



IoT-Enabled Smart Plant Ecosystem Integrating Real-Time Monitoring, Automated Irrigation, and Intelligent Growth Optimization Frameworks

Md. Safaet Hossain^{1*}, Israt Jahan², Jawata Afnan³, Israt Sultana Tanny⁴ Nashid Sultana Mim⁵, Kayes Mahmood⁶

^{1*} Associate Professor. Department of Computer Science and Engineering; Director, IQAC; University of Information Technology and Sciences. Bangladesh

²³⁴⁵⁶ Department of Computer Science and Engineering. University of Information Technology and Sciences. Bangladesh

ARTICLE INFO

Received May 08, 2026

Accepted June 27, 2026

Published June 30, 2026

Keywords:

Digital Twin; Machine

Learning; Predictive

Maintenance; Smart

Mechanical Systems; Industry

4.0

ABSTRACT

Urban plant care is increasingly important for sustainable living, but many users face inconsistent watering, insufficient care knowledge, unsuitable plant selection and delayed disease recognition. This study presents Easy Grow Plants, an integrated web and Internet of Things (IoT) ecosystem that connects plant care guidance, soil-moisture monitoring, automated watering, plant recommendation, image-based plant health assistance, marketplace functions, community interaction and plant exchange. The prototype was implemented using a React frontend, Django REST backend, SQLite database and an Arduino UNO R4 WiFi smart pot with a soil moisture sensor, relay module and DC water pump. Functional, interface, API, IoT connectivity, sensor calibration, watering control and LAN deployment tests were conducted. The results show that the core modules operated together as an integrated academic prototype. The system demonstrates a practical foundation for smart urban gardening, although cloud deployment, multi-device testing and stronger AI validation remain future improvements

How to cite: Hossain, M. S., Jahan, I., Afnan, J., Tanny, I. S., Mim, N. S., & Mahmood, K. (2026). IoT-Enabled Smart Plant Ecosystem Integrating Real-Time Monitoring, Automated Irrigation, and Intelligent Growth Optimization Frameworks. *TechComp Innovations: Journal of Computer Science and Technology*, 3(1), 66–77. <https://doi.org/10.70063/techcompinnovations.v3i1.198>



This is an open-access article distributed under the Creative Commons 4.0 Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. ©2023 by author.

* Corresponding author: safaet_hossain@uits.edu.bd

INTRODUCTION

Urban plant care has become an important part of sustainable living, indoor environmental quality and household greenery. Many people now grow plants on balconies, rooftops, offices and small indoor spaces, yet the available space, time and practical knowledge are often limited. Plant care therefore becomes a continuous decision-making problem rather than a

simple decorative activity. A plant owner must decide when to water, where to place the plant, how to select a suitable plant and how to identify early disease symptoms. When these decisions are made only through memory or visual observation, plants may suffer from overwatering, underwatering, unsuitable light exposure or untreated disease symptoms.

The first major problem addressed in this study is watering inconsistency. Both overwatering and underwatering can weaken roots and reduce plant growth. IoT-based irrigation studies show that soil moisture sensors can support watering decisions based on actual soil condition rather than fixed routine (Boukri et al., 2026). Other monitoring systems use sensor readings to observe plant condition and reduce manual effort. These findings support the use of sensor-based monitoring in a plant care platform.

A second problem is the limited care knowledge of beginner plant owners. Users may buy a plant because of appearance but later discover that the plant does not match the available sunlight, balcony direction, humidity or indoor condition. Plant selection and care guidance must therefore be connected with the purchase and maintenance process. Online plant marketplace research indicates that factors such as price, quality, time-saving, convenience, and reliable delivery services significantly influence consumer decisions when purchasing ornamental plants (Man, 2023).

Plant disease identification creates an additional challenge. Leaf spots, yellowing, wilting and discoloration may occur because of disease, pests, nutrient deficiency or watering error. Manual diagnosis is difficult for non-specialist users. Deep learning and computer vision studies show that plant leaf images can be used to support disease classification and early diagnosis further shows that disease recognition becomes more useful when it is connected with recommendation support (Islam et al., 2025)

The literature shows strong progress in individual areas such as IoT irrigation, plant monitoring, deep learning-based disease detection, and online plant commerce (Chaudhary et al., 2023). However, these solutions are often fragmented (Kumari et al., 2025). A smart irrigation device may monitor soil moisture but may not provide plant marketplace or community support. A disease detection model may classify leaves but may not guide watering or plant selection. An online plant marketplace may support purchasing but may not monitor plant condition after purchase. This fragmentation creates the research gap addressed in this work.

