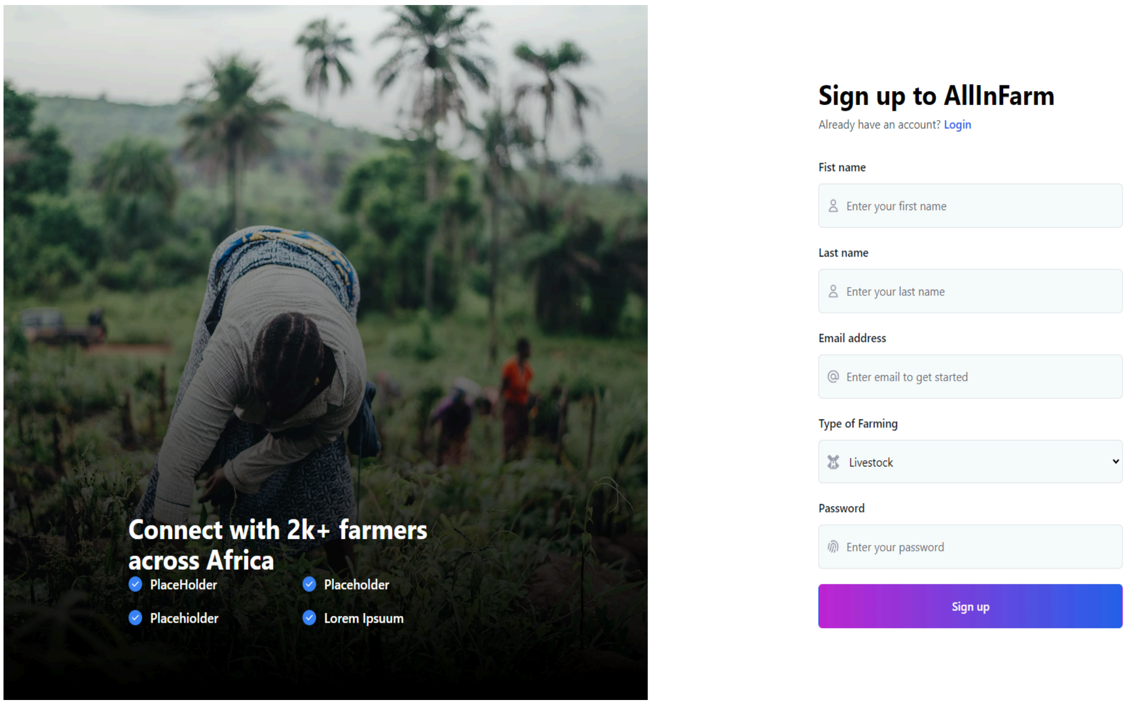
1. ALL IN FARM WEBAPP.

BRIAN KYALO
https://github.com/kyalo3



Sign up to AllInFarm
Already have an account? Login

Fist name
Enter your first name

Last name
Enter your last name

Email address
Enter email to get started

Type of Farming
Livestock

Password
Enter your password

Sign up

Farmers, unite! AllInFarm connects you: share knowledge, boost crops, sell direct. AI assistant, thriving marketplace, live events – the future's here. Join the movement!

2. Team Roles

a. Project manager **-** Coordinates overall project tasks, schedules, and communication.
b. Frontend Developer - Focuses on creating user-friendly interfaces using Next.js and React.
c. Backend Developer - Manages the Firebase Integration, User authentication and the database.

3. Technologies

a. Front-end:
HTML/CSS/JavaScript: These are fundamental technologies for building the structure, style, and interactivity of your web app.
React.js (JavaScript frameworks): modern framework to build dynamic and responsive user interfaces.

b. Back-end:
Node.js with Express: A JavaScript runtime like Node.js paired with the Express framework can be a lightweight and efficient choice for the back-end.

c. Database:
mySQL: For a simple farm web app, mySQL db could be a good fit. It's flexible and can handle different types of data.

d. Server:
Nginx or Apache: Use a web server like Nginx or Apache to serve static files and act as a reverse proxy for your Node.js application

e. Authentication:
JSON Web Tokens (JWT): Implement JWT for secure user authentication.

f. Version Control:
Git: Use Git for version control to track changes in your code

g. Deployment:
Docker: Containerization can help with deploying your application consistently across different environments.

h. API: RESTapi

Trade-offs:
a. Vue.js (Frontend Framework) – Option 1
Tailwind was considered for simplicity, but Next.js chosen for better server-side rendering and project structure.

b. MongoDB (Database) – Option 2

MySQL is known for its performance and efficiency. It uses various optimization techniques , indexing mechanisms, and caching strategies to deliver fast query execution.

4. Challenge Statement.

Problem Statement.

