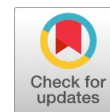


Farmingo: A MERN and ML-Integrated Platform for Smart, Community-Driven Agriculture

Khushbu Jha, Antim Dev Mishra



Abstract: India's agricultural sector, while possessing immense potential, faces persistent challenges that have discouraged farmer participation. Limited access to modern resources, inadequate technical knowledge, and unorganized marketplaces have contributed to these issues. A recent NABARD report (2023) reveals that although 58% of farmers are aware of schemes like the Kisan Credit Card, only 28% successfully access them, and over 76% report not receiving the Minimum Support Prices (MSP), often selling their crops at a loss [1]. This study aims to address these challenges through the development of Farmingo, a unified web-based platform designed to empower farmers with intelligent agricultural insights and facilitate community-driven support. Farmingo has been developed using the MERN stack (MongoDB, Express.js, React.js, Node.js) and integrates Python-based machine learning (ML) models served via Flask APIs. The system provides real-time predictions and recommendations for crop selection and fertiliser application, based on inputs such as soil characteristics, weather patterns, and crop-specific data. Additionally, the platform offers features that enable users to buy, rent, or list farming equipment, as well as share knowledge through lessons, blogs, and videos. To ensure Farmingo's practical relevance, the platform's architecture was designed to seamlessly connect the MERN-based frontend with the ML models, enabling a cohesive and user-friendly experience. Field implementation and user feedback suggest that Farmingo fosters improved decision-making and encourages the adoption of data-driven farming practices among farmers. Unlike existing agri-tech solutions that typically focus on specific aspects of farming, Farmingo adopts a holistic approach by integrating intelligent decision support with a collaborative social environment. This integration aims to bridge knowledge gaps and promote equitable access to resources within India's agricultural ecosystem. The paper elaborates on the system design, implementation of machine learning (ML) algorithms, and real-world use cases encountered during pilot testing. It also outlines future directions for Farmingo, including the incorporation of crop disease detection modules, enhanced visibility of government schemes, and continuous updates to the dataset to improve region-specific agricultural intelligence. The findings of this research highlight the transformative potential of combining technology and community-driven support to bolster India's agricultural sector.

Keywords: Smart Agriculture, MERN (MongoDB, Express.js, React.js, Node.js) Stack, Machine Learning, Crop Recommendation, Fertilizer Prediction, Crop Price Forecasting, Flask API (Application Programming Interface), Community-Driven Farming, Agri Tech, Real-Time Prediction

Manuscript received on 24 May 2025 | First Revised Manuscript received on 31 May 2025 | Second Revised Manuscript received on 17 October 2025 | Manuscript Accepted on 15 November 2025 | Manuscript published on 30 November 2025.

*Correspondence Author(s)

Dr. Antim Dev Mishra*, Associate Professor and Deputy Director, Department of Computer Science and Engineering, IITM Janakpuri, New Delhi (Delhi), India. Email ID: antimdevmishra@gmail.com, ORCID ID: [0009-0000-4936-150X](https://orcid.org/0009-0000-4936-150X)

Khushbu Jha, Department of Computer Science and Engineering, IITM Janakpuri, New Delhi (Delhi), India. Email ID: khushbu3jha@gmail.com

© The Authors. Published by Lattice Science Publication (LSP). This is an open access article under the CC-BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>)

Abbreviations:

NABARD: National Bank for Agriculture and Rural Development
ML: Machine Learning
CRUD: Create, Read, Update, Delete
MSP: Minimum Support Prices
GDP: Gross Domestic Product

I. INTRODUCTION

A. Background and Context

India's agricultural sector employs more than 50% of the country's workforce, yet it contributes only around 17–18% to the GDP (Gross Domestic Product). This disparity stems from long-standing issues, including low profitability, market exploitation, and limited access to modern tools and knowledge. Despite government initiatives, a significant portion of India's farmers, particularly small and marginal ones, remain underserved in terms of credit, storage, insurance, and market access.

According to a recent report by the National Bank for Agriculture and Rural Development (NABARD) [1], only 28% of farmers accessed formal credit despite awareness among 58%, and over 76% reported not receiving Minimum Support Prices (MSP) for their produce. Such gaps highlight the pressing need for intelligent, scalable, and inclusive technological solutions.

B. Problem Statement

While multiple mobile apps and digital solutions exist, most focus on isolated services such as weather updates or product marketplaces. Very few provide an integrated environment where farmers can not only access predictive insights but also share knowledge, manage resources, and engage with a like-minded community.

Farmers also lack tools that are data-driven and personalised to their geography, soil conditions, and market behavior. This disconnect between available technology and user-centric design continues to hinder the potential for technology adoption in the agricultural sector.