Easy Grow Plants was developed as a unified ecosystem for intelligent plant engagement and growth. The platform integrates a React-based web interface, a Django REST backend, database services, and an Arduino UNO R4 WiFi-based smart plant pot. It combines Plant Care Guide, Smart Plant Finder, Plant and Disease Detection, marketplace, cart, checkout, Community Echo, Plant Exchange, and Device Management Dashboard. The IoT device collects soil moisture data and supports manual or automatic watering through a relay-controlled DC pump.

The objective of this study is to design, implement, and evaluate an integrated IoT-enabled plant care ecosystem for urban plant owners, beginner gardeners, plant buyers, sellers, and plant enthusiasts. The main contributions are: (1) integration of plant care, commerce, community, and IoT monitoring in one platform; (2) implementation of sensor-supported watering through a smart plant pot; (3) inclusion of plant recommendation and AI-assisted disease support; (4) secure dashboard-based device management; and (5) validation through functional, API, IoT, sensor, watering, and deployment testing.

The remainder of this article is organized as follows. The Method section describes the research design, system architecture, IoT workflow, hardware design, sensor calibration,

smart features, security, and deployment design. The Result and Discussion section presents prototype evaluation, testing outcomes, comparison with prior studies, limitations, and implications. The Conclusion section summarizes the contribution and identifies future research directions.

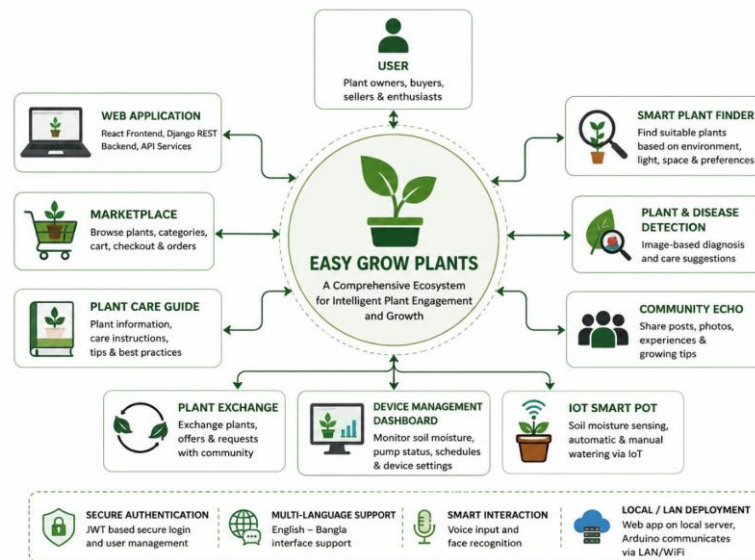


Figure 1. Overview of the Easy Grow Plants ecosystem

RESEARCH METHOD

This study used a design-and-development research approach. The goal was not only to evaluate an isolated algorithm or hardware component, but to construct a functional full-stack ecosystem that connects web services, database models, IoT telemetry, and plant care support. The system was developed as an academic prototype and evaluated through module-level and integration-level testing.

Requirement analysis was derived from the problems identified during the project analysis: watering inconsistency, lack of care knowledge, disease identification difficulty, unsuitable plant selection, fragmented plant commerce, limited community support, and limited real-time monitoring. These requirements were translated into software modules, backend services, database entities, and IoT functions. The functional modules included authentication, marketplace, cart, checkout, Plant Care Guide, Smart Plant Finder, Plant and Disease Detection, Community Echo, Plant Exchange, and Device Management Dashboard.

The system architecture was divided into four connected layers: user interface, backend/API, database, and IoT hardware. The user interface layer was implemented with React and supported navigation across plant care, marketplace, community, exchange, and device pages. The backend layer used Django and Django REST Framework to manage authentication, business logic, user-specific data, marketplace transactions, and device communication. The database layer stored users, products, cart records, orders, community posts, devices, readings, and watering logs. The IoT layer used an Arduino UNO R4 WiFi, soil moisture sensor, relay module, and DC water pump.

