**CHAPTER 1**

**INTRODUCTION**

# 1.1 INTRODUCTION

Agriculture is the backbone of many economies around the world, yet farmers often face challenges in accessing markets and selling their products at fair prices. Traditional methods of selling agricultural products typically involve intermediaries, such as wholesalers and distributors, who increase the cost of goods and reduce the profit margins for farmers. Additionally, the lack of direct communication between farmers and consumers can result in inefficiencies, price inflation, and a lack of transparency in the agricultural supply chain.

With the advancement of technology, there is an opportunity to streamline and improve these processes through digital platforms. A mobile application that connects farmers directly with buyers, including customers and retailers, has the potential to revolutionize the way agricultural products are sold and distributed. This app would allow farmers to list their products, negotiate prices, and complete transactions without the need for intermediaries. On the other side, customers and retailers would have direct access to fresh, quality produce at competitive prices, with the added benefit of negotiating prices and tracking their purchase.

The proposed web application serves as a platform where farmers, customers, and retailers can interact seamlessly. The system supports multiple functionalities, including product listing, price negotiation, secure payments, delivery tracking, and user feedback, all designed to facilitate a smooth and efficient transaction process. This web app is not only aimed at improving the income and efficiency of farmers but also offers customers and retailers a more convenient and transparent method of purchasing agricultural products.

**1.2 HISTORY OVER THE PROJECT**

**Inception:**

The idea for the "Direct Farmer-Customer Access" mobile application originated from the increasing need for transparency and efficiency in the agricultural supply chain. Farmers often face challenges in directly reaching consumers, resulting in reliance on intermediaries who increase costs and limit market access. Recognizing these gaps, the founders envisioned a platform where farmers could list their products, negotiate prices, and handle transactions directly with customers and retailers. The aim was to empower farmers, improve market access for consumers, and reduce inefficiencies in the system.

**Research and Planning:**

The initial phase of the project involved extensive research to understand the pain points of farmers, customers, and retailers in the existing agricultural marketplace. Surveys, interviews, and market analysis were conducted to gather insights into user needs. The research revealed a strong demand for features like direct product listings, price negotiations, secure transactions, and transparent delivery tracking. Based on this data, the project’s scope and objectives were defined, with the goal of providing a seamless, user-friendly platform for all stakeholders.

**Design and Prototyping:**

During the design phase, the team focused on creating wireframes and prototypes to visualize the user interface (UI) and user experience (UX). The primary goal was to make the app simple to use while ensuring that all features, such as product listing, negotia tions, payments, and delivery tracking, were easily accessible. Feedback from early user testing of the prototypes led to design iterations aimed at improving usability, responsiveness, and clarity of the app's navigation. The design also incorporated aspects that would make the system scalable and adaptable for future growth.

**Development:**

The development phase focused on building the core modules of the system:

1. **User Management**: Enabling different roles such as farmers, customers, retailers, and admins, with specific access rights.
2. **Product Listing and Management**: Farmers could list, update, and manage their product details.
3. **Price Negotiation and Communication**: Allowing buyers to offer prices, and enabling real-time communication between buyers and sellers.
4. **Payment System**: A secure, encrypted payment gateway that ensures safe transactions.
5. **Delivery Tracking**: Features that allow both farmers and buyers to track the status of deliveries.

The development followed Agile methodologies to ensure continuous feedback, testing, and refinement. This approach ensured that the app would evolve based on real user input and market needs.

**Testing and Quality Assurance:**

To ensure a secure, bug-free, and user-friendly application, the team conducted thorough testing. This included both manual and automated tests to identify and address any bugs or security vulnerabilities. Additionally, **User Acceptance Testing (UAT)** was carried out, where real users interacted with the app in a controlled environment to validate its functionality, usability, and performance. Feedback from UAT helped identify minor issues that were resolved before the official launch.

**Launch:**

Once testing was completed and the app was refined based on feedback, the application was officially launched to the public. The marketing campaign targeted farmers, customers, and retailers, highlighting the platform’s ability to streamline the agricultural supply chain and provide direct access to fresh produce. The app’s secure payment system and price negotiation features were key selling points in the marketing materials.

**User Feedback and Iteration:**

Following the launch, the team actively collected user feedback through in-app surveys, user reviews, and direct communication. This feedback helped identify areas for improvement, including adding features like additional payment options, more detailed product categorization, and better delivery tracking tools. Based on this ongoing feedback, the team implemented several updates to improve the user experience, performance, and scalability of the platform.

**Future Vision:**

Looking ahead, the project team continues to explore new features and integrations to further enhance the app’s capabilities. Future plans include expanding the platform to include real-time pricing analytics, AI-driven suggestions for pricing and product recommendations, and enhanced support for farmers in managing inventory and sales data. The team is also exploring partnerships with logistics providers to offer integrated delivery solutions. By staying ahead of emerging trends and continuously responding to user needs, the platform aims to remain a leading solution for improving the efficiency and transparency of the agricultural market.

**1.3 DOMAIN INTRODUCTION**

In today’s rapidly evolving agricultural sector, there is a growing need for efficient, transparent, and direct ways for farmers to sell their products to consumers, retailers, and businesses. The domain of agricultural e-commerce focuses on connecting farmers directly with buyers, eliminating intermediaries, reducing costs, and improving the efficiency of the supply chain. It enables farmers to list their products, negotiate prices, process payments, and manage delivery options through a digital platform.

**Key Aspects of the Domain**

**Market Access for Farmers:** A significant challenge for many farmers is the limited access to larger markets. Traditionally, farmers rely on intermediaries like wholesalers and distributors, which reduces their profit margins and limits their reach. Agricultural e-commerce platforms provide farmers with a direct channel to customers and retailers, increasing their market visibility and improving access to better pricing and sales opportunities.

**Price Transparency and Negotiation:** One of the main benefits of agricultural e-commerce is the transparency it provides. Through the platform, buyers (consumers and retailers) can view the products listed by farmers, which promotes price transparency. Additionally, price negotiation tools allow buyers to make offers, and farmers can accept, reject, or counter offers, making the price-setting process more dynamic and fair.

**Direct Communication between Farmers and Buyers:** Communication is vital for successful transactions, especially when it involves perishable goods. Agricultural e-commerce platforms enable real-time messaging and communication tools that allow farmers and buyers to discuss product details, delivery options, and other relevant terms. This direct interaction eliminates the confusion and delays often caused by intermediaries.

**Payment Security:** To foster trust between farmers and buyers, payment security is a key aspect of the agricultural e-commerce domain. Integration with secure payment gateways ensures that both parties are protected during financial transactions. Payment options such as credit/debit cards, digital wallets, and escrow systems are commonly used to offer flexibility and security to users.

**Delivery and Logistics Management:** Delivery of agricultural products, especially fresh produce, requires careful attention to logistics. A major challenge in this domain is ensuring that products are delivered in a timely and safe manner, preserving their quality. E-commerce platforms integrate with third-party logistics providers to offer delivery solutions, and allow users to track the status of their deliveries in real-time.

**Trust and Reputation:** Trust is crucial in online marketplaces, particularly when dealing with perishable goods. A review and rating system within the platform allows both farmers and buyers to evaluate each other’s reliability. Higher ratings help build a reputation, encouraging more transactions and creating a more transparent and trustworthy environment for users.

**User Experience:** As the platform is aimed at a diverse audience, including farmers, customers, and retailers, the user experience must be intuitive and easy to navigate. The interface should be simple enough for farmers, many of whom may not be tech-savvy, while also providing robust features for more advanced users. A well-designed platform makes product listing, browsing, negotiation, and transactions accessible to all users.

**Market Demand:** With the increasing trend towards digitalization and e-commerce in various industries, the demand for agricultural e-commerce platforms is growing. Farmers and businesses are increasingly adopting technology to streamline their operations and access new markets. Similarly, consumers and retailers seek fresh, quality products at competitive prices, which agricultural e-commerce platforms aim to fulfill.

The agricultural e-commerce domain is designed to modernize the way food and agricultural products are traded, making the process more efficient, transparent, and secure for both farmers and buyers. By providing direct access, eliminating middlemen, and offering essential tools for communication, pricing, and payment management, this domain aims to revolutionize the agricultural supply chain.

**1.3 OBJECTIVES**

The primary objective of the "Direct Farmer-Customer Access" web application is to create a seamless and efficient platform that bridges the gap between farmers, customers, and retailers in the agricultural supply chain. The key objectives of the project are as follows:

* **Direct Access to Agricultural Products**: Enable farmers to directly list and sell their agricultural products to customers and retailers, eliminating intermediaries and ensuring better price transparency.
* **Price Negotiation and Transparency**: Facilitate price negotiation between farmers and buyers, allowing customers and retailers to make offers and providing farmers the flexibility to accept, reject, or counter offers.
* **Secure Payment and Transaction Handling**: Implement a secure payment system that supports various payment methods (e.g., credit/debit card, digital wallets, UPI) to ensure smooth transactions. Additionally, provide features like escrow payments to protect both parties in case of disputes.
* **Effective Communication Tools**: Provide real-time messaging and communication tools that allow farmers, customers, and retailers to negotiate prices, discuss product details, and confirm delivery terms, enhancing the transaction process.
* **Delivery and Logistics Integration**: Include options for farmers to manage product delivery, either through their own logistics or third-party services. The app will allow buyers to track the delivery status, ensuring transparency and timely delivery.
* **Rating and Feedback System**: Implement a rating and review system where both buyers and sellers can evaluate each other based on the quality of products and reliability. This helps foster trust and accountability within the platform.
* **User-Friendly Interface**: Develop an intuitive and easy-to-use interface for farmers, customers, and retailers, ensuring that users of varying technical expertise can navigate the platform efficiently. Simplify processes like listing products, searching for items, and completing transactions.
* **Admin Panel for Platform Management**: Build an admin panel that allows administrators to manage product listings, monitor transactions, verify users, and ensure compliance with platform standards. The admin panel also ensures security and proper functioning of the app.
* **Scalability and Performance**: Design the application with scalability in mind to accommodate growing user numbers and increasing transactions, ensuring that the platform can handle peak loads while maintaining performance.
* **Compliance with Regulations**: Ensure that the platform adheres to data protection regulations (such as GDPR, CCPA) to protect users’ personal and financial data, maintaining trust and legal compliance.

The successful implementation of these objectives will create a platform that not only improves the agricultural supply chain but also provides farmers with more control over their sales, while offering buyers fresh and quality produce with transparent pricing.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 Mobile Platforms for Agricultural Marketing:   
Authors: R. Sharma, S. Singh, et al.**

This paper provides a comprehensive analysis of the role mobile platforms play in enhancing agricultural marketing and facilitating direct access to markets for farmers. It explores how mobile applications and digital platforms have revolutionized traditional agricultural marketing by reducing dependency on intermediaries, ensuring better price transparency, and bridging the gap between farmers and consumers.The study highlights several key challenges faced by farmers in conventional agricultural marketing systems, including price manipulation by middlemen, lack of real-time market information, inefficient supply chains, and geographical constraints that limit market access. By leveraging mobile technology, farmers can now directly connect with buyers, negotiate fair prices, and make informed decisions about when and where to sell their produce.

The paper also presents case studies from different regions where mobile platforms have successfully improved market efficiency and farmer incomes. Examples include government-led initiatives, private sector innovations, and non-profit-driven solutions aimed at empowering small-scale farmers.While mobile platforms offer numerous advantages—such as increased price transparency, reduced exploitation, and greater market reach—the paper also discusses potential drawbacks. These include digital literacy barriers, connectivity issues in rural areas, high transaction costs for certain platforms, and concerns regarding data security and fraud.

Overall, the study concludes that mobile technology has immense potential to transform agricultural marketing, provided that there is adequate infrastructure support, proper regulatory frameworks, and continuous efforts to improve digital literacy among farmers. Future research directions include assessing the long-term economic impacts of these platforms, exploring AI-driven market prediction models, and developing policies to ensure equitable access to digital agricultural tools.

**2.2 Application of Mobile Technology to Agriculture for Rural Market Access:**

**Authors: A. Kumar, B. Patel**

This literature review examines the adoption of mobile technology by farmers, particularly in rural areas, to enhance market access and improve their economic outcomes. The paper provides an in-depth discussion on how mobile applications and digital platforms contribute to expanding farmers’ market reach, improving price transparency, and increasing their overall income. By synthesizing findings from various studies and real-world case examples, the authors highlight both the opportunities and challenges associated with integrating mobile technology into rural agricultural systems. The paper presents multiple case studies from different regions, demonstrating how mobile technology has positively impacted farmers’ livelihoods. Notable examples include government-backed initiatives, private sector-led mobile platforms, and partnerships between agricultural cooperatives and telecom companies.

In conclusion, the authors emphasize that while mobile technology has the potential to revolutionize rural agricultural markets, its success depends on improving infrastructure, enhancing digital literacy, and addressing economic and social barriers to adoption. Future research directions include exploring AI-driven market prediction tools, blockchain-based transaction security, and policy recommendations for digital agricultural expansion in rural areas.

**2.3 Effect of Electronic Commerce Platforms on Agricultural Supply Chains:**

**Authors: T. Mehta, R. Jain**

This paper examines the transformative role of e-commerce platforms in enhancing agricultural supply chains, with a particular focus on how digital marketplaces and mobile applications enable farmers to overcome traditional bottlenecks, access broader markets, and increase profitability. Through a case study approach, the authors analyze real-world implementations of e-commerce solutions in agriculture, with a particular emphasis on success stories from countries like India and Kenya.

The paper explores how e-commerce platforms, including mobile applications, have revolutionized agricultural supply chains by Digital Platforms enable farmers to sell directly to consumers, wholesalers, and institutional buyers bypassing intermediaries.

