IT Group Project
Phase 4: System Design

Group 3

Masego Khola

Vuyokazi Mooki

Kelebogile Maema

Mellomey Dzaklidzie

Lebogang Mamabolo

Tshepang Ngulube

Relesego Shabangu

Clerence Ngoepe

Bongane Maluleka

Metchis Mathome

4.1 Introduction

The Farming Management System (FMS) is architected to transform agricultural operations by integrating cutting-edge digital technologies with traditional farming methods. The system design focuses on a multi-layered architecture, encompassing front-end interfaces for user interaction, robust backend services for data processing, and an integrated database for efficient data management. The front-end, developed with modern web and mobile frameworks, ensures a user-friendly experience, allowing farmers to access features such as crop planning, livestock tracking, financial management, and inventory control. The backend is built on scalable technologies, like Node.js or Python, with secure data storage managed through relational databases such as MySQL.

Additionally, FMS incorporates IoT integration using MQTT protocols with sensors connected to Arduino or Raspberry Pi for real-time monitoring of environmental conditions, crop health, and livestock well-being. Advanced analytics powered by Python libraries such as Pandas and Matplotlib enable farmers to make data-driven decisions, leveraging insights on productivity, financial viability, and market trends. The system also prioritizes data security and uptime, implementing strong authentication measures, data encryption, and cloud-based hosting solutions through platforms like AWS, Azure, or GCP. By offering offline capabilities and automated synchronization when connectivity is restored, the FMS provides a resilient, accessible platform to streamline day-to-day farm management while promoting sustainable growth.

4.2 System Design

1. Functional Requirements:

User Management:

- Admins (e.g., farm owners, managers)
- Workers (e.g., labourers, technicians)

Farm Data Management:

- Manage plots, crops, and livestock
- Track seasons and planting cycles

Resource Management:

- Monitor inventory (fertilizers, seeds, feed, tools)
- Manage equipment and machinery

Task Management:

- Assign daily tasks to workers (e.g., planting, fertilizing, harvesting)
- Track task completion and monitor productivity

Weather Integration:

- Suggest optimal planting/ harvesting times based on weather conditions

Financial Management:

- Track farm-related expenses and income

Reporting & Analytics:

- Crop yield predictions
- Soil health and productivity
- Livestock health and productivity

2. Non-Functional Requirements:

- Scalability: Ability to handle a growing number of farms, users, and farm-related data as operations expand.
- Reliability: High system uptime and resilience against failure.
- Security: Ensure secure access and data protection, especially for sensitive financial data.
- Performance: Low latency for real-time data (weather updates, task completion, etc.).
- Usability: Easy-to-use interface for farmers and workers with varying levels of tech literacy.

- Interoperability: Ability to integrate with external systems (e.g., weather APIs, payment gateways)

3. High-Level Architecture:
- Frontend: HTML, CSS, JavaScript, React.js, PHP, Figma.
- Backend: Node.js/Express.js or Python, MySQL, PHP.
- Mobile Platform: React Native
- IoT Integration: MQTT, Arduino/Raspberry Pi for sensors
- Analytics: Python, Pandas, Matplotlib
- Hosting: AWS/Azure/GCP

Database:
- Relational Database (e.g., PostgreSQL, MySQL) for structured farm data (plots, crops, livestock).
- NoSQL Database (e.g., MongoDB) for large-scale storage (weather data, sensor data).
- Business Logic Layer: Manage crop scheduling, task assignments, and financial calculations.

Security Layer:
- Role-based access control (RBAC)

for users.
- Encryption for sensitive data (e.g., financials, user info).
- Two-factor authentication (2FA) for farm managers and admins.
- Monitoring & Logging:
- System monitoring (e.g., uptime, resource usage) via tools like Prometheus.
- Logging user actions and system events using ELK Stack or Cloud
- Logging services.

4. Detailed Module Breakdown:

Farm Management:
- Plot and Crop Module: Allows users to define farm plots, crop types, and planting schedules.
- Livestock Module: Tracks individual animals, feeding schedules, health, and productivity.
- Inventory Management:

- Tracks the use of farm supplies, generates low-stock alerts, and integrates with vendors for purchasing.
- Task Assignment Module:
- Allows farm managers to assign tasks to workers, set deadlines, and track completion status.

Financial Management:

- Tracks expenses for seeds, fertilizers, labour, and machinery, as well as income from sales.
- Reports and Analytics:
- Provides insights on crop yield, cost per acre, livestock performance, and profitability forecasts.

5. Future Enhancements:

- AI/ML for Predictive Analytics: Crop yield prediction, soil health analysis using AI models.
- Drone & Satellite Integration: For aerial farm monitoring and large-scale data collection.
- Blockchain: For transparent, traceable supply chain management in agriculture

4.3 Architectural Design

A. Software Architectural Design:

Components:

1. **Frontend:**

Technologies:

- The frontend will be built using **HTML, CSS, JavaScript, Google APIs** which offers a component-based architecture that supports scalability and reusability of UI components.
- For styling, we will use **CSS-in-JS libraries** and Bootstrap like Styled-Components or SASS.

User Experience Considerations:

- Focus on minimal clicks for core actions.
- Responsive design to support mobile and tablet users.
- User feedback loops will be used to refine the interface for non-tech-savvy users.

2. **Backend:**

- The backend will expose RESTful API endpoints for communication with the frontend and mobile clients.
- These APIs will handle all CRUD operations for entities like farms, crops, livestock, and users.
- A **Java-based microservice** could be developed later for more intensive operations like image processing for crop health monitoring.

3. **Database:**

- **Database Management System**:

- We will use **MySQL** for its ACID-compliance, robust support for relational data, and ability to handle large datasets

Third-Party Integration:

The system will integrate with third-party weather APIs to pull real-time weather data that can be useful for farm operations

B. Hardware Architectural Design for Farm Management System

Components:

1. Servers:
   - **Application Servers**:
   - **Specifications**
     - **CPU**: Quad-core processors (e.g., Intel Xeon or AMD EPYC).
     - **Memory (RAM)**: 16 GB or higher, depending on the scale.
     - **Storage**: SSDs with at least 512 GB for fast access to application data.
     - **Operating System**: Linux-based (Ubuntu or CentOS) for

       stability and performance. o
     - **Purpose**:
     - These servers will run the backend services (API endpoints, business logic) and will be responsible for processing user requests, authentication, and handling IoT device communications.
2. Database Servers:
   a. **Specifications**:

- **CPU**: 8-core processors for handling multiple database queries simultaneously.
- **Memory (RAM)**: 32 GB or more to support complex queries and large datasets.
- **Storage**:
- **Primary Storage**: SSDs (1 TB+) for fast read/write operations on active data.
- **Secondary Storage**: Traditional HDDs for archival and backup purposes.

C. Network Architectural Design for Farm Management System

Components:

1. <u>Network Topology:</u> **Remote Development Setup**:
   - ▪ Team members work remotely and connect to the cloud infrastructure, using VPNs for secure access to the backend servers.
2. <u>Internet Connectivity</u>:
   - **Requirements for Internet Access**:
     - The system requires **high-speed internet** on the farm to ensure data can be transmitted from sensors to the cloud. Each team member working remotely also needs a stable internet connection.
     - **Cloud-Based Connectivity**:
       - Cloud infrastructure is accessed via the internet by team members for development, debugging, and deployment.
3. <u>Security:</u>
   - **Firewalls**:

- Firewalls protect both the farm’s internal network and remote team access.
- **Web Application Firewalls (WAF)** and network firewalls prevent unauthorized access to the farm management system’s servers and APIs.

- **VPNs**:
- o **Virtual Private Networks (VPNs)** enable secure access for team members working remotely. VPNs ensure encrypted communication between team devices and the cloud-based system infrastructure.
- **Encryption Measures**:
  - **Data at Rest**:
    - Data stored locally (on servers or team members' devices) and in the cloud is encrypted using **AES-256** encryption.
  - **Data in Transit**:
    - Communication between IoT devices, servers, and team members working remotely is encrypted using **SSL/TLS**.
  - **Multi-factor Authentication (MFA)** is enforced for remote team members accessing the core system, ensuring secure login practices.

4.4 Physical Design

This physical design provides a comprehensive blueprint for implementing the Farming Management System, ensuring scalability, security, and efficient data management.

1. System Architecture:

The FMS will be designed as a web-based application using a three-tier architecture:
Presentation Tier

- Web browsers (Chrome, Firefox, Safari, etc.)
- Mobile devices (iOS and Android using React Native/Flutter)

Application Tier

- Web Server: Apache or Nginx
- Application Server: Node.js with Express.js or Python with Django
- API Gateway for microservices communication

Data Tier

- Primary Database: MySQL for relational data
- Cache: Redis for session management and caching
- File Storage: AWS S3 or similar for storing images and documents

Additional Components

- Authentication Server: OAuth 2.0 implementation
- Message Queue: RabbitMQ for asynchronous processing
- Monitoring and Logging: ELK Stack (Elasticsearch, Logstash, Kibana)

2. Database Schema:

The MySQL database will consist of the following main tables:

Users

- UserID (PK)
- Username
- Email

- PasswordHash
- Role (User/Admin)
- CreatedAt
- LastLogin

Crops

- CropID (PK)
- Name
- Category
- PlantingDate
- HarvestDate
- FieldID (FK)
- Status

Livestock

- LivestockID (PK)
- Type
- Breed
- BirthDate
- Status
- FieldID (FK)

Fields

- FieldID (PK)
- Name
- Area
- Location
- SoilType
- Status

Inventory

- ItemID (PK)
- Name
- Category
- Quantity
- Unit

- LastUpdated
- Status

Orders

- OrderID (PK)
- UserID (FK)
- OrderDate
- Status
- TotalAmount
- PaymentStatus

OrderItems

- OrderItemID (PK)
- OrderID (FK)
- ItemID (FK)
- Quantity
- UnitPrice

Tasks

- TaskID (PK)
- Title
- Description
- AssignedTo (FK to Users)
- DueDate
- Status

Weather

- WeatherID (PK)
- Date
- Temperature
- Humidity
- Rainfall
- WindSpeed

3. User Interface Components:

Common Components

- Navigation Sidebar
- Header with search bar and user profile
- Footer with copyright and contact information

Specific Pages/Components

3.1. Dashboard

- Overview statistics cards
- Revenue chart
- Yield prediction chart
- Top products chart
- Soil moisture chart
- Recent tasks list
- Weather widget

3.2. Crops Page

- Crop list table
- Add/Edit crop modal
- Crop details view
- Crop calendar

3.3. Livestock Page

- Livestock list table
- Add/Edit livestock modal
- Livestock details view
- Health tracking component

3.4. Inventory Page

- Inventory list table
- Add/Edit inventory item modal
- Low stock alerts
- Inventory usage trends

3.5. Orders Page

- Order list table
- Order details view
- Add/Edit order modal
- Payment processing component

3.6. Fields Page

- Field list table
- Field map view (using GIS integration)
- Add/Edit field modal
- Soil analysis component

3.7. Reports Page

- Report generator
- Data visualization components
- Export functionality (PDF, CSV)

3.8. Settings Page

- User profile management
- System configuration options
- Notification preferences

3.9. Support Page (Helpmate AI ChatBot)

- Chat interface
- Knowledge base integration
- Ticket creation system

4. Integration Points:

- Weather API: Integration for real-time weather data
- Payment Gateway: For processing order payments
- IoT Devices: For field sensors and livestock tracking
- GIS Services: For mapping and geospatial analysis
- AI/ML Services: For yield predictions and image recognition (crop health)

5. Security Measures:

- SSL/TLS encryption for all communications
- Role-based access control (RBAC)
- Input validation and sanitization

    - Regular security audits and penetration testing
    - Secure password hashing (e.g., bcrypt)
    - Two-factor authentication (2FA) for sensitive operations

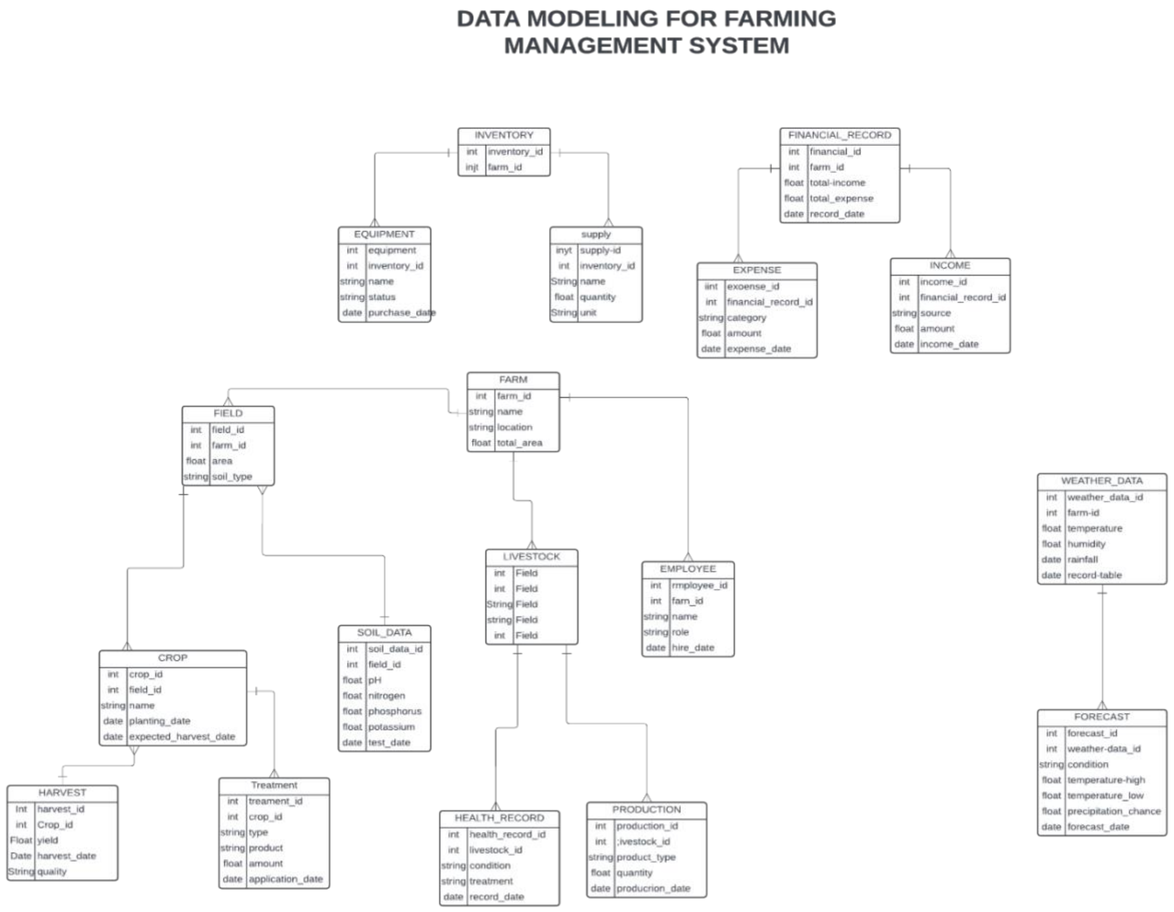
6. Scalability and Performance:

- Use of caching mechanisms (Redis) for frequently accessed data
- Database indexing and query optimization
- Content Delivery Network (CDN) for static assets
- Horizontal scaling of application servers
- Database replication and sharding for high availability

7. Backup and Disaster Recovery:

- Regular automated backups of all databases
- Off-site backup storage
- Disaster recovery plan with defined RPO (Recovery Point Objective) and RTO (Recovery Time Objective)
- Failover mechanisms for critical components

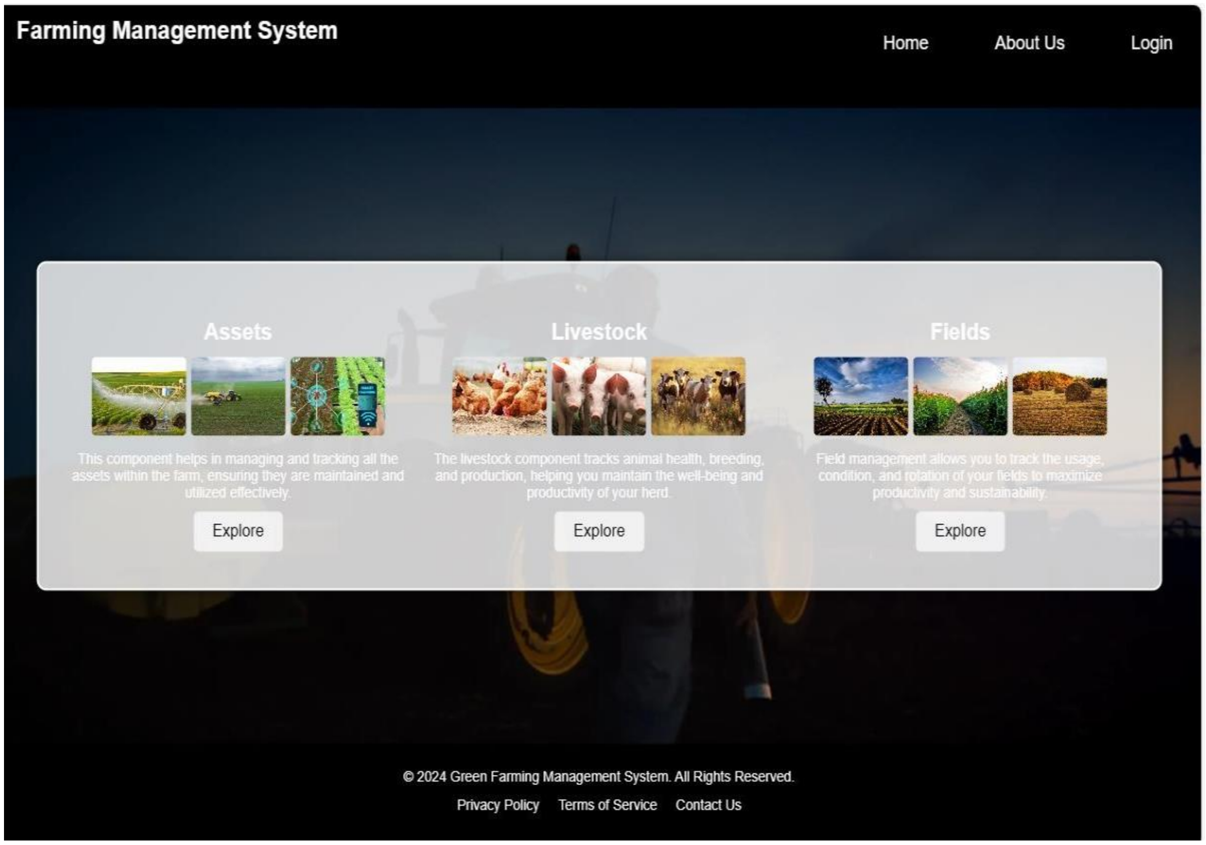
4.5 Database Design


DATA MODELING FOR FARMING MANAGEMENT SYSTEM
INVENTORY: int inventory_id; injt farm_id
FINANCIAL_RECORD: int financial_id; int farm_id; float total-income; float total_expense; date record_date
EQUIPMENT: int equipment; int inventory_id; string name; string status; date purchase_date
supply: inyt supply-id; int inventory_id; String name; float quantity; String unit
EXPENSE: iint exoense_id; int financial_record_id; string category; float amount; date expense_date
INCOME: int income_id; int financial_record_id; string source; float amount; date income_date
FARM: int farm_id; string name; string location; float total_area
FIELD: int field_id; int farm_id; float area; string soil_type
WEATHER_DATA: int weather_data_id; int farm-id; float temperature; float humidity; date rainfall; date record-table
LIVESTOCK: int Field; int Field; String Field; string Field; int Field
EMPLOYEE: int rmployee_id; int farn_id; string name; string role; date hire_date
SOIL_DATA: int soil_data_id; int field_id; float pH; float nitrogen; float phosphorus; float potassium; date test_date
CROP: int crop_id; int field_id; string name; date planting_date; date expected_harvest_date
FORECAST: int forecast_id; int weather-data_id; string condition; float temperature-high; float temperature_low; float precipitation_chance; date forecast_date
HARVEST: Int harvest_id; int Crop_id; Float yield; Date harvest_date; String quality
Treatment: int treament_id; int crop_id; string type; string product; float amount; date application_date
HEALTH_RECORD: int health_record_id; int livestock_id; string condition; string treatment; date record_date
PRODUCTION: int production_id; int ;ivestock_id; string product_type; float quantity; date producrion_date

4.6 Program Design

Pages list:

1. Home
2. About-us
3. Fields
4. Crops information
5. Livestock information
6. Log in
7. Sign-up
8. Admin –dashboard
9. User-dashboard
10. Crops
11. Orders
12. Inventory
13. Support
14. Settings
15. Privacy policy
16. Terms of service
17. Contact us
18. HOME


Farming Management System

Home About Us Login

Assets

This component helps in managing and tracking all the assets within the farm, ensuring they are maintained and utilized effectively.