The control logic followed a telemetry-and-command workflow. The soil moisture sensor captured plant condition data. The Arduino processed the sensor reading, converted it into a user-friendly moisture percentage and sent telemetry to the Django backend through HTTP over WiFi. The backend stored readings and served the latest status to the Device

Management Dashboard. For manual watering, the user pressed a Water Now control in the dashboard; the backend validated the authenticated user and device ownership before forwarding a command to the Arduino. For automatic watering, the system used moisture threshold, schedule time, last watering time, and cooldown logic.

The hardware design separated sensing, control, and pump power for safety. The Arduino received the moisture sensor signal and controlled the relay input, while the pump was powered by an external source rather than directly from the Arduino. This design reduced the risk of overloading the microcontroller. The relay worked as an electrical switch between the low-voltage control side and the pump circuit. The firmware initialized the pump in the off state, limited pump runtime and applied cooldown control to avoid repeated rapid activation.

Sensor calibration was required because raw analog readings are not meaningful to ordinary users. The project used calibrated dry and wet values to map raw readings into moisture percentage. The prototype used DRY_RAW = 375 and WET_RAW = 310 as calibration values. To reduce unstable readings, the firmware used multiple samples, median filtering, and Exponential Moving Average smoothing. These steps made the dashboard value more stable and suitable for threshold-based watering decisions.

The smart feature design connected plant knowledge with user inputs. The Plant Care Guide provided practical guidance on watering, sunlight, soil, fertilizer, and general care. The Smart Plant Finder used balcony direction, sunlight duration, and plant preference to recommend suitable plants. The Plant and Disease Detection module provided image-based plant health assistance. The marketplace and community design connected plant buying with long-term care support, while Plant Exchange supported sustainable sharing among users.

Security and deployment were also considered in the method. JWT authentication protected user accounts and routes. Device ownership checking prevented one user from controlling another user's device. The prototype was deployed on a local area network (LAN) for academic demonstration. The Django server needed to run on a LAN-accessible address so that the Arduino and browser could communicate with the same backend. Cloud hosting, large-scale device management, mobile application, and push notification support were left for future work.