**2.4 ICT Solutions for Agricultural Market Access:**

**Authors: P. Yadav, S. Sharma**

This paper provides a comprehensive review of Information and Communication Technology (ICT) solutions in agriculture, with a particular focus on mobile applications as a key tool for improving farmers' access to markets. The study examines how mobile apps help farmers obtain real-time price information, engage in direct selling, and connect with potential buyers, thereby reducing dependency on intermediaries and increasing profitability. Additionally, the paper evaluates the challenges faced by these digital platforms, using factors such as user engagement, digital literacy, and payment systems as key assessment criteria.

The paper concludes that mobile applications have significantly enhanced agricultural market access by improving price transparency, reducing dependency on middlemen, and providing direct market linkages. However, for widespread adoption, policymakers and developers must address issues related to digital literacy, connectivity, security, and financial accessibility.

**2.5 Agricultural Market Access Enhancement Using Digital Solutions:**

**Authors: M. Thakur, R. Kaur**

This systematic review explores the impact of digital solutions, particularly mobile applications, on improving market access for farmers. It evaluates whether these platforms enhance market transparency, facilitate price discovery, and reduce dependence on middlemen. The study also examines successful implementations of digital agricultural solutions worldwide and identifies key factors that contribute to their effectiveness, including trust, usability, and scalability. The systematic review concludes that digital solutions have the potential to revolutionize agricultural marketing by improving price transparency, reducing reliance on intermediaries, and optimizing supply chains.

**CHAPTER 3**

**REQUIREMENT ANALYSIS**

**3.1 HARDWARE REQUIREMENTS**

* **Processor Name**: Intel Core i5 (or equivalent)
* **Processor Speed**: 3.0 GHz or higher
* **RAM**: 8 GB (minimum)
* **Hard Disk Capacity**: 128 GB (SSD preferred for faster access)
* **Display Device**: 14" to 19" inch monitor (minimum 1080p resolution)
* **Keyboard Type**: PS2 or USB (standard keyboard)
* **Mouse Type**: PS2 or USB (standard mouse)

**3.2 SOFTWARE REQUIREMENTS**

* **Language Used**:
  + JavaScript(for both backend and frontend development)
  + Python (backend development)
* **Server**: Node.js (used for hosting the backend and handling API requests)
* **Database**: MongoDB (NoSQL database for storing user and product data)
* **User Interface Design**: React.js (for building interactive and responsive frontend)
* **Web Browser**: Google Chrome (recommended for testing and usage, but other modern browsers like Firefox and Edge are also supported)

**3.3 SOFTWARE DESCRIPTION**

## **3.3.1 Introduction to JavaScript**

An explanation of exactly what JavaScript is has to begin with Java. Java is a new kind of Web programming language developed by Sun Microsystems. A Java program, or applet, can be loaded by an HTML page and executed by the Java Interpreter, which is embedded into the browser. Java is a complex language, similar to C++. Java is object-oriented and has a wide variety of capabilities; it's also a bit confusing and requires an extensive development cycle. That's where JavaScript comes in. JavaScript is one of a new breed of Web languages called scripting languages. These are simple languages that can be used to add extra features to an otherwise dull and dreary Web page. While Java is intended for programmers, scripting languages make it easy for nonprogrammers to improve a Web page. JavaScript was originally developed by Netscape Corporation for use in its browser, Netscape Navigator. It includes a convenient syntax, flexible variable types, and easy access to the browser's features. It can run on the browser without being compiled; the source code can be placed directly into a Web page. You can program in JavaScript easily; no development tools or compilers are required. You can use the same editor you use to create HTML documents to create JavaScript, and it executes directly on the browser (currently, Netscape or Microsoft Internet Explorer).

JavaScript was originally called Live Script, and was a proprietary feature of the Netscape browser. JavaScript has now been approved by Sun, the developer of Java, as a scripting language to complement Java. Support has also been announced by several other companies. Although useful in working with Java, you'll find that JavaScript can be quite useful in its own right. It can work directly with HTML elements in a Web page, something Java can't handle. It is also simple to use, and you can do quite a bit with just a few JavaScript statements.

**3.3.2 The Advantages of JavaScript**

**An Interpreted Language**: JavaScript is an interpreted language, which requires no compilation steps. This provides an easy development process. The syntax is completely interpreted by the browser just as it interpreted HTML tags.

**Embedded Within HTML**: JavaScript does not requires any special or separate editor for programs to be written edited or compiled. It can be written in any text editor like Notepad, along with appropriate HTML tags, and saved as filename.html.HTML files with embedded JavaScript commands can then be read and interpreted by any browser that is JavaScript enabled.

**Minimal Syntax-Easy to Learn:** By learning just a few commands and simple rules of syntax, complete applications can be built using JavaScript.

**Quick Development:** Because JavaScript does not require time-consuming compilations, scripts can be developed in a short period of time. This is enhanced by the fact many GUI interface features, such as alerts, prompts, confirm boxes, and other GUI elements, are handle by client side JavaScript, the browser and HTML code.

**Design for Simple, Small Programs:** It is well suited to implement simple, small programs (for example, a unit conversion calculator between miles and kilometers or pounds and kilograms).Such programs can be easily written and executed at an acceptable speed using JavaScript. In addition, they can be easily interpreted into a web page.

**Performance:** JavaScript can be written such that the HTML files are fairly compact and quite small. This minimizes storage requirements on the web server and download time for the client. Additionally, because JavaScript are usually include in the same file as the HTML code for a web page, they require fewer separate network accesses.

**Procedural Capabilities:** Every programming language needs to support facilities such as Condition checking, Looping and Branching .JavaScript provides syntax, which can be used to add such procedural capabilities to web page (filename.html) coding.

**Designed for Programming User Events:** JavaScript supports Object/Events based programming JavaScript recognizes when a form **Button** is pressed. This event can have suitable JavaScript code attached, which will executed when the **Button Pressed** event occurs. JavaScript can be used to implement context sensitive help. Whenever an HTML form’s **Mouse** cursor **Mouse Over** a button or a link on the page a helpful and informative massage can be displayed in the status bar at the button of the browser window.

**Easy Debugging and Testing :** Being an interprets language ,scripts in JavaScript are tested line by line, and the errors are also listed as they are encountered ,i.e. an appropriate error message along with the line number is listed for every error that is encountered. It is thus easy to locate errors, make changes, and test it again without the overhead and delay of compiling.

**Platform Independence / Architecture Neutral:** JavaScript is a programming language that is completely independent of the hardware on which it works. It is a language that is understood by any JavaScript enabled browser. Thus, JavaScript application work on any machine that has an appropriate JavaScript enabled browser can be anywhere on the network.

Since each browser is for a specific platform, JavaScript interpretation will be with respect to the specific platform. The browser will add whatever platform specific Information is required to the JavaScript while it interprets the code. Thus, JavaScript is truly platform independent. A JavaScript programmer developed on a UNIX machine will work perfectly well on a Windows machine.

The fact that a platform specific browser , maintained at the client end, does the interpretation of JavaScript , relieves the developer of the responsibility of maintaining multiple source code files for multiple platform.

**3.3.3 React JS**

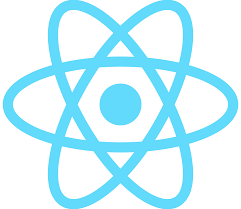


Figure 3.3.1: React JS

React JS is an open-source library used for mobile and web development. It enables developers to build reliable, fast, and scalable web applications. It works on the UI in the application and enables the creation of reusable UI components.

## **Setting up a React Environment:**

## If you have npx and Node.js installed, you can create a React application by using create-react-app.

## If you've previously installed create-react-app globally, it is recommended that you uninstall the package to ensure npx always uses the latest version of create-react-app.

To uninstall, run this command: npm uninstall -g create-react-app.

Run this command to create a React application named my-react-app:

* npx create-react-app my-react-app

The create-react-app will set up everything you need to run a React application.

## **Run the React Application:**

Now you are ready to run your first real React application!

Run this command to move to the my-react-app directory:

* cd my-react-app

Run this command to run the React application my-react-app:

* npm start

A new browser window will pop up with your newly created React App! If not, open your browser and type localhost:3000 in the address bar.

The result:

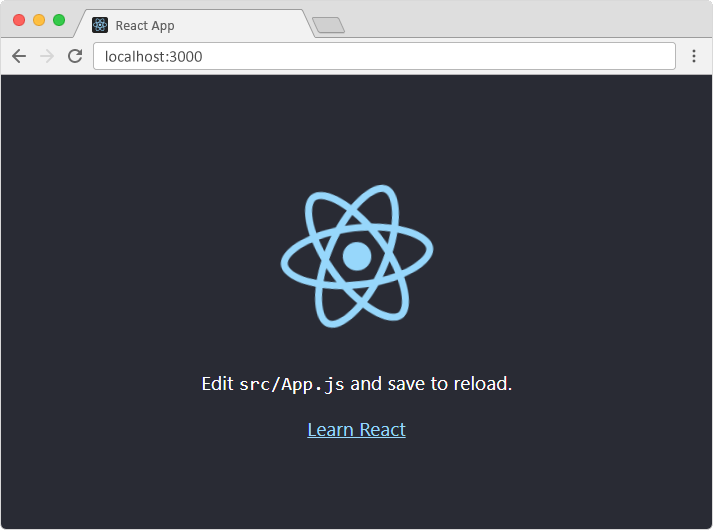


Figure 3.3.2: React JS Installation

**3.3.4 Express JS:**



Figure 3.3.3: Express JS

Express.js is a web application framework for Node.js that allows developers to build web applications and APIs quickly and easily. It provides a simple and flexible way to handle HTTP requests and responses, as well as middleware functions for adding additional functionality to your applications.

**Some of the key features of Express.js include:**

**Routing:** Express.js allows you to define routes for different HTTP methods and URLs, making it easy to handle different requests. You can also handle parameters and query strings in your routes.

**Middleware:** Express.js uses middleware functions to process requests before they reach the route handlers. Middleware can be used for tasks like authentication, request validation, logging, and more.

**Template engines:** Express.js supports a wide variety of template engines, allowing you to render dynamic HTML pages on the server and send them to the client.

**Error handling:** Express.js provides a default error handling mechanism, allowing you to handle errors and send appropriate responses to the client. You can also define your own error handlers for specific routes or middleware.

**Static file serving:** Express.js makes it easy to serve static files, such as HTML, CSS, and images, directly from the file system.

Overall, Express.js is a lightweight and flexible framework that provides all the tools you need to build robust web applications with Node.js. It has a large and active community, with numerous plugins and middleware available to extend its functionality.

**Why Express ?**

Develops Node.js web applications quickly and easily.

It’s simple to set up and personalise.

Allows you to define application routes using HTTP methods and URLs.

Includes a number of middleware modules that can be used to execute additional requests and responses activities.

Simple to interface with a variety of template engines, including Jade, Vash, and EJS.

Allows you to specify a middleware for handling errors.

**Installing Express:**

We can install it with npm. Make sure that you have [Node.js](https://www.geeksforgeeks.org/installation-of-node-js-on-windows/)and[npm](https://www.geeksforgeeks.org/node-js-npm-node-package-manager/)installed.

Step 1: Creating a directory for our project and make that our working directory.

* $ mkdir gfg
* $ cd gfg

Step 2: Using npm init command to create a package.json file for our project.

* $ npm init

This command describes all the dependencies of our project. The file will be updated when adding further Installing Express

Step 3: Now in your *gfg(name of your folder)* folder type the following command line:

* $ npm install express --save

Now let’s understand the working of express.js through an example.

**Project Structure:** It will look like the following.

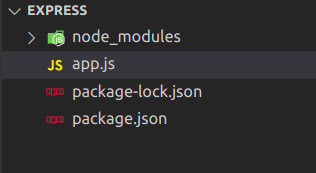


Figure 3.3.4: Installation of Express JS

Example: Write the following code in app.js.

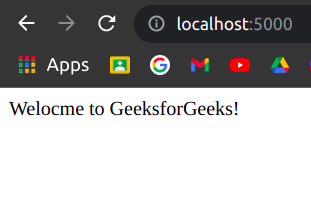
**app.js**

|  |
| --- |
| var express = require('express');  var app = express();  app.get('/', function (req, res) {  res.send("Welcome to GeeksforGeeks!");  });  app.listen(5000); |

Step to run the application: Start the app by typing following command.

* node app.js

**Output:**



**Figure 3.3.5: Output of Node App**

**3.3.5 Node.js**



**Figure 3.3.6: Node JS**

**Node.js is a runtime environment that allows developers to run JavaScript outside of a web browser. It uses the V8 JavaScript engine by Google, which makes it a powerful and efficient tool for building server-side applications. Node.js is particularly well-suited for building scalable network applications and is widely used for web servers, real-time applications, and command-line scripts. It provides a vast ecosystem of packages and modules through the npm package manager, which allows developers to easily add functionality to their applications.**

## **What is Node.js?**

* Node.js is an open source server environment
* Node.js is free
* Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
* Node.js uses JavaScript on the server

**Why Node.js?**

**Node.js uses asynchronous programming!**

A common task for a web server can be to open a file on the server and return the content to the client.

Here is how PHP or ASP handles a file request:

1. Sends the task to the computer's file system.
2. Waits while the file system opens and reads the file.
3. Returns the content to the client.
4. Ready to handle the next request.

Here is how Node.js handles a file request:

1. Sends the task to the computer's file system.
2. Ready to handle the next request.
3. When the file system has opened and read the file, the server returns the content to the client.

Node.js eliminates the waiting, and simply continues with the next request.