Explore

Livestock

The livestock component tracks animal health, breeding, and production, helping you maintain the well-being and productivity of your herd.

Explore

Fields

Field management allows you to track the usage, condition, and rotation of your fields to maximize productivity and sustainability.

Explore

© 2024 Green Farming Management System. All Rights Reserved.

Privacy Policy Terms of Service Contact Us

19.

Explanation:

Overview of the HTML code:

- The webpage belongs to a Green Farming Management System, containing a navigation bar, background video, transparent sections highlighting different farming components (Assets, Livestock, Fields), and a footer.

Program Design with Pseudocode:

1. Initialize Webpage Structure:
   ○ Define the document as an HTML file. ○ Specify language as English. ○ Set the webpage's metadata (character encoding and viewport settings).

```
START
  Initialize HTML document with lang="en"
  Set charset to UTF-8 for proper character encoding
  Define viewport for responsive design
  Set title of the webpage
  Link external CSS for styling
END
```

Create Navigation Bar:

- Create a navigation bar with the website logo and links to "Home," "About Us," and "Login" pages.
- Use a list for navigation items

```
START
  Create a <nav> element
  Inside <nav>, add a logo with the name "Farming Management System"
  Create a list of links:
1. Home (link to home.html)
2. About Us (link to about-us.html)
3. Login (link to login.html)
END
```

Add Video Background:

- Implement a video background that plays automatically and loops with no sound.

```
css
START
  Create a <div> for video background
  Inside the <div>, add a <video> tag
    Set video to autoplay, muted, and loop
    Link video source (background.mp4)
  Provide fallback text for browsers that don’t support video
```

END

Create Transparent Containers for Components:

- Add transparent containers for different components (Assets, Livestock, Fields).
- Each container has:
  - A heading with the component's name.
  - Images related to the component.
  - A description of the component.
  - An "Explore" button linking to a detailed page for that component.

```
START
  Create a <div> for transparent containers
  Inside the <div>, create a <div> for each component:
    Component: Assets
      Display images for assets (e.g., equipment, machinery)
      Write description about managing assets
      Add button to navigate to assets.html
    Component: Livestock
      Display images for livestock (e.g., cows, sheep)
      Write description about tracking livestock health
      Add button to navigate to livestock.html
    Component: Fields
      Display images for fields (e.g., crops, soil)
      Write description about managing fields
      Add button to navigate to fields.html
END
```

Create Footer:

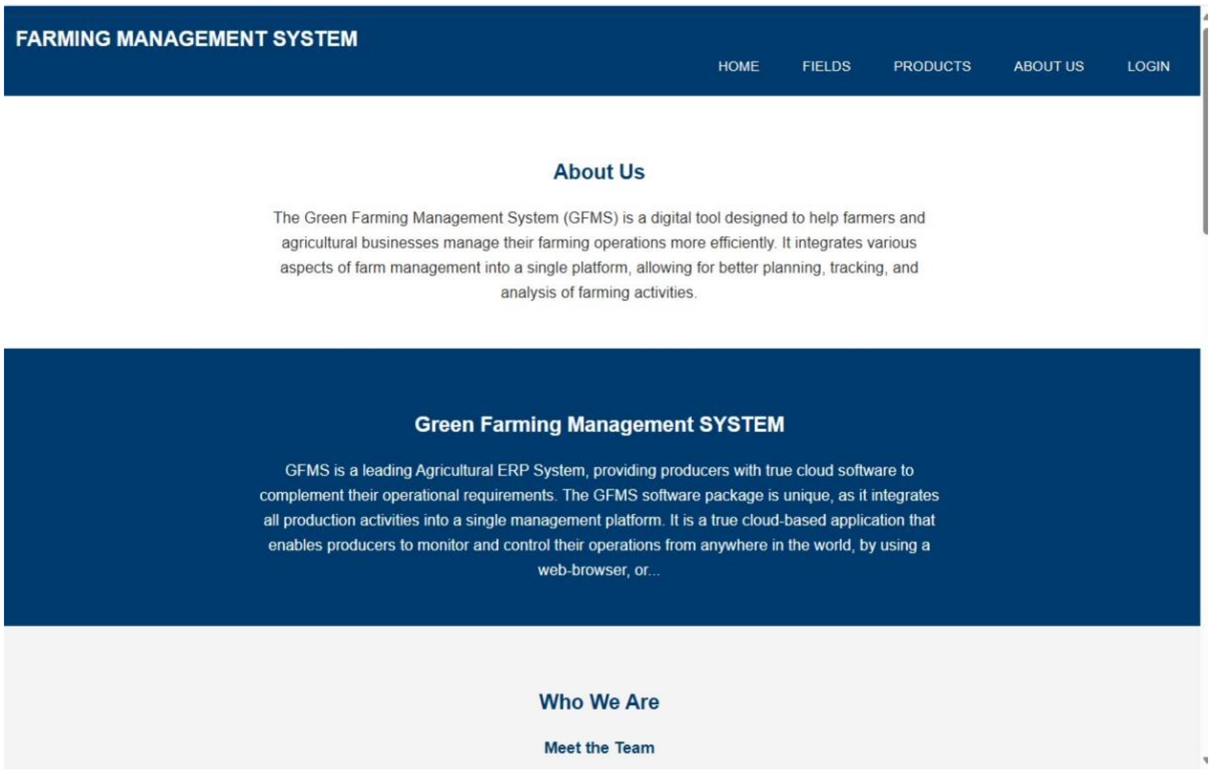
- Add a footer containing the copyright notice, privacy policy, terms of service, and contact information.

```
START
    Create a <footer> element
    Inside <footer>, add copyright information
    Create a list of footer links:
1. Privacy Policy (link to privacy.html)
2. Terms of Service (link to Ts&Cs.html)
3. Contact Us (link to contact.html)
END
```

Key Points:

- Navigation: Provides user-friendly links for easy access to different sections.
- Video Background: Adds visual appeal with an auto-playing video.
- Component Sections: Users can explore the various aspects of the farming management system, such as assets, livestock, and fields.
- Footer: Contains legal and contact information, contributing to the professionalism of the page.

<u>About-us</u>


FARMING MANAGEMENT SYSTEM

About Us

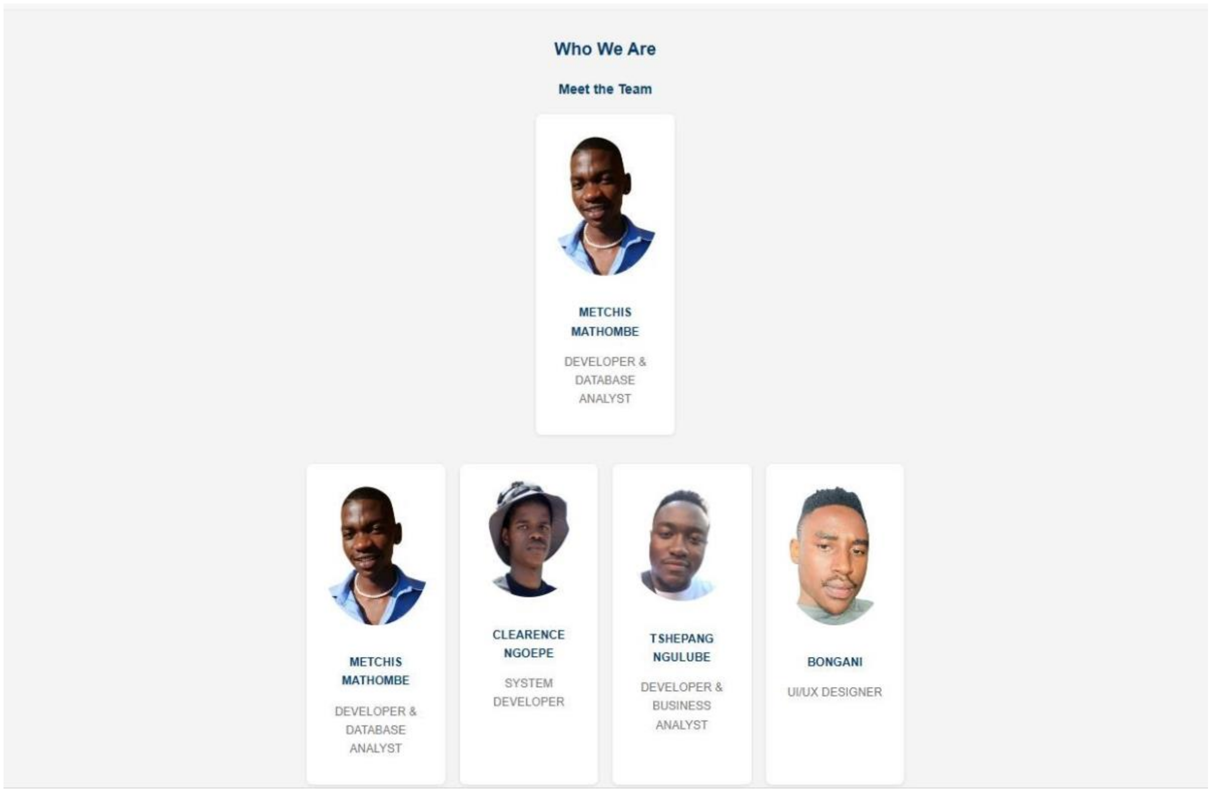
The Green Farming Management System (GFMS) is a digital tool designed to help farmers and agricultural businesses manage their farming operations more efficiently. It integrates various aspects of farm management into a single platform, allowing for better planning, tracking, and analysis of farming activities.

Green Farming Management SYSTEM

GFMS is a leading Agricultural ERP System, providing producers with true cloud software to complement their operational requirements. The GFMS software package is unique, as it integrates all production activities into a single management platform. It is a true cloud-based application that enables producers to monitor and control their operations from anywhere in the world, by using a web-browser, or...

Who We Are

Meet the Team


Who We Are

Meet the Team

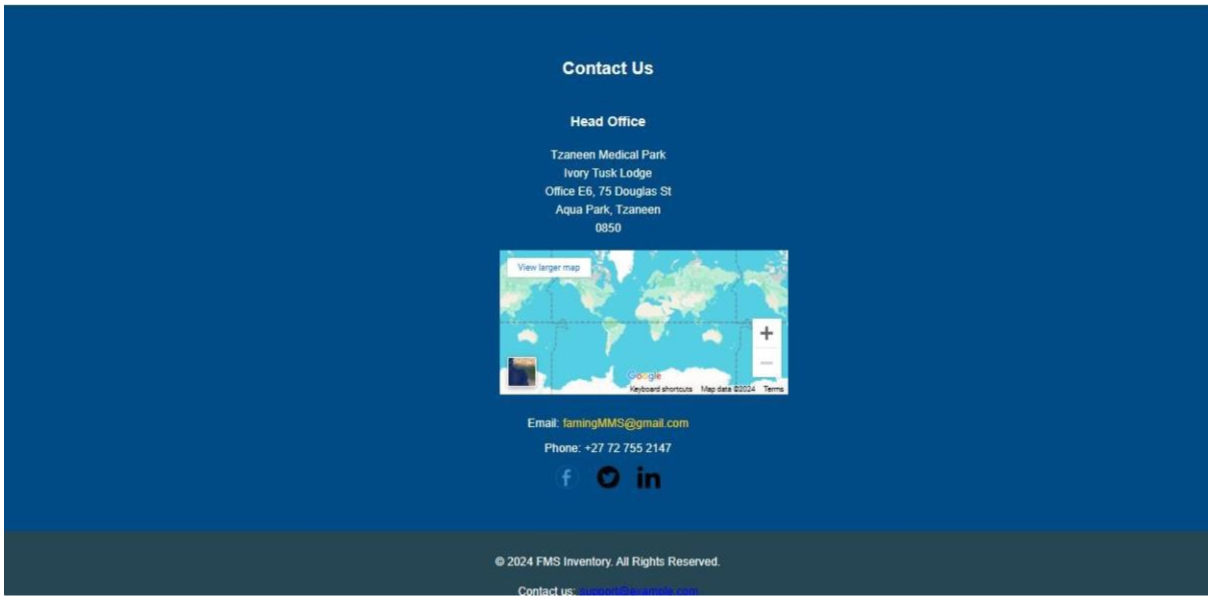
METCHIS MATHOMBE
DEVELOPER & DATABASE ANALYST

METCHIS MATHOMBE
DEVELOPER & DATABASE ANALYST

CLEARENCE NGOEPE
SYSTEM DEVELOPER

TSHEPANG NGULUBE
DEVELOPER & BUSINESS ANALYST

BONGANI
UI/UX DESIGNER


Contact Us

Head Office

Tzaneen Medical Park
Ivory Tusk Lodge
Office E6, 75 Douglas St
Aqua Park, Tzaneen
0850

Email: famingMMS@gmail.com

Phone: +27 72 755 2147

© 2024 FMS Inventory. All Rights Reserved.

Contact us: support@example.com

Explanation:

Program Design (Pseudocode)

1. Initialize Webpage Structure:
    - Define the document as an HTML file.
    - Specify language as English.
    - Set the webpage's metadata (character encoding, viewport for mobile devices).
    - Link an external stylesheet for page styling (about1.css).

Create Header and Navigation Bar:

- Add a header with a navigation bar that includes:
    - Website logo ("FARMING MANAGEMENT SYSTEM").
    - Links to "Home," "Fields," "Products" (with dropdown menu), "About Us," and "Login" pages.

Create "About Us" Section:

- Add a section describing the Green Farming Management System (GFMS).
- Provide an overview of the platform's features and benefits

Create Platform Description Section:

- Add a section that provides details about the GFMS platform.

- Explain the platform’s ERP features, cloud capabilities, and benefits for producers.

Create "Meet the Team" Section:

- Add a section introducing the team behind the Green Farming Management System.
- Display members' names, roles, and images.
- Structure the team into three rows:
    1. First row: 1 team member.
    2. Second row: 4 team members.
    3. Third row: 5 team members.