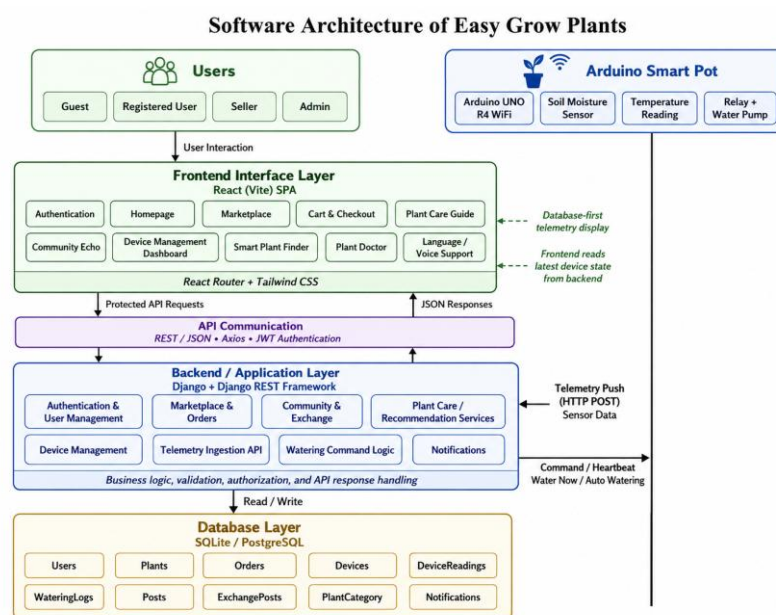


Figure 2. Software architecture of Easy Grow Plants

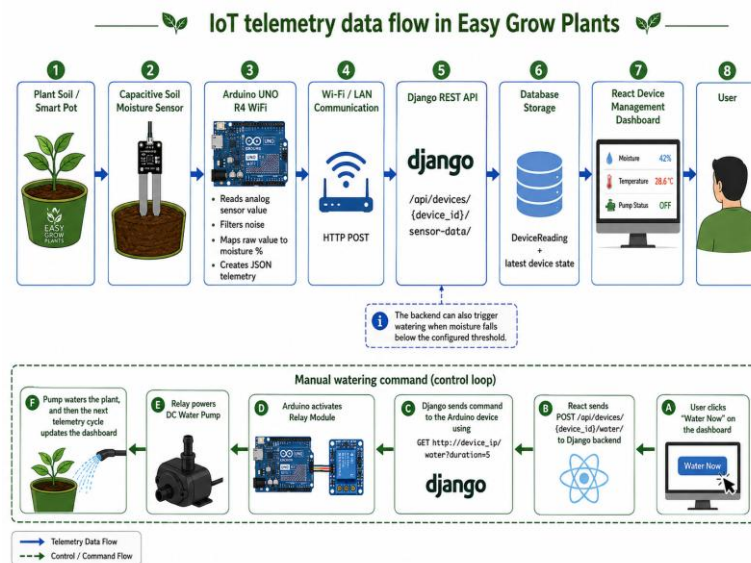


Figure 3. IoT telemetry data flow and watering workflow

Table 1. Main system layers and responsibilities

Layer	Main Components	Responsibility
User interface	React, Vite, Tailwind CSS, React Router, Axios	Provides pages and interaction for plant care, marketplace, community and device dashboard.
Backend/API	Django, Django REST Framework, Simple JWT	Handles authentication, business logic, API routing and IoT communication.
Database	SQLite and Django ORM models	Stores users, plants, products, orders, devices, readings and watering logs.
IoT hardware	Arduino UNO R4 WiFi, soil moisture sensor, relay, DC pump	Collects moisture data and performs manual or automatic watering.
Deployment	Local/LAN server and browser access	Supports prototype demonstration and communication between browser, backend and Arduino.

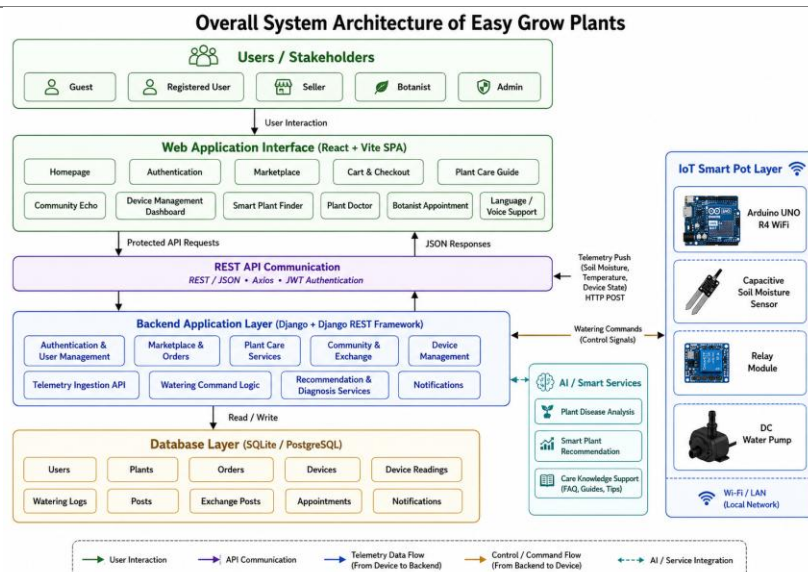


Figure 4. Overall system architecture of Easy Grow Plants

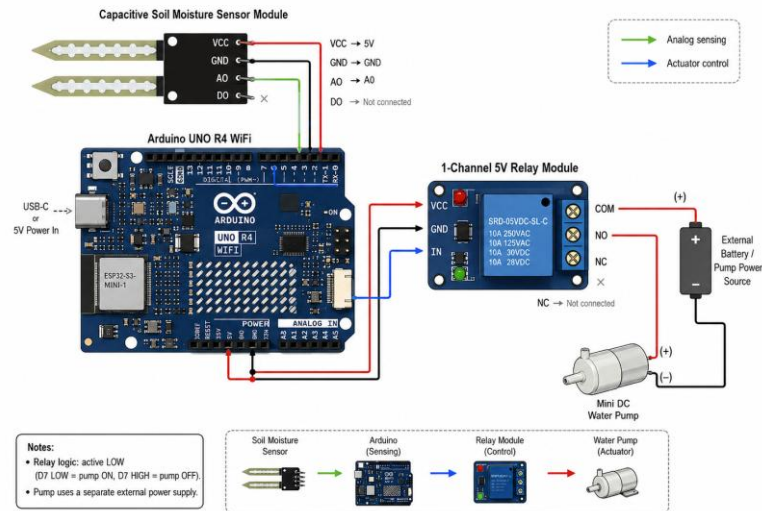


Figure 5. Circuit design of the Arduino smart plant watering device

Table 2. Sensor calibration and watering safety design.

