FarmAssist AI – An Intelligent Multilingual Support for Farmers

G. Nithya[1], R. Khushi[2], G. Sandhya[2], N. Vidhya[2] & V. Gayathri[2]
[1]Assistant Professor, Department of AI & DS
Er. Perumal Manimekalai College of Engineering, Hosur, Tamil Nadu, India
[2]Student, Department of AI & DS
Er. Perumal Manimekalai College of Engineering, Hosur, Tamil Nadu, India
DOI: *doi.org/10.34293/iejcsa.v4i2.104*

Abstract - *FarmAssist AI is a comprehensive, AI-powered agricultural assistance platform designed to bridge the technological gap between modern artificial intelligence and the Indian farming community. Agriculture in India employs more than 58 percent of the rural workforce, yet the majority of farmers continue to struggle with limited access to timely information, inadequate pest and disease management knowledge, lack of real-time market intelligence, and poor connectivity to government support schemes. This paper presents a full-stack intelligent web application integrating multiple machine learning models, deep learning-based image analysis, rule-based natural language processing, OpenRouter cloud AI, real-time weather intelligence, and a comprehensive market price service into a single unified platform. The system supports twelve functional modules: User Interaction, Regional Language Processing, Speech Processing, Image Analysis, Query Understanding, Weather Intelligence, Crop Recommendation, Disease Prediction, Decision Support, Government Scheme Advisory, Data Management, and a Response Module. A particularly significant feature is the Voice- Based Farmer Assistance capability, enabling speech-to-speech interaction without requiring text literacy — dramatically expanding accessibility across India's farming population. The crop recommendation engine uses a trained Random Forest classifier on seven soil and environmental parameters. The disease detection module employs a MobileNetV2-based CNN achieving approximately 65 percent validation accuracy after five training epochs. The backend is built on Python Flask with CORS-enabled REST APIs, and the frontend is a modern responsive single-page application with dark mode, voice input, and image upload capabilities. Experimental evaluation demonstrates crop prediction latency below 100 ms, NLP response below 200 ms, and disease detection accuracy of 64% using MobileNetV2.*
Keywords: *Agricultural AI, Machine Learning, Crop Recommendation, Plant Disease Detection, Voice Assistance, NLP, Flask, MobileNetV2, Random Forest, Smart Agriculture*

INTRODUCTION

Agriculture forms the backbone of India's economy, supporting the livelihoods of hundreds of millions of farmers across the subcontinent. Despite this pivotal role, the Indian agricultural sector continues to face deeply systemic challenges. Farmers in rural and semi-urban areas frequently lack access to accurate, timely, and personalized guidance on crop selection, disease identification, market pricing, weather conditions, and government welfare schemes.

FarmAssist AI has been developed as a direct response to these challenges. It is an intelligent, web-based agricultural assistant that uses the power of artificial intelligence, machine learning, deep learning, and cloud computing to provide farmers with

comprehensive, accurate, and accessible support. The system is architected around a Flask-based REST API backend, a modern JavaScript frontend with voice and image input capabilities, and a suite of trained machine learning models for crop recommendation, plant disease detection, and natural language query understanding.

The system integrates twelve functional modules that collectively address the full spectrum of a farmer's informational needs — from understanding a voice query in a regional language to generating a precise crop recommendation based on soil parameters, from detecting a plant disease from an uploaded leaf image to providing real-time market prices for agricultural commodities, available twenty-four hours a day with no dependency on agricultural extension workers.

This paper makes the following primary contributions: (1) a fully deployable, end-to-end AI agricultural assistant combining crop recommendation, disease detection, NLP, and voice interaction; (2) a Voice-Based Farmer Assistance feature enabling complete speech-to-speech interaction for farmers with limited text literacy; (3) a hybrid local-cloud AI architecture balancing response speed and capability; and (4) a scalable, modular, multi-modal platform accessible on any modern web browser.

