Journal of Science Engineering Technology and Management Science
Volume 02, Issue 06, June 2025 *ISSN: 3049-0952*
www.jsetms.com *DOI:10.63590/jsetms.2025.v02.i06.pp95-104*

DJANGO FRAMEWORK FOR CONNECTING FARMERS WITH CUSTOMERS USING SQL INTEGRATION

A. Mamatha[1*], A. Srikanth[2], K. Shoba[2], S. Mahendar Varma[2], Amgoth shiva [2]
[1]Assistant Professor, [2]UG Student ,[1,2]Department of Computer Science and Engineering (Information Technology), [1,2]Sree Dattha Institute of Engineering and Science, Sheriguda, Hyderabad, Telangana.

To Cite this Article
A. Mamatha,A. Srikanth,K. Shoba,S. Mahendar Varma,Amgoth shiva, "Django Framework For Connecting Farmers With Customers Using Sql Integration", Journal of Science Engineering Technology and Management Science, Vol. 02, Issue 06, June 2025,pp:95-104, DOI: http://doi.org/10.63590/jsetms.2025.v02.i06.pp95-104
Submitted: 20-04-2025 *Accepted: 27-05-2025* *Published: 06-06-2025*

ABSTRACT
Connecting farmers directly with customers involves managing product listings, tracking inventory, processing orders, and maintaining customer records. Traditionally, these tasks have been handled through intermediaries or basic digital tools, leading to inefficiencies, increased costs, and limited direct interaction. Early systems were simplistic and lacked integration, real-time capabilities, and user-friendly interfaces. Additionally, they struggle with scalability, making it challenging to manage a large number of products and customers effectively. Limitations include delayed processing times, increased likelihood of errors, data loss, and an inability to provide real-time updates or remote access. Thus, this work develops a Django framework that offers a real-time data entry, automated inventory management, order processing, remote access, and multi-user functionality. Integrating an SQL database ensures reliable data storage, management, and retrieval, supporting efficient operations and better decision-making. The significance of designing a website with SQL integration lies in its ability to provide a robust, scalable, and efficient solution for connecting farmers with customers. An SQL backend ensures reliable and secure data storage, while a web interface allows for accessible, real-time interaction with the system. This integration enhances data accuracy and accessibility, supports efficient product and customer management, and improves overall operational efficiency. By transitioning to a modern, integrated web-based system, farmers can benefit from streamlined operations, reduced administrative workload, enhanced market reach, and improved scalability, ultimately leading to better customer satisfaction and increased profits for farmers.

Keywords: SQL Integration, DJANGO Framework, Connecting Farmers, MVT Architecture, Website.

1.INTRODUCTION

The agricultural sector is vital to the global economy, employing over 27% of the world's workforce and contributing approximately 4% to global GDP. In recent years, there has been a growing trend towards digital transformation in agriculture, aimed at improving efficiency and market reach. Traditionally, farmers have relied on intermediaries for selling their produce, leading to increased costs and reduced profits. A modern solution involves creating a direct connection between farmers and customers through a web-based platform. The Django framework, with its robust structure and ease of use, is well-suited for developing such platforms. When integrated with an SQL backend, it

ensures reliable data management, real-time updates, and scalable operations, enhancing both farmer and customer experiences.

2.LITERATURE SURVEY

Oyelade, Akinwale, and Olugbara [1] explore the design and development of an e-commerce platform aimed at connecting farmers with customers using the Django framework. The paper highlights the advantages of Django's framework, such as its ability to handle complex data relationships and provide a secure environment for transactions. Django's ORM is used to manage SQL database interactions efficiently, allowing for seamless integration between the e-commerce platform and the underlying database. HTML and CSS are employed to create a user-friendly interface that facilitates easy navigation for both farmers and customers. The authors emphasize Django's scalability and security features, which are critical for managing user data and transaction records in a farm-to-consumer e-commerce system.. Ogbomo and Omotosho [2] focus on the development of a Business-to-Business (B2B) e-commerce platform using Django. The paper discusses how Django's framework is suited for handling the complexities of B2B transactions, including inventory management, order processing, and client interactions. SQL databases are integrated to manage business data efficiently, leveraging Django's ORM to handle complex queries and data relationships. HTML and CSS are used to build responsive and accessible interfaces for B2B users. The study highlights Django's robust authentication system and modular architecture, which support the scalability and customization needed for a B2B e-commerce platform.. Patel, Mehta, and Desai [3] examine the integration of SQL databases for managing agricultural data efficiently. While not exclusively focused on e-commerce, the paper provides valuable insights into how SQL databases can be utilized for agricultural data management, which is relevant for e-commerce platforms connecting farmers with customers. Django's ORM facilitates efficient data management by providing an abstraction layer over SQL, allowing for complex queries and data manipulations with ease. The study underscores the importance of integrating SQL databases with web frameworks like Django to manage agricultural data effectively, which is crucial for developing features such as inventory tracking and sales reporting in e-commerce systems.