Design element	Implementation role
Dry and wet calibration values	Convert raw analog sensor reading into moisture percentage.
Multiple sampling	Reduces dependence on a single unstable reading.
Median filtering	Removes sudden outlier readings from the sensor stream.
EMA smoothing	Makes displayed moisture values change gradually.
Moisture threshold	Supports automatic watering decision.
External pump power	Prevents pump load from being supplied directly by Arduino.
Runtime and cooldown limits	Avoids overwatering and repeated rapid pump activation.
Ownership validation	Ensures authenticated users control only their registered devices.

System Implementation

The prototype implementation followed the system design described above. The frontend was implemented using React components and route-based navigation. Main user-facing interfaces included the home page, registration page, login page, marketplace, cart, checkout, Plant Care Guide, Smart Plant Finder, Plant and Disease Detection, Community Echo, Plant Exchange, Plant Doctor and Device Management Dashboard. The interface also supported English-Bengali access and voice-based interaction for better usability.

The backend was implemented using Django and Django REST Framework. API groups were created for authentication, marketplace, cart and order handling, plant care features, community and exchange records, device registration, device readings and watering logs. JSON Web Token authentication was used to protect routes and user-specific data. SQLite was used as the development database because it is simple and suitable for a local academic prototype.

The IoT firmware was implemented on Arduino UNO R4 WiFi. The device read soil moisture sensor values, processed the readings, sent telemetry to the backend and responded

to watering commands. The dashboard displayed the latest soil moisture value, pump status, auto-watering setting, schedule information and connectivity status. Manual watering and automatic watering were implemented as controlled workflows rather than uncontrolled pump switching.