C. Objective and Scope of Farmingo

This paper introduces **Farmingo**, a MERN and ML-integrated web platform that provides a unified solution to address the challenges above. Its key objectives are:

- To provide ML-based crop and fertilizer recommendations
- To offer a resource exchange platform (buy/rent farming tools)
- To enable farmers to share experiences via blogs and short videos
- To visualise market trends for better planning



- To eventually build a dataset for further agri-intelligence training

Farming is not just a product but a vision to transform farming into a more data-driven, collaborative, and profitable ecosystem.

II. LITERATURE REVIEW

A. Existing AgriTech Solutions

Various agri-tech applications have been introduced in India over the past decade, including Kisan Suvidha, AgriApp, and IFFCO Kisan, which offer services such as weather updates, mandi prices, and expert advice. However, these platforms often operate in silos, addressing isolated problems rather than providing an integrated ecosystem [2].

B. Smart Agriculture and Machine Learning

According to a report by NITI Aayog, the adoption of AI and ML in Indian agriculture remains low despite its transformative potential. Real-time data analysis using ML can significantly enhance crop yields, pest control, and resource optimization [3].

C. Identified Gaps

Despite progress in technology:

- Platforms often lack real-time personalization
- There's limited community support for peer learning
- Predictive systems aren't tightly integrated with user-friendly UIs
- Farmers often lack digital literacy to navigate complex tools

D. Farmingo's Contribution

Farming addresses these gaps through:

- ML-based crop and fertilizer recommendations
- A user-friendly dashboard built on the MERN stack
- A social platform for farmers to share blogs, shorts, and lessons
- Tools for visualizing price trends and managing resources

Unlike other tools, Farmingo combines learning, prediction, and commerce on a single platform, tailored specifically for Indian farmers.

III. METHODOLOGY

A. Overall Architecture

Farming adopts a decoupled architecture using the MERN stack (MongoDB, Express.js, React.js, Node.js) for the web application and integrates a Flask-based machine learning (ML) service. This approach ensures that the web operations and machine learning tasks are separated, allowing for increased flexibility, scalability, and modularity. The machine learning service is accessed through REST API calls, allowing the frontend to directly request predictions in real time, which streamlines the overall user experience.

B. Frontend (React.js)

The frontend of Farmingo is developed using React.js [4], providing a component-based architecture that allows the reuse of components such as:

- **Signup/Login Forms:** Handles user registration and authentication.

- **Dashboard:** Displays key information such as recommended crops, fertilizers, price predictions, and user's past activities.
- **Product Listings:** Shows products for rent or sale in a marketplace format.
- **Blog/Video Uploads:** Users can share farming tips and success stories.
- **ML Tools:** Provides interfaces for the crop, fertilizer, and price prediction tools.

Routing is managed by React Router v6, ensuring smooth navigation between different views. Axios and Fetch are used for making asynchronous requests to both the Express.js backend and Flask-based ML endpoints. The user interface is fully responsive, providing a seamless experience across both desktop and mobile platforms.

C. Backend (Node.js + Express.js)

The backend is built using Node.js [5] with Express.js to handle business logic, authentication, and interactions with the MongoDB database. RESTful APIs are responsible for:

- Managing user accounts and sessions.
- Handling CRUD operations for tools and products.
- Storing and retrieving lessons and videos.
- Implementing JWT-based authentication for secure login and session management.

The backend serves as the core of the application, processing incoming requests, interacting with the database, and serving the necessary data to the frontend.

D. Database (MongoDB)

Farming uses MongoDB, a NoSQL database, to store various types of data:

- **User Information:** Includes user details such as name, email, hashed password, and liked content.
- **Product Listings:** Contains details about tools and products available for rent or purchase, including price, rental status, and owner information.
- **Lessons and Videos:** Stores learning materials uploaded by users.

MongoDB's flexible schema allows easy storage of diverse data types, and collections are interconnected with embedded references for efficient querying and enhanced performance.

E. Machine Learning (Flask + Python)

The machine learning layer is built using Flask [6], which serves as the interface between the frontend and the ML models. The ML models predict crop suitability, fertilizer recommendations, and price predictions based on user inputs. The machine learning system consists of:

i. Crop Recommendation Model:

Objective:

The crop recommendation model suggests the most suitable crop based on environmental factors and soil nutrients.

Input Features:

- N (Nitrogen)
- K (Potassium)
- P (Phosphorus)
- Temperature
- Humidity
- pH
- Rainfall



Output:

The model predicts the crop name.