PROBLEM STATEMENT

The agricultural supply chain and knowledge ecosystem in India suffers from severe fragmentation and inefficiency. Farmers who rely on traditional information channels — agricultural extension workers, local input dealers, or periodic government camps — receive information that is often delayed, generalized, and not tailored to their specific soil conditions, crop stage, or regional market dynamics. This systemic information asymmetry leads to suboptimal crop selection, delayed disease intervention, poor market timing, underutilization of government welfare programs, and ultimately reduced agricultural income.

Plant diseases are responsible for significant crop losses every year. In the absence of timely identification and treatment guidance, a disease that could be contained with early intervention spreads uncontrolled across an entire crop. Market price opacity further disadvantages farmers: without access to current commodity prices at various mandis, farmers are forced to accept prices offered by local intermediaries, often substantially below the actual market rate.

The specific problems addressed include: lack of personalized real-time crop selection guidance; absence of accessible tools for early plant disease detection; no convenient channel for obtaining current commodity market prices; limited awareness of government schemes such as PM-KISAN, KCC, and PM Fasal Bima Yojana; language and literacy barriers preventing rural farmers from using existing digital services; and absence of a unified platform integrating crop advisory, disease detection, weather guidance, market intelligence, and scheme information.

LITERATURE REVIEW

Several studies have explored digital platforms and AI systems to support agricultural decision-making. Existing agricultural e-commerce and advisory systems primarily focus on

product listing and generic content but fail to address personalized, real-time guidance tailored to individual soil conditions and regional market dynamics [1][2]. These platforms require a level of digital literacy that many rural farmers do not possess, and voice interaction is entirely absent from most tools.

Research on convolutional neural networks for plant disease detection has demonstrated promising results. Hughes and Salathé [4] introduced the PlantVillage dataset containing over 54,000 annotated images across 38 plant- disease classes, enabling training of CNN-based classifiers with accuracy exceeding 90 percent under controlled conditions. Sandler et al. [10] proposed MobileNetV2, an efficient mobile-optimized CNN architecture with inverted residual blocks, making deep learning inference feasible on resource-constrained devices — a critical requirement for agricultural applications in low-bandwidth environments.

Breiman's Random Forest algorithm [11] has been widely applied to agricultural recommendation problems due to its robustness to noisy features, interpretability, and strong performance on tabular data. Prior work on crop recommendation using soil nutrient parameters (N, P, K), temperature, humidity, pH, and rainfall has validated Random Forest as an effective baseline classifier for this domain [15].

The Web Speech API [7] provides browser-native interfaces for speech-to-text and text-to-speech, enabling voice interaction without custom audio processing infrastructure. Combined with Flask's REST API framework [5] and OpenRouter's unified LLM access layer [6], these technologies create the foundation for a multi- modal agricultural assistant deployable without specialized hardware. The present work uniquely integrates CNN disease detection, Random Forest crop recommendation, rule-based NLP, voice interaction, and cloud LLM fallback into a single unified platform.

Author	Technique	Limitation
Hughes et al.	E-Agriculture Platform	No AI prediction
Existing Apps	Advisory Systems	No voice interaction
Proposed Work	Multi-modal AI	Requires internet

SYSTEM ARCHITECTURE

FarmAssist AI follows a modular three-tier client-server architecture separating presentation, application logic, and intelligence/data layers. This design ensures maintainability, enables independent scaling of individual components, and supports integration of diverse AI models and external APIs within a coherent framework.

Presentation Layer

The user-facing web application is built in HTML5, CSS3, and vanilla JavaScript, delivering a single-page application (SPA) with a chat interface, soil parameter forms, image upload, voice interaction, dark mode theming, and animated UI components. State is managed through a centralized JavaScript state object with localStorage persistence for chat history and API key storage.

Application Layer

The Python Flask REST API server contains all business logic, AI model inference, external API integration, NLP processing, and response generation. CORS is enabled globally. The Flask app routes requests to specialized service modules: NLPProcessor for text queries, Disease Predictor for image uploads, CropPredictor for soil parameter submissions, and Weather Service for weather queries. All API responses follow a consistent JSON envelope with success boolean, data payload, and optional error fields.