**What Can Node.js Do?**

* Node.js can generate dynamic page content
* Node.js can create, open, read, write, delete, and close files on the server
* Node.js can collect form data
* Node.js can add, delete, modify data in your database

**What is a Node.js File?**

* Node.js files contain tasks that will be executed on certain events
* A typical event is someone trying to access a port on the server
* Node.js files must be initiated on the server before having any effect
* Node.js files have extension ".js"

**3.3.6 mongoDB:**



Figure 3.3.7: MongoDB

MongoDB is a NoSQL document database that is based on the concept of collections and documents. It stores data in a flexible schema format called BSON (Binary JSON), allowing for dynamic and nested data structures. MongoDB provides high scalability and performance, making it suitable for large-scale applications that require fast and efficient data access.

Some key features of MongoDB include:

**Flexible Data Model:** MongoDB's flexible document model allows for easy representation of complex data structures, with support for nested arrays and documents.

Replication and High Availability: MongoDB supports replica sets, which are groups of database nodes that maintain multiple copies of the data for high availability and data redundancy.

**Sharding:** MongoDB can distribute data across multiple machines using sharding, which allows for horizontal scaling and improved performance.

Indexing and Querying: MongoDB supports indexing on any field and provides powerful query capabilities, including support for complex joins and aggregations.

**Transactions:** MongoDB now supports multi-document transactions, allowing for consistent and atomic operations across multiple documents.

**Horizontal Scalability:** MongoDB can be scaled horizontally by adding more nodes to a cluster, enabling the database to handle increased traffic and workload.

MongoDB uses a JSON-like query language called MongoDB Query Language (MQL) for querying and manipulating data. It also provides drivers for various programming languages, making it easy to integrate MongoDB into different applications and frameworks.

Overall, MongoDB is a popular choice for developers and organizations due to its ease of use, flexibility, and scalability, making it well-suited for a wide range of use cases, including web and mobile applications, content management systems, real-time analytics, and IoT (Internet of Things) applications.

**Create a Database:**

* From the databases tab, click the Create Database button to bring up the Create Database dialogue.
* In the dialog, enter the name of the database to create and its first collection. Both the database name and the collection name are required.
* If you want to create a [capped collection](https://www.mongodb.com/docs/manual/core/capped-collections/), select the **Capped Collection** checkbox and enter the maximum bytes.
* If you want to use [custom collation](https://www.mongodb.com/docs/manual/reference/collation/#collation-document) on the collection, select the **Use Custom Collation** checkbox and select the desired collation settings.
* If your deployment is connected using **In-Use Encryption**, you can use [Queryable Encryption](https://www.mongodb.com/docs/v6.0/core/queryable-encryption/" \t "_self) on the newly created collection.
* Check the **Queryable Encryption** option and indicate the following encryption properties:

[**Encrypted Fields.**](https://www.mongodb.com/docs/v6.0/core/queryable-encryption/fundamentals/encrypt-and-query/)

(Optional) [KMS Provider.](https://www.mongodb.com/docs/v6.0/core/queryable-encryption/fundamentals/kms-providers/)

(Optional) [Key Encryption Key.](https://www.mongodb.com/docs/v6.0/core/queryable-encryption/fundamentals/keys-key-vaults/)

* Click *Create Database* to create the database and its first collection. Drop a Database.
* Click the trash can icon for the database.
* From the Databases tab, to delete a database, click on the trash can icon for that database. A confirmation dialog appears.
* Confirm the database to delete.
* In the dialog, enter the name of the database to delete.
* Click Drop Database to delete the database.

**Databases Tab**

The **Databases** tab lists the existing databases for your MongoDB deployment. To access the **Databases** tab, click the cluster name in the upper left corner and select the **Databases** tab.

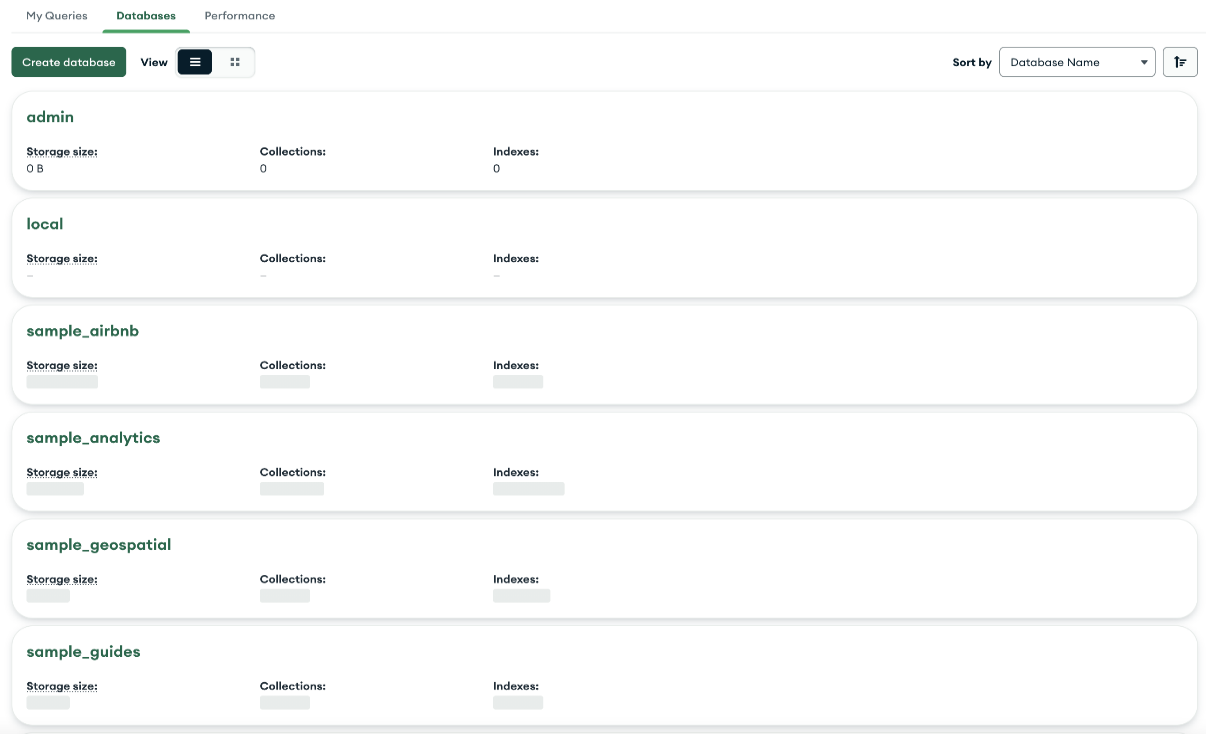


Figure 3.3.8: Database Tab

**3.4 TESTING OF PRODUCT**

**Unit testing in Node.js**

Unit testing is a software testing method in which individual pieces of code (usually the smallest piece of code that can be logically isolated in a system) are tested in isolation. Unit tests should be isolated so that there are no external dependencies.

First, unit testing makes it easier to identify bugs in code. Appropriate test cases should be written for every piece of code to ensure that they meet specifications and provide the desired output. Any changes that result in failing tests will indicate that an error or bug has been introduced. Additionally, unit testing makes it easier to narrow down the offending piece of code.

Second, unit tests act as self-documentation. A new team member can gain a better understanding of the code base by going through unit tests.

Third, the debugging process is made a lot easier. This is because when the test fails, the focus will be on the latest changes made.

Fourth, refactoring code is made easier, because changes can be verified using tests to ensure that the unit being tested still behaves in the desired manner.

Finally, costs that would be incurred fixing bugs or system outages are reduced.

Testing frameworks provide a set of reusable components or modules, such as test runners and utilities, for running automated tests. The testing framework is responsible for:

1. Describing the format used to convey test expectations
2. Creating a way of accessing the application or code to be tested
3. Executing the tests
4. Reporting test results

They are particularly useful when tests are an integral part of your continuous integration process. Frameworks are built for a specific type of testing: unit, integration, functional, or combinations of these.

**What makes a good Node.js testing framework?**

There are a thousand and one testing frameworks out there. To pick something that works for your use case, you need to evaluate each framework based on your project needs and how effective you consider it to be for your team.

Below are six key characteristics of a strong Node.JS testing framework:

1. Ease of setup: getting up and running with your tests should take a minimal amount of effort
2. Well-supported: there is plenty of excellent documentation and communities to get help
3. Wide array of feature sets: the framework has things such as matchers, spies, and mocking built in
4. Speed: for tests that are CPU-bound, choosing the right framework can save you a lot of time during test runs
5. Ease of reporting: coverage reports should be easy to generate using built-in reporting and external reporting libraries should be easy to integrate
6. Ease of integration: a good testing library should be easy to integrate into your continuous integration process

We’ll compare four of these Node.js unit testing frameworks:

1. Mocha
2. Jest
3. Jasmine
4. AVA

**1.Mocha**

[Mocha](https://mochajs.org/) has been around for quite a while; it was initially released in November 2011. However, unlike other frameworks like Jest and Jasmine, it relies on third-party assertions, mocking, and spying tools like [Sinon](https://sinonjs.org/) and [Chai](https://www.chaijs.com/). It is very extensible and has a lot of plugins, extensions, and libraries designed to run on top of it.

**Pros**

* Highly extensible with support for various assertion and mocking libraries
* Easy asynchronous testing
* Adding support for generators to test suites is relatively easy. Using the [co-mocha](https://www.npmjs.com/package/co-mocha) package, all you have to do is require it in your tests and you’re ready to use generators
* Supported by some CI servers and plugins for others

**Cons**

* The use of extra libraries can introduce configuration complexity and increases maintenance work
* No auto-mocking

**2.Jest**

[Jest](https://jestjs.io/) is a JavaScript testing framework developed and regularly maintained by Facebook. Its popularity has grown steadily since 2016, when only six percent of respondents to that year’s “[State of JS](http://2016.stateofjs.com/2016/testing/)” survey said they had used Jest before and would use it again. This figure climbed to [a quarter of respondents in 2017](https://2017.stateofjs.com/2017/testing/results/) before reaching [40 percent in 2018](https://2018.stateofjs.com/testing/overview/). As of the most recent edition, a whopping 73 percent of JavaScript developers had tried Jest and plan to use it again.

**Pros**

* Comprehensive [documentation](https://jestjs.io/docs/en/getting-started) includes detailed instructions to help you set up testing, write various types of tests, and use its many features, as well as great examples
* Easy setup with flexible and easy configuration and less boilerplate code than other frameworks
* Parallel test running enabled
* Optimal performance: tests are parallelized by running them in their own processes to maximize performance
* Useful features such as snapshots, coverage, and test watching

**Cons**

* Displays multiple messages for the same error
* It can require more dependencies during initial setup (e.g., [Babel](https://blog.logrocket.com/why-you-dont-need-babel/))

**3.Jasmine**

Developed by Pivotal Labs and released in 2010, Jasmine has been around for a lot longer than Jest. It aims to run on any JavaScript-enabled platform and is highly flexible and compatible with a variety of other testing frameworks and libraries, including Sinon and Chai. Due to its longevity, it has developed a significant community and enjoys ample support with loads of libraries, blog articles, and tutorials.

**Pros**

* Simple to set up — Jasmine has a CLI tool that creates a spec folder and a JSON configuration file, so with one command you’re ready to start testing your code
* Thoroughly tested, documented, and supported by numerous tutorials on how to use it.
* Behaviour-driven development focused with descriptive syntax
* Supported by many CI servers with plugins available for those that don’t have out-of-the box support

**Cons**

* Unfriendly error logs
* Test files must have a specific suffix (e.g., spec.js)
* Assertion library is not as rich as Chai

**4.AVA**

Minimalism is the focus of AVA. It has a simple API while still supporting advanced features. It achieves its blazing speed by running tests in parallel as separate Node processes. Unlike other testing frameworks such as Jest and Jasmine, it does not create test global.

**Pros**

* Easy to use. To install and setup AVA, all you have to do is run npm init ava
* Parallel test running
* Native ES6/ES7 support
* Built-in support for async functions
* If a promise is returned, you don’t need to end the test yourself; it will end when the promise resolves

**Cons**

* AVA is relatively new. The community is still growing and there isn’t a lot of documentation or tutorials like other testing frameworks
* AVA has a lot of open issues on GitHub

***Jenkins***

[Jenkins](https://www.softwaretestinghelp.com/integration-of-jenkins-with-selenium-webdriver/) is a continuous integration system that can hook into your version control (e.g. git) and automatically execute mocha any time a commit occurs. This means your product is being tested every time a change occurs.

**3.5 FEASIBILITY TESTING** :

Feasibility study is carried out based on many purposes to analyse whether software product will be right in terms of development, implantation, contribution of project to the organization etc.

**Types of Feasibility Study:**

The feasibility study mainly concentrates on below five mentioned areas. Among these Economic Feasibility Study is most important part of the feasibility analysis and Legal Feasibility Study is less considered feasibility analysis.

**1.Technical Feasibility**-

In Technical Feasibility current resources both hardware software along with required technology are analysed/assessed to develop project. This technical feasibility study gives report whether there exists correct required resources and technologies which will be used for project development. Along with this, feasibility study also analyses technical skills and capabilities of technical team, existing technology can be used or not, maintenance and up-gradation is easy or not for chosen technology etc.

**2.Operational Feasibility-**

In Operational Feasibility degree of providing service to requirements is analysed along with how much easy product will be to operate and maintenance after deployment. Along with this other operational scopes are determining usability of product, Determining suggested solution by software development team is acceptable or not etc.

**3.Economic Feasibility-**

In Economic Feasibility study cost and benefit of the project is analysed. Means under this feasibility study a detail analysis is carried out what will be cost of the project for development which includes all required cost for final development like hardware and software resource required, design and development cost and operational cost and so on. After that it is analysed whether project will be beneficial in terms of finance for organization or not.

**4.Legal Feasibilty-**

In Legal Feasibility study project is analysed in legality point of view. This includes analysing barriers of legal implementation of project, data protection acts or social media laws, project certificate, license, copyright etc. Overall it can be said that Legal Feasibility Study is study to know if proposed project conform legal and ethical requirements.