Algorithms:

Decision Tree Classifier: Achieved **90% accuracy**. Decision Trees [7] Use a hierarchical structure to split the data based on the input features, which helps to classify the crop accurately.

```
from sklearn.tree import DecisionTreeClassifier

DecisionTree = DecisionTreeClassifier(criterion="entropy", random_state=2, max_depth=5)

DecisionTree.fit(Xtrain, Ytrain)

predicted_values = DecisionTree.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
print("DecisionTree's Accuracy is: ", x*100)

DecisionTree's Accuracy is: 90.0
```

[Fig.1: Accuracy from the Decision Tree]

Naive Bayes Classifier: Achieved **99% accuracy**. The Naive Bayes [8] The algorithm assumes that all features are conditionally independent and applies Bayes' Theorem for classification.

```
from sklearn.naive_bayes import GaussianNB

NaiveBayes = GaussianNB()

NaiveBayes.fit(Xtrain, Ytrain)

predicted_values = NaiveBayes.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
print("Naive Bayes's Accuracy is: ", x*100)

Naive Bayes's Accuracy is: 99.0909090909091
```

[Fig.2: Accuracy from Naïve Bayes's]

Model Evaluation: The models are evaluated based on **accuracy** and **precision**, with Naive Bayes performing better due to its assumption of feature independence, which simplifies the learning process and leads to higher accuracy.

ii. Fertilizer Recommendation Model

Objective:

The fertilizer recommendation model suggests the optimal fertilizer for a particular crop based on various soil and environmental features.

Input Features:

- Temperature
- Humidity
- Soil Moisture
- Soil Type
- Crop Type
- N, K, P (Nutrient levels)

Output:

Recommended fertilizer.

Algorithm:

Random Forest Classifier: This model uses the **Random Forest** algorithm [9], which achieved an accuracy of **96.66%**. Random Forest performs well by training multiple trees on random subsets of the data and aggregating their outputs to increase accuracy and robustness.

```
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators=100, criterion = 'gini', random_state= 42)
classifier.fit(X_train, y_train)

RandomForestClassifier(random_state=42)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

y_pred = classifier.predict(X_test)

classifier.score(X_test, y_test)

0.9666666666666667
```

[Fig.3: Accuracy from Random Forest]

iii. Crop Price Prediction Model

Objective:

This model predicts the market price of crops, helping farmers make informed decisions about when to sell their produce.

Input Features:

- **Month:** The month in which the price is being predicted.
- **Commodity Name:** Name of the crop (e.g., Maize, Wheat, Rice).
- **Avg Modal Price:** The average market price.
- **Avg Min Price:** The lowest recorded price.
- **Avg Max Price:** The highest recorded price.
- **Calculation Type:** Price calculation method.
- **Change:** The price change percentage over time.

Output:

Predicted Market Price for all Months of the Selected Crop.

Algorithm:

Random Forest Regressor: The Random Forest algorithm [9], an ensemble method, achieved 93% accuracy. This algorithm works by creating multiple decision trees during training and outputs the average prediction from all the trees. This improves accuracy and reduces overfitting.

```
In [65]: model = RandomForestRegressor(n_estimators=100, random_state=42)
         model.fit(X_train, y_train)

Out[65]: RandomForestRegressor(random_state=42)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [66]: y_pred = model.predict(X_test)
         print("R² Score:", r2_score(y_test, y_pred))
         print("RMSE:", mean_squared_error(y_test, y_pred))

R² Score: 0.9324835347568879
RMSE: 1946386.984945914
```

[Fig.4: Accuracy from Random Forest]

Data Preprocessing:

- **Missing Data:** Imputation techniques or removal of incomplete data.
- **Feature Scaling:** Standardization or Min-Max scaling applied to features like temperature, pH level, and age.
- **Categorical Encoding:** One-Hot Encoding for categorical features such as crop type and tool brand.

Model Training:

- Models are trained using agricultural datasets, including crop suitability, soil conditions, and product pricing data.

Model Evaluation:

- Models are evaluated using metrics such as Accuracy, Precision, and Recall (for classification tasks), or MAE and RMSE (for regression tasks).

Model Deployment:

- **Flask serves the Models via API endpoints:**
- **/Predict-crop:** Returns crop recommendations.
- **/Recommend-fertiliser:** Returns fertiliser suggestions.
- **/Predict-price:** Returns the predicted price of farming tools.

Libraries Used:

- Scikit-learn [11] for model training and evaluation.



Farmingo: A MERN and ML-Integrated Platform for Smart, Community-Driven Agriculture

- Numpy & Pandas for data manipulation.
- Flask for serving the models.
- Joblib for model serialization.

F. Continuous Learning and Model Updates

To ensure the models remain effective, continuous learning is implemented through the following mechanisms:

- **User Feedback:** Users can provide feedback on predictions (e.g., whether the crop recommendation was accurate or the price prediction was realistic).
- **Data Updates:** Models are regularly retrained with new data to keep them up to date with changing farming conditions and market trends.