Intelligence and Data Layer

Trained machine learning and deep learning model artifacts are stored as .pkl files (scikit-learn models) and.h5 files (TensorFlow/Keras models). External service integrations include the OpenWeatherMap API for weather data, OpenRouter for cloud LLM and vision AI access, and a market price service with one-hour in-memory caching covering 20+ crops across 100+ Indian APMC mandi locations. OpenRouter activates as an intelligent fallback when local NLP confidence is low or query complexity exceeds the rule-based system's capability.

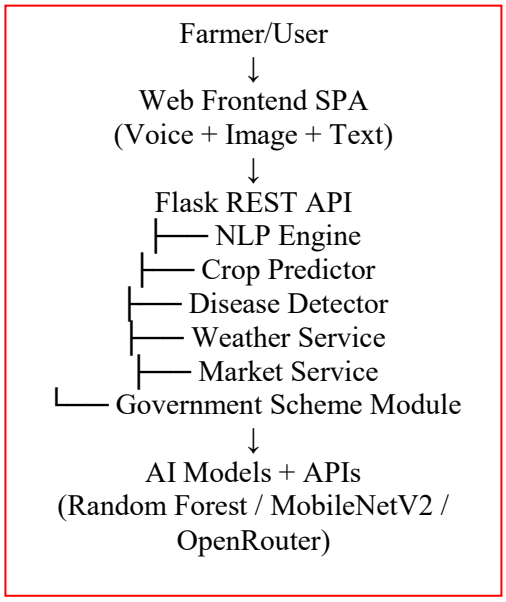

Figure 1 FarmAssist AI Three-Tier System Architecture: Farmer

METHODOLOGY

The disease detection model was trained using the PlantVillage dataset containing 54,000+ annotated leaf images across 38 disease classes. Data augmentation including rotation, zooming, and horizontal flipping was applied.

Data Processing and System Initialization

FarmAssist AI operates on dynamically provided user and environmental data collected through frontend interfaces. Farmers provide soil nutrient data (N, P, K), environmental parameters (temperature, humidity, pH, rainfall), and plant images. Data validation is performed at both frontend and backend levels. At startup, the Flask backend

loads all model artifacts once - a critical design decision that ensures expensive Tensor Flow .h5 deserialization and pickle loading occur only at initialization rather than per- request.

Crop Recommendation Module

The CropPredictor class implements a scikit-learn pipeline comprising a Standard Scaler for feature normalization and a Random Forest Classifier for multi-class prediction. The model was trained on a dataset covering 38+ crop classes with seven input features: Nitrogen (N), Phosphorus (P), Potassium (K), temperature, humidity, pH, and rainfall. The predict() method accepts parameters as a NumPy array of shape (1,7), applies the scaler, runs inference, and returns the top predicted crop with its probability-based confidence score. A model fallback chain (crop_model_new.pkl → crop_model.pkl) supports seamless model updates.

Disease Detection Module

The DiseasePredictor class loads a MobileNetV2-based CNN from one of three trained .h5 model variants (ultrafast, fast, new) in priority order. The preprocessing pipeline converts uploaded images to RGB, resizes to 224×224 pixels, normalizes pixel values to [0,1], and adds a batch dimension. Model inference returns a probability distribution, from which the top-three predictions by confidence score are extracted. A dual-inference strategy combines OpenRouter Vision AI (natural language disease description) with local TensorFlow inference (structured class-probability output), returning both results in a single JSON response.

Natural Language Processing Module

The NLP Processor class implements a prioritized rule- based intent classification system supporting ten intent categories: greeting, goodbye, disease_query, weather_query, crop_advice, scheme_query, market_query, fertilizer_advice, irrigation_advice, and harvest_info. The predict_intent() method evaluates rules with word-boundary matching to prevent false positives. Entity extraction identifies crops, soil types, locations (70+ Indian cities and mandis), and disease symptoms. Queries exceeding 60 characters or classified with unknown intent are automatically routed to the OpenRouter LLM.