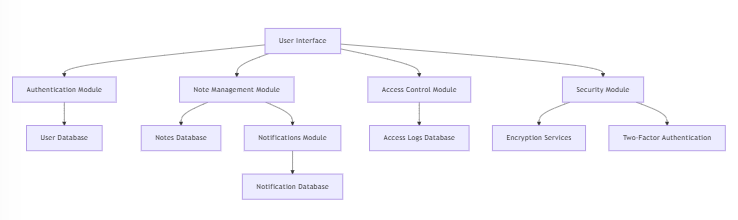
**Aim of feasibility study:**

* The overall objective of the organization are covered and contributed by the system or not.
* The implementation of the system be done using current technology or not.
* can the system be integrated with the other system which are already exist.
* Improves project teams’ focus
* Identifies new opportunities
* Provides valuable information for a “go/no-go” decision
* Narrows the business alternatives

**CHAPTER 4**

**DESIGN SPECIFICATIONS**

**4.1 SYSTEM ARCHITECTURE**



This diagram represents the architecture of a web or software application with a structured modular approach, focusing on authentication, note management, access control, and security. The User Interface (UI) serves as the central point for interaction, connecting different functional modules.

**Explanation:**

**User Interface**

* The primary access point for users to interact with the application.
* Connects to various modules that handle authentication, note management, access control, and security.
* It connects to four modules:

1. Authentication Module
2. Note Management Module
3. Access Control Module
4. Security Module

**1.Authentication Module:**

This module is responsible for user identity verification, ensuring that only authorized users can access the application.

* Stores user credentials (e.g., email, passwords, authentication tokens).
* May support different login methods (e.g., email/password, social login, OAuth).

**2. Note Management Module:**

This module handles the creation, storage, and management of user-generated notes

or documents.

* Users can create, edit, delete, and share notes.
* Organizing notes into categories, tags, or folders.
* Setting reminders and automated alerts.

**3. Access Control Module:**

This module ensures that users have appropriate permissions to access or modify data.

* Defines **user roles and permissions** (e.g., Admin, Editor, Viewer).
* Ensures **data privacy** by preventing unauthorized users from accessing sensitive information.
* Logs **audit trails** for security and compliance.

**4. Security Module:**

This module provides additional security measures to protect user data and prevent

breaches.

* Prevents unauthorized access with strong security mechanisms.
* Ensures **data confidentiality** using encryption.
* Provides **user identity verification** beyond just passwords.

**4.2 MODULES**

* User Management
* Product Management
* Transaction Management
* Search and Filter
* Negotiation Management
* Payment Integration
* Admin Control Panel

**4.3 MODULE DESCRIPTION**

**1. User Management**

* **Account Creation and Authentication**: Allows users (farmers, customers, retailers, and admins) to register for an account and securely authenticate their identity through login credentials or social media logins (e.g., Google, Facebook).
* **Session Management**: Tracks and manages user sessions, ensuring users remain logged in while providing the option to log out and terminate any suspicious or inactive sessions.
* **Role-Based Access Control**: Implements role-based access control to manage permissions and access levels effectively, ensuring that each user type (farmer, customer, retailer, admin) has access to the appropriate features and data.
* **Data Privacy Compliance**: Ensures compliance with regulations such as GDPR and CCPA, allowing users to manage their data rights and ensuring that personal information is handled securely and with transparency.
* **User Activity Monitoring**: Logs user activity within the application, helping monitor usage patterns and detect anomalies or unusual behavior, thus enhancing the overall security of the platform.

**2. Product Management**

* **Product Listing and Management**: Allows farmers to list agricultural products, update product details (such as name, quantity, price, and description), and manage their inventory effectively.
* **Price Setting and Updates**: Farmers can set the initial price for their products and make changes to the pricing based on market conditions or buyer negotiations.
* **Product Deletion and Editing**: Gives farmers the ability to remove products from the platform or make edits to product information when necessary.

**3. Transaction Management**

* **Negotiation Handling**: Manages the process of price negotiations between buyers and sellers, including the ability to send offers, accept offers, and make counteroffers.
* **Payment Processing**: Handles the secure processing of payments through integrated payment gateways, ensuring that transactions are completed safely and efficiently.
* **Transaction History**: Allows both buyers and sellers to track and review their transaction history, providing transparency and clarity on past purchases and sales.

**4. Search and Filter**

* **Product Search**: Allows customers and retailers to search for specific products based on criteria such as product name, type, or category.
* **Filter Options**: Provides users with filtering options like price range, location, and product type, helping them narrow down their search and find the best matches for their needs.

**5. Negotiation Management**

* **Price Offer Management**: Allows customers and retailers to make price offers on products and send them to farmers for approval, rejection, or counteroffer.
* **Counteroffer System**: Enables farmers to send counteroffers back to the buyers, creating a dynamic negotiation process until both parties agree on a final price.
* **Negotiation History**: Tracks the history of price negotiations, providing both farmers and buyers with a record of all interactions during the negotiation process.

**6. Payment Integration**

* **Secure Payment Gateway**: Integrates with secure payment gateways (e.g., credit/debit cards, digital wallets, UPI) to ensure the safe and efficient processing of payments between buyers and sellers.
* **Transaction Confirmation**: Sends payment confirmations to both parties after a successful transaction, ensuring that the payment process is transparent and reliable.
* **Escrow Payment System**: Optionally integrates escrow payments, where the payment is held until the buyer confirms receipt of the product, adding a layer of security for both parties.

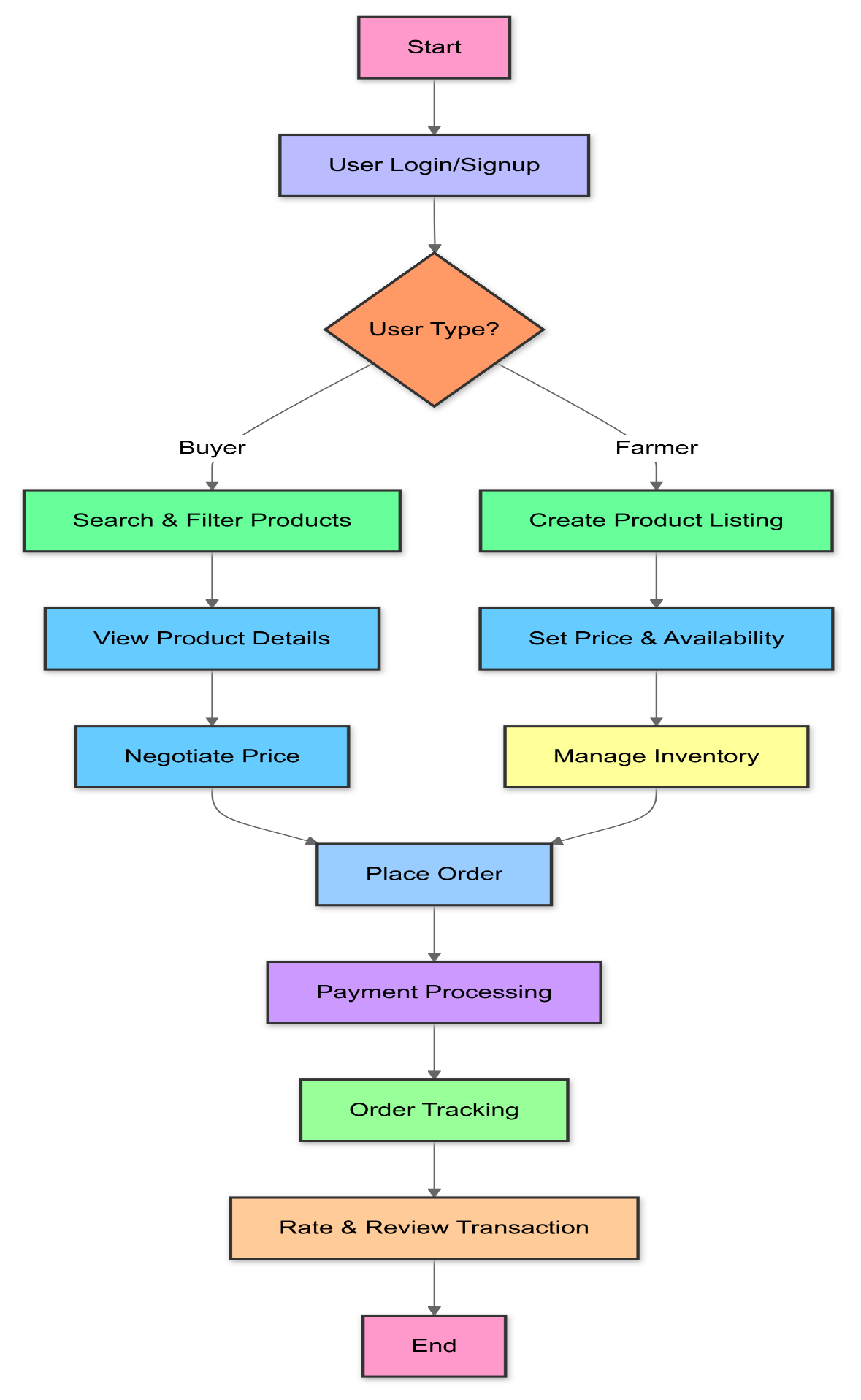
**7. Admin Control Panel**

* **Product Verification**: Admins can review and verify product listings to ensure compliance with quality standards and platform guidelines.
* **User Management**: Admins can manage user accounts by approving or suspending users, handling disputes, and ensuring that all users are adhering to platform rules.
* **Transaction Monitoring**: Allows admins to monitor transactions for fraud or suspicious activity and take appropriate action if needed.
* **Platform Security and Monitoring**: Ensures the overall security of the platform by monitoring for vulnerabilities, ensuring that the payment system functions properly, and managing data security.
* **Reporting and Insights**: Admins can access detailed reports on platform activity, user behavior, and financial transactions, helping with decision-making and platform improvement.

**CHAPTER 5**

**METHODOLOGY**

The workflow of a direct farmer-to-customer web application, enabling efficient price negotiation and streamlined transactions between buyers and farmers can be explained using the flow chart. The flow chart is given below:

****

Flowchart 5.1: workflow of a direct farmer-to-customer web application

* 1. **WORKING PROCEDURE OF WEB APP**

**1.Start:**

The Start node in the flowchart represents the beginning of the user journey in the farmer-to-customer web application. This step initiates the workflow, where users (either farmers or buyers) enter the platform to perform actions like listing products, searching for items, negotiating prices, and placing orders.

At this stage:

* The user accesses the platform (via a website or mobile app).
* They are presented with an option to log in or sign up to continue.

**2.User Login/Signup:**

The **User Login/Signup** step is where users authenticate themselves to access the platform. This ensures that only **registered users** (either buyers or farmers) can use the system’s features like product listing, price negotiation, and order placement.

**i. New Users (Signup Process)**

If the user is new, they must **register** by providing:

* Personal Information (e.g., Name, Email, Phone Number)
* Account Type Selection (Buyer or Farmer)
* Login Credentials (Username & Password)
* Verification (Email or OTP authentication for security)

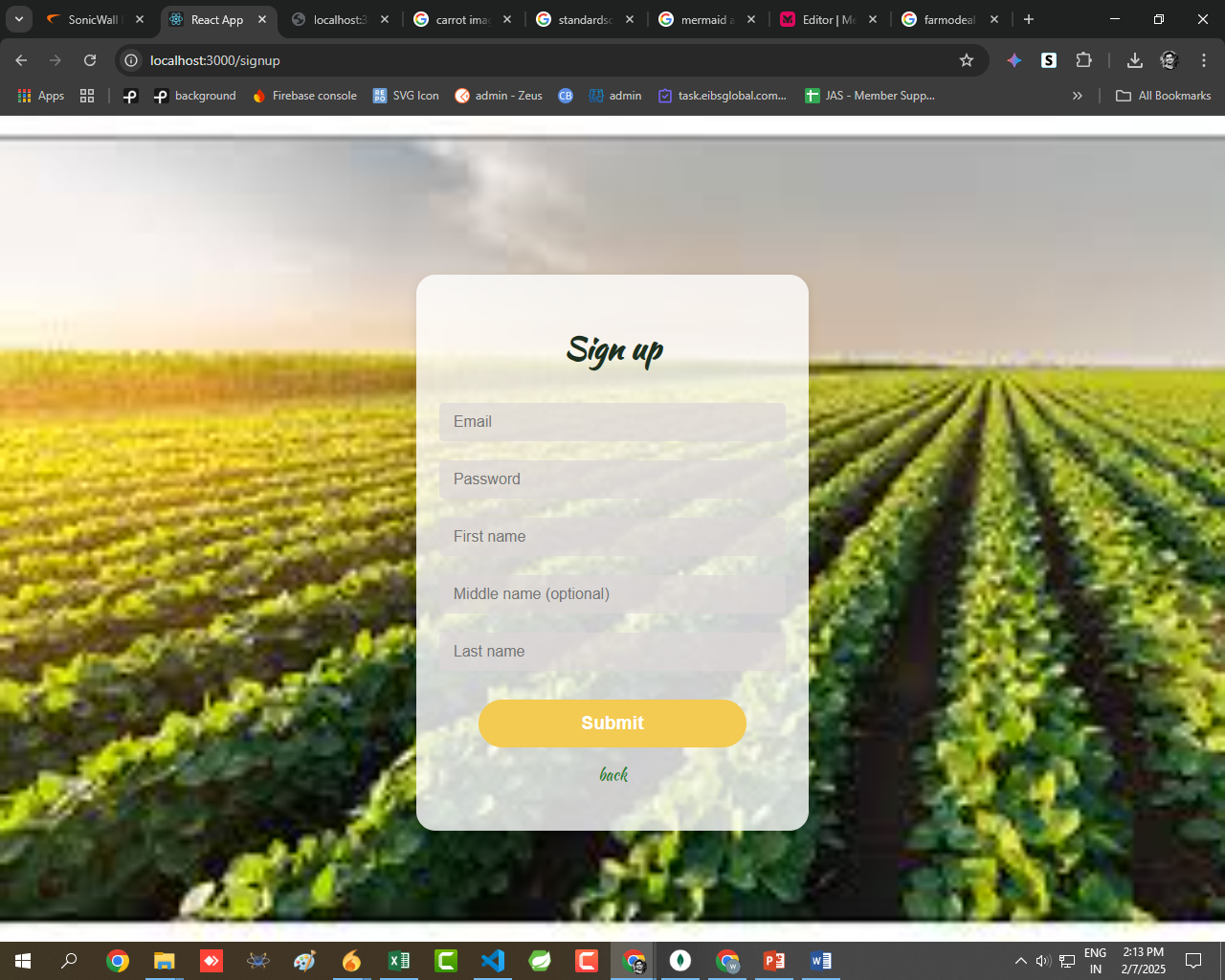


Figure 5.5.1: Register Page

**ii. Existing Users (Login Process)**

If the user already has an account, they can log in by entering:

* Username or Email
* Password
* Two-Factor Authentication (if enabled)

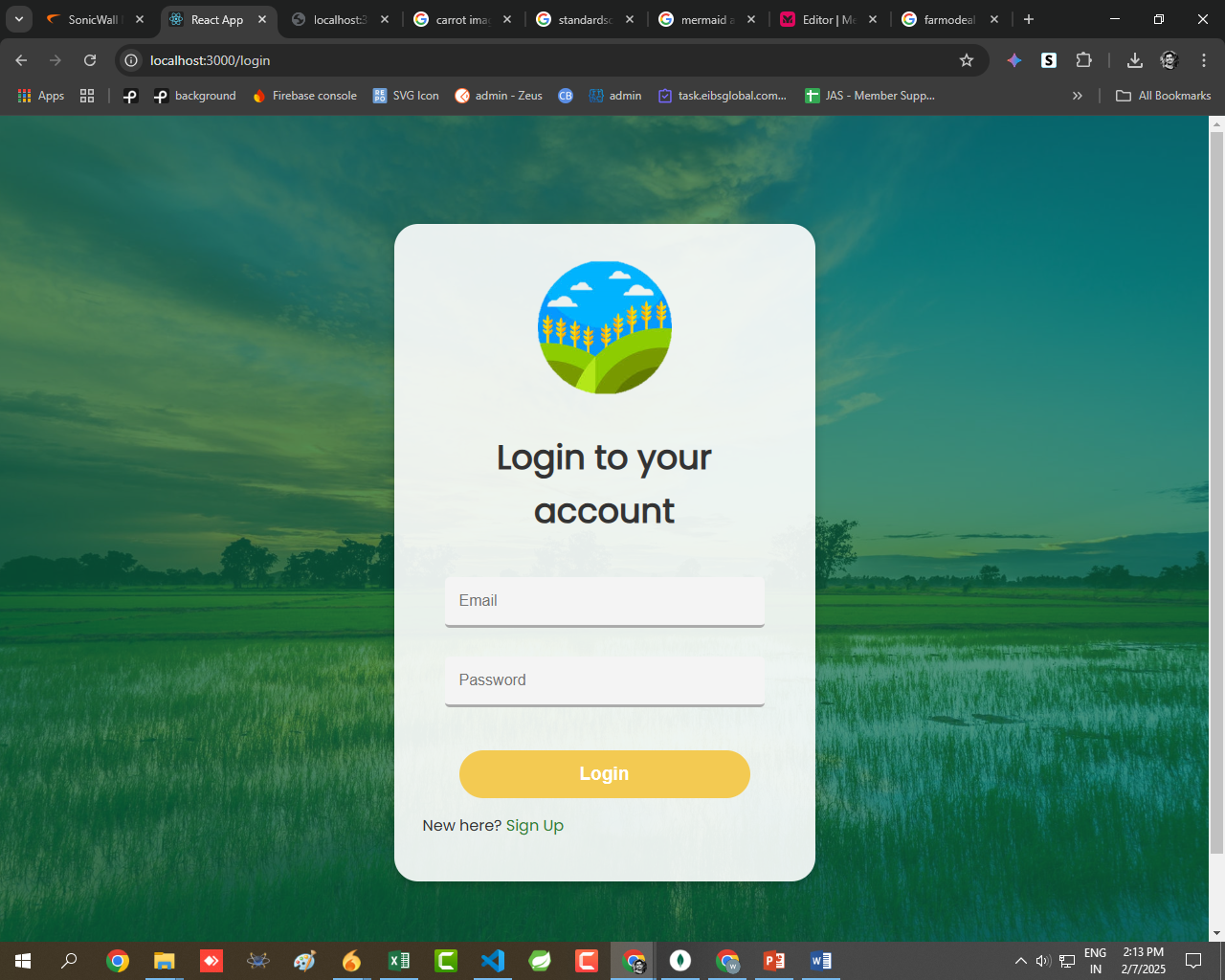


Figure 5.5.2: Login Page

Once you successfully logged in then it will goes to home page

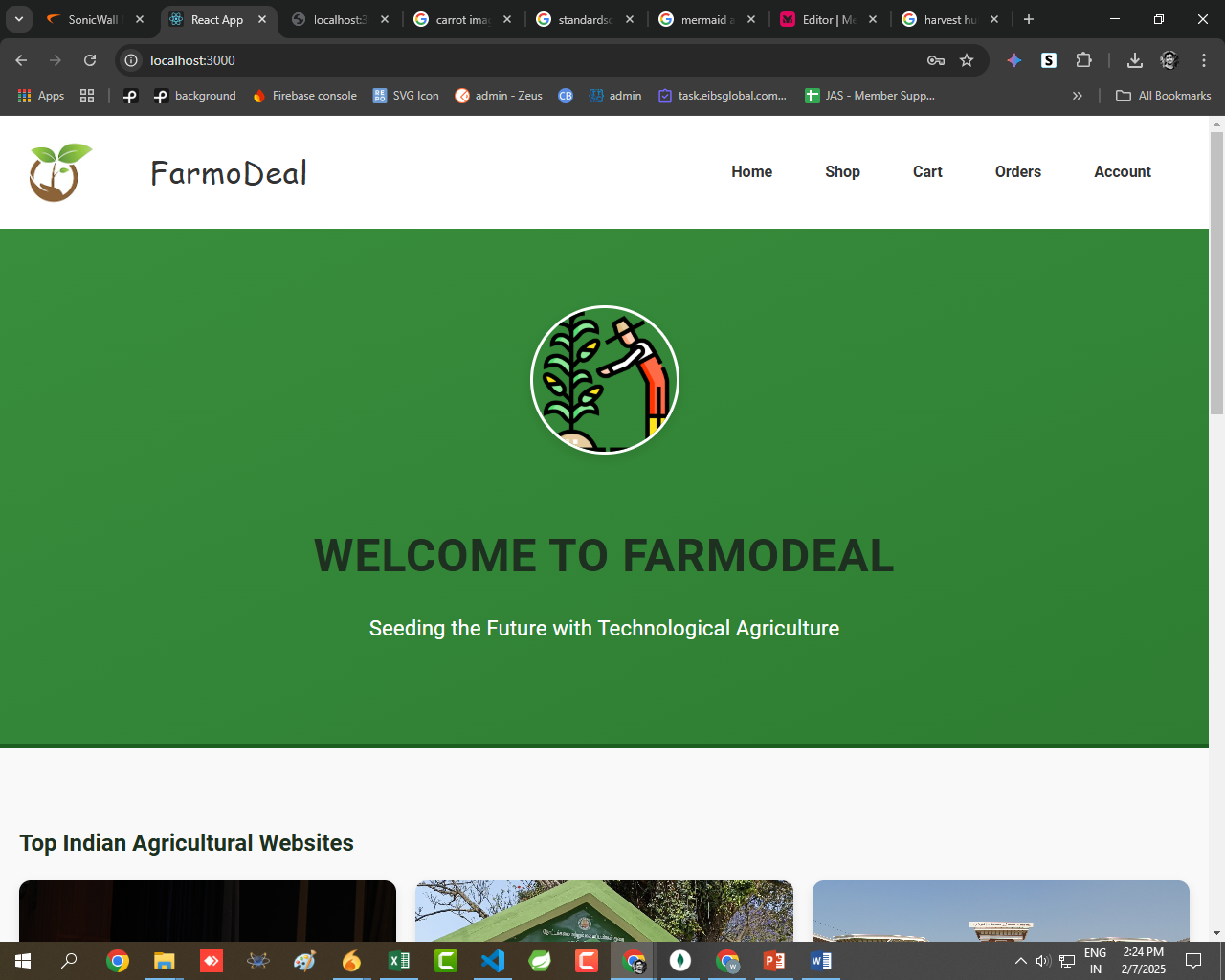


Figure 5.5.3: Home Page

**3.User Type?**

The User Type? step is a decision point in the flowchart where the system determines whether the user is a Buyer or a Farmer after logging in.

**1. Buyer**

If the user selects Buyer, they can:

1. Search & Filter Products
2. View Product Details
3. Negotiate Price

**1.Search & Filter Products**

The Search & Filter Products step allows buyers to efficiently find the products they need by browsing through the available listings. This feature enhances the user experience by making product discovery quick and convenient.

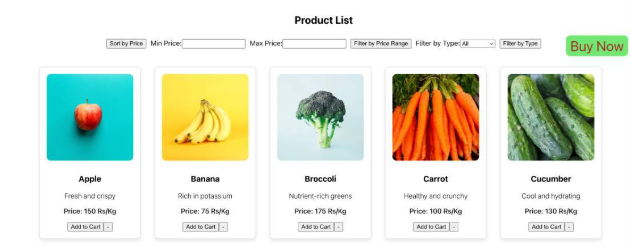


Figure 5.5.4: Search& Filter Page

**2.View Product Details**

The **View Product Details** step allows buyers to see detailed information about a selected product before making a purchase or negotiating a price. This ensures buyers can make informed decisions.

* **Product Description** – Detailed information, including name, category, and usage.
* **Pricing Information** – The listed price and any available discounts.
* **Images & Videos** – High-quality images or videos showcasing the product.
* **Seller Information** – Details about the farmer, including location and ratings.
* **Stock Availability** – Displays whether the product is in stock or available for pre-order.

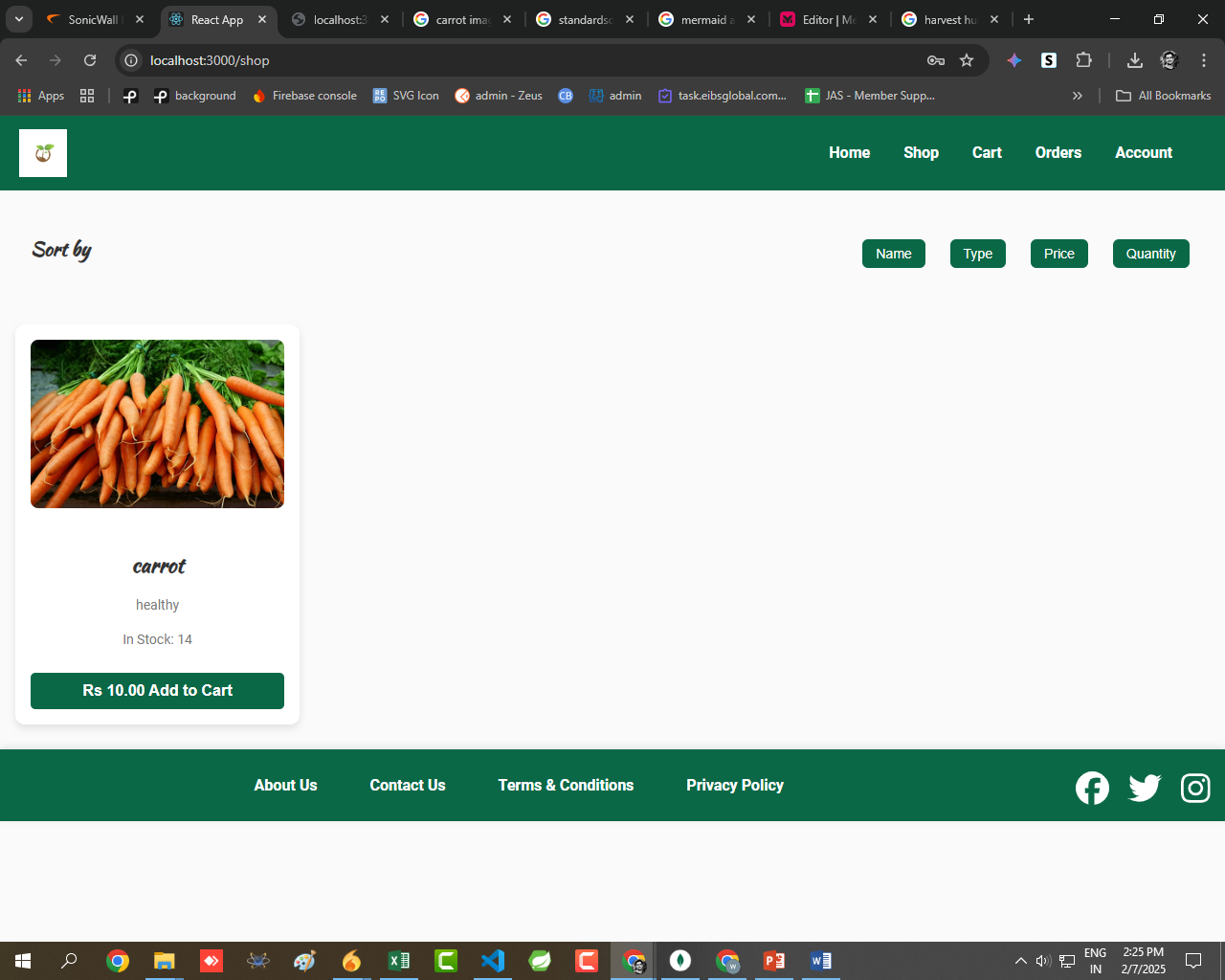


Figure 5.5.5: Product Details

**3.Negotiate Price**

The Negotiate Price step allows buyers and farmers to communicate directly and agree on a mutually beneficial price before placing an order. This feature gives buyers flexibility to get better deals and farmers control over their pricing.