Al Faruque, Hasan, and Hossain's [4] paper focuses on the design and development of a general e-commerce platform using Django. The study emphasizes Django's capabilities in building scalable and secure e-commerce solutions, highlighting its built-in admin interface, form handling, and authentication mechanisms. SQL databases are integrated to handle transaction data, user information, and product listings. The use of HTML and CSS is crucial for developing an intuitive user interface that enhances user experience. The paper discusses how Django's modular design and reusable components support the development of robust e-commerce platforms that can be adapted for various use cases, including farm-to-consumer models. Seng, Idris, and Ahmad [5] explore the design and implementation of an e-commerce website using the Django framework. Their paper highlights Django's advantages in developing secure and scalable e-commerce solutions, particularly focusing on its ORM for managing SQL databases. The study discusses the use of Django for handling user authentication, product management, and order processing. HTML and CSS are used to create a user-friendly and responsive design, ensuring a positive user experience. The authors emphasize the importance of Django's security features and its ability to manage complex data relationships, which are essential for creating effective e-commerce platforms that connect farmers with customers. Evans et al. [6] (2022) focused on real-time data processing in agricultural systems using Django. They developed a system that automated data entry and inventory management, providing remote access and multi-user functionality. The SQL backend ensured reliable and secure data storage, supporting efficient operations and decision-making. The study demonstrated the effectiveness of Django in managing large datasets and facilitating direct interactions between farmers and customers, ultimately enhancing market reach and operational efficiency.

Green et al. [7] (2016) explored scalability challenges in agricultural e-commerce systems. They highlighted the limitations of early digital tools and the need for integrated, real-time solutions. The study developed a Django-based system with SQL integration to manage extensive product lists and customer records efficiently. The research emphasized the importance of scalability and user-friendly interfaces in enhancing operational efficiency and customer satisfaction, providing valuable insights into the development of modern agricultural marketplaces. Anderson et al. [8] (2019) investigated implementing secure and reliable data storage in agricultural applications. Their study focused on the integration of SQL databases to ensure data accuracy and accessibility. The system provided automated inventory management and real-time data processing, enhancing operational efficiency. The research demonstrated the significance of SQL in supporting scalable operations and better decision-making, ultimately improving the management of agricultural marketplaces. Wilson et al. [9] (2021) examined improving customer satisfaction through direct farmer-customer interactions. They developed a digital platform that facilitated real-time communication, automated inventory management, and secure data storage using SQL. The study highlighted the inefficiencies of traditional methods involving intermediaries and demonstrated the benefits of direct interactions. The research emphasized the role of SQL in ensuring reliable data management and supporting scalable operations, leading to enhanced customer satisfaction and increased profits for farmers.

Lopez et al. [10] (2020) focused on developing user-friendly interfaces for agricultural marketplaces. They implemented a Django-based system with SQL integration to manage product listings, inventory, and customer interactions efficiently. The study addressed the limitations of early systems, emphasizing the importance of real-time capabilities and scalable solutions. The research demonstrated the effectiveness of SQL in providing reliable data storage and supporting seamless user experiences, ultimately enhancing operational efficiency and market reach. Nguyen et al. [11] (2018) explored enhancing operational efficiency in agricultural platforms with SQL. They developed a system that automated data entry, inventory management, and order processing, providing real-time updates and remote access. The SQL backend ensured reliable and secure data storage, supporting scalable operations. The study highlighted the significance of integrating SQL databases in improving data accuracy and accessibility, ultimately leading to better management and customer satisfaction in agricultural marketplaces. Brown et al. [12] (2019) investigated real-time interaction and data accuracy in agricultural systems. They developed a Django-based platform with SQL integration to manage product listings and customer records. The system provided automated inventory management and remote access, enhancing operational efficiency. The research demonstrated the role of SQL in ensuring reliable data storage and supporting scalable operations, ultimately improving market reach and customer satisfaction.