G. API Integration and Data Flow

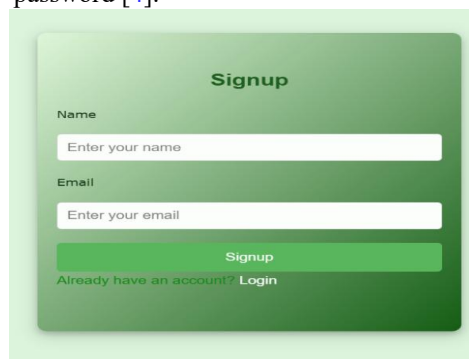
The communication flow between the frontend, backend, and Machine Learning [12] service is as follows:

- **Frontend → Flask (Machine Learning Prediction):** The frontend sends HTTP (Hyper Text Transfer Protocol) POST requests containing user data (e.g., environmental conditions, crop type, product details) to Flask API endpoints, which respond with crop, fertilizer, or price predictions.
- **Frontend → Express (CRUD (Create, Read, Update, Delete) operations):** Product-related data (e.g., brand, condition, location) is handled by the Express.js backend, interacting with the MongoDB database to manage product listings, tools for rent/sale, and user-generated content (e.g., lessons, shorts and videos).

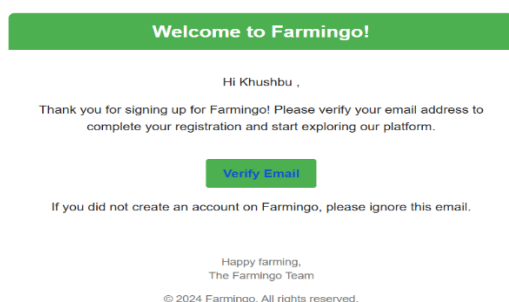
H. System Flow

Farmingo follows a precise, step-by-step user interaction flow from signup to accessing core features:

1. **Signup:** Users enter basic details (name, email), receive an email verification link, and set their password [4].

The image shows a green-themed 'Signup' form. It has input fields for 'Name' and 'Email', each with a placeholder 'Enter your name' and 'Enter your email' respectively. Below these is a green 'Signup' button. At the bottom, there is a link 'Already have an account? Login'.

[Fig.5: Signup Page]

The image shows an email verification page. At the top, it says 'Welcome to Farmingo!' in a green box. Below, it says 'Hi Khushbu,'. Then, it says 'Thank you for signing up for Farmingo! Please verify your email address to complete your registration and start exploring our platform.' There is a green 'Verify Email' button. At the bottom, it says 'If you did not create an account on Farmingo, please ignore this email.' and 'Happy farming, The Farmingo Team'.

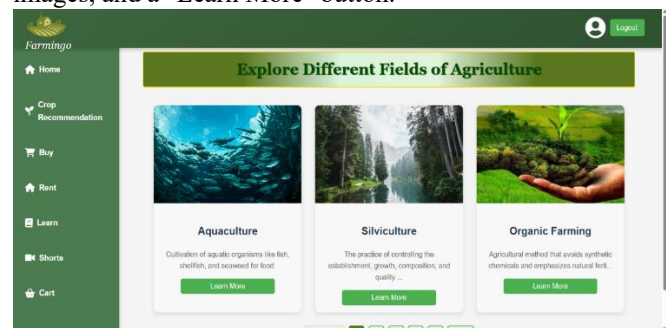
[Fig.6: Email Verification]

2. **Login:** Users log in to access the platform.

The image shows a green-themed 'Login' form. It has input fields for 'Email' and 'Password', each with a placeholder 'Enter your email' and 'Enter password' respectively. Below these is a green 'Login' button. At the bottom, there is a link 'Don't have an account? Signup'.

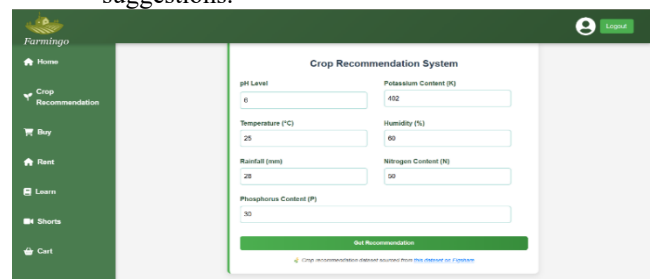
[Fig.7: Login Form]

Home Page: The top section displays a list of all types of agriculture (e.g., horticulture, aquaculture, floriculture, organic farming, etc.). Each category is presented in a card format, featuring a brief introduction, relevant icons or images, and a "Learn More" button.