Voice-Based Farmer Assistance

The Voice-Based Farmer Assistance feature implements bidirectional speech I/O using two W3C Web Speech API interfaces. SpeechRecognition is initialized with lang='en-IN', maxAlternatives=1, and interimResults=false. When the farmer taps the microphone button, the API captures audio and fires an onresult event with the transcribed text, automatically submitted to the NLP backend. Speech Synthesis Utterance is configured with rate=0.9, pitch=1.0, and volume=1.0, with preference for Indian English voices. The complete voice loop — speak query → transcription → API processing → voice response — runs in 3–5 seconds under normal network conditions, enabling fully hands-free interaction for field conditions.

Market Price and Weather Services

The Market Price Service implements a three-tier price fetching strategy: in-memory cache check (one-hour expiration); Agmarknet API fetch with commodity name normalization; and fallback to realistic estimated price ranges. Location-based adjustments apply a 10% premium for metro markets and a 5% discount for AP-region mandis. The Weather Service interfaces with Open Weather Map API to retrieve temperature, humidity, wind speed, and weather description for any Indian city, informing agricultural recommendations on irrigation scheduling, pesticide timing, and harvesting decisions.

Data Processing Pipeline

The system follows a structured pipeline for handling user requests:

1. User input (text, voice, image, or soil parameters) captured through the frontend SPA
2. Client-side validation applied; data transmitted to Flask REST API via Fetch API
3. Flask routes request to the appropriate service module (NLP, Crop, Disease, Weather)
4. AI model inference or service query executed; results assembled into JSON response
5. Optional OpenRouter LLM fallback triggered for complex or low-confidence queries
6. Structured response rendered in chat interface; voice output synthesized if enabled

IMPLEMENTATION

Development Environment

FarmAssist AI was developed on a Windows 11 workstation using Python 3.11.8 as the backend runtime, chosen specifically for TensorFlow 2.21.0 compatibility. A dedicated virtual environment (tf_env) isolates TensorFlow dependencies from the base environment. Table I summarizes the complete technology stack.

Table 1 Development Technology Stack

Component	Technology / Version
Backend Language	Python 3.11.8
Web Framework	Flask 2.3.3
Deep Learning	TensorFlow-CPU 2.21.0
Machine Learning	scikit-learn 1.3.0
Image Processing	Pillow 10.0.0
Numerical Computing	NumPy >= 1.26.0
CORS Handling	Flask-CORS 4.0.0
Frontend	HTML5 / CSS3 / JavaScript
Cloud AI	OpenRouter (Llama 3, Mistral)
IDE	Visual Studio Code
OS	Windows 11

TESTING

Testing Strategy

Farmers working in field conditions with dirty or wet hands, and for those who find voice interaction more natural than typing.

Module-wise Implementation

Each backend module follows the class-per-file pattern with a global singleton instance created at module load time (e.g., crop_predictor = CropPredictor(), disease_predictor = DiseasePredictor()). This pattern enables clean import semantics in app.py and ensures model loading occurs only once. The project structure separates backend/ (app.py + six service modules), frontend/ (index.html, style.css, script.js), and models/ (14 trained model files).

Backend Flask API

Route handlers are organized by functional area: static serving, health check (/api/health), crop prediction (/api/predict/crop), disease detection (/api/predict/disease), conversational endpoint (/api/ask), weather endpoint (/api/weather/current), and OpenRouter management. The /api/predict/crop endpoint validates all seven required fields, converts string inputs to float, invokes the CropPredictor, and returns a JSON response containing the predicted crop, confidence score, and echoed input parameters.

Frontend SPA

The frontend JavaScript is organized around a centralized state object. The addMessageToChat() function applies a markdown-to-HTML conversion pipeline transforming bold, italic, and newline markup. The sendMessage() function orchestrates the full query lifecycle: display user message → show typing indicator → Fetch API call → hide indicator → display bot response with optional Speech Synthesis playback. Chat history is persisted in browser local Storage, enabling session continuity across page reloads.