Create Contact Information Section:

- Add a section with the company's contact information, including:
    - Address (Head Office).
    - Embedded Google Map for location.
    - Email and phone contact details.
    - Social media links (Facebook, Twitter, LinkedIn)

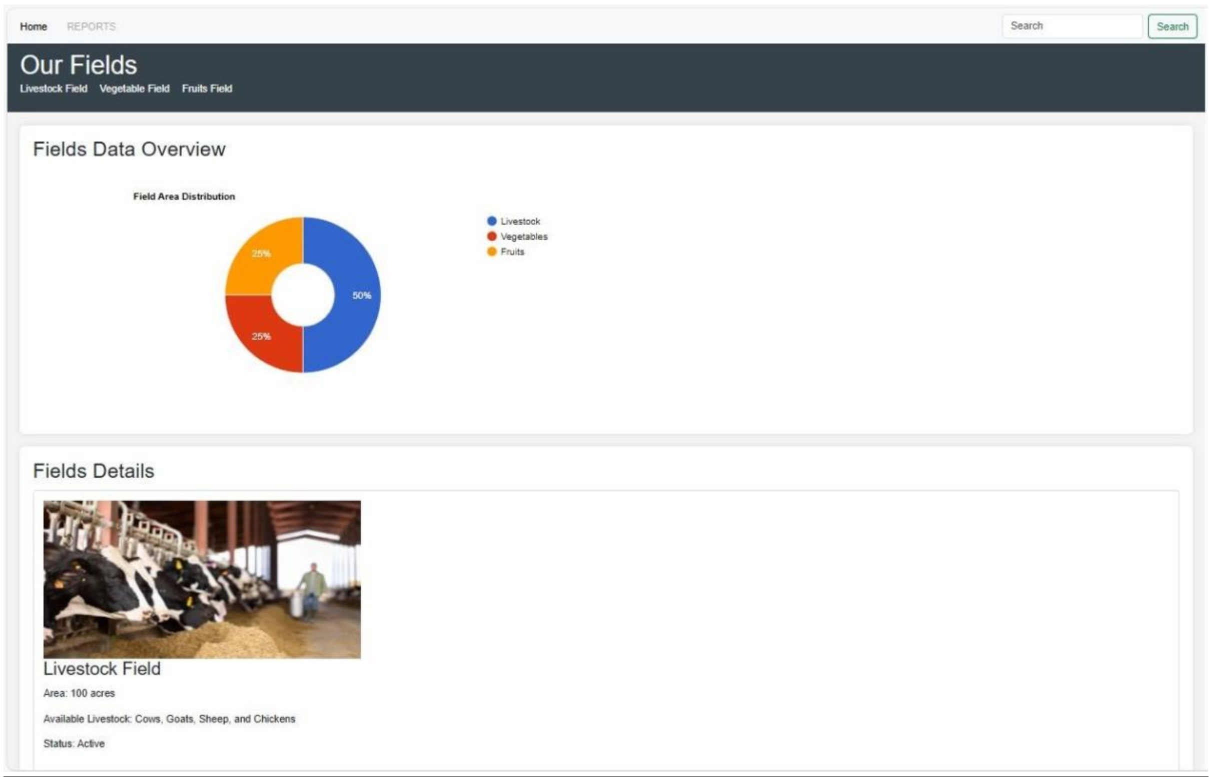
Create Footer:

- Add a footer containing:
    - Copyright notice.
    - Email contact link

```
START
  Initialize HTML structure
  Set language to "en"
  Define meta charset as "UTF-8"
  Set viewport for mobile responsiveness
  Link external CSS file (about1.css)

  Create <header> for the navigation bar
Inside <nav>:
        Add logo: "FARMING MANAGEMENT SYSTEM"
        Add links:
          Home
          Fields
          Products (with dropdown: Crops, Inventory, Livestock)
          About Us
```

```
            Login

   Create "About Us" section
      Add heading: "About Us"
      Write paragraph about the GFMS tool

   Create "Platform Description" section
      Add heading: "Green Farming Management SYSTEM"
      Write paragraph about cloud-based ERP system

   Create "Meet the Team" section
      Add heading: "Who We Are"
      Add sub-heading: "Meet the Team"

      For each team row:
         Row 1: 1 team member (image, name, role)
         Row 2: 4 team members (image, name, role)
         Row 3: 5 team members (image, name, role)

   Create "Contact Us" section
      Add heading: "Contact Us"
      Add office address and Google Map
      Add contact details (email, phone)
      Add social media links (Facebook, Twitter, LinkedIn)

   Create footer section
      Add copyright information
      Add email contact link

   Link JavaScript file (about-us.js)
END
```

Fields



Our Fields

Livestock Field Vegetable Field Fruits Field

Fields Data Overview

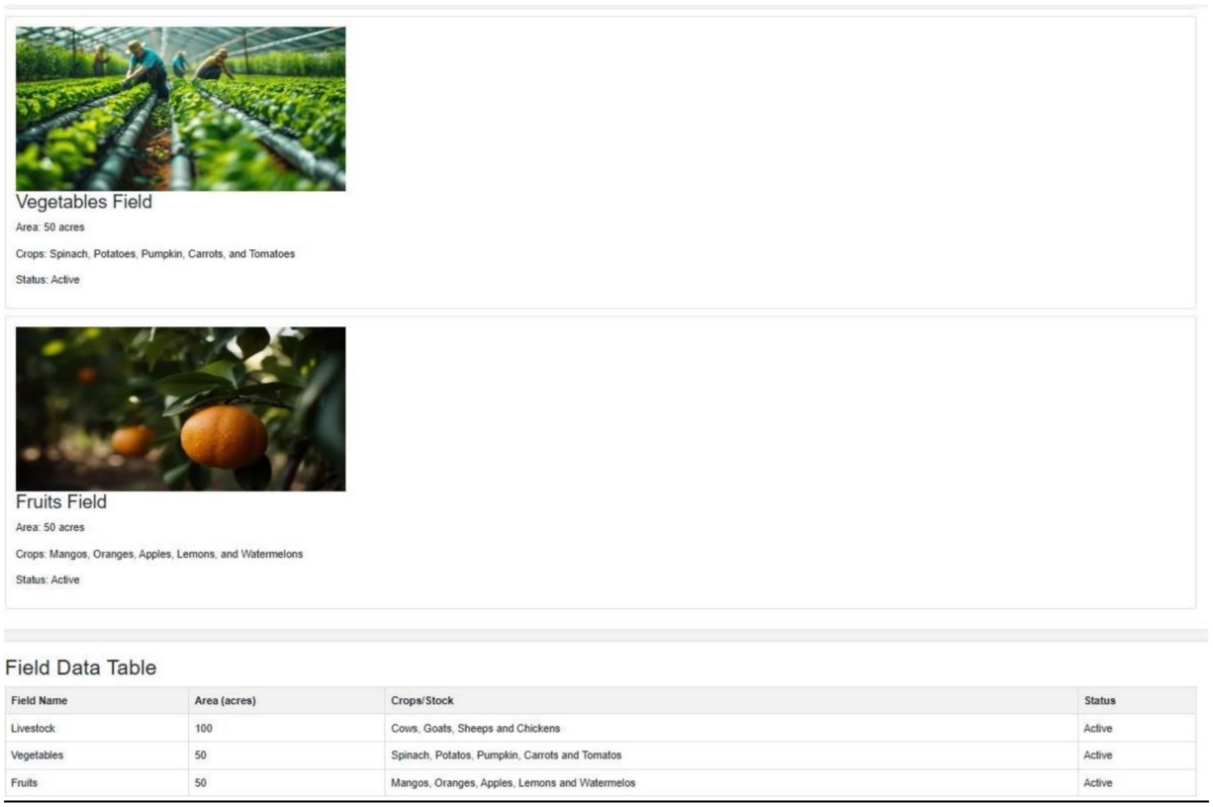
Fields Details

Livestock Field

Area: 100 acres

Available Livestock: Cows, Goats, Sheep, and Chickens

Status: Active



Vegetables Field

Area: 50 acres

Crops: Spinach, Potatoes, Pumpkin, Carrots, and Tomatoes

Status: Active

Fruits Field

Area: 50 acres

Crops: Mangos, Oranges, Apples, Lemons, and Watermelons

Status: Active

Field Data Table

Field Name	Area (acres)	Crops/Stock	Status
Livestock	100	Cows, Goats, Sheeps and Chickens	Active
Vegetables	50	Spinach, Potatos, Pumpkin, Carrots and Tomatos	Active
Fruits	50	Mangos, Oranges, Apples, Lemons and Watermelos	Active

Explanation:

Program Design Overview

This webpage for an "Agricultural Store" focuses on presenting fields used for agricultural purposes, such as livestock, vegetables, and fruits. The design integrates a chart, a detailed list of fields, and a table summarizing the key attributes of each field. It also allows for navigation to other parts of the site, such as reports, and includes a search feature.

Key Features:

- Navigation Bar: Offers quick access to the Home page, a (disabled) link to Reports, and a search form.
- Fields Section: Divides the fields into livestock, vegetables, and fruits, providing details on the area, crops/stock, and status.
- Google Chart Integration: Provides a visual overview of the field data in a pie chart.
- Field Data Table: Summarizes the fields' attributes, such as name, area, crops/stock, and status, in a tabular format.
- Footer: Contains contact information and copyright details.

1. HTML Layout Initialization

- Define the doctype, set lang as English, and include necessary meta tags for character encoding and viewport responsiveness.
- Link external CSS files for layout and Google Charts for visualization.

2. Navbar Setup

- Create a navigation bar with the following:
  - "Home" link ○ A disabled "Reports" link ○ A search form with an input box and search button.

3. Header Section

- Display a header titled "Our Fields."
- Create a sub-navigation with links to:
  - "Livestock Field" ○ "Vegetable Field" ○ "Fruits Field."

4. Main Content

- Chart Section:
  - Include a section with a header "Fields Data Overview."
    ○ Create a div to hold the Google Chart for field data.
- Fields Data Section:
  - Define the layout for each field (Livestock, Vegetables, Fruits):
    - Display the field's image.
    - Show the area in acres, the crops or livestock, and the field's status.
- Table Section:
  - Display field data in a table with columns: Field Name, Area (acres), Crops/Stock, and Status.

5. Footer

- Provide contact details, a copyright message, and a link to email support.

6. JavaScript File

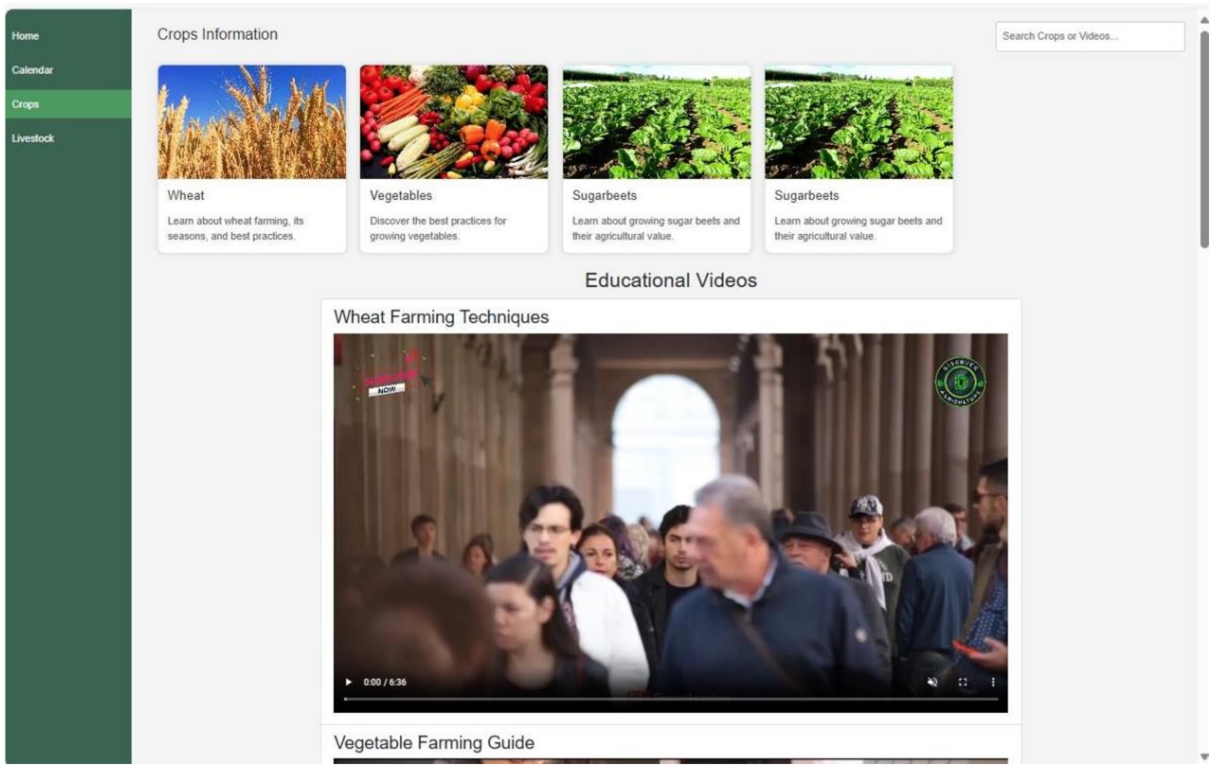
- In scripts.js, load the Google Charts library and implement a function to display a pie chart using field data.

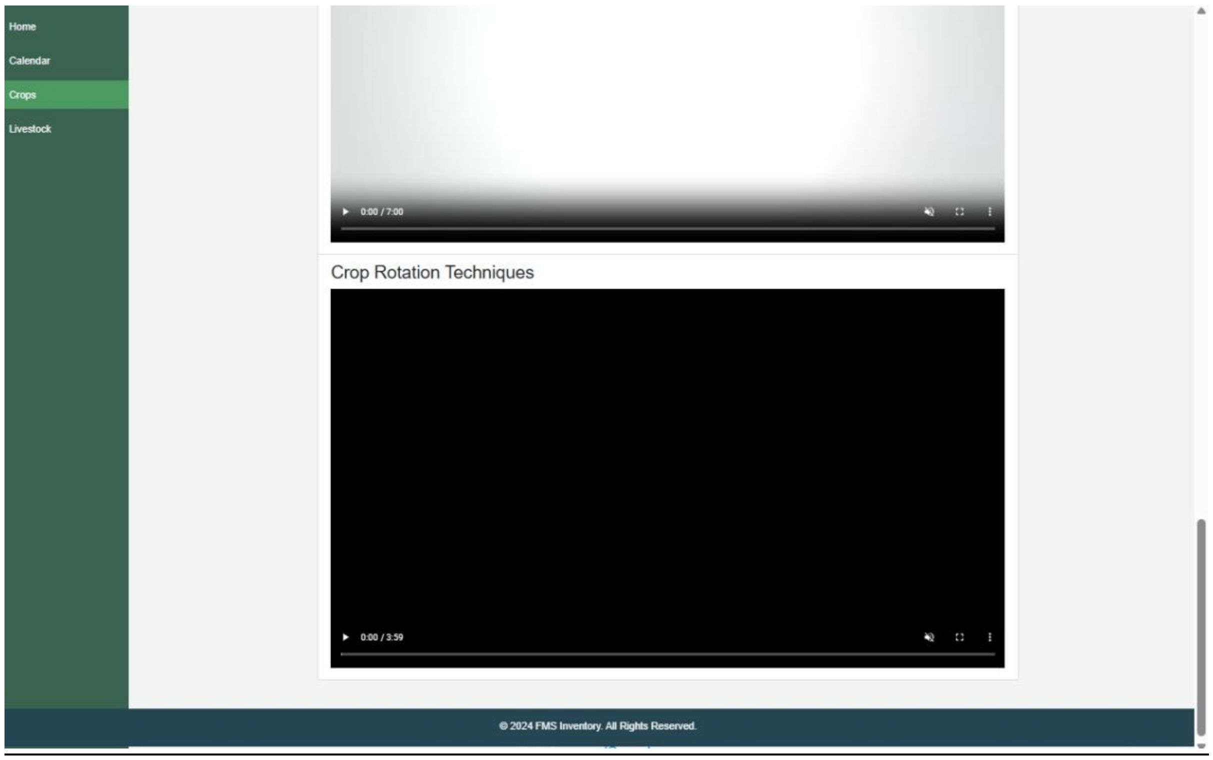
```
// HTML page setup
START HTML Document
    DEFINE DOCTYPE as html
    SET language to en (English)

    // Head section: Meta, title, and styles
    IN HEADER:
        SET character encoding to UTF-8
        SET viewport for mobile responsiveness
        SET page title to "Fields Page - Agricultural Store"
        LINK external stylesheets and Google Charts

// Body section setup
START Body
    // Navbar setup
    DEFINE Navigation bar
        ADD Home link
        ADD disabled Reports link
        ADD Search form with input and button

    // Header section for Fields Overview
    DISPLAY Page title "Our Fields"
    CREATE Navigation links to Livestock, Vegetables, and Fruits fields
```

```
// Main content section for charts, field details, and table
START Main Content:
  // Chart section
  CREATE "Fields Data Overview" title
  SET container for Google Chart visualization

  // Field Details section
  CREATE Livestock Field block:
    SHOW livestock image
    DISPLAY area, livestock types, and status
  CREATE Vegetable Field block:
    SHOW vegetable field image
DISPLAY area, crops, and status
  CREATE Fruit Field block:
    SHOW fruit field image
    DISPLAY area, crops, and status

  // Table section for field data summary
  CREATE Table with columns:
    Field Name, Area (acres), Crops/Stock, Status
  INSERT field rows (Livestock, Vegetables, Fruits)

END Main Content

// Footer section
START Footer
```

```
    DISPLAY Contact Information and Copyright
    ADD support email link

// Link JavaScript for Google Charts and any additional scripts
END Body

// JavaScript: Google Charts setup
LOAD Google Charts library
CREATE function to load and draw pie chart using field data
DISPLAY chart in the specified div
```

Crops information”


Home
Calendar
Crops
Livestock
Crops Information
Search Crops or Videos...
Wheat
Learn about wheat farming, its seasons, and best practices.
Vegetables
Discover the best practices for growing vegetables.
Sugarbeets
Learn about growing sugar beets and their agricultural value.
Sugarbeets
Learn about growing sugar beets and their agricultural value.
Educational Videos
Wheat Farming Techniques
0:00 / 6:36
Vegetable Farming Guide



Explanation:

The program is designed as a webpage for the "Crops" section of a Farming Management System (FMS). It provides a user interface where users can explore information about various crops, watch educational videos related to farming, and search for specific crops or videos using a search functionality. The webpage includes navigation to different sections of the FMS like "Home," "Calendar," and "Livestock," ensuring that users can easily navigate the system. The key features include:

- Sidebar Navigation: Links to various sections of the FMS, helping users move between pages.
- Main Content: Displays crop cards and educational videos related to farming techniques.
- Search Functionality: Allows users to search for crops or videos based on keywords.
- Responsive Layout: The page uses Bootstrap for responsive design, ensuring it works across different devices.

Pseudo Code

Below is a breakdown of the program structure, represented as pseudo code:

HTML Structure:

1. Header Section:
    - Display a header with a title and a search bar.
    - Implement a search input for filtering crops and videos.
2. Sidebar Navigation:
    - Create a sidebar navigation with links to "Home", "Calendar", "Crops", and "Livestock" sections.
3. Crops Cards Section:
    - For each crop, display a card containing:
        - An image of the crop.
        - A brief description and a link to further information.
4. Videos Section:
    - For each video, display:
        - A title.
        - An embedded video related to farming techniques.
5. Search Functionality:
    - Use JavaScript to implement a search feature that filters both crop cards and videos based on user input.

CSS:

- Use an external CSS file (crops.css) to style the page and ensure a clean and visually appealing layout.

JavaScript (Search Function):

1. Define Variables:
    - Input for search query.
    - Elements representing crop cards and videos.
2. Loop Through Crop Cards:
    - For each card, check if its title matches the search query.
    - If the title matches, display the card, otherwise hide it.
3. Loop Through Videos:

o For each video, check if its title matches the search query.
o If the title matches, display the video, otherwise hide it.

Start

1. Load the webpage with:

a. Header containing search bar and title.

b. Sidebar navigation with links to different sections.

c. Main content area for crop cards and educational videos.

2. Display Crop Cards:

For each crop:

a. Show an image of the crop.

b. Show a title and description.

c. Provide a link for further information.

3. Display Educational Videos:

For each video:

a. Show the title.

b. Embed the video with controls to play/pause.

c. Provide any additional information.