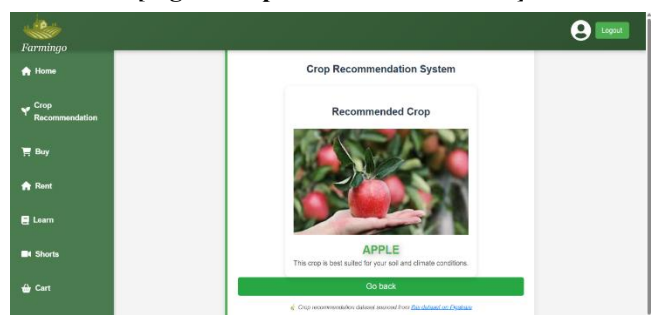
The image shows the home page of the Farmingo platform. It has a green header with the 'Farmingo' logo and a 'Logout' button. Below the header is a section titled 'Explore Different Fields of Agriculture'. It contains three cards: 'Aquaculture' (with an image of fish), 'Silviculture' (with an image of a forest), and 'Organic Farming' (with an image of a hand holding a plant). Each card has a brief description and a 'Learn More' button.

[Fig.8: Home Page]

3. **Crop Recommendation:** Users input environmental data, and the system returns crop suggestions.

The image shows the 'Crop Recommendation System' form. It has input fields for 'pH Level' (6), 'Potassium Content (K)' (402), 'Temperature (°C)' (25), 'Humidity (%)' (60), 'Rainfall (mm)' (28), 'Nitrogen Content (N)' (50), and 'Phosphorus Content (P)' (30). Below these is a green 'Get Recommendation' button. At the bottom, there is a link 'Crop recommendation based on your data. Click here to download the data'.

[Fig.9: Crop recommendation for]

The image shows the output of the 'Crop Recommendation System'. It displays a 'Recommended Crop' section with an image of an apple and the text 'APPLE'. Below this, it says 'This crop is best suited for your soil and climate conditions.' and a green 'Go back' button. At the bottom, there is a link 'Crop recommendation based on your data. Click here to download the data'.

[Fig.10: Crop Recommendation Form Output]

4. **Fertilizer Recommendation:** Users input crop and soil data, and the system returns suitable fertilizer recommendations.



[Fig.11: Fertilizer Recommendations Form]

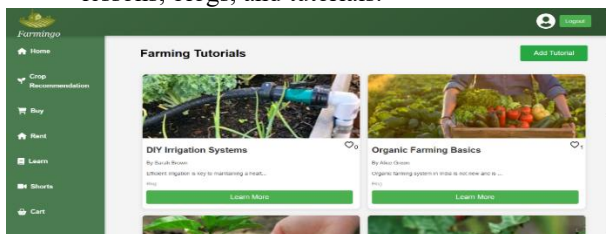
5. **Crop Price Prediction:** Users enter the crop name, and the system displays a line graph showing the predicted price trend of that crop across all months.

[Fig.12: Crop Price Prediction]

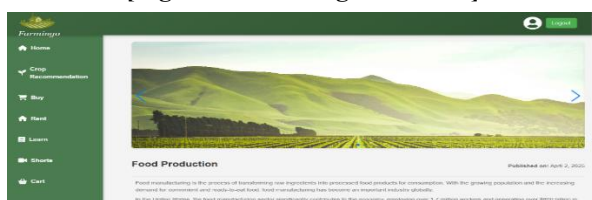


[Fig.13: Crop Price Prediction Output]

6. **Learning Module:** Users can upload, view, and like lessons, blogs, and tutorials.

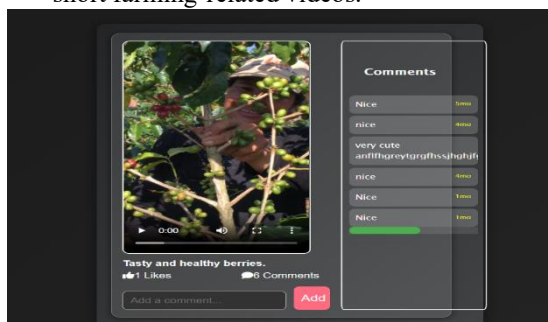


[Fig.14: Learn Blog and Video]



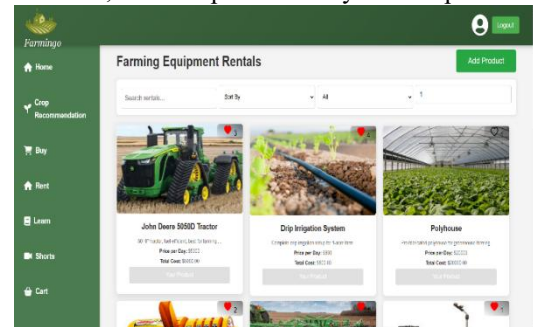
[Fig.15: Blog]

7. **Shorts:** Users can post, view, like, and comment on short farming-related videos.