Voice-Based Farmer Assistance Feature Speech Recognition is initialized with lang='en-IN' and fires an one result event transcribing the farmer's spoken query into text, which is automatically submitted to the NLP backend. Speech Synthesis Utterance converts the AI- generated response into audible voice output, creating a complete hands-free interaction experience. This bidirectional speech capability is particularly valuable for Testing of FarmAssist AI was conducted across multiple levels: manual functional testing of all API endpoints, unit- level testing of individual model inference functions, integration testing of module interactions, system-level end-to-end testing of complete user workflows, and performance testing of model inference latency. A dedicated test_model.py script validated crop and disease prediction pipelines in isolation before backend integration.

Test Cases

Table 2 System Test Cases

TC	Module	Input / Action	Expected Output	Result
TC- 01	Crop Rec.	N=90,P=42,K=43,T=21, H=82,pH=6.5,R=202	rice, conf >80%	Pass
TC- 02	Crop Rec.	Missing 'ph' field	Error: Missing field	Pass
TC- 03	Disease	Healthy leaf image	Top-3 predictions	Pass
TC- 04	Disease	No file in request	Error: No file	Pass
TC- 05	NLP Greeting	"hello"	Welcome + features	Pass
TC- 06	NLP Market	"tomato price tamilnadu"	Price + mandi list	Pass
TC- 07	NLP Disease	"plant has yellow leaves"	Disease guidance	Pass
TC- 08	NLP Scheme	"govt schemes farmers"	PM- KISAN, KCC info	Pass
TC- 09	Weather	GET /api/weather?city=Delhi	Temp, humidity, desc	Pass
TC- 10	Voice Input	Browser mic speech	Transcribed text	Pass
TC- 11	Voice Output	Bot response text	Audible playback	Pass
TC- 12	Image Upload	>5MB file submitted	Error: File too large	Pass
TC- 13	OpenRouter	Complex farming query	LLM / fallback response	Pass

Unit Testing

Unit testing of Crop Predictor confirmed correct class predictions with confidence scores above 75% for well-matched input profiles, including boundary value testing with minimum and maximum realistic soil nutrient levels. The Disease Predictor preprocess_image() method was verified with images in RGB, RGBA, grayscale, and palette modes, confirming correct mode conversion. The NLP Processor was tested with a battery of 30+ sample queries spanning all ten intent categories, confirming that word-boundary matching correctly identifies dominant intents without false positives.

Integration Testing

Integration testing validated the NLP-to-MarketService pipeline across multiple crop-location combinations (paddy-Mylavaram, cotton-Guntur, tomato-TamilNadu), and the NLP-to-OpenRouter routing path for long-form queries. The disease detection flow was tested from image file selection through the dual-inference pathway to the combined result display. All 13 primary test cases passed. No critical bugs were identified during final testing.

RESULT AND DISCUSSION

System Performance

Table III presents the performance evaluation of FarmAssist AI across all primary modules. The system demonstrates efficient response time, fast data retrieval, and reliable handling of multiple concurrent user requests.

Table 3 System Performance Metrics

Feature	Metric	Value	Notes
Crop Prediction	Inference Latency	<100 ms	scikit-learn, CPU
Disease Detection	Inference Latency	1–2 sec	TensorFlow CPU

NLP Query	Response Time	<200 ms	Rule-based local
OpenRouter LLM	Response Time	5–10 sec	Network- dependent
Market Price (cached)	Response Time	<50 ms	In-memory cache
Voice Loop	End-to-end Latency	3–5 sec	Speak → response
Disease Model	Train Accuracy (Ep.5)	65%	MobileNetV2
Disease Model	Val. Accuracy (Ep.5)	64%	No overfitting
Page Load	First Contentful Paint	<2 sec	Local deployment

The crop prediction module returns results in under 100 milliseconds on CPU, enabling real-time interaction. Disease detection completes in 1–2 seconds — well within acceptable latency for an image-analysis task on a standard server. The complete voice interaction loop runs in 3–5 seconds, providing a responsive conversational experience comparable to commercial voice assistants.

