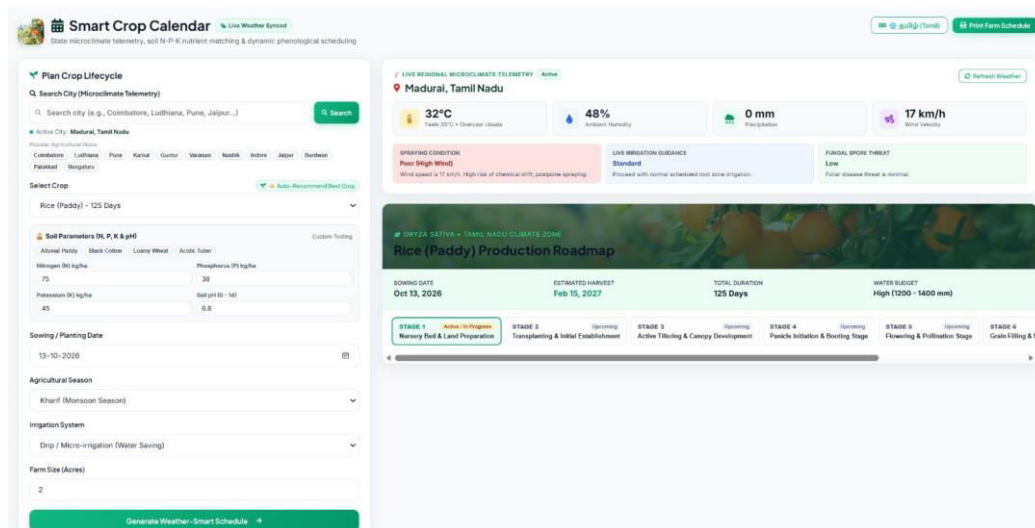


Fig.6.Smart Crop Calendar

D. AI Chat Assistant Result

Fig. 7 shows the Agronomy Assistant. Users can type a question or select a quick prompt, such as treating yellowing leaves in paddy organically or the best irrigation interval for cotton. The query is processed by the RAG pipeline, and the assistant replies in natural language using the retrieved knowledge-base content.

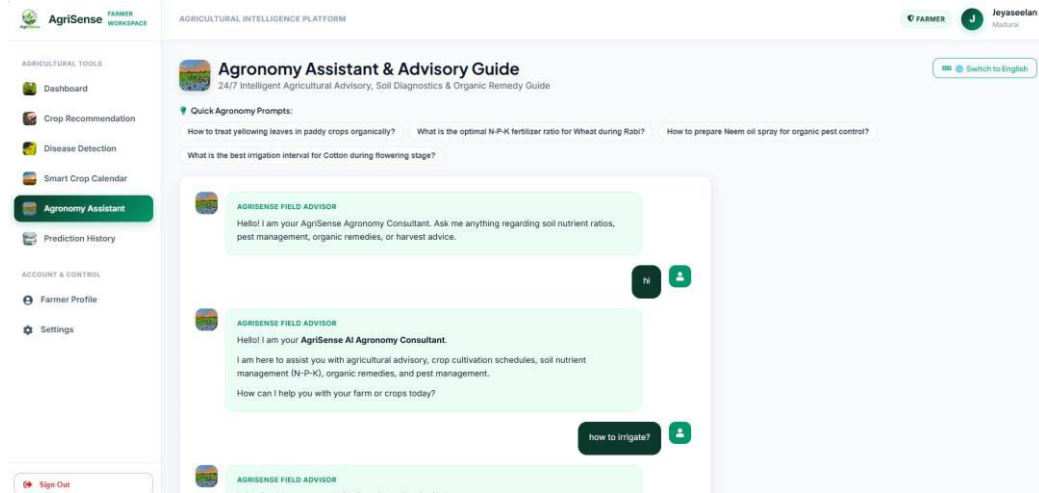


Fig.7.AI Chat Assistant response

E. System Testing

Testing covered unit, functional, integration, performance and usability aspects. Unit tests checked registration, login, crop recommendation, disease detection, the calendar and the chatbot. Functional tests are summarized in Table I. Integration tests covered React.js → Express.js, Express.js → Python ML services, Python → TabM, Python → OpenCV and CNN, backend → MongoDB, RAG → ChromaDB → LLM, and authentication → protected features. Performance checks considered frontend response, API response, model inference, chatbot response and database operations, and usability checks considered navigation, input entry, image upload and clarity of results.