[Fig.16: Shorts]

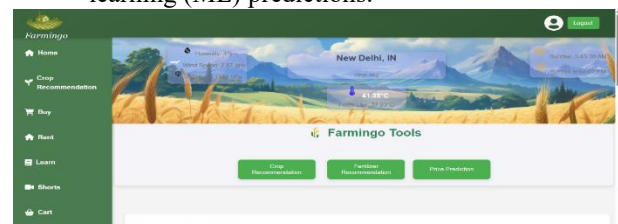
8. **Buy & Rent:** Users can add their products for sale or rent, explore other users' listings, rent or buy items, and like products they find helpful.



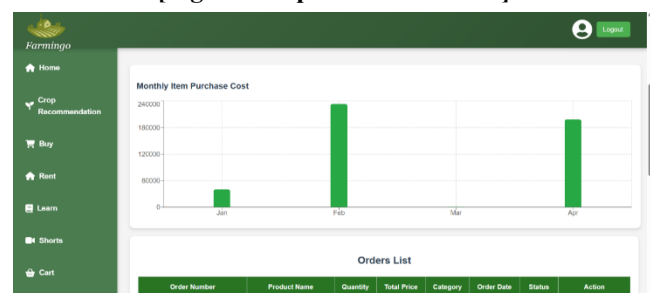
[Fig.17: Product for Rents]

[Fig.18: Add Product for Rent]

9. **Dashboard:** Users can view their activity, including purchased and rented tools, as well as machine learning (ML) predictions.



[Fig.19: Crop Price Prediction]



[Fig.20: Month Wise Sale]

Order Number	Product Name	Quantity	Total Price	Category	Order Date	Status	Action
67e6d77a6b3059188161638f	Drip Irrigation Kit	1	¥1000	Irrigation	4/3/2025	Ordered	Mark as Shipped
67e6d77a6b3059188161638e	Organic Wheat Seeds	1	¥2000	Seeds	4/3/2025	Ordered	Mark as Shipped
67e6d77a6b3059188161638d	Organic Tomato Seeds	1	¥1500	Seeds	4/3/2025	Ordered	Mark as Shipped

Rental Number	Product	Quantity	Rental Days	Cost	Rental Date	Status	Action
67e6d77a6b3059188161638f	John Deere 5050 Tractor	1	1	¥5000	4/3/2025	Ordered	Mark as Shipped
67e6d77a6b3059188161638e	Polyhouse	1	1	¥20000	4/3/2025	Ordered	Mark as Shipped

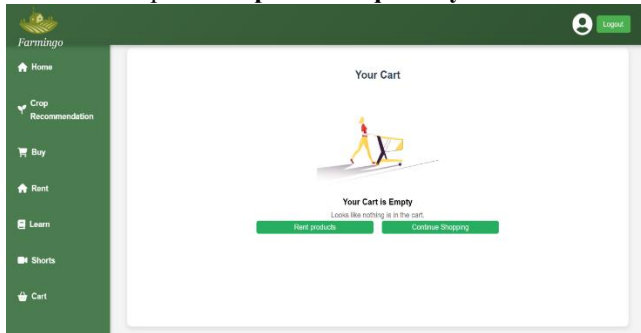
[Fig.21: List of Items for Rent and Buy]

10. **Cart:**
If the cart is empty, a message like "Empty

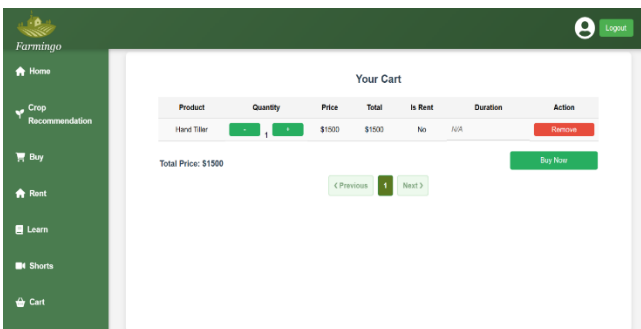
Cart” is shown.

If it contains items, a **tabular format** displays:

- List of items
- Product name, price, quantity
- An option to **update the quantity** for each item.



[Fig.22: Empty Cart]



[Fig.23: Cart with Item]

I. Tools and Technologies Used

Category	Technologies / Tools
Frontend	React.js, Axios, React Router
Backend	Node.js, Express.js, JWT (JSON Web Token)
ML Integration	Python, Flask, Scikit-learn
Database	MongoDB (Atlas or Local)
Deployment	Vercel (Frontend), Render (Backend)
Testing	Postman, MongoDB Compass, VS Code

IV. RESULTS AND EVALUATION

A. System Usability

Initial testing of Farmingo was conducted through simulated user interactions and internal feedback. The platform was evaluated based on its responsiveness, ease of navigation, and accessibility of features, including the marketplace, machine learning tools, and learning modules.