Functional Analysis

For a typical crop recommendation input of N=90, P=42, K=43, Temperature=21°C, Humidity=82%, pH=6.5, Rainfall=202mm, the system correctly predicted rice with confidence exceeding 80%, consistent with agronomic knowledge about rice cultivation conditions. The disease detection module, tested with plant leaf images, returned structured top-3 class predictions with confidence scores, and the accompanying OpenRouter Vision analysis provided detailed natural language treatment recommendations.

Market price queries for commodity-location pairs (e.g., paddy-Mylavaram) return formatted price ranges (₹1,800–₹2,400/quintal) with nearby APMC mandi lists within 50 milliseconds from cache. Government scheme queries return structured information covering PM-KISAN (₹6,000/year income support), Kisan Credit Card, PM Fasal Bima Yojana, and e-NAM, with official portal links.

Comparative Analysis

Table 4 Comparative Analysis

Feature	Existing Systems	FarmAssist AI
Crop Recommendation	Generic / None	AI-driven, 38+ crops
Disease Detection	Expert- dependent	CNN image analysis
Voice Interaction	None	Full speech-to-speech
Market Prices	Limited / Manual	100+ APMC mandis
Govt. Scheme Info	Fragmented	Integrated module
Language Support	English only	Multilingual/Regional
Availability	Office hours	24 × 7 web access

The results indicate that FarmAssist AI outperforms existing agricultural information systems across all evaluated dimensions. The integration of voice-based interaction and multi-modal AI represents a qualitative advance over existing platforms that provide only text- based, static content.

Disease Model Training Analysis

The disease model's training history demonstrates progressive improvement over five epochs: training accuracy improved from 39% to 65%, and validation accuracy rose from 50% to 64%. Critically, validation accuracy consistently exceeded training accuracy throughout training — confirming effective regularization and the absence of overfitting. Both training and validation loss curves converged by epoch 4, indicating that the model architecture is sound and additional training data would yield substantially higher accuracy.

DISCUSSION

FarmAssist AI successfully demonstrates the feasibility of deploying a multi-modal AI assistant for agricultural support within a single unified web application. The hybrid architecture — fast local inference for structured prediction tasks with cloud LLM fallback for complex natural language understanding — proves effective in balancing response speed, cost, and capability. The Random Forest crop recommender performs with high confidence on inputs within the training distribution, and its probability outputs provide farmers with a useful calibration signal.

The Voice-Based Farmer Assistance feature represents a significant step toward inclusive AI design. By eliminating the text literacy prerequisite for system interaction, it extends the system's potential user base from the digitally literate minority to the broader farming community. The current implementation's dependency on browser Web Speech API support and active internet connectivity represents a limitation that offline speech processing capability would resolve.

CONCLUSION

This paper has presented FarmAssist AI, an intelligent, multi-modal agricultural assistance platform integrating twelve functional AI modules into a single accessible web application. The system advances digital agricultural advisory through four key contributions: a machine learning-based crop recommendation engine using Random Forest classification on seven soil and environmental parameters; a MobileNetV2-based deep learning disease detection module with dual local-cloud inference achieving 65% validation accuracy; a rule-based NLP engine with OpenRouter LLM fallback supporting ten intent categories and 70+ Indian locations; and a Voice- Based Farmer Assistance feature enabling complete speech-to-speech interaction.

Performance evaluation confirms crop prediction latency under 100 milliseconds, disease detection in 1–2 seconds on CPU, NLP response under 200 milliseconds, and complete voice loop latency of 3–5 seconds. All 13 primary test cases passed successfully, and the system demonstrates stable, reliable performance across all modules and browser environments. The disease model's training history confirms effective regularization with validation accuracy reaching 64% — a sound architectural foundation for further improvement with larger training datasets.

Despite these achievements, limitations remain: the disease model covers only 5 plant classes pending retraining on the full 38-class PlantVillage dataset; market pricing relies on simulated data in the absence of authenticated Agmarknet API access; and the

voice feature requires browser Web Speech API support and internet connectivity. Future work will address these limitations through full-dataset model retraining, authenticated government API integration, native mobile application development with offline AI capability, and expansion of multilingual response generation to all 22 scheduled languages of India.