4. Implement Search Functionality:

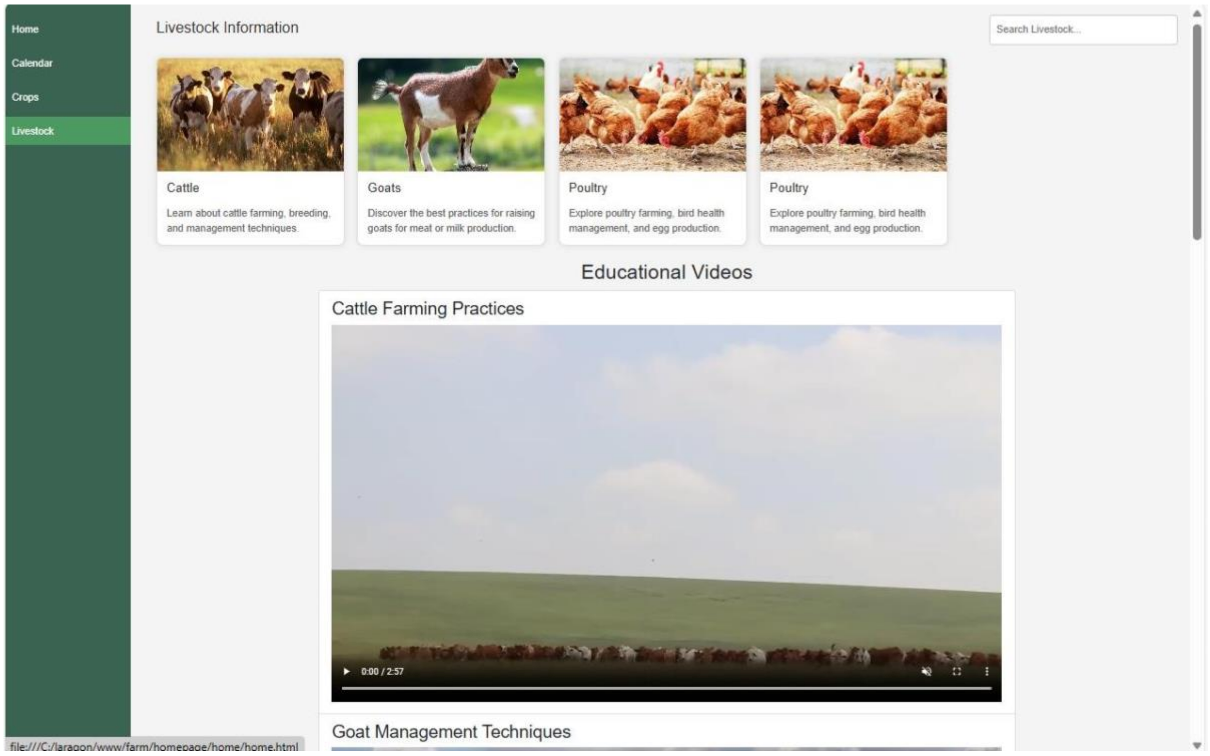
a. Capture the search query from the search input field.

b. Convert the query to lowercase for comparison.

c. Loop through each crop card:

```
i.  If the card title contains the search query, display the card.
ii. Otherwise, hide the card.
   d. Loop through each video:
i.  If the video title contains the search query, display the video.
ii. Otherwise, hide the video.


5. Handle User Interaction:
   a. When the user types in the search bar, update the visibility of cards and videos dynamically.


End
```

<u>Livestock information:</u>



Livestock Information

Cattle

Learn about cattle farming, breeding, and management techniques.

Goats

Discover the best practices for raising goats for meat or milk production.

Poultry

Explore poultry farming, bird health management, and egg production.

Poultry

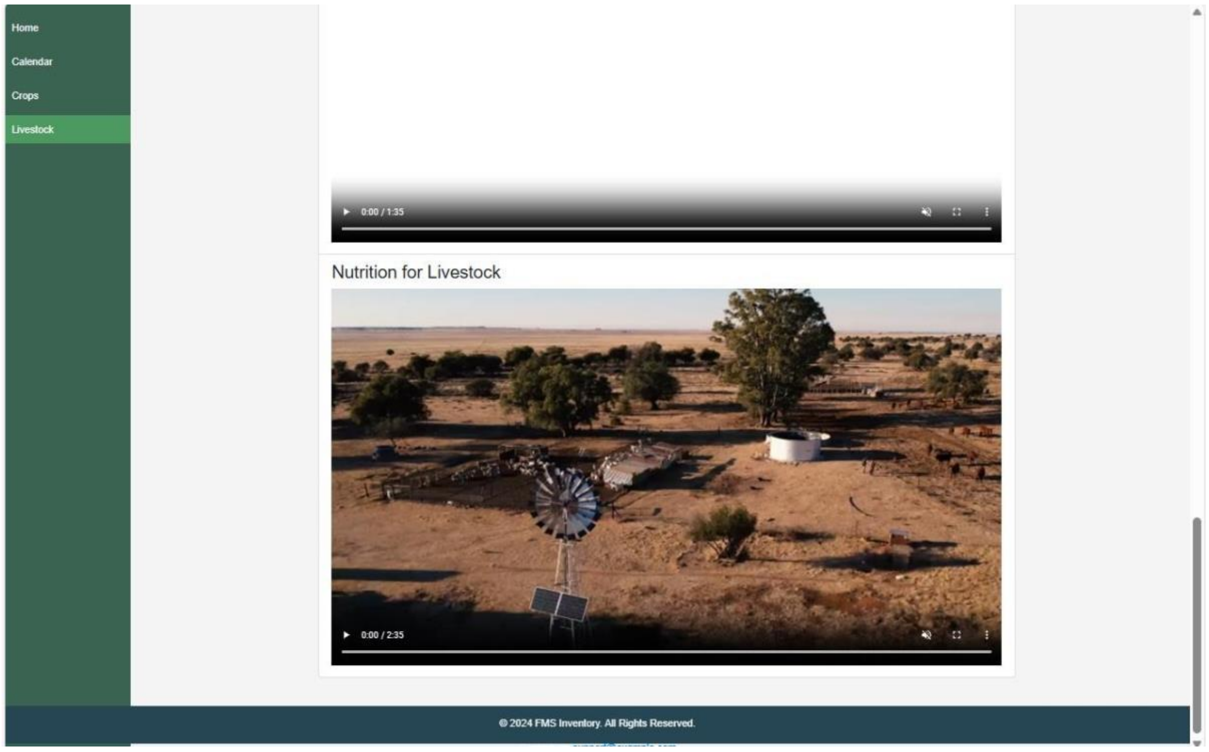
Explore poultry farming, bird health management, and egg production.

Educational Videos

Cattle Farming Practices

0:00 / 2:57

Goat Management Techniques



<u>Explanation:</u>

Program Design Overview

This HTML program is designed to serve as a part of a Farming Management System focused specifically on livestock information. The structure follows a basic layout with a sidebar navigation menu and a main content area that includes sections for livestock cards and educational videos.

Key Components

1. HTML Structure:
    - Head Section: Contains meta tags for character set and viewport settings, a title, and links to CSS stylesheets (Bootstrap and a custom stylesheet).
    - Body Section:
        - Sidebar Navigation: Provides links to different sections of the farming management system (Home, Calendar, Crops, Livestock).
        - Main Content Section:
        - A header with a title and a search bar for filtering livestock information.

▪ A card container displaying various livestock types, including images and descriptions.
▪ A video container showcasing educational videos related to livestock management.

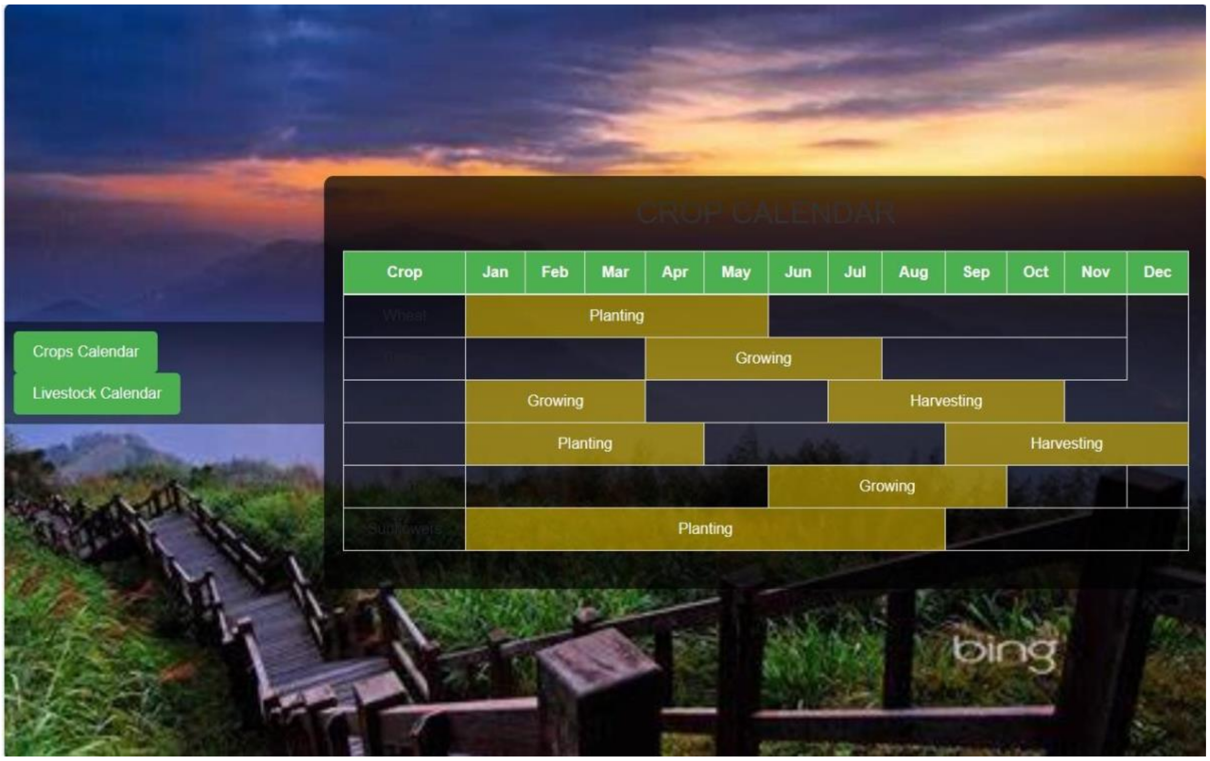
2. Search Functionality:
   - Allows users to search for specific livestock information or videos based on their input.
3. Footer: Provides copyright information and a contact email.

```
START PROGRAM


DEFINE FUNCTION initialize()
   LOAD CSS stylesheets
   DISPLAY sidebar navigation
   DISPLAY main content with header and search bar
   DISPLAY livestock cards
   DISPLAY educational videos
END FUNCTION


DEFINE FUNCTION searchFunction()
   // Get input from search bar
   SET input to the value of 'searchBar'
   CONVERT input to lowercase and store in filter


   // Get all livestock cards and videos
   SET cards to all elements with class 'card' in 'cardsContainer'
   SET videos to all elements with class 'list-group-item' in 'videosContainer'


   // Loop through livestock cards
```

```
    FOR EACH card IN cards DO
        // Get the title attribute of the card
        SET txtValue to the data-title attribute of card (converted to lowercase)
        // Show or hide the card based on filter match
        IF txtValue contains filter THEN
            DISPLAY card
        ELSE
            HIDE card
        END IF
    END FOR

    // Loop through educational videos
    FOR EACH video IN videos DO
        // Get the title attribute of the video
        SET txtValue to the data-title attribute of video (converted to lowercase)
        // Show or hide the video based on filter match
        IF txtValue contains filter THEN
            DISPLAY video
        ELSE
            HIDE video
        END IF
    END FOR
END FUNCTION

// Call the initialize function to set up the page
CALL initialize()
```

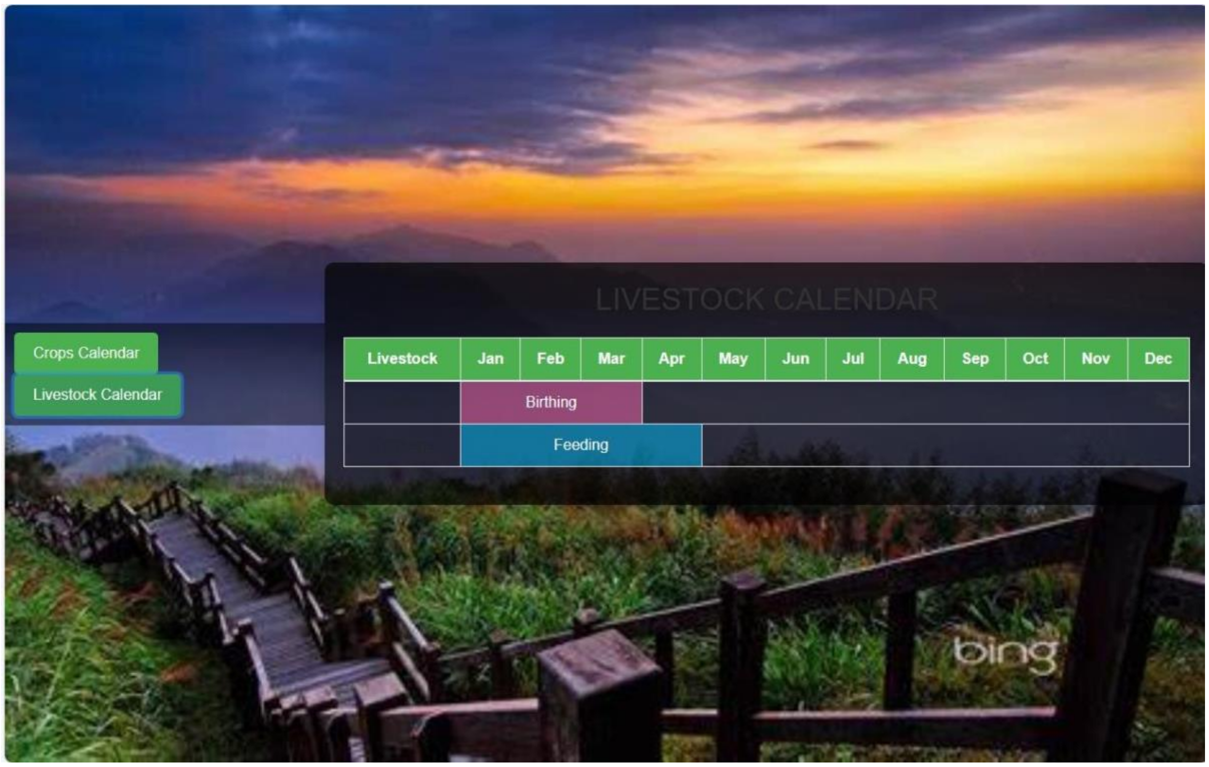
END PROGRAM

Calender:



CROP CALENDAR

Crop	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Wheat	Planting											
[illegible]				Growing								
	Growing						Harvesting					
	Planting								Harvesting			
						Growing						
Sunflowers	Planting											



LIVESTOCK CALENDAR

Livestock	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
	Birthing											
	Feeding											

<u>Explanation:</u>

Program Design

1. Purpose: The purpose of this HTML code is to create a calendar that displays seasonal activities for crops and livestock on a farming management system webpage. Users can switch between the crops and livestock calendars.

2. Structure:
   - o The page is structured with a navigation bar at the top, followed by two main sections: one for the crop calendar and one for the livestock calendar. o Each calendar is displayed in a table format with months as columns and activities as rows.
   - o Specific activities like planting, growing, birthing, and feeding are highlighted with color-coded backgrounds.
3. Style:
   - o The page uses Bootstrap for styling and responsiveness. Additional CSS is included in the <style> section to customize the appearance, such as setting background images and colors for different activities.
   - o Tooltips provide additional information when users hover over certain activities.
4. Interactivity:
   - o The JavaScript file (calender.js) is linked but not included in the provided code. This file is likely responsible for handling the interactive elements, such as toggling between the crops and livestock calendars.

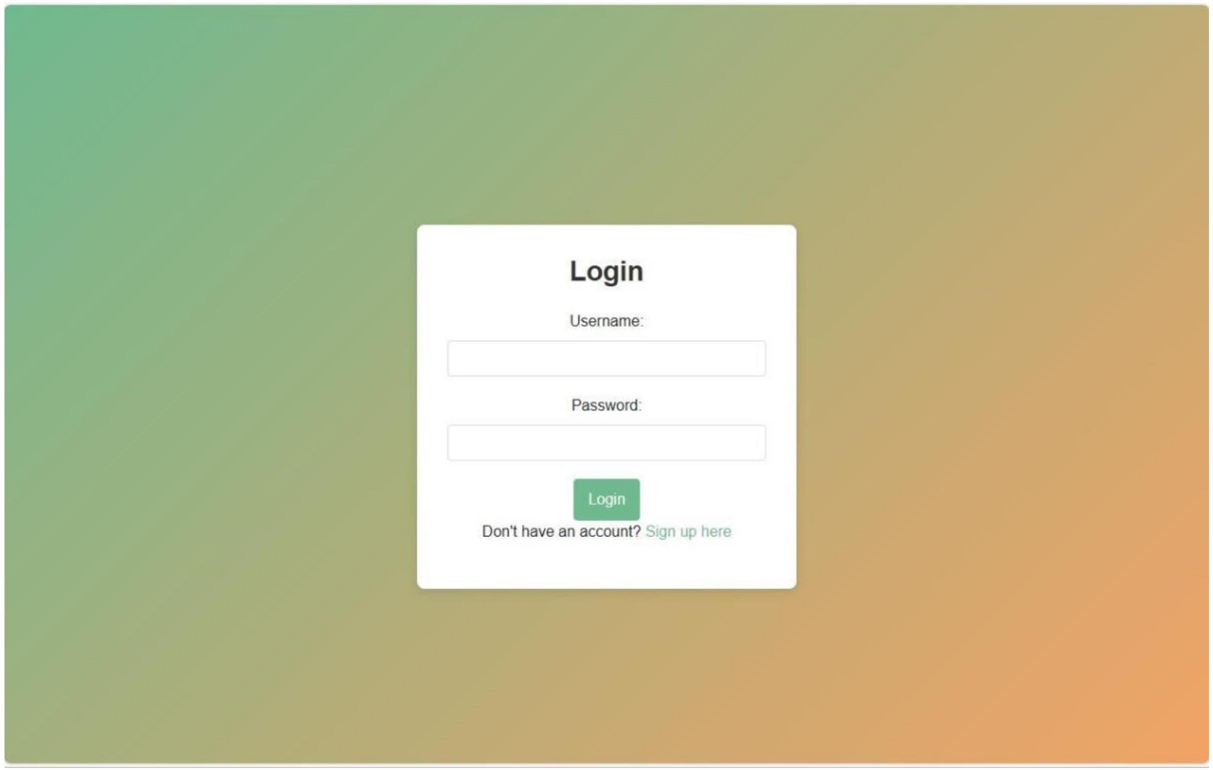
Pseudocode

```
START


FUNCTION Main

  SET up HTML document structure

  ADD metadata for character set and viewport

  ADD title "Calendar"


  INCLUDE Bootstrap CSS

  INCLUDE custom CSS for styling
```

```
SET body background image and styles

CREATE navigation bar with buttons for Crop Calendar and Livestock Calendar

CREATE Crop Calendar section
  DISPLAY title "CROP CALENDAR"
  CREATE a table for crops
    ADD header row with month names
    FOR each crop in crops data
      ADD a row for the crop
      ADD columns for each month
      IF activity is "planting" THEN
        HIGHLIGHT with background color and tooltip
      ELSE IF activity is "growing" THEN
        HIGHLIGHT with different background color and tooltip
      ELSE IF activity is "harvesting" THEN
        HIGHLIGHT with another background color and tooltip
      ENDIF
    ENDFOR
  END TABLE
END Crop Calendar section

CREATE Livestock Calendar section (initially hidden)
  DISPLAY title "LIVESTOCK CALENDAR"
  CREATE a table for livestock
    ADD header row with month names
```

```
      FOR each livestock type in livestock data
        ADD a row for livestock type
        ADD columns for each month
        IF activity is "birthing" THEN
          HIGHLIGHT with background color and tooltip
        ELSE IF activity is "feeding" THEN
          HIGHLIGHT with different background color and tooltip
        ENDIF
      ENDFOR
    END TABLE
  END Livestock Calendar section

  LINK JavaScript file for interactivity

END FUNCTION
```