Feedback from testers highlighted the intuitive layout and real-time nature of crop predictions as highly impactful. The integration of social features such as blogs, videos, and shorts increased engagement and knowledge exchange among users.

B. Machine Learning Performance

Two ML models were deployed and tested using small but diverse datasets:

Model	Algorithm	Accuracy	Notes
Crop Recommendation	Naive Bayes	99%	Based on soil pH, rainfall, humidity, and temperature
Fertilizer Recommendation	Random Forest	96.66%	Trained on crop type, soil condition, and nutrient levels
Crop Price Prediction	Random Forest	93%	Predicts month-wise crop price trends using historical market data

Both models returned predictions within ~1 second via Flask API, making them suitable for real-time use. While the models are functional, larger datasets and region-specific training could further improve accuracy.

C. Functional Coverage

Farmingo’s major features performed reliably in internal tests:

- Add/Buy/Rent flow → tested with multiple products
- ML tool integration → consistent predictions and no server errors
- Blog/Shorts upload → verified with text and media content
- Dashboard tracking → dynamic display of liked, rented, and listed items

D. Observed Limitations

- No real-time weather or soil data integration (user must input manually)
- Models trained on limited datasets; lack of region-wise tuning
- Rural users may require offline support or language localization

These limitations are acknowledged and targeted for future enhancement.

V. DISCUSSION

A. Strengths of the Platform

Farmingo successfully demonstrates how a combination of machine learning and full-stack web technologies can solve multiple pain points in the agricultural sector. Key strengths of the platform include:

- **Integrated Functionality:** Unlike many existing apps that focus only on weather or market prices, Farmingo provides a unified platform where farmers can learn, share, transact, and make data-driven decisions.
- **Direct ML Integration with Frontend:** The ability to call ML models directly from the frontend ensures a seamless and interactive user experience.
- **Community Features:** The inclusion of blogs, short videos, and comment sections fosters peer-to-peer learning and farmer engagement.
- **Modular Design:** The architecture enables individual modules (e.g., prediction, rental) to be updated or expanded independently, resulting in a highly scalable system.

B. Key Learnings

During development, several crucial technical and domain insights emerged:

- **ML model selection** should be based on both accuracy and real-time performance.
- **Flask APIs**, when directly integrated with the frontend, require CORS and security handling.
- Farmers often need **visual, simple interfaces**, so UI design must minimize complexity.
- Testers found that combining ML predictions with marketplace



functionality helped them make faster and better-informed decisions.

C. Limitations

While the platform shows strong promise, a few limitations were observed:

- **Static ML (Machine Learning) inputs:** The system currently depends on user-entered parameters. Integration with real-time soil and weather APIs could improve usability and accuracy.
- **Limited dataset scope:** ML (Machine Learning) models are trained on generalized datasets; regional-specific datasets would enhance precision.
- **Language barrier:** The platform is currently only available in English; multilingual support will be essential for widespread adoption in rural areas.

D. Comparison with Existing Tools

Compared to standalone apps like AgriApp or Kisan Suvidha, Farmingo [10] offers a broader ecosystem where:

- Knowledge sharing is two-way (not just reading from experts)
- ML(Machine Learning) predictions are real-time and interactive
- Resource management (buy/rent/list) is built into the same system

This makes it not just an information app but an innovative farming ecosystem.

VI. CONCLUSION AND FUTURE WORK

A. Conclusion

Farmingo presents a modern, user-centric solution to some of the most pressing challenges faced by Indian farmers. By integrating machine learning directly into a full-stack MERN web application, it offers intelligent crop and fertilizer recommendations, a dynamic equipment marketplace, and a socially driven knowledge-sharing environment.

Unlike existing tools that are feature-specific or static, Farmingo offers a unified experience that is interactive, real-time, and adaptable. Its modular design and direct ML integration demonstrate that accessible, data-driven agriculture is both feasible and scalable, even in resource-constrained rural settings.

Initial testing shows strong usability and system stability, with both technical feedback and feature reception being positive. The inclusion of short-form videos and lessons enhances community engagement and paves the way for grassroots agri-education.

Shortly, the real-time data collected through user interactions on the platform can be utilized to train more advanced machine learning models. This evolving dataset has the potential to support automated decision-making and move us closer to achieving fully automated, intelligent farming systems.