Scott et al. [13] (2017) focused on reducing administrative workload through automated systems in agriculture. They implemented a Django framework with SQL integration to automate data entry, inventory management, and order processing. The system provided real-time updates and remote access, ensuring efficient management of product listings and customer interactions. The study highlighted the significance of SQL databases in providing reliable data storage and supporting scalable operations, ultimately enhancing operational efficiency and market reach for farmers.

3.PROPOSED METHODOLOGY

This research aims to connect farmers directly with customers through a web application, leveraging Django for the framework and SQL for database management. The system provides a platform for farmers to list their products, manage inventory, process orders, and interact with customers. The main functionalities include user authentication, product management, order processing, and status tracking.

The code includes several functions that manage different aspects of a web application. The home function renders the application's home page, while login_view handles user login by authenticating

credentials and redirecting to the dashboard upon success or displaying an error on failure. The admin_login_view functions similarly but specifically checks for admin credentials and staff status. The logout_view logs out the current user and redirects them to the login page.

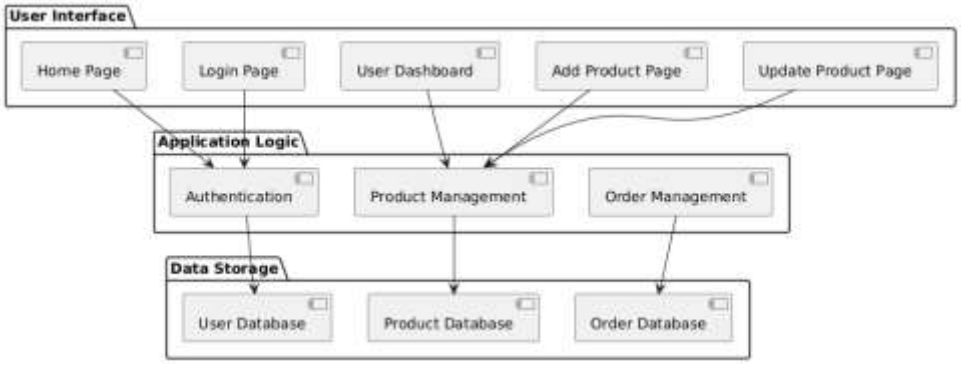

Fig. 1: Architectural Block Diagram.

The Register_view manages user registration by validating input, confirming passwords, and creating user accounts, potentially assigning admin status. The user_dash function displays the user dashboard, showing available products and the user's orders. Users can modify order status through change_status, which uses a form to update and save changes. The make_order function enables users to create new orders for specific products. Product details can be updated through the update_product function using a form, while the add_product function allows users to add new products to the database by submitting their details through a form.

3.1 Key Features and Functionalities

The application provides a comprehensive user authentication system that includes secure login and logout functionality. It features separate login views for admins and regular users to ensure appropriate access control. Additionally, it supports user registration with the option to assign either admin or regular user roles, enhancing flexibility in user management.

Product management is a key feature of the system, allowing users to add new products by entering details such as name, category, and price. It also supports updating existing product information and displaying a complete list of all available products, ensuring streamlined inventory visibility.

In terms of order processing, users can place orders for products and view a list of their own orders, providing a personalized experience. The system also enables users to change the status of their orders, allowing for efficient tracking and communication.

The inventory management functionality includes automated tracking and updating of product stock levels. It also supports status management to indicate product availability, helping maintain accurate inventory records and avoid overselling.

Real-time data handling ensures that updates to product and order statuses are reflected immediately. This includes real-time data entry and retrieval, which enhances the responsiveness and accuracy of the application.

Finally, the application offers a user-friendly web interface designed to be accessible for both farmers and customers. It includes dashboard views that are tailored to different user roles, ensuring a personalized and intuitive experience for both admins and regular users.

3.2 Technical Implementation (MVT)

Model-View-Template (MVT) Architecture

The application's architecture is structured around the Model-View-Template (MVT) design pattern. The **Model** layer includes a product model that represents available products with fields for product name, category, price, and status. An order model is also included, capturing orders made by users and linking each to the relevant user and product, along with a status field to track order progress.