Log in :



Explanation:

Program Design

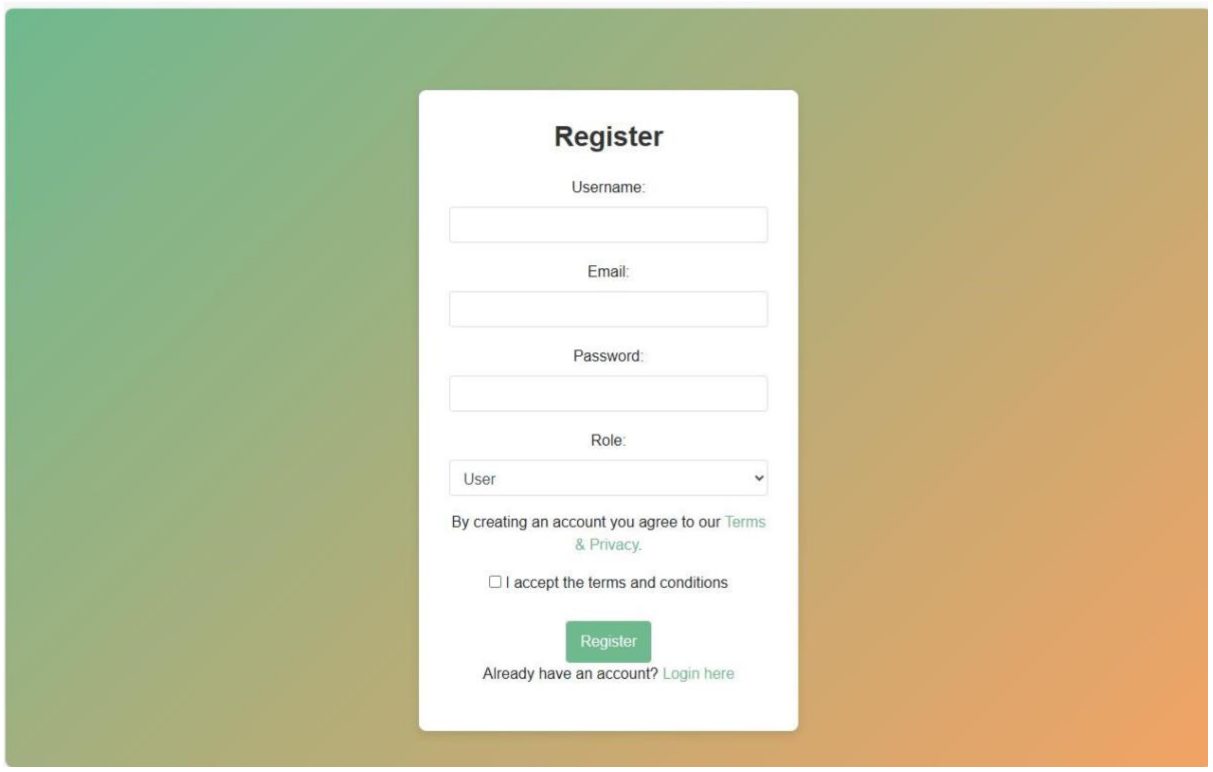
1. Purpose: The HTML code is designed to create a login page where users can enter their credentials to access the application. The design includes input fields for username and password, as well as a link to a signup page for new users.
2. Structure:
    - The page is structured with a <head> section for metadata and linked resources, and a <body> section containing the login form and relevant instructions.
    - The form sends data to a login.php file for processing.
3. Style:
    - The page uses Bootstrap for responsive design and additional custom CSS for aesthetics, including a gradient background, container styling, and form element styles.
    - It employs transitions and hover effects to enhance user experience.
4. Interactivity:
    - JavaScript is used to handle the form submission event. It prevents the default form submission, simulates a role-based response, and changes

the background color based on the user role before finally submitting the form. Pseudocode

```
START


FUNCTION Main
  SET up HTML document structure
  ADD metadata for character set and viewport
  ADD title "Login"

  INCLUDE Bootstrap CSS
  INCLUDE custom CSS for styling

  SET body background gradient and styles
  CENTER content in the viewport

  CREATE container for login form
    DISPLAY title "Login"
    CREATE a form with action "login.php" and method "POST"
      CREATE input field for "username"
        SET required attribute
      CREATE input field for "password"
        SET required attribute
      CREATE a button for form submission
    ADD link to sign up for new users

  LINK JavaScript for dynamic behavior
```

```
 DEFINE JavaScript function to handle form submission
    PREVENT default form submission
    COLLECT username and password from form inputs

    // Simulate backend role checking
    SET role based on username (admin/user)

    CHANGE background color based on role
    WAIT for 500 milliseconds
    SUBMIT the form

END FUNCTION
```

Sign up:



Register

Username:

Email:

Password:

Role:

User

By creating an account you agree to our Terms & Privacy.

☐ I accept the terms and conditions

Register

Already have an account? Login here

Explanation:

Program Design

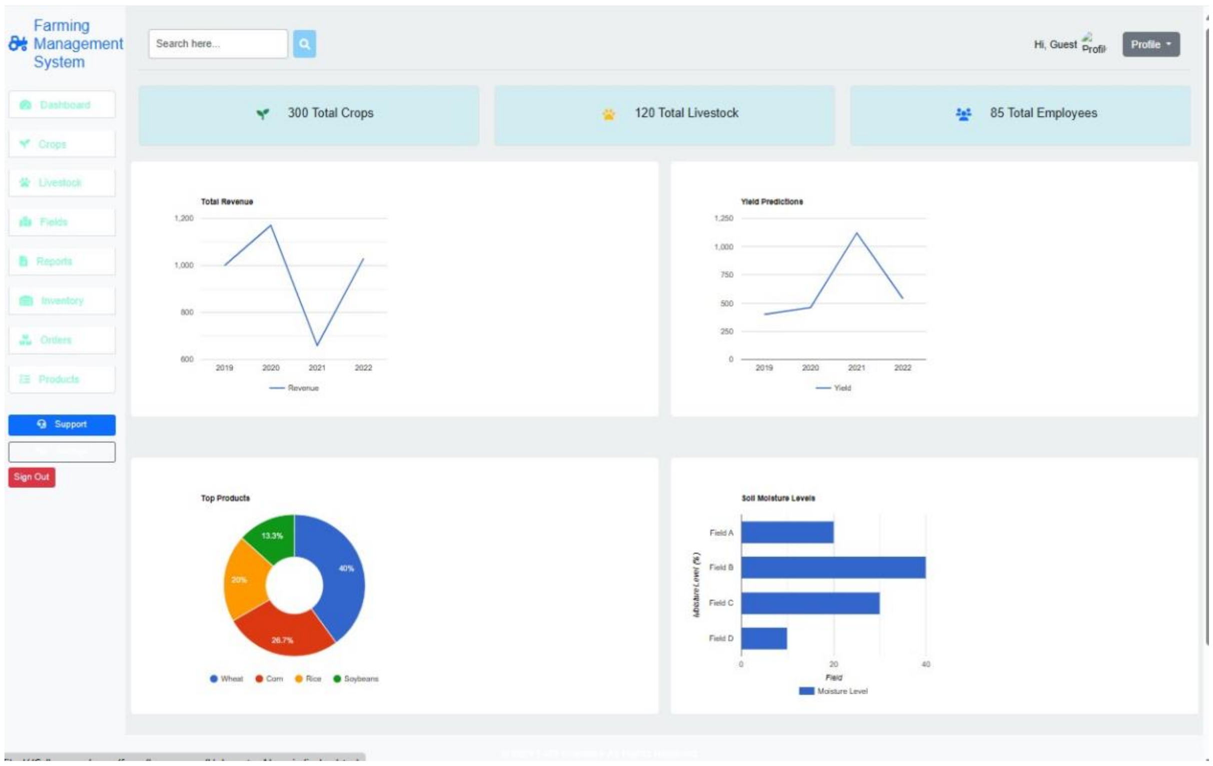
1. Purpose: The HTML code is intended to create a signup page where users can register for an account by providing their username, email, password, and selecting a role (user or admin).

2. Structure:
   - The page consists of a <head> section for metadata, styles, and linked resources, and a <body> section containing the registration form and relevant instructions.
   - The form submits user data to a signup.php file for processing.
3. Style:
   - The page uses Bootstrap for responsive design and custom CSS for styling, including a gradient background, a white container for the form, and styling for form elements and buttons.
4. User Interaction:
   - The form includes input fields for user information and has a checkbox for accepting terms and conditions, which enhances user engagement and compliance.

Pseudocode

```
START


FUNCTION Main
  SET up HTML document structure
  ADD metadata for character set and viewport
  ADD title "Signup"


  INCLUDE Bootstrap CSS
  INCLUDE custom CSS for styling


  SET body background gradient and styles
  CENTER content in the viewport
  CREATE container for signup form
    DISPLAY title "Register"
    CREATE a form with action "signup.php" and method "POST"
      CREATE input field for "username"
        SET required attribute
      CREATE input field for "email"
        SET required attribute
      CREATE input field for "password"
        SET required attribute
      CREATE a dropdown select for "role"
        ADD options "User" and "Admin"
        SET required attribute
      DISPLAY terms and conditions text with link
      CREATE checkbox for "terms" with required attribute
      CREATE a button for form submission
    ADD link to login for existing users

END FUNCTION
```

Admin-dashboard:



Explanation:

Program Design for the Dashboard Code

The given HTML code outlines a web-based dashboard for a Farming Management System (FMS). It is designed to provide an interactive and informative interface for users, enabling them to manage various aspects of farming, such as crops, livestock, inventory, and tasks.

Key Components:

1. HTML Structure:
   - The code is organized into semantic sections: a sidebar for navigation, a main content area for displaying statistics, charts, and tasks, and a footer.
   - Each section utilizes Bootstrap for responsive design, ensuring that the layout adapts to different screen sizes.
2. User Interface Elements:
   - Sidebar: Contains navigation links to various sections like Dashboard, Crops, Livestock, Reports, and Inventory. It also includes buttons for user settings and signing out.
   - Main Content:

- Header: Features a search bar and user profile display, allowing users to search for items and access profile settings.
- Stats Overview: Displays key statistics (e.g., total crops, livestock, employees) using iconography and simple text.
- Charts Section: Designed for visual data representation, displaying revenue, yield predictions, top products, and soil moisture levels using Google Charts.
- Tasks and Weather Section: Provides a list of tasks and current weather conditions relevant to farming.

3. Modals:
   - A modal is implemented for editing user profiles, providing fields for updating personal information and uploading a profile picture.
4. Interactivity:
   - JavaScript functions handle user interactions, including fetching data from localStorage, signing out, changing background colors of statistics, searching for items, and rendering charts.
5. Styling:
   - Uses Bootstrap for responsive styling and Font Awesome for icons. Custom CSS is added for specific elements like chart sizing and transition effects.

```
START


FUNCTION onPageLoad()
  SET username = getItemFromLocalStorage('username')
  IF username EXISTS THEN
    DISPLAY 'Hi, ' + username IN user-name element
    FILL profile fields with stored user data
    SET profile picture to stored image or default
  ELSE
    DISPLAY "Hi, Guest" IN user-name element
  ENDIF


  CALL changeStatCardColors()
```

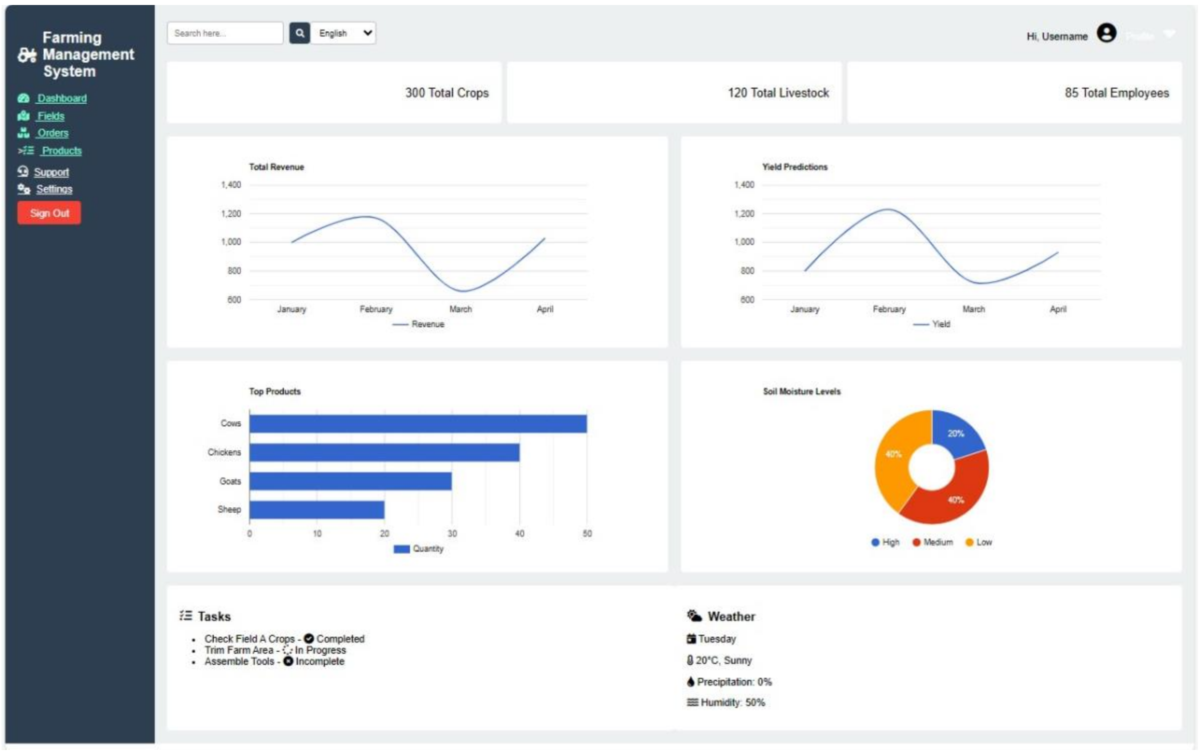
```
END FUNCTION

FUNCTION signOut()
   REMOVE user data from localStorage
   REDIRECT to 'home.html'
END FUNCTION

FUNCTION changeStatCardColors()
   SET colors = ['#f8d7da', '#d1ecf1', '#d4edda', '#fff3cd', '#cce5ff', '#e2e3e5']
INITIALIZE index = 0

   REPEAT EVERY 5 seconds
      SELECT all stat cards
      FOR EACH card IN stat cards DO
         SET card background color to colors[index]
      END FOR
      INCREMENT index (CIRCULAR)
   END REPEAT
END FUNCTION

FUNCTION setupSearchButton()
   ADD event listener for search button CLICK
   ON CLICK:
      GET search query from search bar
      DISPLAY alert with search query (functionality not implemented)
END FUNCTION
```

```
FUNCTION setupCharts()
LOAD Google Charts API
ON Load:
    CALL drawRevenueChart()
    CALL drawYieldChart()
    CALL drawTopProductsChart()
    CALL drawSoilMoistureChart()
END FUNCTION

FUNCTION drawRevenueChart()
  CREATE data table for revenue
  SET options for chart
  DRAW LineChart in 'revenue-chart' element
END FUNCTION

FUNCTION drawYieldChart()
  CREATE data table for yield
  SET options for chart
  DRAW LineChart in 'yield-chart' element
END FUNCTION

FUNCTION drawTopProductsChart()
  CREATE data table for top products
  SET options for chart
  DRAW PieChart in 'top-products-chart' element
```

```
END FUNCTION


FUNCTION drawSoilMoistureChart()
  CREATE data table for soil moisture levels
  SET options for chart
  DRAW BarChart in 'soil-moisture-chart' element
END FUNCTION


FUNCTION setupProfileForm()    ADD event
listener for profile form SUBMIT    ON
SUBMIT:
    PREVENT default form submission
    GET form values
    IF profile picture is uploaded THEN
      READ image as Data URL
      STORE image in localStorage
      UPDATE profile picture displayed
    ENDIF
    STORE other form data in localStorage
    CLOSE modal
END FUNCTION


CALL onPageLoad()
CALL setupSearchButton()
CALL setupCharts()
CALL setupProfileForm()
```

END

User dashboard:



300 Total Crops

120 Total Livestock

85 Total Employees

Tasks

- Check Field A Crops - Completed
- Trim Farm Area - In Progress
- Assemble Tools - Incomplete

Weather

Tuesday

20°C, Sunny

Precipitation: 0%

Humidity: 50%

Explanation:

Program Design for the Dashboard Code

The given HTML code outlines a web-based dashboard for a Farming Management System (FMS). It is designed to provide an interactive and informative interface for users, enabling them to manage various aspects of farming, such as crops, livestock, inventory, and tasks.

Key Components:

1. HTML Structure:
   - o The code is organized into semantic sections: a sidebar for navigation, a main content area for displaying statistics, charts, and tasks, and a footer.
   - o Each section utilizes Bootstrap for responsive design, ensuring that the layout adapts to different screen sizes.