Buyers can propose a lower price, and farmers can accept, reject, or counter the offer. Farmers may offer lower prices for larger purchases. The diagram shows below:

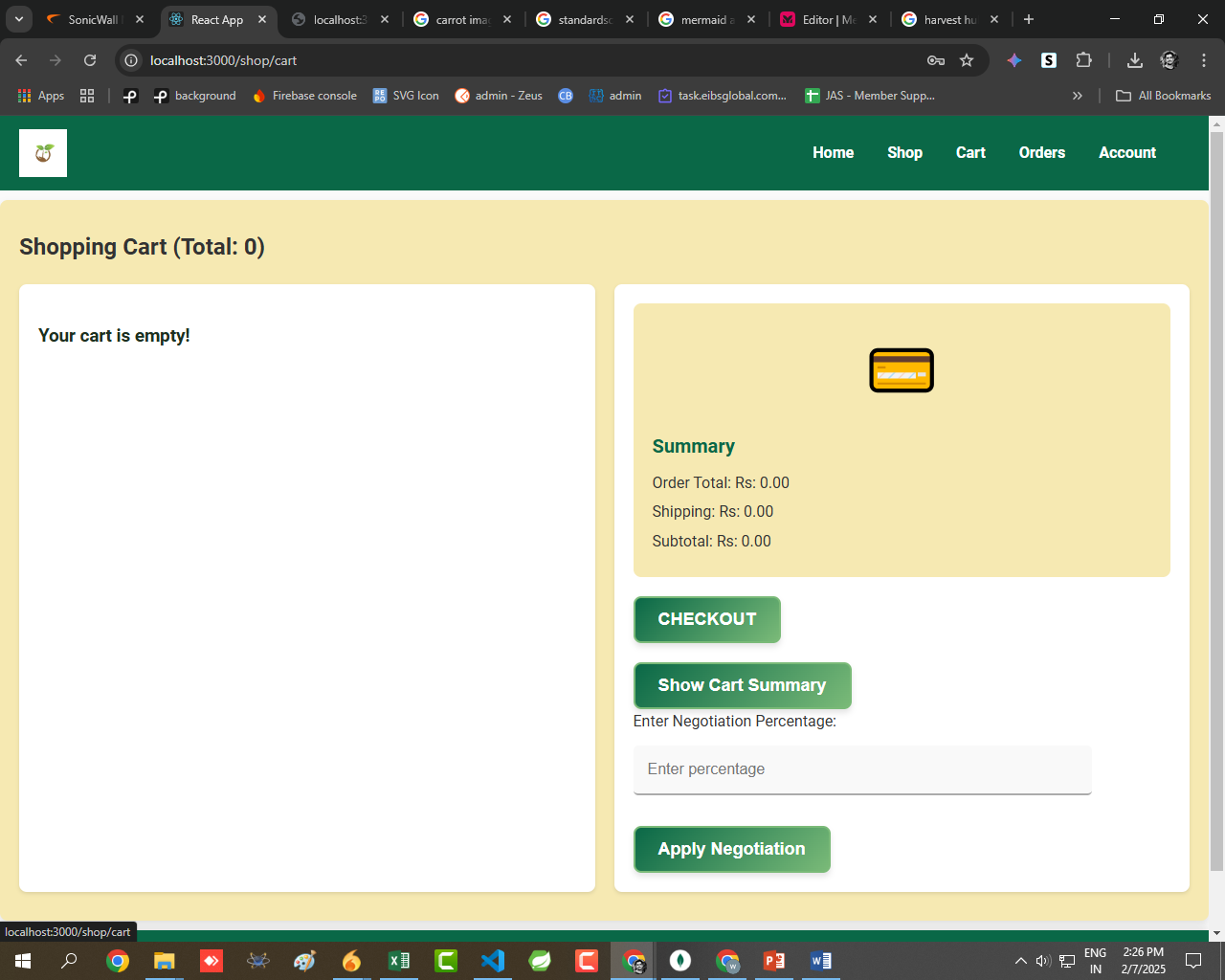


Figure 5.5.6: Price Negotiation

**2. Farmer**

If the user selects Farmer, they can:

1. Create Product Listing
2. Set Price & Availability
3. Manage Inventory

Once the user chooses their role, they are directed to the corresponding actions. This step ensures that users access the features relevant to their needs.

**1.Create Product Listing**

The Create Product Listing step allows farmers to showcase their products on the platform, making them available for buyers to view and purchase. This is a crucial step for farmers to manage their sales and reach potential customers directly.

1. Product Name & Description – Farmers provide a clear title and details about the product.
2. Category Selection – Farmers categorize products (e.g., Fruits, Vegetables, Dairy).
3. Upload Images/Videos – High-quality images or videos to attract buyers.
4. Pricing & Discounts – Farmers set the selling price and any available discounts.
5. Stock Availability – Indicate the quantity available for sale.

**2.Set Price & Availability**

The Set Price & Availability step allows farmers to define the cost of their products and manage stock levels, ensuring buyers get accurate and up-to-date information.

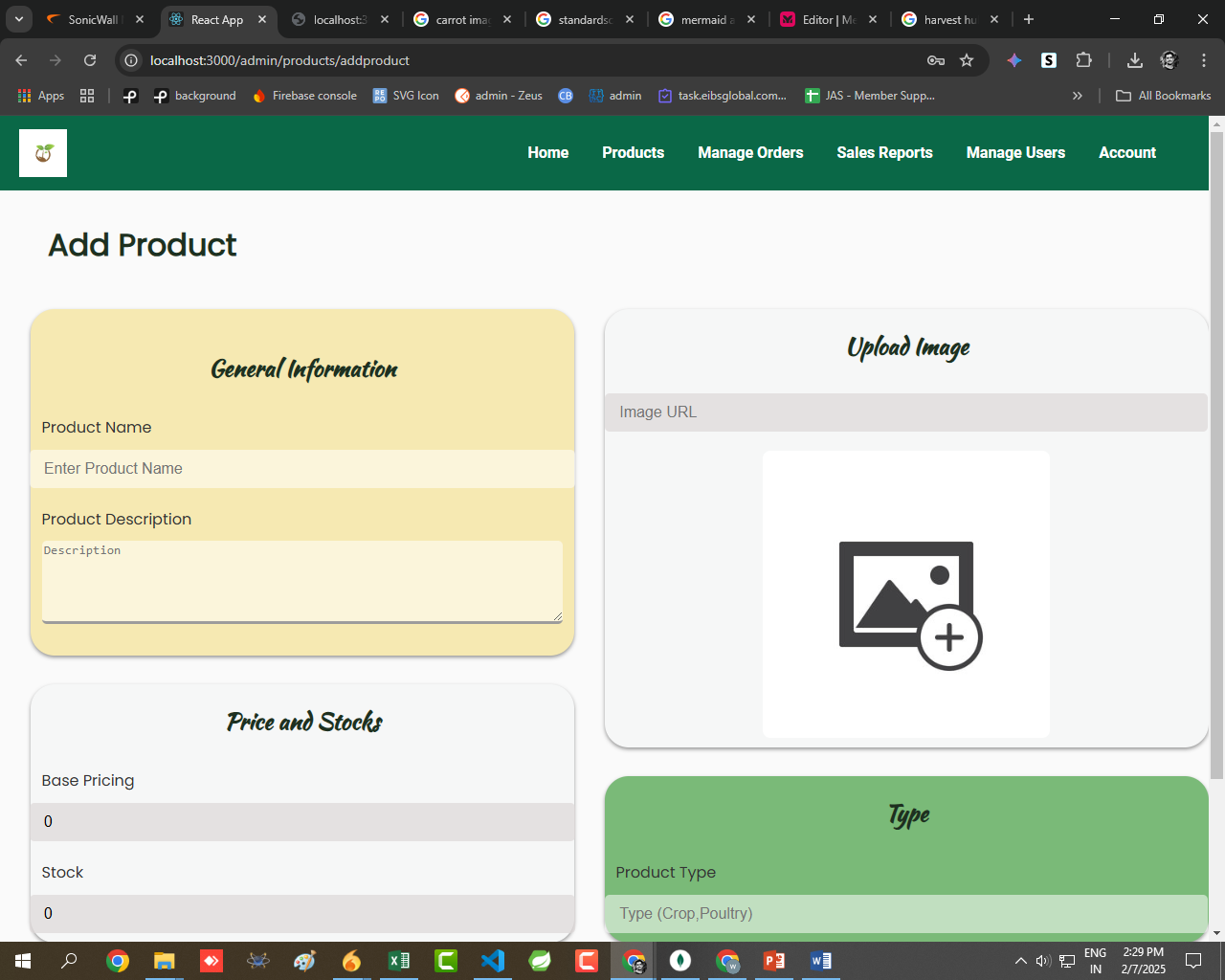


Figure 5.5.7: Product Listing & Setting Price

**3.Manage Inventory**

The Manage Inventory step allows farmers to track, update, and maintain their stock levels to ensure product availability and prevent overselling. This feature helps farmers keep their listings accurate and reliable for buyers.

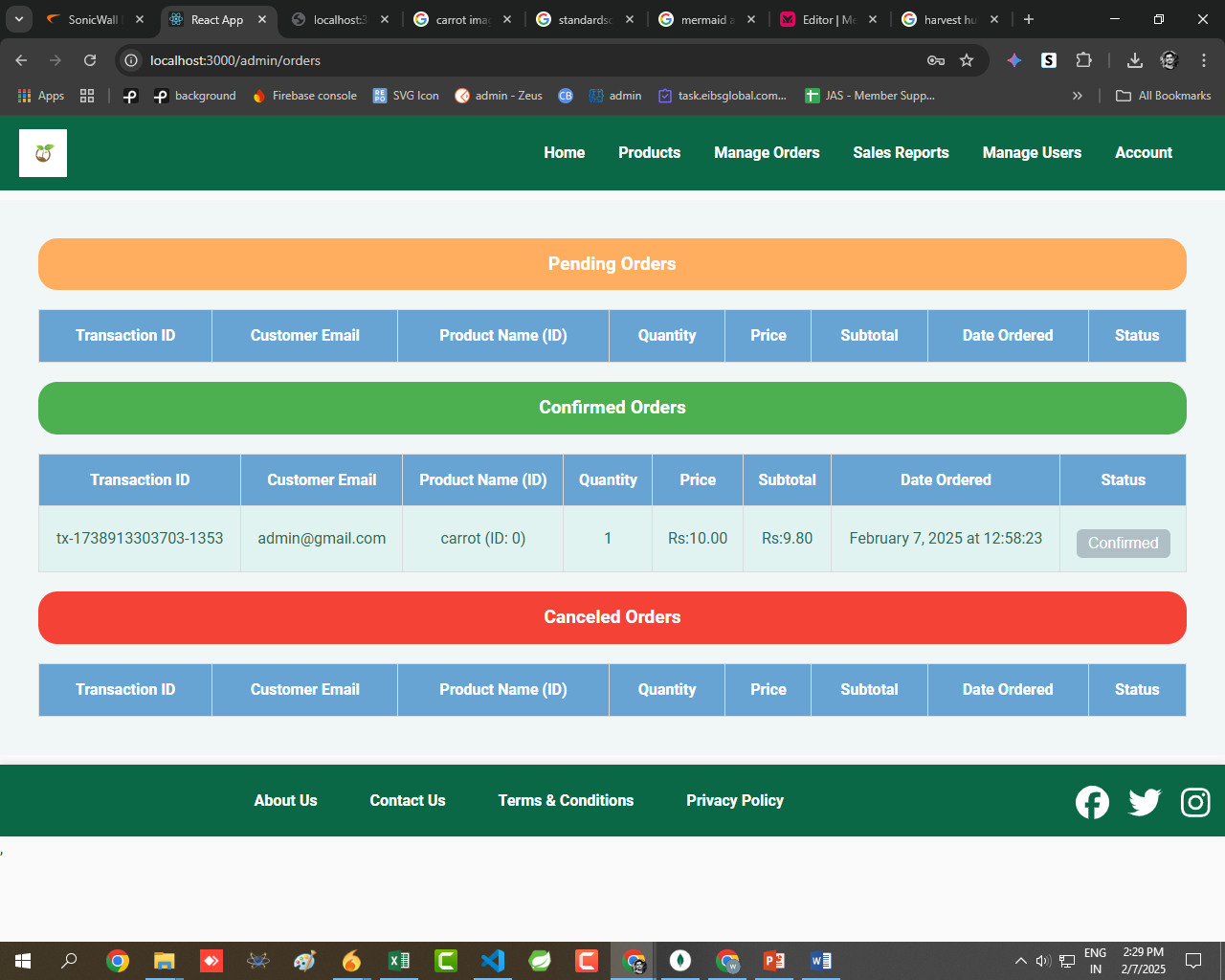


Figure 5.5.8: Manage Product

**Sales Report**

The Sales Report feature provides farmers with insights into their sales performance, helping them track revenue, analyze trends, and make informed business decisions.