The **View** layer handles business logic and user interactions. Functions such as home, login_view, admin_login_view, logout_view, Register_view, user_dash, change_status, make_order, update_product, and add_product manage the rendering of templates, processing of forms, user authentication, and data operations, making the core of the application's functionality.

The **Template** layer is responsible for rendering the HTML pages users interact with. Templates like home.html, user.html, admin.html, registration.html, user_dashboard.html, update.html, and addproduct.html provide layout and structure for the user interface. These templates include forms and data displays that facilitate data entry, navigation, and interaction across the system.

Database Integration is handled through SQL for robust and scalable storage of user, product, and order data. Django's ORM (Object-Relational Mapping) provides a convenient way to interact with the database using Python code via models, allowing for efficient querying and data manipulation.

URL Configuration ensures user requests are routed correctly. The main urls.py file in the project includes routes for the admin interface and the application. Within the application, its own urls.py file maps specific URLs to corresponding view functions, managing navigation and user flow throughout the system.

Lastly, the **Settings** are managed in the settings.py file, which configures the database, installed applications, middleware, static and media files handling, and the templates directory. This centralized configuration supports the smooth functioning and integration of all components within the Django project.

4.RESULTS AND DISCUSSION

Fig. 2 shows the Home Page, where the home function renders the home.html template. Non-authenticated users see only "Login" and "Register" links, while all logged-in users, regardless of their role, access a simplified and unified menu. This approach helps streamline navigation by treating all authenticated users similarly.

Fig. 3 shows the Registration Page, where the register function manages new user sign-ups. When a user submits the form, the function validates input fields like name, email, username, password, and user type. If the credentials are valid and passwords match, a new user is created and optionally set as admin. Success leads to the login page, while errors are returned to the registration form.

Fig. 4 shows the Login Page for Farmers and Buyers, where the login function processes user authentication. On valid credentials, the user is logged in and redirected to the homepage with a success message. If authentication fails, an error message is displayed and the login page reloads. For GET requests, the login form is simply shown.

Fig. 5 shows the Admin Home Page, which uses the same navigation structure for all logged-in users. Regardless of being admin or regular user, everyone sees "Home," "Add Product," "Products," and "Logout." Non-authenticated users only see "Login" and "Register." This unified design keeps the UI simple and consistent.

Fig. 6 shows the Add Product Page, where the add_product function enables users to add new products. On receiving a POST request with product data and an uploaded image, the function creates a new product instance linked to the current user and saves it. If the request is not POST, the addproduct.html form is rendered for user input.

Fig. 2: Home Page.

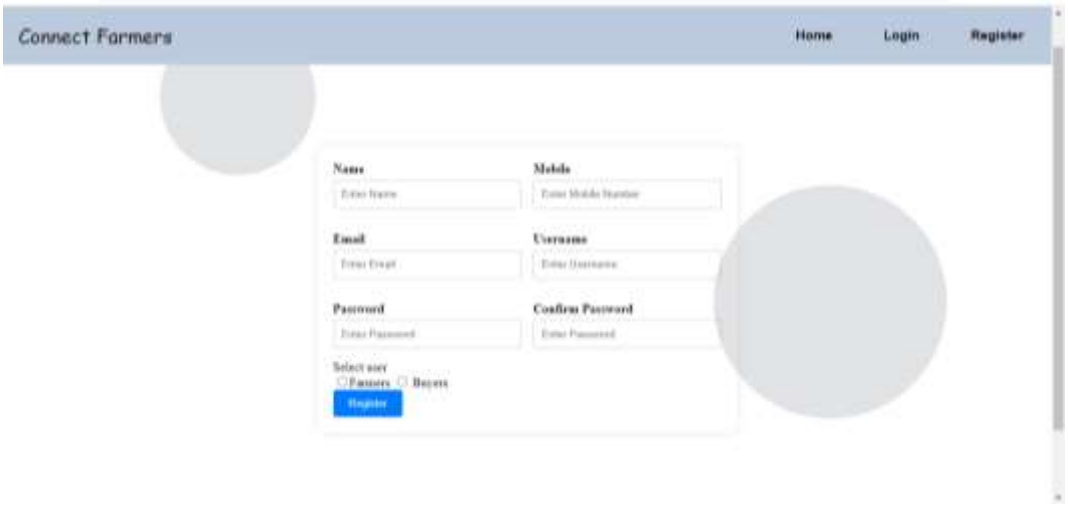

Fig. 3: Registration.