2. User Interface Elements:
   - Sidebar: Contains navigation links to various sections like Dashboard, Crops, Livestock, Reports, and Inventory. It also includes buttons for user settings and signing out.
   - Main Content:
     - Header: Features a search bar and user profile display, allowing users to search for items and access profile settings.
     - Stats Overview: Displays key statistics (e.g., total crops, livestock, employees) using iconography and simple text.
     - Charts Section: Designed for visual data representation, displaying revenue, yield predictions, top products, and soil moisture levels using Google Charts.
     - Tasks and Weather Section: Provides a list of tasks and current weather conditions relevant to farming.
3. Modals:
   - A modal is implemented for editing user profiles, providing fields for updating personal information and uploading a profile picture.
4. Interactivity:
   - JavaScript functions handle user interactions, including fetching data from localStorage, signing out, changing background colors of statistics, searching for items, and rendering charts.
5. Styling:
   - Uses Bootstrap for responsive styling and Font Awesome for icons. Custom CSS is added for specific elements like chart sizing and transition effects.

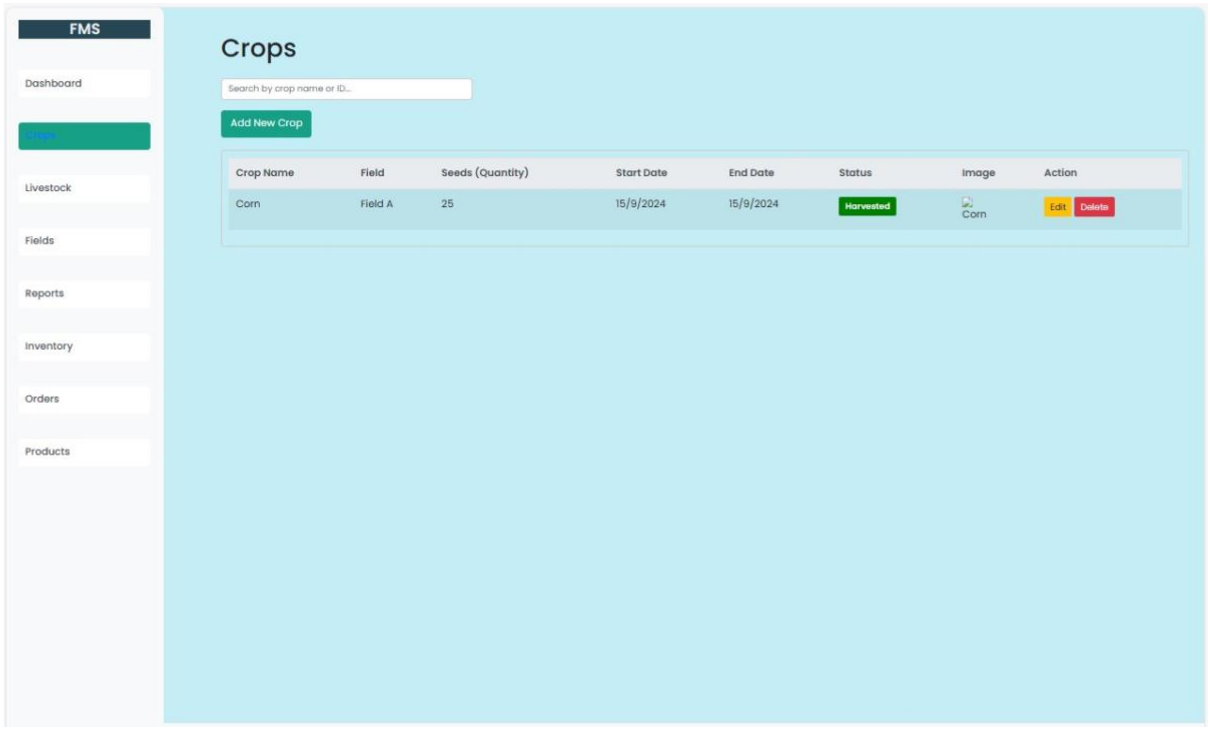
```
START


FUNCTION onPageLoad()
  SET username = getItemFromLocalStorage('username')
  IF username EXISTS THEN
    DISPLAY 'Hi, ' + username IN user-name element
    FILL profile fields with stored user data
    SET profile picture to stored image or default
  ELSE
```

```
    DISPLAY "Hi, Guest" IN user-name element
  ENDIF

  CALL changeStatCardColors()
END FUNCTION

FUNCTION signOut()
  REMOVE user data from localStorage
  REDIRECT to 'home.html'
END FUNCTION

FUNCTION changeStatCardColors()
  SET colors = ['#f8d7da', '#d1ecf1', '#d4edda', '#fff3cd', '#cce5ff', '#e2e3e5']
INITIALIZE index = 0

  REPEAT EVERY 5 seconds
    SELECT all stat cards
    FOR EACH card IN stat cards DO
      SET card background color to colors[index]
    END FOR
    INCREMENT index (CIRCULAR)
  END REPEAT
END FUNCTION

FUNCTION setupSearchButton()
  ADD event listener for search button CLICK
  ON CLICK:
      GET search query from search bar
      DISPLAY alert with search query (functionality not implemented)
END FUNCTION

FUNCTION setupCharts()
LOAD Google Charts API
ON Load:
      CALL drawRevenueChart()
      CALL drawYieldChart()
      CALL drawTopProductsChart()
      CALL drawSoilMoistureChart()
END FUNCTION

FUNCTION drawRevenueChart()
   CREATE data table for revenue
   SET options for chart
   DRAW LineChart in 'revenue-chart' element
END FUNCTION

FUNCTION drawYieldChart()
   CREATE data table for yield
   SET options for chart
   DRAW LineChart in 'yield-chart' element
END FUNCTION
```

```
FUNCTION drawTopProductsChart()
  CREATE data table for top products
  SET options for chart
  DRAW PieChart in 'top-products-chart' element
END FUNCTION

FUNCTION drawSoilMoistureChart()
  CREATE data table for soil moisture levels
  SET options for chart
  DRAW BarChart in 'soil-moisture-chart' element
END FUNCTION

FUNCTION setupProfileForm()    ADD event
listener for profile form SUBMIT    ON
SUBMIT:
    PREVENT default form submission
    GET form values
    IF profile picture is uploaded THEN
      READ image as Data URL
      STORE image in localStorage
      UPDATE profile picture displayed
    ENDIF
    STORE other form data in localStorage
    CLOSE modal
END FUNCTION
```

```
CALL onPageLoad()

CALL setupSearchButton()

CALL setupCharts()

CALL setupProfileForm()


END
```

Crops:



Crop Name	Field	Seeds (Quantity)	Start Date	End Date	Status	Image	Action
Corn	Field A	25	15/9/2024	15/9/2024	Harvested	Corn	Edit Delete

Explanation:

Program Design Overview

1. HTML Structure: The page is structured using HTML5 with Bootstrap for styling. It includes a sidebar for navigation, a main content area for displaying crops, a form for adding new crops, and a footer.
2. CSS Styling: Custom CSS is applied for layout and design aesthetics, such as sidebar styles, table formatting, and popup form design.
3. JavaScript Functionality: Although the JavaScript functionalities (like search functionality and handling the popup form) are not fully detailed in the provided

code, references to functions such as searchCrops() suggest there will be JavaScript to enhance interactivity.

4. Forms and Data Handling: The form for adding a new crop includes fields for crop details and an image upload option. It uses the POST method to submit data to add_crop.php.

5. Responsive Design: The use of Bootstrap ensures that the page is responsive and adjusts to different screen sizes.

```
START Program "FMS - Crops Page"


// Define the main layout structure

FUNCTION createPage():

  SET up HTML structure

  SET up Bootstrap styles

  INCLUDE sidebar navigation

  INCLUDE main content area with title

  INCLUDE search bar for filtering crops

  INCLUDE button for adding a new crop

  INCLUDE crops table for displaying crops

  INCLUDE footer section


// Define sidebar navigation

FUNCTION createSidebar():

  CREATE sidebar element

  ADD links to dashboard, crops, livestock, fields, reports, inventory, and orders


// Define main content

FUNCTION createMainContent():

  DISPLAY title "Crops"
```

```
    ADD search input with onkeyup event to call searchCrops()
    ADD button "Add New Crop" with onclick event to show add crop form
    CREATE crops table with headers:
- Crop Name
- Field
- Seeds (Quantity)
- Start Date
- End Date
- Status
- Image
- Action
    POPULATE table rows with crop data

// Define search functionality
FUNCTION searchCrops():
    GET input from search bar
    FILTER crops in table based on search input
    UPDATE visible rows in crops table

// Define add crop form functionality
FUNCTION showAddCropForm():
    DISPLAY popup form for adding a new crop
    SET form fields: Crop Name, Field, Seeds, Start Date, End Date, Status, Image, Notes

FUNCTION addNewCrop():
ON form submission:
    VALIDATE form inputs
    IF valid THEN
        SEND data to "add_crop.php"
        CLOSE popup
        REFRESH crops table
    ELSE
        DISPLAY error message

// Define cancel button functionality
FUNCTION cancelAddCrop():
    CLOSE popup without saving changes

// Define footer
FUNCTION createFooter():
    DISPLAY footer with copyright information and contact details

// Call main functions to construct the
page createPage() createSidebar()
createMainContent() createFooter()
```

 createPage: This is the main function that sets up the overall structure of the HTML page, integrating all components (sidebar, main content, footer).

 createSidebar: This function creates the sidebar navigation menu with links to other sections of the application.

 createMainContent: It defines the main area of the page, including the title, search bar, button for adding new crops, and a table to display crop data.

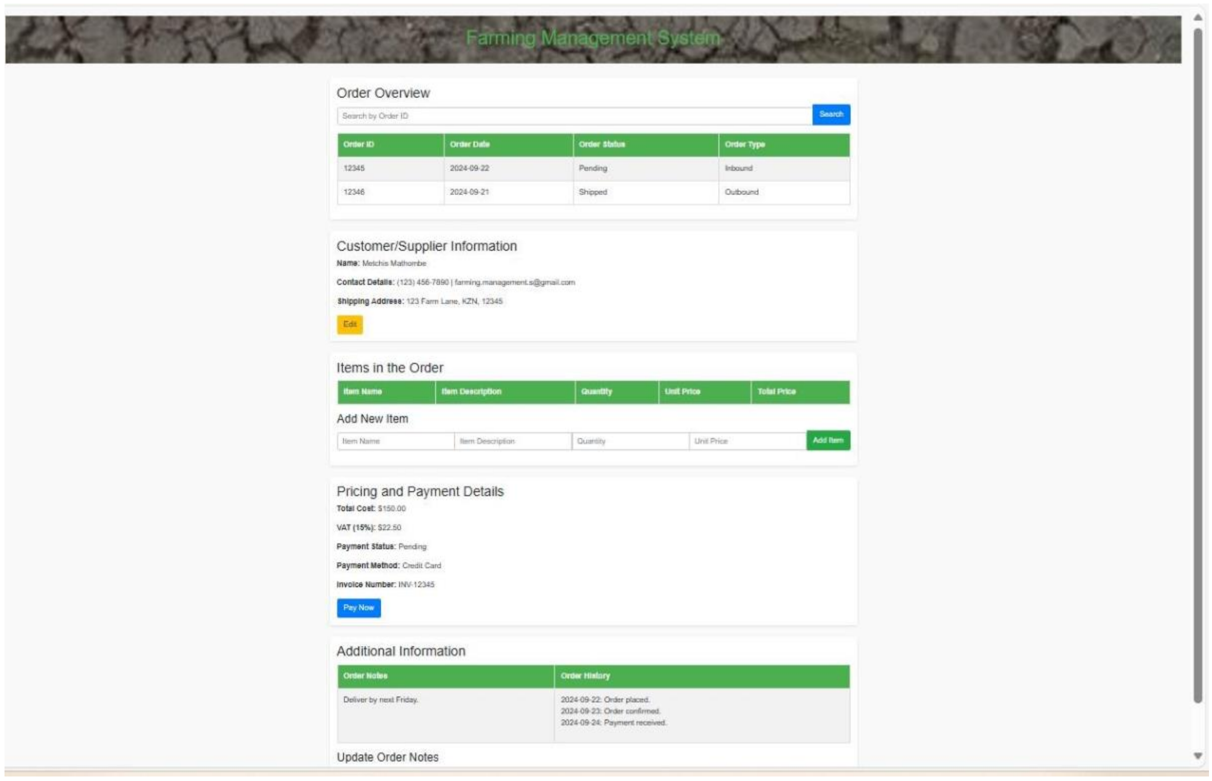
 searchCrops: This function handles the search functionality by filtering crops in the table based on user input.

- showAddCropForm: It displays a popup form for users to enter details about a new crop.
- addNewCrop: This function manages the submission of the new crop data to the server and handles validation.
- cancelAddCrop: This function allows users to cancel adding a new crop, closing the popup

without saving. createFooter: It defines the footer section of the page, including

contact information.

Order :



Farming Management System

Order Overview

Search by Order ID | Search

Order ID	Order Date	Order Status	Order Type
12345	2024-09-22	Pending	Inbound
12346	2024-09-21	Shipped	Outbound

Customer/Supplier Information

Name: Melchis Mathombe

Contact Details: (123) 456-7890 | farming.management.s@gmail.com

Shipping Address: 123 Farm Lane, KZN, 12345

Edit

Items in the Order

Item Name	Item Description	Quantity	Unit Price	Total Price

Add New Item

Item Name | Item Description | Quantity | Unit Price | Add Item

Pricing and Payment Details

Total Cost: $150.00

VAT (15%): $22.50

Payment Status: Pending

Payment Method: Credit Card

Invoice Number: INV-12345

Pay Now

Additional Information

Order Notes	Order History
Deliver by next Friday.	2024-09-22: Order placed. 2024-09-23: Order confirmed. 2024-09-24: Payment received.

Update Order Notes

Explanation:

Program Design

The "Orders Page" consists of several sections to manage and display orders, customer/supplier information, items in orders, pricing/payment details, and additional information. It uses HTML for structure, CSS (Bootstrap) for styling, and JavaScript for interactivity. Here's a summary of each component:

1. Header: Displays the title of the system.

2. Order Overview:
    - A search bar to filter orders by Order ID.
    - A table displaying orders with their details (ID, Date, Status, Type).
3. Customer/Supplier Information:
    - Displays the name, contact details, and shipping address of the customer/supplier.
    - An "Edit" button that opens a modal to modify this information.
4. Items in the Order:
    - A table for displaying items in the current order.
    - Input fields to add new items, including name, description, quantity, and unit price.
5. Pricing and Payment Details:
    - Displays the total cost, VAT, payment status, payment method, and invoice number.
    - A "Pay Now" button to initiate payment (the functionality is not fully defined in the provided code).
6. Additional Information:
    - Displays order notes and order history. o Input field and button to update order note

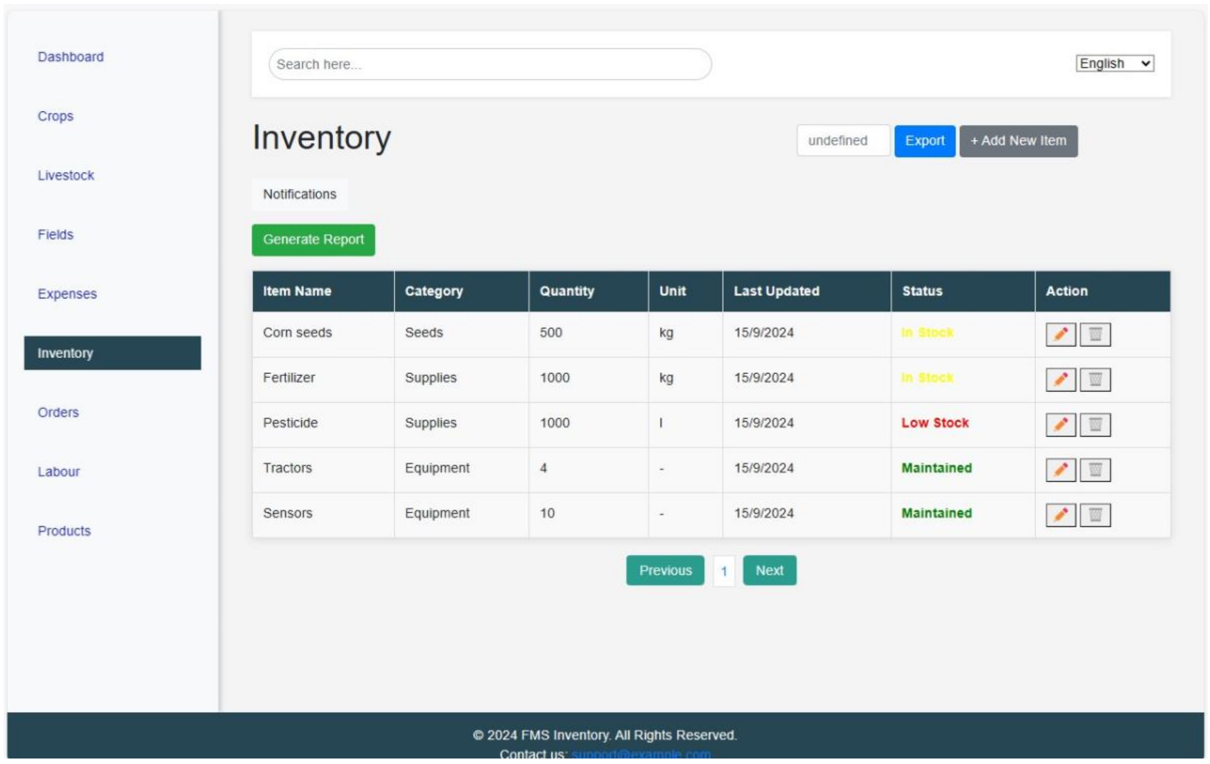
```
START Program OrdersPage


// Initialize the page

LOAD HTML structure

LOAD CSS styles (Bootstrap and custom)

LOAD JavaScript libraries (jQuery, Bootstrap)


// Header Section

DISPLAY header with title "Farming Management System"


// Main Content Section

CREATE container for main content
```

```
// Order Overview Section
DISPLAY "Order Overview" header
CREATE input group for search functionality
  INPUT: Order ID (search-bar)
  BUTTON: Search (search-button)
DISPLAY orders in a table with columns: Order ID, Order Date, Order Status, Order Type

// Customer/Supplier Information Section
DISPLAY "Customer/Supplier Information" header
DISPLAY customer/supplier details (Name, Contact Details, Shipping Address)
BUTTON: Edit (opens edit modal)

// Edit Modal Functionality
IF Edit button is clicked THEN
  OPEN edit modal
  DISPLAY current customer/supplier details in input fields
  BUTTON: Save changes (saves updated information)

// Items in Order Section
DISPLAY "Items in the Order" header
CREATE table for displaying items with columns: Item Name, Item Description, Quantity,
Unit Price, Total Price
CREATE input group for adding new items
  INPUTS: Item Name, Item Description, Quantity, Unit Price
  BUTTON: Add Item (adds new item to the items table)

// Pricing and Payment Details Section
```

```
DISPLAY "Pricing and Payment Details" header
SHOW total cost, VAT, payment status, payment method, and invoice number
BUTTON: Pay Now (initiates payment process)

// Additional Information Section
DISPLAY "Additional Information" header
CREATE table for order notes and order history
SHOW current order notes
BUTTON: Update Notes (updates order notes)

// JavaScript Functionality
ON Add Item button click:
   GET values from item input fields
   CALCULATE total price (Quantity * Unit Price)
   ADD new item to items table
   CLEAR input fields

ON Search button click:
   GET search value from search-bar
   FILTER orders in orders table based on Order ID

ON Save Changes button click:
   GET updated information from edit input fields
   UPDATE displayed customer/supplier information
   CLOSE edit modal
```

END Program OrdersPage

Inventory:



Inventory

Item Name	Category	Quantity	Unit	Last Updated	Status	Action
Corn seeds	Seeds	500	kg	15/9/2024	In Stock	
Fertilizer	Supplies	1000	kg	15/9/2024	In Stock	
Pesticide	Supplies	1000	l	15/9/2024	Low Stock	
Tractors	Equipment	4	-	15/9/2024	Maintained	
Sensors	Equipment	10	-	15/9/2024	Maintained	

© 2024 FMS Inventory. All Rights Reserved.
Contact us: support@example.com

Explanation:

Program Design

The FMS Inventory webpage is designed to manage inventory for a farming management system. It includes features like searching, adding items, notifications, generating reports, and pagination. The user interface is built using HTML and styled with Bootstrap, while JavaScript handles dynamic functionalities like modal forms and event handling.