B. Future Work

The future roadmap for Farmingo includes several enhancements aimed at improving scalability, accuracy, and inclusivity:

- **Crop Disease Detection:** Incorporate image-based deep learning models for early-stage disease identification using farmer-uploaded leaf photos.
- **Weed Recognition & Treatment:** Use object detection to identify and classify common weeds in crop fields.
- **Real-Time Weather & Soil Integration:** Connect with government or third-party APIs to fetch live soil moisture, rainfall, and temperature data for more accurate predictions.
- **Government Scheme Integration:** Automatically show relevant schemes and subsidies to users based on their crop and region.
- **Multilingual Support & Offline Mode:** To improve adoption in rural areas, Farmingo will support regional languages and offline data caching.
- **Data-Driven AI (Artificial Intelligence) Growth:** Use anonymized farmer inputs to build a real-time agricultural dataset, which can be used to train more innovative and region-specific ML models.

With these developments, Farmingo aspires to evolve into a full-fledged **agri-intelligence assistant**, reshaping the future of farming in India through innovative, collaborative, and accessible technology.

DECLARATION STATEMENT

After aggregating input from all authors, I must verify the accuracy of the following information as the article's author.

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Funding Support:** This article has not been funded by any organizations or agencies. This independence ensures that the research is conducted with objectivity and without any external influence.
- **Ethical Approval and Consent to Participate:** The content of this article does not necessitate ethical approval or consent to participate with supporting documentation.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Research Involving Human Participants and/or Animals:** This article does not contain any studies with human participants or animals performed by any of the authors. The data used in the study were anonymised and obtained with the necessary institutional permissions.
- **Author's Contributions:** The authorship of this article is contributed equally to all participating individuals.

REFERENCE

1. NABARD, "NABARD RESEARCH STUDY - 41," NABARD, 2023. <https://www.nabard.org/auth/writereaddata/tender/1201243818assessing-the-state-of-affairs-in-indian-agriculture-with-a-focus-on-credit-insurance-and-storage-marketing.pdf>
2. Ifco, "kisan-sanchar-limited," [Online]. Available: <https://www.ifco.in/en/kisan-sanchar-limited>. [Accessed 2025]. <https://www.ifco.in/en/kisan-sanchar-limited>
3. N. Aayog, "Intelligent Inputs Revolutionising Agriculture," NITI Aayog, 2021. <https://www.niti.gov.in/artificial-intelligence-revolutionising-agriculture>
4. React, "React," [Online]. Available: <https://reactjs.org>.
5. Node.js, [Online]. Available: <https://nodejs.org>. [Accessed 2025]. <https://nodejs.org/en>



6. "Flask," [Online]. Available: <https://flask.palletsprojects.com>.
7. J. R. Quinlan, "Decision Tree," Induction of Decision Trees, vol. Machine Learning, pp. 81-106, 1986.
DOI: <https://doi.org/10.1007/BF00116251>
8. N. Bayes, "Zhang, H.," The Optimality of Naive Bayes, pp. 562-567, 2004.
<https://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf>
9. L. Breiman, "Random Forests," vol. 45, pp. 5-32, 2001.
DOI: <https://doi.org/10.1023/A:1010933404324>
10. "Kisan-sanchar-limited," Iffco, [online]. Available: <https://www.iffco.in/en/kisan-sanchar-limited>.
11. Scikit-learn Developers," [Online]. Available: <https://scikit-learn.org>.
12. Mishra AD, Thakral B, Jijja A, Sharma N. Real-time Vital Signs Monitoring and Data Management Using a Low-Cost IoT-based Health Monitoring System. Journal of Health Management. 2024;26(3):449-459. DOI: <https://doi.org/10.1177/09720634241246926>

AUTHOR'S PROFILE



Dr. Antim Dev Mishra, MTech, PhD, is an academic professional with 12+ years of teaching and 4 years of industry experience in IoT, ML, VLSI, Data Analytics, Embedded Systems, and Computer Science. Currently, Associate Professor and Deputy Director at IITM Group of Colleges, Janakpuri, Delhi (affiliated with MDU/GGSIPU), he leads curriculum development and the Incubation and Innovation Cell. His research on IoT-based health monitoring has resulted in 12 journal articles, 16 conference papers, and 8 Indian patents. As a former Joint Program Director at Sushant University and an NTA Observer, he is passionate about integrating technology, academia, and community solutions.



Ms Khushbu Jha, pursuing a B.Tech in Computer Science and Engineering (CSE) from Maharishi Dayanand University (8 CGPA), has completed internships in web development, data science, and SaaS applications. My technical skills include web development (HTML, CSS, JavaScript, React, Node.js, Express, MongoDB), data science (Python, Beautiful Soup, Pandas, etc.), and data structures and algorithms (DSA) with C++. Projects include a social media cooking app, a quiz app, and ML-based Parkinson's detection and car price prediction models. I have also contributed to the placement cell and organized a hackathon. I am enrolled in the Full Stack Career Program by Skill Academy (Textbook) to enhance my skills further.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of the Lattice Science Publication (LSP)/ journal and/ or the editor(s). The Lattice Science Publication (LSP)/ journal and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.