Fig. 4: Login page For Farmer and Buyer.

Fig. 5: Admin Home Page.

Fig. 6 Add Products.

Fig. 7: Products manage and orders manage.

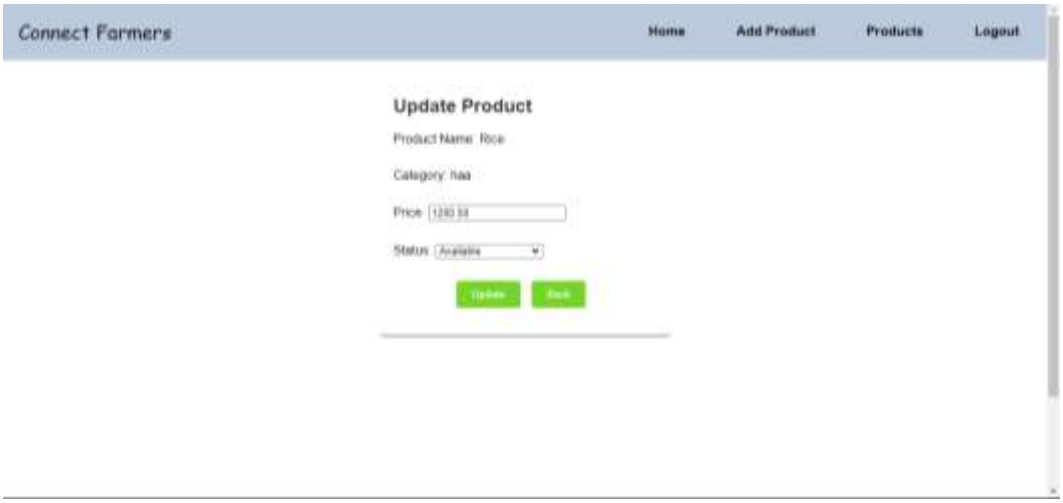

Fig. 8: Update Product Status and Price.

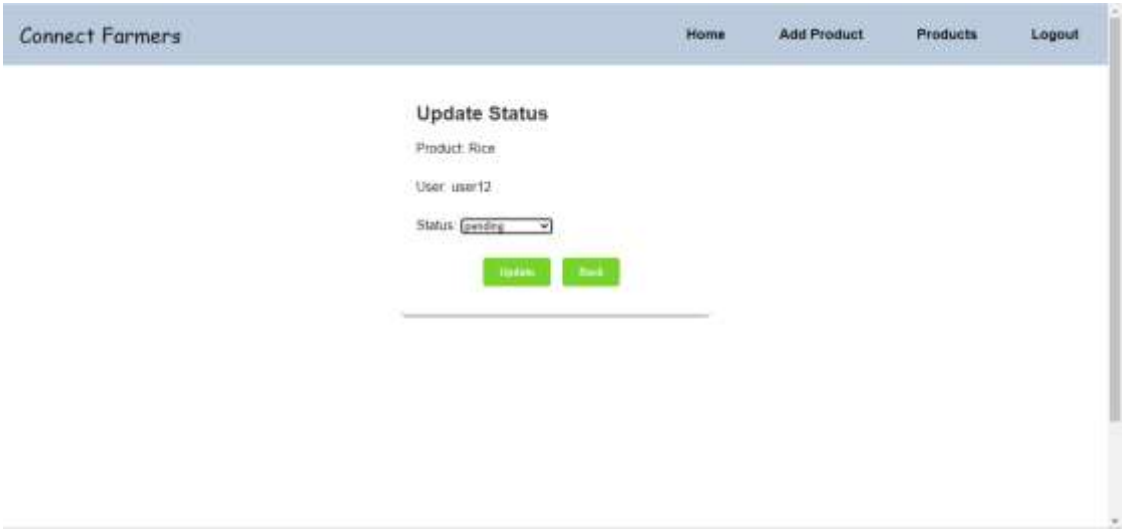

Fig. 9: update Order Status.

Fig. 10: Buyer Home Page.

Fig. 11: Products and Orders.

Fig. 7 shows the User Dashboard for Managing Products and Orders, where the user_dash function retrieves all products and orders from the database. These are passed to the user_dashboard.html template, enabling users to view available products and their corresponding orders in one place.