TABLE I. FUNCTIONAL TEST CASES

ID	Input / Action	Expected Result
FT01	Enter valid soil and environmental values	Suitable crop recommendation is displayed
FT02	Enter invalid or incomplete crop parameters	Input validation message is displayed
FT03	Upload a valid plant leaf image	Disease prediction and confidence are displayed
FT04	Select a crop in the calendar	Crop-specific farming activities are displayed
FT05	Enter an agriculture-related question	Relevant chatbot response is generated
FT06	Login with valid credentials	User is authenticated successfully
FT07	Access prediction history	Previously stored records are displayed

F. Discussion

AgriSense combines machine-learning prediction, image processing, rule-based scheduling and knowledge retrieval in one interface, so users do not have to switch between separate applications. Its limitations should be noted. Numerical test results (pass/fail counts and response times) were not recorded, so the system has not been quantitatively evaluated; recommendations and disease predictions depend on input quality, the training data and the supported disease classes; chatbot answers depend on the knowledge base; and the calendar provides general guidance. Further testing with diverse inputs and real field data is needed.

V. CONCLUSION AND FUTURE ENHANCEMENT

AgriSense is an integrated AI-powered smart farming assistant that provides crop recommendation using TabM, plant disease detection using OpenCV and a custom PyTorch CNN, a rule-based Smart Crop Calendar and a RAG-based AI Chat Assistant, supported by JWT and bcrypt authentication and MongoDB history management. By offering these functions through one React.js, Node.js/Express.js and Python-based application, it gives farmers convenient access to crop suggestions, preliminary disease identification, cultivation schedules and agricultural guidance, and is intended

to assist rather than replace professional advice and field-based soil testing.

Future work includes real-time weather APIs and IoT-based soil sensors for more context-aware recommendations; larger and more diverse datasets, additional disease categories and images captured under varied lighting to improve the TabM and CNN models; GPS-based location support and regional information in the calendar; wider multilingual and voice-based interaction in languages such as Tamil, Telugu, Kannada and Malayalam; a larger verified agricultural knowledge base; and cloud deployment, improved data backup and interactive dashboards for reviewing past predictions. These are proposed enhancements, and systematic quantitative evaluation is also planned.

REFERENCES

- [1] E. Moupojou, A. Tagne, F. Retraint, A. Tadonkemwa, W. Dongmo, H. Tapamo and M. Nkenlifack, "FieldPlant: A Dataset of Field Plant Images for Plant Disease Detection and Classification With Deep Learning," IEEE Access, vol. 11, pp. 35398–35410, 2023.
- [2] Ashwitha and Latha, "Crop Recommendation and Yield Estimation Using Machine Learning," Journal of Mobile Multimedia, vol. 18, no. 3, pp. 861–884, 2022.
- [3] R. Pradhan, D. Joshi, S. Maurya, R. Gupta and A. Pm, "Intelligent Crop Recommendation System Including Plant-Based Diseases Detection," International Journal of Engineering Research & Technology (IJERT), vol. 15, no. 4, Apr. 2026, Paper ID: IJERTV15IS041017, doi: 10.5281/zenodo.19640831.
- [4] Liu and Wang, "Plant Diseases and Pests Detection Based on Deep Learning: A Review," Plant Methods, vol. 17, Art. 22, 2021.
- [5] D. P. Hughes and M. Salathé, "An Open Access Repository of Images on Plant Health to Enable the Development of Mobile Disease Diagnostics," 2015.
- [6] React Documentation, "React: The Library for Web and Native User Interfaces," [Online]. Available: <https://react.dev>
- [7] Vite Documentation, "Vite: Next Generation Frontend Tooling," [Online]. Available: <https://vite.dev>
- [8] Node.js Documentation, "Node.js," [Online]. Available: <https://nodejs.org>
- [9] Express Documentation, "Express: Fast, Unopinionated, Minimalist Web Framework for Node.js," [Online]. Available: <https://expressjs.com>
- [10] MongoDB, Inc., "MongoDB Manual," [Online]. Available: <https://www.mongodb.com/docs>
- [11] Y. Gorishniy et al., "TabM: Advancing Tabular Deep Learning with Parameter-Efficient Ensembling," in Proc. Int. Conf. Learning Representations (ICLR), 2025.
- [12] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Advances in Neural Information Processing Systems, vol. 32, 2019.
- [13] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.
- [14] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.
- [15] Chroma Documentation, "Chroma: The Open-Source Embedding Database," [Online]. Available: <https://docs.trychroma.com>
- [16] Groq, "Groq API Reference," [Online]. Available: <https://console.groq.com/docs>
- [17] N. Provos and D. Mazières, "A Future-Adaptable Password Scheme," in Proc. USENIX Annual Technical Conference, 1999.
- [18] M. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)," IETF RFC 7519, 2015.
- [19] OWASP Foundation, "Password Storage Cheat Sheet," [Online]. Available: <https://cheatsheetseries.owasp.org>



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 9940 572 462  6381 907 438  ijirset@gmail.com



www.ijirset.com

Scan to save the contact details