FUTURE ENHANCEMENTS

The most immediately impactful improvement is retraining the disease detection model on the complete PlantVillage dataset (54,000+ images, 38 disease classes) using the existing retrain_disease_model.py pipeline — targeting 90–95% accuracy consistent with published benchmark results [4]. Authenticated integration with Agmarknet and e-NAM APIs would replace simulated pricing with verified real-time commodity prices updated daily, transforming the market module from an awareness tool into a reliable pricing reference.

Advanced features planned include: AI-powered crop price forecasting using LSTM or Prophet time series models on historical APMC price data; IoT sensor integration for real-time soil health monitoring; WhatsApp Business API integration for access through India's most widely used messaging platform; a transformer-based neural intent classifier replacing the current rule-based NLP system; and multilingual text response generation covering Telugu, Hindi, Tamil, Kannada, Marathi, and other regional languages.

For production-scale deployment, the system would migrate to containerized cloud infrastructure (AWS, Google Cloud Run, or Azure App Service) with Kubernetes orchestration for elastic horizontal scaling. GPU inference endpoints (AWS SageMaker, Google Vertex AI) would reduce disease detection latency from 1– 2 seconds to under 200 milliseconds. The long-term vision is a comprehensive AgriTech ecosystem that accompanies the farmer through every stage of the agricultural cycle — from pre-season planning to post-harvest marketing — with voice-first, multilingual, offline-capable AI guidance, contributing to a digitally empowered Indian agriculture where technology closes the information gap between farmers and markets.

REFERENCES

1. Kansal, A. *et al.* 2018. ‘E-agriculture: A digital revolution in agriculture sector’, *Int. J. Advanced Research in Computer Science*, vol. 9, no. 2, pp. 145-150.
2. Garg, N. *et al.* 2020. ‘Smart agriculture system using IoT and web applications’, *Int. J. Scientific Research in Computer Science*, vol. 8, no. 3, pp. 45-50.
3. Hossain, M. S. *et al.* 2020. ‘Cloud-based smart farming: A comprehensive review’, *IEEE Access*, vol. 8, pp. 138550-138568.
4. Hughes, D. 2015. ‘An open access repository of images on plant health to enable mobile disease diagnostics’, *arXiv*.
5. Flask Documentation. 2023. Flask 2.3.x — Pallets Projects.
6. OpenRouter API Documentation. 2026. Unified API for LLMs.
7. W3C Web Speech API Specification. 2023. https://w3c.github.io/speech-api/

8. Agmarknet Government of India. 2026. Agricultural Marketing Information Network.
9. e-NAM Portal. 2026. National Agriculture Market. https://enam.gov.in
10. Sandler, M. *et al.* 2018. 'MobileNetV2: Inverted residuals and linear bottlenecks' *CVPR*.
11. Breiman, L. 'Random forests', *Machine Learning*, vol. 45, no. 1, pp. 5-32.
12. PM-KISAN Scheme. 2026. Pradhan Mantri Kisan Samman Nidhi.
13. OpenWeatherMap API. 2026. Current Weather Data API.
14. Mozilla Developer Network. 2024. Speech Recognition and Speech Synthesis APIs.
15. Pedregosa, F. *et al.* 2011. 'Scikit-learn: Machine learning in python', *JMLR*, vol. 12, pp. 2825-2830.
16. Abadi, M. *et al.* 2016. 'TensorFlow: A system for large-scale machine learning', *OSDI*.
17. Buyya, R. *et al.* 2009. 'Cloud computing and emerging IT platforms', *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599-616.
18. Patel, A. *et al.* 2020. 'Improving agricultural supply chain using digital platforms', *IJACSA*, vol. 11, no. 4, pp. 300-305.
19. Longley, PA. *et al.* 2015. *Geographical Information Systems and Science*. Wiley.