Key Components:

1. Sidebar Navigation:
   - Contains links to different sections like Dashboard, Crops, Livestock, Orders, etc. ○ Organized as a vertical menu using Bootstrap's flex layout.
2. Main Content:
   - Includes an inventory table where the list of items is displayed.

- o Allows users to search through inventory, add new items via a modal form, and export the inventory.
- o Supports pagination for navigating through multiple inventory pages.

3. Notifications:
   - o A notification area shows messages related to inventory updates. o Clicking the notification button reveals a dropdown where the notifications will be displayed.
4. Add New Item Modal:
   - o A form allows users to add new items by providing item details such as name, category, quantity, unit, and status.
5. Language Support:
   - o The webpage is multilingual, allowing users to select their preferred language from a dropdown menu.
6. Inventory Table:
   - o Displays current inventory with columns for item name, category, quantity, unit, last updated, and status. o Users can interact with the inventory through buttons (e.g., adding new items).
7. Footer:
   - o Contains copyright information and a support email link.

```
START


1. SETUP page layout using Bootstrap:

- Create a sidebar with links to different sections.

- Create a main content area with inventory table and action buttons.


2. SETUP Sidebar:

  - Add links for sections like Dashboard, Crops, Livestock, Orders, etc.


3. IN main content area:

- Create a header with search bar and user info (including language selection).

- Add buttons for exporting inventory and adding new items.
```

4. INVENTORY Section:

- Add "Inventory" title.

- Add a search bar for filtering inventory items.

- Create a button for exporting data.

- Create a button that triggers a modal to add new items.

5. ADD New Item Modal:

- Include form fields for item name, category, quantity, unit, and status.

- Submit form to add new item.

6. INVENTORY Table:

- Add a table to display current inventory with columns for:

- Item Name

- Category

- Quantity

- Unit

- Last Updated

- Status

- Action (e.g., Edit/Delete)

7. ADD Pagination Controls:

  - Include buttons for navigating between pages.

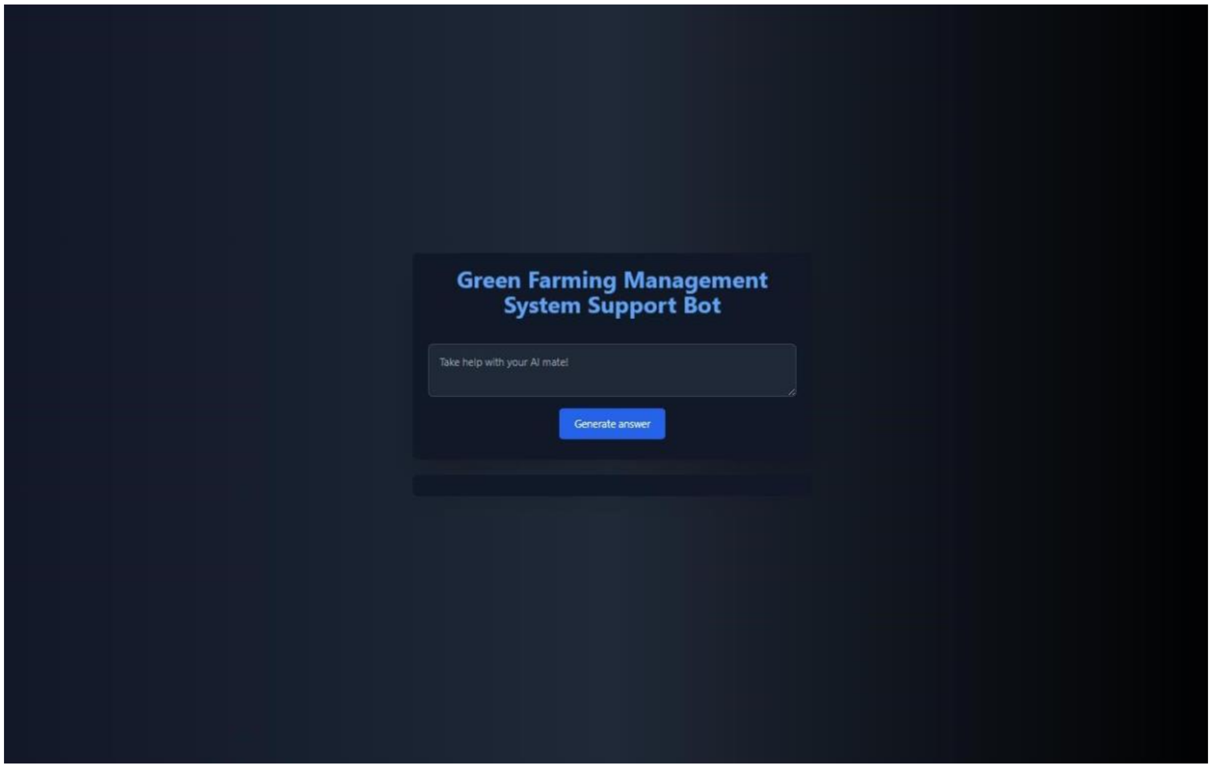
8. ADD Notification Section:

- Create a notification area to display inventory-related updates.

- Button to toggle viewing notifications.

```
9. ADD Language Support:
- Provide dropdown to select different languages.
- Display page text based on selected language.

10. ADD Footer:
   - Include copyright information and contact support email.

11. HANDLE Button Events:
          - On "Add New Item" click, show the modal.
          - On "Export" click, trigger export functionality.
          - On pagination button click, load next/previous page of inventory.

END
```

Support:


Green Farming Management System Support Bot
Take help with your AI mate!
Generate answer

Explanation:

Program Design

The provided HTML code represents a simple webpage structure for the Helpmate AI ChatBot. It serves as the entry point for a web application where users can interact with an AI chatbot to get answers to their queries.

Key Components:

1. HTML Structure:
   - The document is structured with a <!doctype html> declaration, indicating that it is an HTML5 document. ○ The <head> section contains metadata about the webpage, such as the title, description, keywords, author, and Open Graph properties for social media sharing.
2. Meta Tags:
   - The meta tags provide information about the content of the page and its author. This includes:
     - Charset: Specifies the character encoding for the document (UTF-8).

    - Viewport: Ensures the page is responsive and scales well on different devices.
    - Description and Keywords: Used for SEO to improve discoverability on search engines.
    - Open Graph Tags: Helps in rendering a preview when shared on social media.
3. Favicon:
    - The <link rel="icon"> tag specifies a favicon for the webpage, which is displayed in the browser tab.
4. Root Element:
    - The <div id="root"></div> serves as a mounting point for the React application, where the main content will be rendered.
5. JavaScript:
    - The script tag loads the main JavaScript file (main.jsx) that likely initializes the React application and renders the chatbot interface.

```
START


1. SETUP HTML Document:

  - Declare the document type as HTML5.


2. HEAD Section:

  -    SET character encoding to UTF-8.

  -    SET favicon to "Helpmate-AI.png".

  -    SET viewport to ensure responsiveness.

  -    ADD meta description for SEO.

  -    ADD keywords for SEO.

  -    SET author to "Ashutosh Singh".

  -    ADD Open Graph properties for social media sharing:

  -    Set image URL.

  -    Set page URL.
```

- SET page title to "Green Farming Management System Support Bot".

3. BODY Section:

   - CREATE a root div with id="root" to mount the React application.

4. SCRIPT Section:

   - LOAD the main JavaScript file (`/src/main.jsx`) using module type.

END

 Separation of Concerns: The use of HTML for structure, metadata for SEO, and a dedicated JavaScript file for application logic promotes clean separation and maintainability.

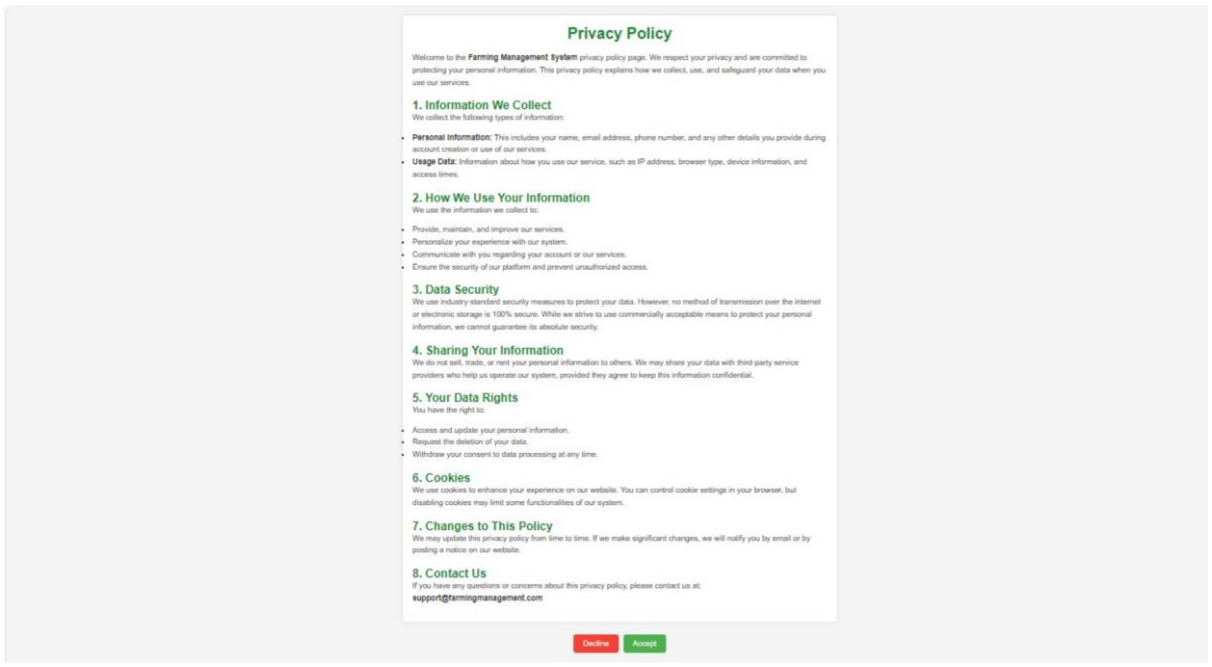
 Responsive Design: The inclusion of the viewport meta tag ensures that the application is user-friendly across various devices.

 SEO Optimization: Adding meta descriptions and keywords enhances the visibility of the application in search engine results, making it easier for users to discover the chatbot.

 React Integration: The structure indicates that the application is likely built with React, a popular JavaScript library for building user interfaces, allowing for dynamic and responsive interaction with the chatbot. The root div serves as the mounting point for the React component, where all chat functionalities will be handl

Settings:

Terms of service:



Privacy Policy

Welcome to the **Farming Management System** privacy policy page. We respect your privacy and are committed to protecting your personal information. This privacy policy explains how we collect, use, and safeguard your data when you use our services.

1. Information We Collect

We collect the following types of information:

- **Personal Information:** This includes your name, email address, phone number, and any other details you provide during account creation or use of our services.
- **Usage Data:** Information about how you use our service, such as IP address, browser type, device information, and access times.

2. How We Use Your Information

We use the information we collect to:

- Provide, maintain, and improve our services.
- Personalize your experience with our system.
- Communicate with you regarding your account or our services.
- Ensure the security of our platform and prevent unauthorized access.

3. Data Security

We use industry-standard security measures to protect your data. However, no method of transmission over the internet or electronic storage is 100% secure. While we strive to use commercially acceptable means to protect your personal information, we cannot guarantee its absolute security.

4. Sharing Your Information

We do not sell, trade, or rent your personal information to others. We may share your data with third-party service providers who help us operate our system, provided they agree to keep this information confidential.

5. Your Data Rights

You have the right to:

- Access and update your personal information.
- Request the deletion of your data.
- Withdraw your consent to data processing at any time.

6. Cookies

We use cookies to enhance your experience on our website. You can control cookie settings in your browser, but disabling cookies may limit some functionalities of our system.

7. Changes to This Policy

We may update this privacy policy from time to time. If we make significant changes, we will notify you by email or by posting a notice on our website.

8. Contact Us

If you have any questions or concerns about this privacy policy, please contact us at:
support@farmingmanagement.com

Explanation:

Program Design

The provided HTML code represents a Privacy Policy page for a Farming Management System. This page informs users about how their personal information is collected, used, and protected when they interact with the system.

Key Components:

1. HTML Structure:
   - The document begins with a <!DOCTYPE html> declaration, indicating that it is an HTML5 document.
2. Head Section:
   - The <head> section includes metadata, such as character encoding, viewport settings for responsiveness, and the page title.
   - It also includes links to external resources:
     - Google Fonts for typography.
     - A CSS file (privacy.css) for styling.
3. Body Section:
   - The main content is wrapped in a <div class="container">, which holds all the policy text, including headings and paragraphs detailing the privacy policy.

- o The policy is organized into sections with headers (<h2>) and lists (<ul> and <li>) for clarity and easy navigation.

4. User Rights and Information:
   - o The policy outlines the types of information collected, how it is used, data security measures, rights of the users, and cookie usage.
   - o It includes a section on changes to the policy and provides contact information for user inquiries.
5. Action Buttons:
   - o Two buttons, Accept and Decline, allow users to provide their consent or refusal regarding the privacy policy. o A JavaScript file (privacy.js) is included at the end of the body to handle user interactions with the buttons.

```
START


1. SETUP HTML Document:

  - Declare the document type as HTML5.


2. HEAD Section:

  -    SET character encoding to UTF-8.

  -    SET viewport for responsive design.

  -    SET page title to "Privacy Policy - Farming Management System".

  -    LINK to Google Fonts for typography.

  -    LINK to external CSS stylesheet ("privacy.css") for styling.


3. BODY Section:

  -    CREATE a container div to hold the policy content.

  -    ADD a header ("Privacy Policy") and an introductory paragraph explaining the policy.

  -    CREATE sections for the privacy policy with headings and content:

  -    Section 1: Information We Collect (list personal and usage data).

  -    Section 2: How We Use Your Information (list usage purposes).
-       Section 3: Data Security (explain security measures).
-       Section 4: Sharing Your Information (state sharing policy).
-       Section 5: Your Data Rights (list user rights).
-       Section 6: Cookies (explain cookie usage).
-       Section 7: Changes to This Policy (describe update process).
-       Section 8: Contact Us (provide contact information).

4. BUTTONS Section:
  - CREATE buttons for user actions ("Decline" and "Accept").

5. SCRIPT Section:
  - LINK to JavaScript file ("privacy.js") for handling button interactions.

END
```

- User-Friendly Structure: The privacy policy is organized into clear sections, making it easy for users to read and understand their rights and the information that is being collected.

- Responsive Design: The inclusion of the viewport meta tag and external CSS ensures that the page is responsive and visually appealing across various devices.

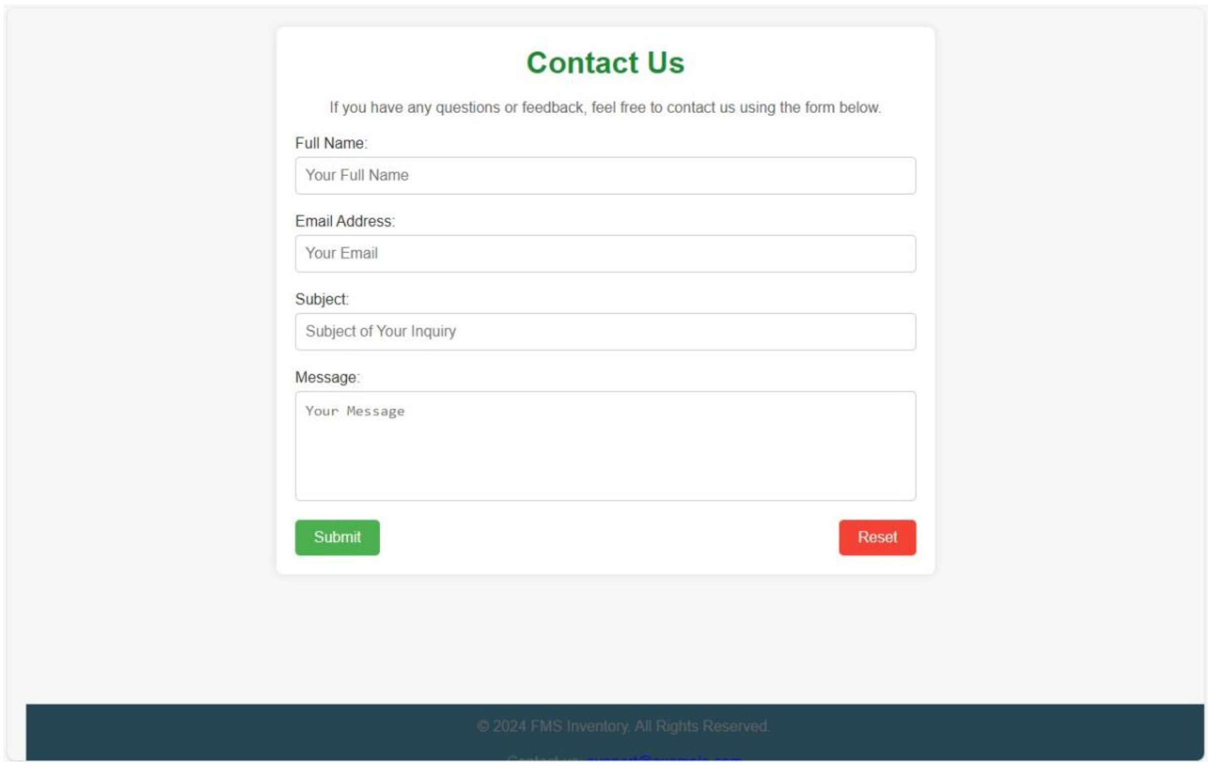
- Use of Lists: Lists are employed to break down complex information into manageable pieces, enhancing readability.