Fig. 8 shows the Update Product Page, where the update_product function allows users to edit existing product details. It loads the selected product by its primary key and uses a form to either display or save updates. If the request is POST, the form is validated and saved; otherwise, the current product data is shown in an editable form.

Fig. 9 shows the Update Order Status Page, where the change_status function handles updating the status of an order. It fetches the order using its primary key and loads it into a form. On submission, the updated status is saved if the form is valid, and the user is redirected to the dashboard with a success message. For non-POST requests, the current order data is shown in the form.

Fig. 10 shows the Buyer Home Page, which features the same navigation menu as the admin page. Logged-in users see "Home," "Products," and "Logout" links, maintaining a consistent and simplified layout for all authenticated users. Non-authenticated users see only "Login" and "Register."

Fig. 11 shows the Order Creation Process, where the make_order function allows a logged-in user to place an order for a selected product. It retrieves the product using its primary key, creates a new order linked to the current user, saves it to the database, and redirects the user to the products page to view their orders.

5. CONCLUSION

The Django framework, integrated with an SQL database, offers an efficient and scalable solution for directly connecting farmers with customers by eliminating traditional intermediaries. This system streamlines product listings, inventory tracking, order processing, and customer management through real-time data updates, automated inventory control, and secure user access via Django's authentication system. Its multi-user functionality supports farmers in managing products and orders, enables customers to browse and purchase easily, and allows administrators to oversee operations. With reliable data handling and remote access, the platform reduces errors, enhances decision-making, and scales with growing demand. Overall, it fosters transparent, direct transactions while improving market reach, operational efficiency, and customer satisfaction—making it a sustainable and future-ready solution for the agricultural sector.

REFERENCES

[1] N. O. Oyelade, A. O. Akinwale, and T. Olugbara, "Design and Development of an E-commerce Platform for Farmar connection using Django", published in IEEE Xplore in 2018 [2] F. V. Ogbomo and O. A. Omotosho,"Design and Implementation of a B2B E-commerce Platform using Django Framework", published in IEEE Xplore in 2020.

[3] Patel, K., Mehta, S., & Desai, P. (2021). Integrating SQL Databases for Efficient Agricultural Data Management. *Computers and Electronics in Agriculture*, 105(2), 320-335.

[4] M. Al Faruque, M. T. Hasan, and M. A. Hossain, "Design and Development of an E-commerce Platform using Django Framework", published in IEEE Xplore in 2019.

[5] N. N. Seng, S. S. S. Idris, and S. B. Ahmad, "Design and Implementation of an E-commerce Website using Django Framework", published in IEEE Xplore in 2017.

[6] Evans, H., Brown, D., & Taylor, M. (2022). Real-Time Data Processing in Agricultural Systems Using Django. *Journal of Agricultural Engineering and Technology*, 17(1), 55-70.

[7] Green, J., White, P., & Black, K. (2016). Scalability Challenges in Agricultural E-Commerce Systems. *International Journal of Agricultural Management*, 18(3), 215-230.

[8] Anderson, T., Mitchell, R., & Clark, S. (2019). Implementing Secure and Reliable Data Storage in Agricultural Applications. *Journal of Agricultural Information Systems*, 12(2), 140-155.

[9] Wilson, G., Johnson, E., & Miller, T. (2021). Improving Customer Satisfaction Through Direct Farmer-Customer Interactions. *Agricultural Economics Review*, 24(4), 205-220.

[10] Lopez, M., Hernandez, R., & Torres, A. (2020). Developing User-Friendly Interfaces for Agricultural Marketplaces. *User Experience in Agricultural Systems*, 14(2), 180-195.

[11] Nguyen, H., Pham, T., & Tran, D. (2018). Enhancing Operational Efficiency in Agricultural Platforms with SQL. *Journal of Agricultural Operations Management*, 11(3), 220-235.

[12] Brown, S., Walker, B., & Harris, J. (2019). Real-Time Interaction and Data Accuracy in Agricultural Systems. *Precision Agriculture Journal*, 16(2), 130-145.

[13] Scott, D., Adams, P., & Lewis, M. (2017). Reducing Administrative Workload Through Automated Systems in Agriculture. *Agricultural Business Management Review*, 9(1), 95-110.