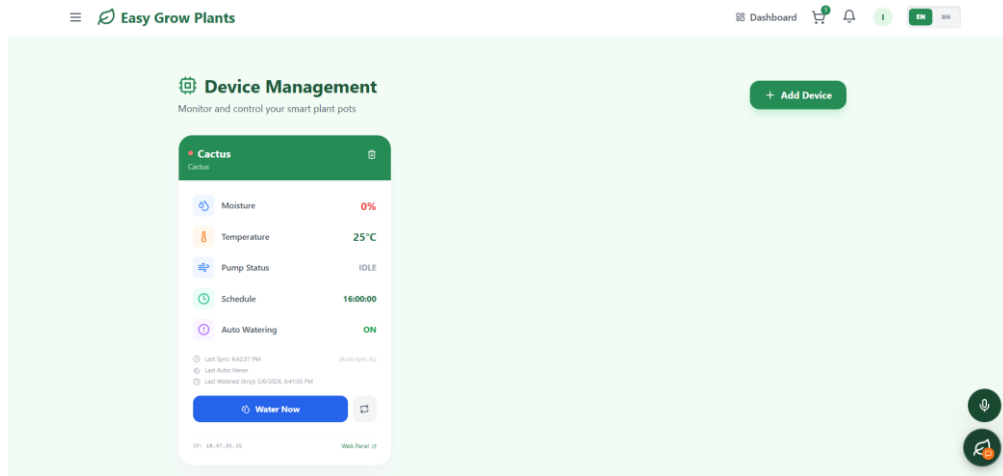


Figure 6. Device Management Dashboard interface

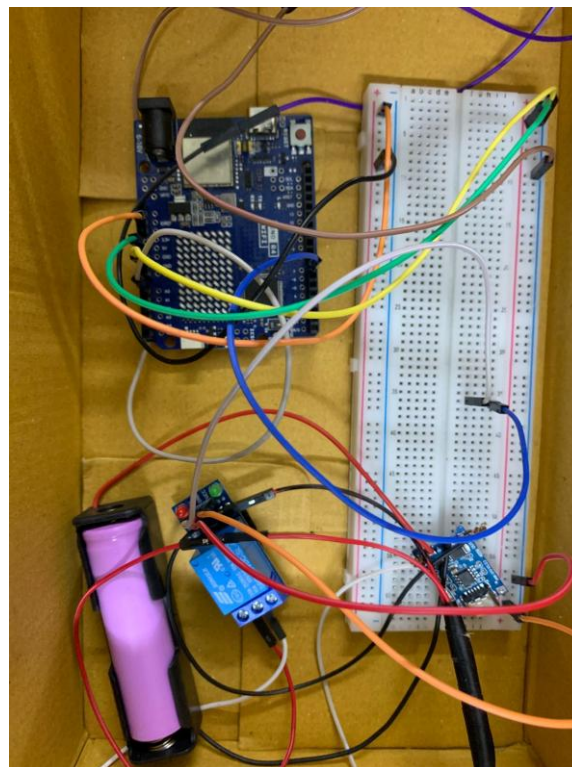


Figure 7. Arduino smart pot hardware connection

RESULT AND DISCUSSION

The prototype was evaluated through functional testing, interface testing, backend/API verification, IoT connectivity testing, sensor calibration review, watering control testing, and LAN deployment analysis. The evaluation focused on whether the integrated modules worked together as a complete academic prototype. Since the project is an engineering design

prototype, the results emphasize functional correctness, module integration, data flow, and practical limitations rather than statistical hypothesis testing.

Frontend testing showed that the major user-facing pages loaded and responded as expected. Registration and login supported authenticated access, while protected pages required valid user sessions. Marketplace pages allowed plant browsing, searching, cart updates, and the checkout workflow. These results are important because plant commerce is part of the complete care journey rather than a separate module.

Plant care and smart feature testing confirmed that the Plant Care Guide, Smart Plant Finder, Plant and Disease Detection, Community Echo, and Plant Exchange modules were accessible from the user interface. The Smart Plant Finder supported simple user inputs such as direction and sunlight duration. The disease detection interface allowed image-based input and returned plant health assistance. This implementation is consistent with previous disease detection and recommendation studies, which show that image-based models and recommendation support can assist users (Ahad et al., 2023; Bandi et al., 2023; Borhani et al., 2022; Carion et al., 2020). Backend and database testing confirmed that the Django REST APIs handled requests and returned structured responses. Authentication, marketplace, device, community, and plant care APIs were tested for expected responses. Database models stored the major entities required by the platform, including user accounts, products, device readings, and watering logs. This backend structure allowed the IoT device, dashboard, and marketplace modules to work through one service layer.

IoT testing demonstrated the main value of the project. The Arduino smart pot connected to WiFi, collected soil moisture readings, and transmitted telemetry to the backend. The dashboard could display plant condition data, while watering commands activated the relay-controlled pump. Manual watering worked through a dashboard-to-backend-to-device command path. Automatic watering used moisture threshold, schedule, and cooldown conditions. This supports the literature that soil moisture-based irrigation can reduce manual effort and improve watering decisions (Alshammari et al., 2022; Chougui et al., 2022; Chowdhury et al., 2023). Calibration testing showed that raw sensor values needed mapping and filtering before being useful to users. The use of dry/wet calibration values, multiple samples, median filtering, and EMA smoothing produced a more understandable moisture percentage. This is an important design decision because ordinary users are unlikely to interpret raw analog values correctly. Presenting moisture percentage on the dashboard makes the system more practical for non-technical plant owners.

LAN deployment testing showed that the prototype could operate in a local network environment. The browser accessed the web interface, and the Arduino communicated with the Django backend through the LAN IP. However, this design also exposed a limitation: the Arduino and server must remain on the same network. If the IP address changes, the firmware or configuration may need adjustment. This limitation supports the future need for cloud deployment and remote device communication.

Compared with previous studies, Easy Grow Plants extends the scope of single-purpose plant care systems. IoT irrigation studies mainly focus on watering (Alshammari et al., 2022; Chougui et al., 2022; Chowdhury et al., 2023), disease detection studies mainly focus on classification or model accuracy (Ahad et al., 2023; Bandi et al., 2023; Carion et al., 2020), and online plant sales studies mainly focus on e-commerce design and customer experience (Caron et al., 2021). The present system combines these functions into one ecosystem, allowing plant monitoring, watering, care guidance, buying, disease assistance, community discussion, and plant exchange to support one complete user workflow.