The spark for AllInFarm came from witnessing the challenges faced by African farmers firsthand. Scattered resources, limited communication, and lack of market access were stifling their potential. We saw a critical need for a platform that could:

a. Bridge the knowledge gap - Connect farmers to share best practices, troubleshoot problems,and access essential information.

b. Empower through community – Build a vibrant forum for mutual support,collaboration, and fostering a sense of belonging.

c. Unlock economic opportunities – Provide a marketplace to sell produce, find buyers and suppliers, and break free from exploitative middlemen.

- Target Users : Farmers seeking a comprehensive tool for their income.
- Locale: The app is relevant globally and not dependent on a specific locale.

5. Risks.

- Technical Risks: Possible challenges in REST API integration; safeguards include thorough testing and alternative services if needed.
- Non-technical Risks: User adoption challenges; strategies involve user education and a seamless onboarding process.

6. Infrastructure.

I managed to create the repository and work on the project locally.

- Deployment Strategy: Deploy the Next.js app on Netlify for continuous deployment.
- Data Population: Initial data populated through user inputs; persistent storage.
- Testing: - Automated testing using Jest for unit testing and Cypress for end-to-end testing.
- Tools:
  a. Version Control System – Github for collaboration and hosting of my git repos.
  b. Monitoring and logging – Datadog for monitoring and analytics for infrastructures.

7. Existing Solutions.

Crop managements systems:

a. Track and manage information related to different crops, including planting dates, harvest dates, and crop rotation schedules.

b. Provide insights into crop yield, growth stages, and health.

c. Inventory Management:Keep track of farm equipment, supplies, and resources.

d. Manage inventory levels, reorder points, and supplier information.

MVP SECTION.

3. APIs

1. **User Management:**

/api/users
GET: Retrieve a list of users (accessible only by the superuser).
POST: Create a new user (accessible only by the superuser).

2. **Client Management:**

/api/clients
GET: Retrieve a list of clients.
POST: Create a new client (accessible only by the superuser).

3. **Farm Information:**

/api/farm
GET: Retrieve information about the farm.
PUT: Update farm details (accessible only by the superuser).

4. **User Profile:**

/api/user/profile
GET: Retrieve the profile information of the logged-in user.
PUT: Update the user's profile.

5. **Farm Products:**

/api/products
GET: Retrieve a list of farm products.
POST: Add a new product (accessible only by the superuser).

6. **Orders:**

/api/orders
GET: Retrieve a list of orders for the logged-in user.
POST: Place a new order.

7. **Superuser Dashboard:**

/api/superuser/dashboard
GET: Retrieve analytics and key information for the superuser.

8. **Weather Information:**

/api/weather
GET: Retrieve current weather information for the farm location.

9. **Client Feedback:**

/api/feedback
POST: Allow clients to submit feedback on the farm products or services.

10. **Authentication and Authorization:**

/api/auth/login
POST: Log in and obtain an authentication token.
/api/auth/logout
POST: Log out and invalidate the authentication token.
/api/auth/register
POST: Register a new user.

<u>For Superuser:</u>

1. Create a new user/ client/.
2. Retrieve a list of all users/ client/ farm-related data.
3. Update user/ client/ darm-data information.
4. Delete a user/ client/ farm-related data.

<u>For Users:</u>

1. View farm information.
2. Task management - Create tasks, mark tasks, view lists of pending tasks.
3. Weather Integration - a weather API to provide weather forecasts.

<u>For Clients:</u>

1. Order products - place order, view order history.
2. Request services - view service request history.
3. Feedback - submit feedback on products and services.

<u>General:</u>

1. Authentication and Authorization:

   User login.

   Token-based authentication.

   Authorization checks to ensure users have the right access levels.

2. Notifications:

   Send notifications (email, SMS) for important events (e.g., completed tasks, order status).

3. Reports and Analytics:
   Generate reports on farm productivity, sales, etc.
   Analytics to provide insights into farm operations.

4. Integration with External APIs:
   Integrate with external APIs for features like market prices, soil quality analysis, etc.

4. Data Modelling.

Entities:

1. User:
   - user_id (PK)
   - username
   - password_hash
   - email
   - role (Superuser, Client, etc.)

2. Client:
   - client_id (PK)
   - user_id (FK)
   - name
   - contact_number
   - address

3. Product:
   - product_id (PK)
   - name
   - description
   - price
4. Order:
   - order_id (PK)
   - user_id (FK)
   - order_date
   - total_amount

5. OrderItem:
    - order_item_id (PK)
    - order_id (FK)
    - product_id (FK)
    - quantity
    - subtotal

6. Feedback:
    - feedback_id (PK)
    - user_id (FK)
    - feedback_text
    - timestamp

7. Farm:
    - farm_id (PK)
    - name
    - location
    - description

8. Weather:
    - weather_id (PK)
    - farm_id (FK)
    - temperature
    - humidity
    - wind_speed
    - timestamp

Relations:

1. One User can have One Client (One-to-One).
2. One User can place Many Orders (One-to-Many).
3. One Order can have Many OrderItems (One-to-Many).
4. One User can provide Many Feedback (One-to-Many).
5. One Farm can have Many Weather entries (One-to-Many).