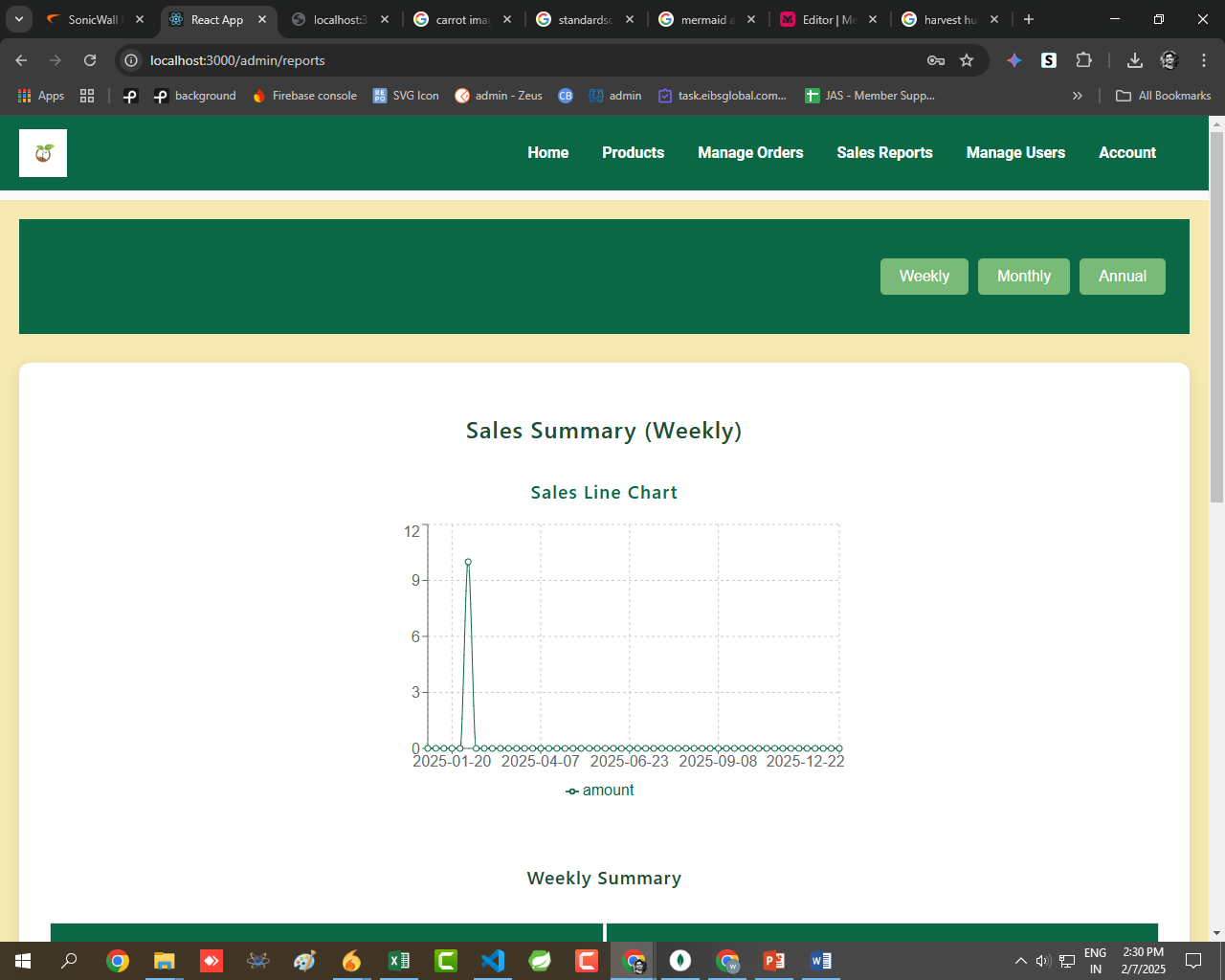


Figure 5.5.9 Sales Report

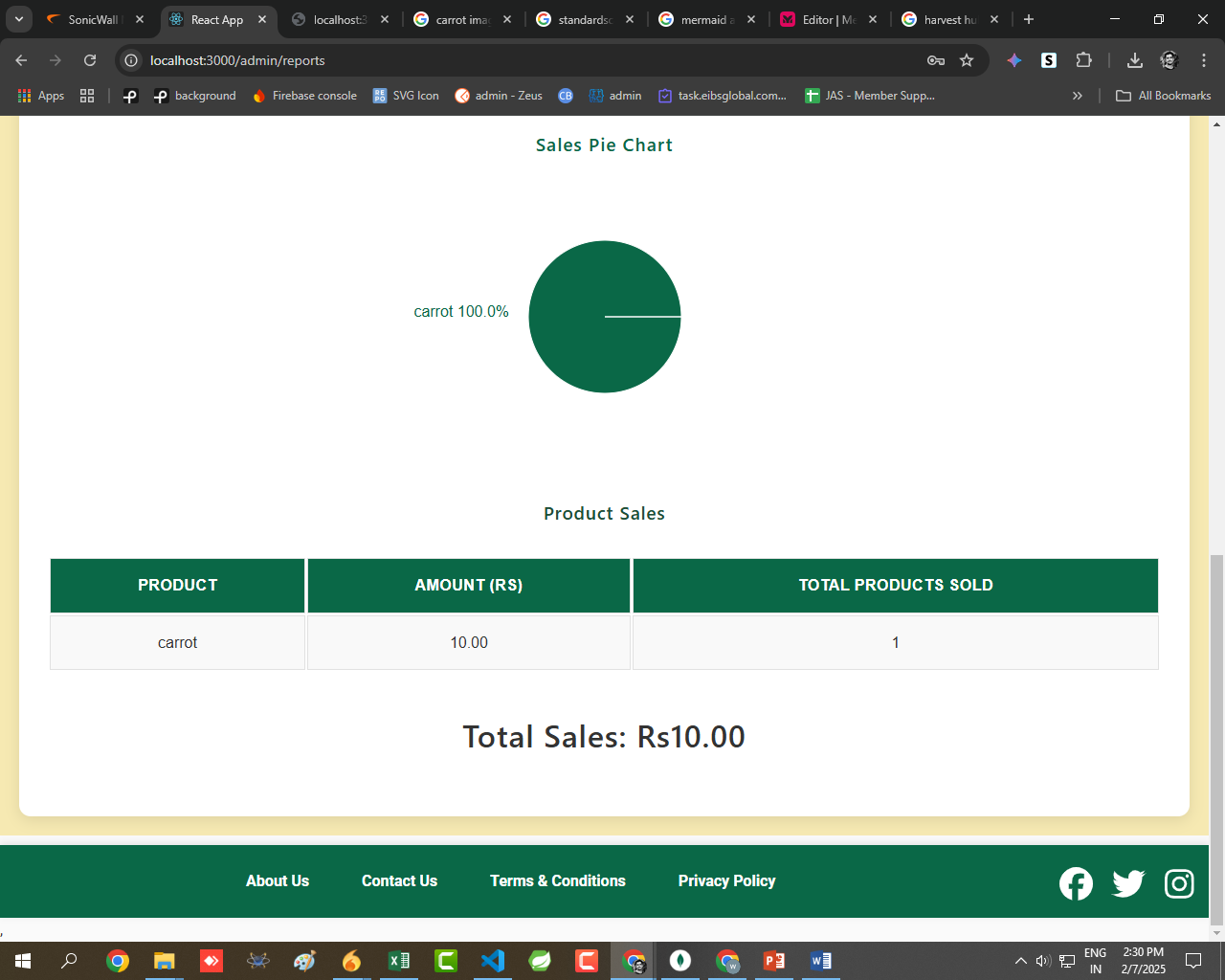


Figure 5.5.10: Total Sales

**Manage Users**

The Manage Users feature allows administrators or platform owners to oversee and control user activities, ensuring a smooth and secure experience for both buyers and farmers.

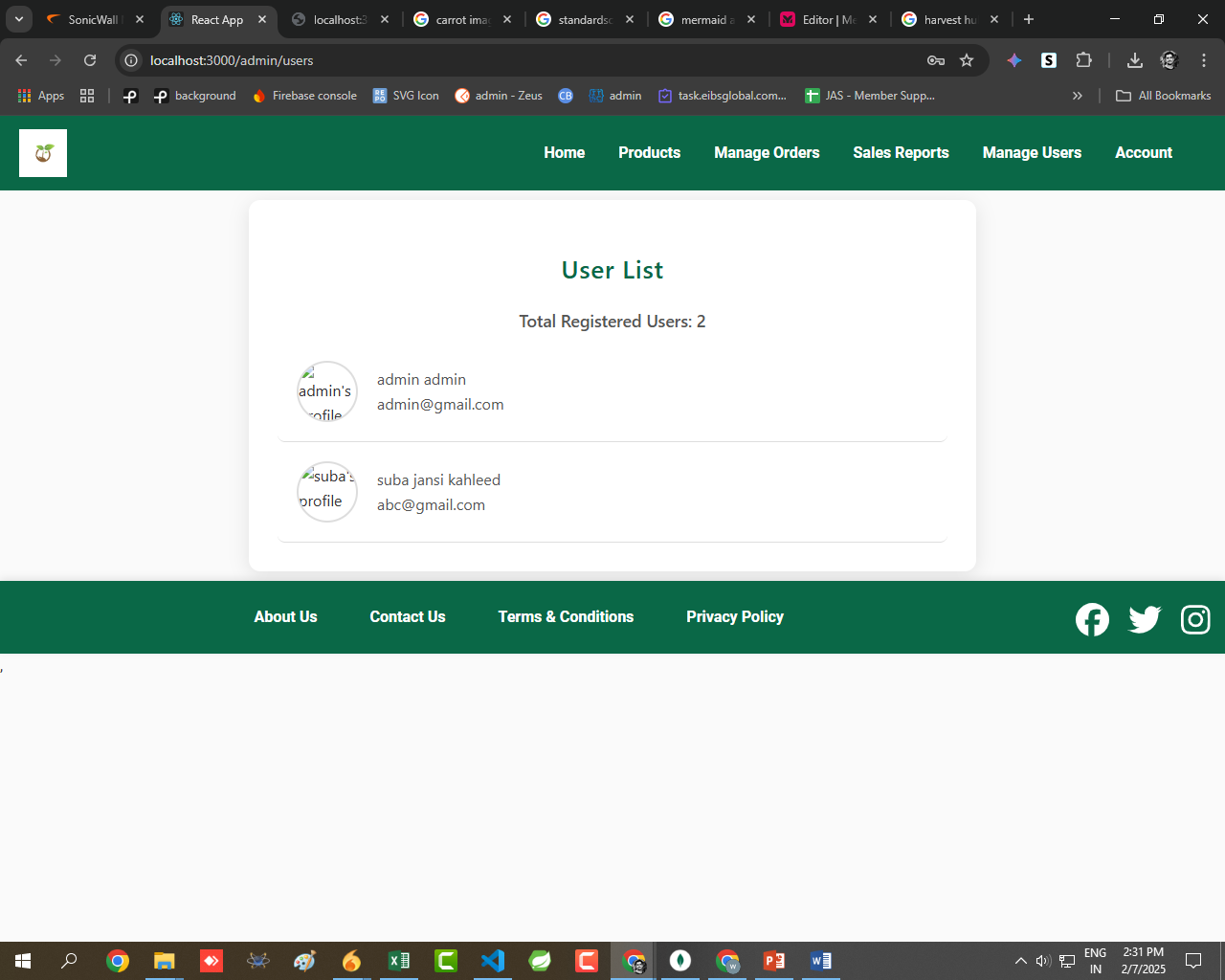


Figure 5.5.11: Manage Users

**Admin Page**

The Admin Page is a central dashboard that allows administrators to manage and monitor the entire platform, ensuring smooth operations for farmers, buyers, and transactions.

Figure 5.5.12: Admin Page

**CHAPTER 6**

**RESULT ANALYSIS**

**6.1 EXISTING SYSTEM**

In the current agricultural marketplace, there are a variety of platforms available that facilitate online buying and selling of agricultural products. However, these systems often involve intermediaries, leading to higher costs and a lack of transparency in pricing. Some existing systems also do not allow direct communication between farmers and customers, resulting in inefficiencies and missed opportunities for price negotiation. Furthermore, the lack of robust features for inventory management, product verification, and delivery tracking creates a fragmented user experience.

**Key Features of the Existing System:**

* **Third-Party Platforms**: Many farmers use third-party platforms to list their products for sale, where buyers can browse available products. However, these platforms often charge high fees or commissions, leaving farmers with a small share of the profit.
* **Limited Communication**: Direct communication between farmers and customers is often not available. Buyers and farmers might have to rely on external tools like phone calls or emails to negotiate prices and discuss product details.
* **Lack of Control for Farmers**: Farmers have limited control over their product listings. They often cannot update, edit, or remove listings once they are posted.
* **Non-Transparent Pricing**: In many cases, farmers cannot negotiate prices directly with buyers, leading to price disparities and frustrations for both parties.
  + 1. **Disadvantages of Existing System**

1. **Intermediaries and High Fees**: Third-party platforms charge significant fees and commissions, making it less profitable for farmers. This also reduces the transparency in pricing and causes farmers to lose control over their revenue.

**2.Limited Communication Channels**: Without direct communication tools built into the platform, buyers and farmers are forced to rely on external methods like phone calls or messaging apps, which complicates transactions and introduces the risk of missed messages or miscommunication.

**3.Lack of Inventory and Product Management**: Farmers often have no tools to manage their inventory, update product details, or track product sales effectively. This results in a disorganized process, especially when dealing with bulk orders.

**4.Inability to Negotiate Prices**: Most platforms do not support price negotiation between buyers and sellers, leading to a rigid pricing structure that might not benefit both parties.