The prototype still has limitations. Testing was conducted primarily in a local/LAN environment and at an academic demonstration scale. Single-device testing limits the ability to claim large-scale reliability. Low-cost soil moisture sensors can vary depending on placement, soil type, and water distribution. The Plant and Disease Detection module should also be validated with stronger datasets and real-world plant images before being treated as a robust diagnostic tool. These limitations were reported honestly to avoid overstating the system's readiness for production deployment.

Table 3. Overall testing strategy and observed result.

Test area	Purpose	Observed result
Frontend and authentication	Check page loading, login, registration and protected routes.	Major pages and authentication workflow operated as expected.
Marketplace workflow	Verify browsing, cart and checkout functions.	Product browsing, cart update and checkout pages worked in prototype form.
Plant care and smart features	Check Plant Care Guide, Smart Finder, disease support and community modules.	Modules were accessible and supported the intended user workflow.
Backend/API/database	Validate API response and data storage.	Django REST APIs returned structured responses and stored required data.
IoT device and dashboard	Verify WiFi connection, telemetry and dashboard status.	Arduino sent moisture data and dashboard showed device status.
Watering control	Test manual and automatic watering logic.	Relay-controlled pump activation worked with safety constraints.
LAN deployment	Test local network communication.	Browser and Arduino communicated with backend through LAN configuration.

Table 4. Problems encountered and applied solutions.

Problem	Impact on system	Applied solution
LAN IP dependency	Arduino could not communicate if backend used localhost only.	Backend configured to run on LAN-accessible address.
Sensor jitter	Dashboard moisture value could fluctuate.	Multiple sampling, median filtering and EMA smoothing were used.
Pump power limitation	Direct pump supply from Arduino could damage the board.	External pump power and relay switching were used.
Repeated watering risk	Overwatering could occur if pump activated too frequently.	Controlled runtime and cooldown logic were applied.

User-device access risk	Unauthorized device control could occur.	JWT authentication and ownership validation were implemented.
Prototype deployment limitation	System could not yet support remote access.	Cloud deployment was identified as future work.

CONCLUSION

This study presents Easy Grow Plants as an integrated IoT-enabled ecosystem for smart plant care, automated watering, and intelligent plant engagement. The system addressed the fragmentation found in many existing solutions (Soibam, 2025) by connecting plant care guidance, smart plant recommendation, image-based plant health assistance, marketplace services, community interaction, plant exchange, and IoT-based device monitoring within one platform.

The developed prototype achieved the main research objectives. It implemented a React-based frontend, Django REST backend, database layer, and Arduino UNO R4 WiFi-based smart plant pot. It supported soil moisture monitoring, manual watering, automatic watering, device dashboard visualization, marketplace workflow, Plant Care Guide, Smart Plant Finder, Plant and Disease Detection, Community Echo, and Plant Exchange. Testing confirmed that the core modules worked together as an integrated academic prototype.

The work advances the field by demonstrating that urban plant care can be treated as a complete ownership workflow rather than a collection of isolated digital tools. The project also shows the practical value of combining IoT monitoring with web services, care knowledge, e-commerce, and social engagement. Future work should focus on cloud deployment, mobile application support, push notifications, multi-device scalability, weather integration, improved hardware packaging, stronger disease detection validation, and machine learning-based watering prediction.

LIMITATIONS AND FUTURE RESEARCH RECOMMENDATIONS

This study has several limitations that should be considered when interpreting the findings. The research primarily relied on qualitative content analysis and secondary data sources instead of direct industrial experimentation or quantitative validation. Consequently, the proposed framework may not fully represent the operational complexity of all industrial environments and mechanical systems. In addition, challenges related to data interoperability, cybersecurity protection, and computational latency remain significant barriers to the large-scale implementation of intelligent digital twin systems. The dependence on existing literature may also limit the generalizability of the conclusions across different industrial sectors and technological infrastructures.