**6.2 PROPOSED SYSTEM**

The **Direct Farmer-Customer Access** platform is designed to solve the above-mentioned issues and create a direct, transparent, and secure environment for agricultural transactions. The platform will connect farmers directly with customers and retailers, enabling them to negotiate prices, manage their listings, and handle secure transactions—all within the same system.

The platform will focus on providing a seamless and secure experience with user-friendly features and enhanced control for both farmers and buyers. It will also support payment integration, delivery tracking, and an admin panel for monitoring and managing the platform.

**Key Features of the Proposed System:**

1. **Direct Communication Channels**: The system will include real-time messaging or calling features, allowing farmers, customers, and retailers to directly communicate and negotiate terms like price, delivery, and quality, reducing the need for external communication tools.
2. **Price Negotiation**: Both buyers and sellers will be able to negotiate prices in real-time. Buyers (whether individuals or retailers) can make offers below the listed price, which farmers can either accept, reject, or counter. This will ensure fair pricing for both parties.
3. **Product Listing Management**: Farmers will have full control over their product listings, allowing them to add new products, update prices, and edit product descriptions. Farmers can also delete products that are no longer available.
4. **Secure Payment System**: The platform will integrate a secure payment gateway that supports multiple payment methods (credit/debit cards, digital wallets, UPI). Additionally, an escrow payment system will be available, where the buyer’s payment is held until the product is confirmed as received by the buyer.
5. **Inventory Management**: Farmers can manage their inventory through the platform, keeping track of stock levels, pricing, and order history. This will help them stay organized and prevent stock-outs or overstocking issues.
6. **Delivery and Logistics Integration**: After payment confirmation, the platform will facilitate delivery either by the farmer or through third-party logistics services. Both the farmer and the buyer will be able to track the product’s delivery status within the app.
7. **Rating and Feedback System**: After each transaction, buyers and sellers can rate each other. This will help build trust and accountability, as users can assess the reliability of their counterparts before engaging in a transaction.
8. **Admin Panel**: The platform will feature an admin panel where administrators can verify products, monitor transactions, and manage users. Admins will also be able to manage disputes and ensure compliance with platform guidelines.
9. **User Profiles and Dashboards**: Both farmers and buyers will have their own profiles and dashboards to view transaction histories, product listings, and communication logs, providing a streamlined experience for users to manage their activities.

**6.2.1 Advantages of the Proposed System:**

1. **Empowered Farmers**: Farmers will have complete control over their product listings, enabling them to manage their products and prices directly. This removes the reliance on third-party platforms and increases their profit margins.
2. **Improved Transparency**: The ability for farmers and buyers to negotiate prices directly will increase transparency in transactions. Both parties will have a clearer understanding of pricing and delivery terms, promoting fairness and satisfaction.
3. **Secure Payments and Fraud Prevention**: The integrated secure payment system with features like escrow payments will ensure that both farmers and buyers are protected from fraudulent activity. Funds will only be released when the buyer confirms receipt of the product.
4. **Efficient Communication**: Direct messaging and calling features will eliminate the need for third-party communication apps, reducing miscommunication and improving the efficiency of the transaction process.
5. **Scalability and Flexibility**: The platform is designed to scale and handle a growing number of users and products. Farmers can easily expand their listings, and buyers can access a wide range of agricultural products in one place.
6. **Enhanced User Experience**: With features like inventory management, product updates, and a streamlined checkout process, the platform will offer a smooth and efficient experience for both farmers and buyers, making agricultural transactions easy and hassle-free.
7. **Comprehensive Monitoring and Admin Control**: Admins will have full oversight of the platform, ensuring proper functioning, verifying products, and preventing fraudulent activities. The system will also allow admins to manage disputes, ensuring that both parties have a fair and secure experience.

The proposed system addresses the key limitations of existing systems by focusing on user control, secure transactions, and efficient communication. By doing so, it will create a more streamlined and transparent platform for farmers, customers, and retailers in the agricultural marketplace.

**CHAPTER 7**

**CONCLUSION**

**7.1 CONCLUSION**

* In conclusion, the **Farmer-Customer Access** mobile application is a comprehensive solution designed to bridge the gap between farmers, customers, and retailers by providing a seamless, efficient, and secure platform for agricultural product transactions.
* This app not only allows farmers to manage their inventory with ease but also empowers them to negotiate fair prices, interact directly with customers, and take control of their sales processes.
* The platform is built to promote transparency and foster trust by offering a clear and straightforward channel of communication between all parties involved.
* For customers, whether individual buyers or retailers, the application provides an intuitive interface for browsing a wide range of agricultural products, negotiating prices in real-time, and completing transactions securely.
* With features that allow customers to track deliveries, provide feedback, and access transaction histories, the app ensures a smooth and reliable shopping experience.
* Furthermore, the integration of advanced security measures, including secure payment gateways, two-factor authentication, and encrypted data, ensures that all transactions are safeguarded from unauthorized access.
* The inclusion of efficient communication tools, such as in-app messaging and push notifications, promotes ongoing engagement and provides users with real-time updates on their transactions.

**7.2 SCOPE OF PROJECT**

* **User Management**:
  + Enabling farmers, customers, and retailers to create accounts, authenticate securely, and manage profiles to ensure a personalized experience for each user type.
* **Product Listing and Management**:
  + Providing farmers with the ability to upload, manage, and update product listings, enabling them to set initial prices and negotiate with customers.
* **Price Negotiation**:
  + Allowing buyers (customers/retailers) to negotiate prices with farmers through real-time communication, helping both parties arrive at a mutually agreeable price point.
* **Transaction Handling**:
  + Facilitating secure payments through multiple payment methods and ensuring both buyer and seller have access to a transaction history, enhancing trust in the system.
* **Delivery Management**:
  + Offering delivery tracking features so buyers and sellers can follow the status of their orders, improving the overall customer experience.
* **Feedback and Ratings**:
  + Providing a rating and review system for both buyers and sellers to maintain transparency and trust in the marketplace.

**7.3 FUTURE SCOPE**

* **Mobile Application Development**:
  + Expanding the platform to include dedicated mobile applications for iOS and Android, enabling farmers, customers, and retailers to manage their transactions, communicate, and track orders conveniently on the go.
* **Advanced Search and Filtering**:
  + Introducing enhanced search functionalities that allow users to filter products by categories, price range, location, and other relevant criteria, making it easier for customers to find what they need.
* **Bulk Order Management**:
  + Developing additional features that streamline the bulk ordering process for retailers, providing more efficient ways to negotiate and purchase large quantities of agricultural products.
* **Integration with Third-Party Logistics**:
  + Exploring integration with third-party logistics and delivery services to optimize the delivery process, ensuring faster and more reliable shipping for customers.
* **Personalized User Dashboards**:
  + Creating personalized dashboards for farmers, customers, and retailers that allow them to track their activities, manage products, and view transaction histories, improving the user experience.
* **Multi-Language Support**:
  + Expanding the app to support multiple languages, allowing farmers and customers from various regions and languages to use the platform more easily.
* **Sustainability Features**:
  + Introducing features to highlight eco-friendly or organic products, allowing users to easily identify and purchase sustainable products, supporting environmentally conscious farming practices.
* **Data Analytics for Farmers**:
  + Implementing data analytics tools to provide farmers with insights into market trends, customer preferences, and inventory management, helping them make informed decisions to optimize sales and production.

**References**

1. C. B. S. Maior, M. das C. Moura, and I. D. Lins, "Particle swarm-optimized support vector machines and preprocessing techniques for remaining useful life estimation of bearings," *Eksploatacja i Niezawodnosc - Maintenance and Reliability*, vol. 21, no. 4, pp. 610–619, Sep. 2019, doi: 10.17531/ein.2019.4.10.
2. C. B. S. Maior et al., "Seroprevalence of SARS-CoV-2 on health professionals via Bayesian estimation: a Brazilian case study before and after vaccines," *Acta Trop*, vol. 233, p. 106551, Sep. 2022, doi: 10.1016/j.actatropica.2022.106551.
3. C. B. S. Maior et al., "Seroprevalence of SARS-CoV-2 on health professionals via Bayesian estimation: a Brazilian case study before and after vaccines," *Acta Trop*, vol. 233, p. 106551, Sep. 2022, doi: 10.1016/j.actatropica.2022.106551.
4. C. B. S. Maior, L. M. M. Araújo, I. D. Lins, M. D. C. Moura, and E. L. Droguett, "Prognostics and Health Management of Rotating Machinery via Quantum Machine Learning," *IEEE Access*, vol. 11, pp. 25132–25151, 2023, doi: 10.1109/ACCESS.2023.3255417.
5. C. Okoh, R. Roy, J. Mehnen, and L. Redding, "Overview of Remaining Useful Life Prediction Techniques in Throughlife Engineering Services," *Procedia CIRP*, vol. 16, pp. 158–163, 2014, doi: 10.1016/j.procir.2014.02.006.
6. F. C. Zegarra, J. Vargas-Machuca, and A. M. Coronado, "Tool wear and remaining useful life (RUL) prediction based on reduced feature set and Bayesian hyperparameter optimization," *Production Engineering*, Oct. 2021, doi: 10.1007/s11740-021-01086-8.
7. M. Tanwar and N. Raghavan, "Lubricating Oil Remaining Useful Life Prediction Using Multi-Output Gaussian Process Regression," *IEEE Access*, vol. 8, pp. 128897–128907, 2020, doi: 10.1109/ACCESS.2020.3008328.
8. C. B. S. Maior, M. das C. Moura, I. D. Lins, E. L. Droguett, and H. H. L. Diniz, "Remaining Useful Life Estimation by Empirical Mode Decomposition and Support Vector Machine," *IEEE Latin America Transactions*, vol. 14, no. 11, pp. 4603–4610, Nov. 2016, doi: 10.1109/TLA.2016.7795836.
9. K. Xu, M. Xie, L. C. Tang, and S. L. Ho, "Application of neural networks in forecasting engine systems reliability," *Appl Soft Comput*, vol. 2, no. 4, pp. 255–268, Feb. 2003, doi: 10.1016/S1568-4946(02)00059-5.
10. M. D. C. Moura, E. Zio, I. D. Lins, and E. Droguett, "Failure and reliability prediction by support vector machines regression of time series data," *Reliab Eng Syst Saf*, vol. 96, no. 11, pp. 1527–1534, 2011, doi: 10.1016/j.ress.2011.06.006.
11. S. L. Ho, M. Xie, and T. N. Goh, "A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction," *Comput Ind Eng*, vol. 42, no. 2–4, pp. 371–375, Apr. 2002, doi: 10.1016/S0360-8352(02)00036-0.
12. Z. Tian, L. Wong, and N. Safaei, "A neural network approach for remaining useful life prediction utilizing both failure and suspension histories," *Mech Syst Signal Process*, vol. 24, no. 5, pp. 1542–1555, Jul. 2010, doi: 10.1016/j.ymssp.2009.11.005.
13. C. Ordóñez, F. Sánchez Lasheras, J. Roca-Pardiñas, and F. J. de C. Juez, "A hybrid ARIMA–SVM model for the study of the remaining useful life of aircraft engines," *J Comput Appl Math*, vol. 346, pp. 184–191, Jan. 2019, doi: 10.1016/j.cam.2018.07.008.
14. P. J. García Nieto, E. García-Gonzalo, F. Sánchez Lasheras, and F. J. de Cos Juez, "Hybrid PSO–SVM-based method for forecasting of the remaining useful life for aircraft engines and evaluation of its reliability," *Reliab Eng Syst Saf*, vol. 138, pp. 219–231, Jun. 2015, doi: 10.1016/j.ress.2015.02.001.
15. L. Wang, D. Zhou, H. Zhang, W. Zhang, and J. Chen, "Application of Relative Entropy and Gradient Boosting Decision Tree to Fault Prognosis in Electronic Circuits," *Symmetry (Basel)*, vol. 10, no. 10, p. 495, Oct. 2018, doi: 10.3390/sym10100495.
16. T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, "Machine learning in manufacturing: advantages, challenges, and applications," *Prod Manuf Res*, vol. 4, no. 1, pp. 23–45, Jan. 2016, doi: 10.1080/21693277.2016.1192517.
17. N. E. Huang et al., "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1971, pp. 903–995, Mar. 1998, doi: 10.1098/rspa.1998.0193.
18. B. Sales da Cunha, M. das Chagas Moura, C. Souto Maior, A. Cláudia Negreiros, and I. Didier Lins, "A comparison between computer vision- and deep learning-based models for automated concrete crack detection," *Proc Inst Mech Eng O J Risk Reliab*, p. 1748006X2211409, Dec. 2022, doi: 10.1177/1748006X221140966.
19. F.-J. Yang, "An Extended Idea about Decision Trees," in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, Dec. 2019, pp. 349–354. doi: 10.1109/CSCI49370.2019.00068.
20. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer New York, 2009, doi: 10.1007/978-0-387-84858-7.
21. F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011, doi: 10.1007/s13398-014-0173-7.2.
22. S. Zhang, D. Cheng, Z. Deng, M. Zong, and X. Deng, "A novel k NN algorithm with data-driven k parameter computation," *Pattern Recognit Lett*, vol. 109, pp. 44–54, Jul. 2018, doi: 10.1016/j.patrec.2017.09.036.
23. R. Rojas, "The Backpropagation Algorithm," in *Neural Networks*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 149–182. doi: 10.1007/978-3-642-61068-4\_7.
24. C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006.
25. A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Stat Comput*, vol. 14, no. 3, pp. 199–222, Aug. 2004, doi: 10.1023/B:STCO.0000035301.49549.88.
26. V. Kecman, "Support Vector Machines – An Introduction," in *Support Vector Machines: Theory and Applications*, 2005, pp. 1–47. doi: 10.1007/10984697\_1.