Future research should focus on developing quantitative experimental models and real-world industrial validation for intelligent digital twin frameworks. Researchers are encouraged to explore advanced machine learning approaches such as deep reinforcement learning, federated learning, and explainable artificial intelligence for predictive maintenance applications. Further studies should also investigate cybersecurity enhancement mechanisms, edge computing optimization, and sustainable industrial data management strategies. Comparative investigations across manufacturing, transportation, and energy sectors may

provide deeper insights into the adaptability and scalability of intelligent predictive systems. Therefore, continuous technological innovation and interdisciplinary collaboration are essential to improving the efficiency, reliability, and sustainability of smart mechanical systems in future industrial ecosystems.

REFERENCES

- Ahad, M. T., Li, Y., Song, B., & Bhuiyan, T. (2023). Comparison of CNN-based deep learning architectures for rice diseases classification. *Artificial Intelligence in Agriculture*, 9.
- Alshammari, H., Gasmi, K., Ben Ltaifa, I., Krichen, M., Ben Ammar, L., & Mahmood, M. A. (2022). Olive disease classification based on vision transformer and CNN models. *Computational Intelligence and Neuroscience*, 2022.
- Bandi, R., Swamy, S., & Arvind, C. S. (2023). Leaf disease severity classification with explainable artificial intelligence using transformer networks. *International Journal of Advanced Technology and Engineering Exploration*, 10(100), 278.
- Borhani, Y., Khoramdel, J., & Najafi, E. (2022). A deep learning based approach for automated plant disease classification using vision transformer. *Scientific Reports*, 12(1), 11554–11554.
- Boukri, Y., Hamici, H. S., Mansour, R. F., Maamar, A. E. T., Ghoneim, S. S. M., Paramasivam, P., Hashim, M. A., & Hussein, E. E. (2026). Analysis and experimental implementation of affordable smart irrigation system using IoT to reduce agricultural costs and minimize water usage. *Applied Water Science*, 16(2). <https://doi.org/10.1007/s13201-025-02727-4>
- Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. In *arXiv (Cornell University)* (pp. 213–229). Cornell University. https://doi.org/10.1007/978-3-030-58452-8_13
- Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021). Emerging Properties in Self-Supervised Vision Transformers. *arXiv (Cornell University)*, 9630–9640. <https://doi.org/10.1109/iccv48922.2021.00951>
- Chaudhary, A., Sharma, A., Kumar, M., & Kaushik, H. (2023). Bringing the Garden Online: An Exploration of Best Practices for E-Commerce Plant Sales. *International Journal for Research in Applied Science and Engineering Technology*, 11(5), 607–618. <https://doi.org/10.22214/ijraset.2023.51441>
- Chougui, A., Moussaoui, A., & Moussaoui, A. (2022). Plant-leaf diseases classification using CNN, CBAM and vision transformer. *Proceedings of the 5th International Symposium on Informatics and Its Applications (ISIA)*, 1–6.
- Chowdhury, M. J. U., Mou, Z. I., Afrin, R., & Kibria, S. (2023). Plant leaf disease detection and classification using deep learning: A review and a proposed system on Bangladesh's perspective. *International Journal of Science and Business*, 28, 193–204.
- Homaei, M., Gutiérrez, Ó. M., Núñez, J. C. S., Ávila, M., & Caro, A. (2024). A review of digital twins and their application in cybersecurity based on artificial intelligence. *Artificial Intelligence Review*, 57(8). <https://doi.org/10.1007/s10462-024-10805-3>
- Islam, M., Azad, A., Arman, S. E., Alyami, S. A., & Hasan, M. M. (2025). PlantCareNet: an advanced system to recognize plant diseases with dual-mode recommendations for prevention. *Plant Methods*, 21(1), 52–52. <https://doi.org/10.1186/s13007-025-01366-9>

- Kumari, C. S., -, D. S. B., -, M. F. R., -, N. M. K., & -, S. R. J. (2025). Detailed Analysis of Smart IoT-Based Plant Care Systems. *International Journal on Science and Technology*, 16(2). <https://doi.org/10.71097/ijst.v16.i2.6460>
- Man, N. (2023). Consumers' Preference in Online Purchasing of Ornamental Plants in the Klang Valley, Malaysia. *Malaysian Journal of Agricultural Economics*. <https://doi.org/10.36877/mjae.a0000436>
- Soibam, S. (2025). Design and Development of a Smart Plant Assistant. *International Journal for Research in Applied Science and Engineering Technology*, 13(7), 2022–2028. <https://doi.org/10.22214/ijraset.2025.73298>
- .