- Contact Information: Providing a contact email fosters transparency and allows users to reach out with questions or concerns, promoting trust.

- Consent Mechanism: The inclusion of Accept and Decline buttons gives users control over their data, which is essential for compliance with privacy regulations.

- JavaScript Integration: The inclusion of a JavaScript file allows for potential interactivity, such as handling the user's consent choice and possibly redirecting them based on their selection.

Contact us:



Contact Us

If you have any questions or feedback, feel free to contact us using the form below.

Full Name: Your Full Name

Email Address: Your Email

Subject: Subject of Your Inquiry

Message: Your Message

© 2024 FMS Inventory. All Rights Reserved.

Explanation:

Program Design

The provided HTML code represents a Contact Us page for a Farming Management System. This page enables users to reach out with questions or feedback through a contact form.

Key Components:

1. HTML Structure:
   - The document starts with a <!DOCTYPE html> declaration, indicating that it is an HTML5 document.
2. Head Section:
   - The <head> section includes metadata, such as character encoding, viewport settings for responsiveness, and the page title.
   - It includes links to external resources:
     - Google Fonts for typography.
     - A CSS file (contact.css) for styling.
3. Body Section:

- The main content is wrapped in a <div class="container">, which holds the header, introductory text, and contact form.
- The contact form includes fields for the user's name, email address, subject, and message. Each field has a corresponding label for accessibility.

4. Form Functionality:
   - The form has the following components:
     - Input Fields: Users enter their full name, email address, subject, and message.
     - Form Buttons: Two buttons are provided—Submit to send the form and Reset to clear the form fields.
   - The required attribute ensures that all fields must be filled out before submission.
5. Footer Section:
   - A footer provides copyright information and a contact email link for additional support.
6. JavaScript Integration:
   - A JavaScript file (contact.js) is included at the end of the body for handling form submission and any interactive elements.

```
START

1. SETUP HTML Document:
  - Declare the document type as HTML5.

2. HEAD Section:
- SET character encoding to UTF-8.
- SET viewport for responsive design.
- SET page title to "Contact Us - Farming Management System".
- LINK to Google Fonts for typography.
- LINK to external CSS stylesheet ("contact.css") for styling.

3. BODY Section:
```

- CREATE a container div to hold the contact form content.

- ADD a header ("Contact Us") and an introductory paragraph encouraging user interaction.

4. FORM Section:

- CREATE a form element with an ID of "contactForm".

- ADD the following input fields:

- Field for full name:

- LABEL for "Full Name" linked to input.

- INPUT of type text for name with placeholder and required attribute.

- Field for email address:

- LABEL for "Email Address" linked to input.

- INPUT of type email for email with placeholder and required attribute.

- Field for subject:

- LABEL for "Subject" linked to input.

- INPUT of type text for subject with placeholder and required attribute.

- Field for message:

- LABEL for "Message" linked to textarea.

- TEXTAREA for message input with rows and required attribute.

- CREATE a div for form buttons:

- ADD a Submit button of type "submit" with an ID "submit".

- ADD a Reset button to clear the form.

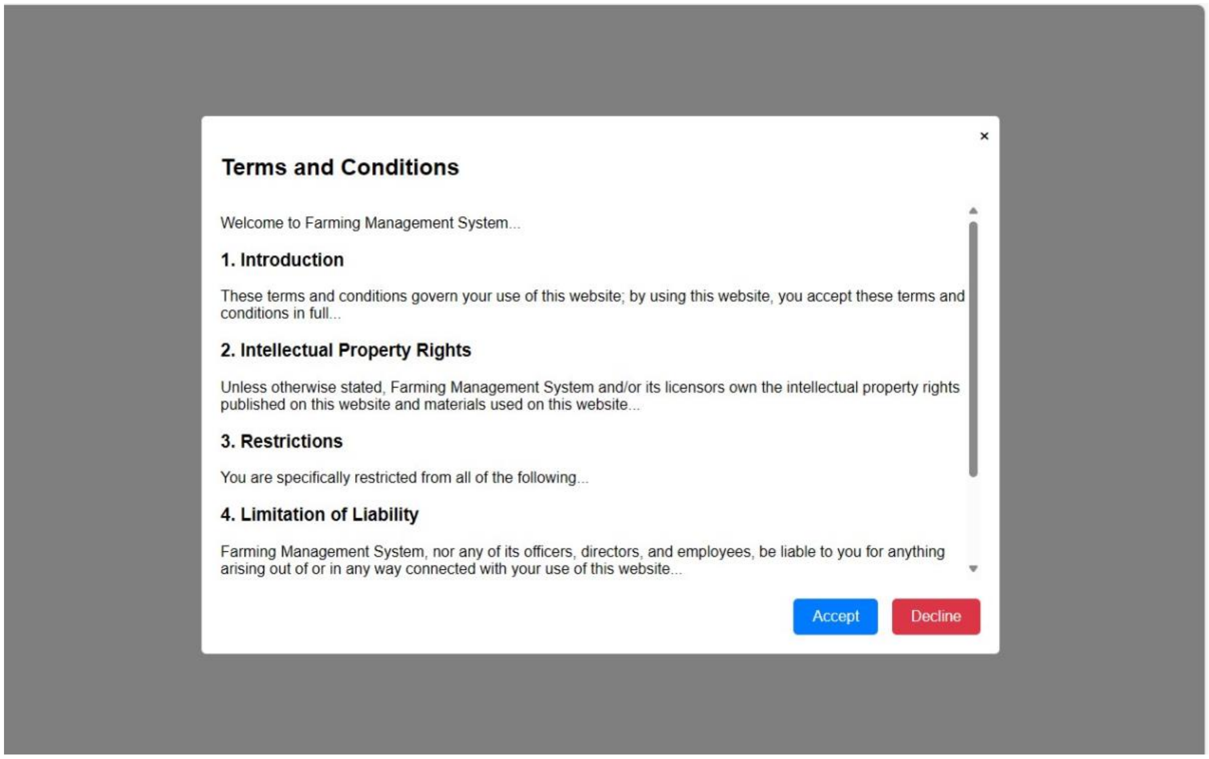
5. FOOTER Section:

- CREATE a footer with copyright information and a contact email link.

Explanation of Design Choices

- User-Friendly Layout: The page is designed to be intuitive, with clear labels for each input field, making it easy for users to provide their information.
- Responsive Design: The use of the viewport meta tag ensures that the page adapts to various device sizes, enhancing user experience.
- Accessibility: The for attribute in labels is associated with the corresponding input fields, improving accessibility for screen readers.
- Form Validation: The required attribute on the input fields enforces that all fields must be completed before the form can be submitted, reducing the chances of incomplete submissions.
- Call to Action: The introductory text invites users to engage with the system, enhancing user interaction.
- Contact Information: The footer provides additional support by displaying a contact email, reinforcing user confidence in the system's support structure.
- JavaScript Integration: The inclusion of a JavaScript file allows for handling form submission, validation, and any additional interactive elements, enhancing the overall functionality of the page.

Terms and condition:



Terms and Conditions

Welcome to Farming Management System...

1. Introduction

These terms and conditions govern your use of this website; by using this website, you accept these terms and conditions in full...

2. Intellectual Property Rights

Unless otherwise stated, Farming Management System and/or its licensors own the intellectual property rights published on this website and materials used on this website...

3. Restrictions

You are specifically restricted from all of the following...

4. Limitation of Liability

Farming Management System, nor any of its officers, directors, and employees, be liable to you for anything arising out of or in any way connected with your use of this website...

Explanation:

Program Design

The provided HTML code implements a Terms and Conditions modal along with a user registration form for the Farming Management System. This design enables users to review the terms and conditions before they can complete their registration.

Key Components:

1. HTML Structure:
   - The document starts with a <!DOCTYPE html> declaration for HTML5.
   - The <head> section includes character encoding, viewport settings for responsiveness, the page title, and a link to an external CSS file (tnc.css) for styling.
2. Terms Modal:
   - A <div> with an ID of terms-modal contains the terms and conditions content.
   - Within this modal:
     - A close button (<span>) allows users to dismiss the modal.
     - A heading and sections outline the terms and conditions, including:
     - Introduction
     - Intellectual Property Rights
     - Restrictions
     - Limitation of Liability
     - Indemnification
   - Two buttons—Accept and Decline—enable users to agree or disagree with the terms.
3. Registration Form:
   - A separate <div> with an ID of content contains a form for user registration.
   - The form includes:
     - Input fields for Username, Email, and Password.
     - A dropdown menu for selecting a Role (User or Admin).
     - A checkbox to confirm acceptance of the terms and conditions.
     - A link to the Terms and Privacy policy. ▪ A Register button to submit the form.
4. JavaScript Integration:
   - A JavaScript file (tnc.js) is linked at the end of the body, which handles the modal functionality and form interactions.

```
START


1. SETUP HTML Document:
   - Declare the document type as HTML5.


2. HEAD Section:
- SET character encoding to UTF-8.
- SET viewport for responsive design.
- SET page title to "Terms and Conditions".
- LINK to external CSS stylesheet ("tnc.css") for styling.


3. BODY Section:
- CREATE a modal for Terms and Conditions:
- DIV with ID "terms-modal" and class "modal".
- DIV with class "modal-content".
- ADD close button with ID "close-btn" for dismissing the modal.
- ADD heading "Terms and Conditions".
- CREATE a section with the terms text:
- INCLUDE paragraphs and headings detailing the terms and conditions.
- CREATE a div for modal buttons:
- ADD Accept button with ID "accept-btn".
- ADD Decline button with ID "decline-btn".


4. REGISTRATION FORM Section:
- CREATE a div with ID "content" and class "content hidden".
- CREATE a form that submits to "signup.php" using POST method.
- ADD form-group for username:
- LABEL for "Username" linked to input field.
- INPUT of type text for username with required attribute.
- ADD form-group for email:
- LABEL for "Email" linked to input field.
- INPUT of type email for email with required attribute.
- ADD form-group for password:
- LABEL for "Password" linked to input field.
- INPUT of type password for password with required attribute.
- ADD form-group for role:
- LABEL for "Role" linked to select field.
- SELECT dropdown for user roles (User, Admin) with required attribute.
- ADD paragraph linking to Terms and Privacy.
- ADD checkbox for terms acceptance with ID "terms" and label.
- ADD a submit button to register the user.

5. SCRIPT Section:
  - LINK to JavaScript file ("tnc.js") for handling modal and form interactions.

END
```

▯ User Experience: The modal window provides a focused view of the terms and conditions, preventing users from navigating away from the registration process. This enhances user engagement and ensures users read the terms before accepting.

▯ Responsive Design: The inclusion of the viewport meta tag ensures that the page is optimized for various screen sizes, making it accessible on mobile devices.

▯ Accessibility: The use of labels linked to input fields improves accessibility for users relying on screen readers, ensuring that all fields are clearly defined.

- Form Validation: The required attribute on input fields ensures that users fill out essential information before submitting, reducing errors in form submission.

- Confirmation of Terms: The checkbox for accepting the terms is crucial to ensure that users agree to the conditions of use, adding a layer of legal protection for the service provider.

- Clear Call to Action: The registration button encourages user engagement and leads users to create an account once they have accepted the terms.

- JavaScript Integration: Linking to a JavaScript file allows for dynamic interactions, such as showing/hiding the modal and handling user inputs effectively.

4.7 Interface Design

Designing an interface for a **Farm Management System** involves ensuring a user-friendly and intuitive experience for managing farm activities, such as livestock, crops, fields, inventory, and financial records. The system allows users (farmers, managers, or workers) to input and retrieve data efficiently. Here's an approach for the **Menu Interface**, **Input Design**, and **Output Design** for the system.

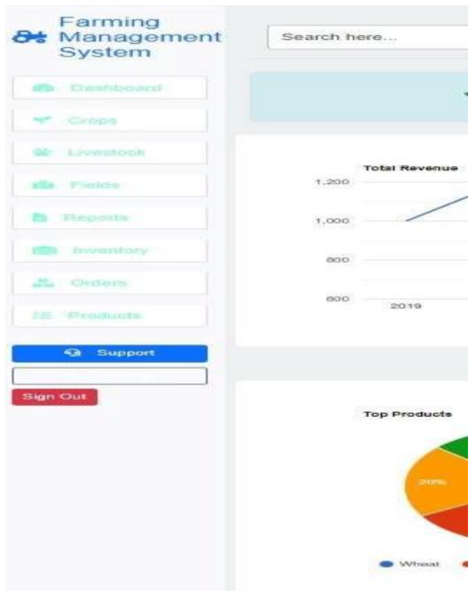
1. Menu Interface Design

The **Menu Interface** is the main navigational component that give users access to the different functionalities of the farm management system.

Features to Include in the Menu:

- **Dashboard**: Summary of all farm activities.
- **Livestock Management**: Add, edit, or view livestock details.
- **Crop Management**: Add, track, and view crop details.
- **Field Management**: Manage fields and monitor crop rotation.
- **Labor & Inventory**: Manages and monitor equipment.
- **Expenses**: Record income/expenses and view reports.
- **Reports**: Generate and view summaries of farm data.
- **Orders:** Manages and track the orders made buy the customers
- **User Management**: Manage users and roles (admin, workers, etc.).
- **Products**: All The record of the products that are available in store.

Dashboard Menu:



Menu Interface in a GUI (Example):

- **Dashboard**: A homepage that shows critical stats (total livestock, field utilization, weather forecasts, etc.).
- **Livestock**: Submenu with buttons like:
  - Add Livestock
  - View Livestock
  - Manage Livestock Records
- **Crops**: Submenu with buttons like:
  - Add Crop
  - View Crop Status
  - Crop History
- **Labor & Inventory:** Submenu with options for monitoring and tracking machinery usage.

- **Fields:** show all the active and available fields.

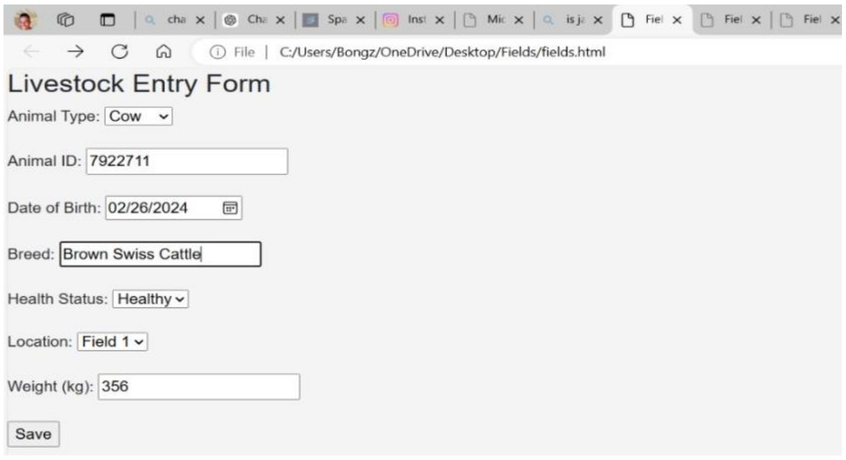
2. Input Design

The **Input Design** is essential for entering data about the farm's various operations. The design should be clear, concise, and user-friendly, ensuring that all required fields are covered.

Input Forms to Include:

1. **Livestock Entry Form**

- Animal Type (dropdown)
- Animal ID (text field)
- Date of Birth (date picker)
- Breed (text field)
- Health Status (dropdown or text)
- Location (dropdown for fields or barns)
- Weight (numeric input)
- [Save] Button


C:/Users/Bongz/OneDrive/Desktop/Fields/fields.html

Livestock Entry Form

Animal Type: Cow

Animal ID: 7922711

Date of Birth: 02/26/2024

Breed: Brown Swiss Cattle

Health Status: Healthy

Location: Field 1

Weight (kg): 356

Save

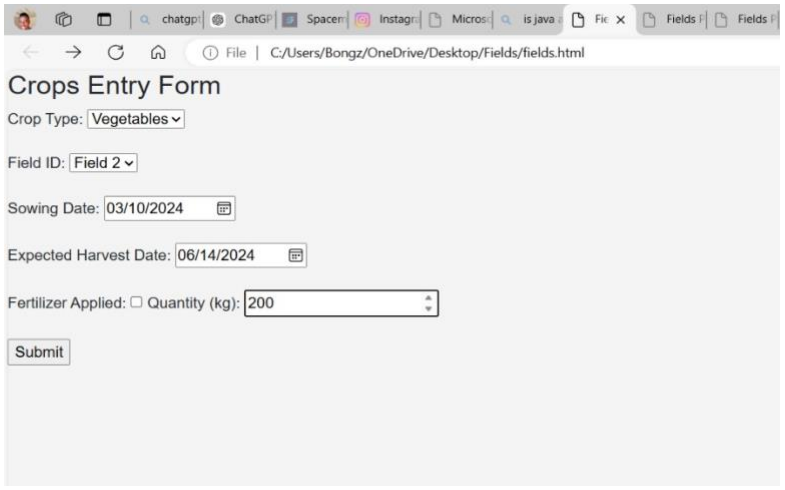
2. **Crop Entry Form**

- Crop Type (dropdown)
- Field ID (dropdown)
- Sowing Date (date picker)
- Fertilizer Applied

  (checkbox with input for

  quantity)
- [Submit] Button
- Expected Harvest

  Date(date picker)



Crops Entry Form

Crop Type: Vegetables

Field ID: Field 2

Sowing Date: 03/10/2024

Expected Harvest Date: 06/14/2024

Fertilizer Applied: ☐ Quantity (kg): 200

Submit

3. **Field Management Input Form** o

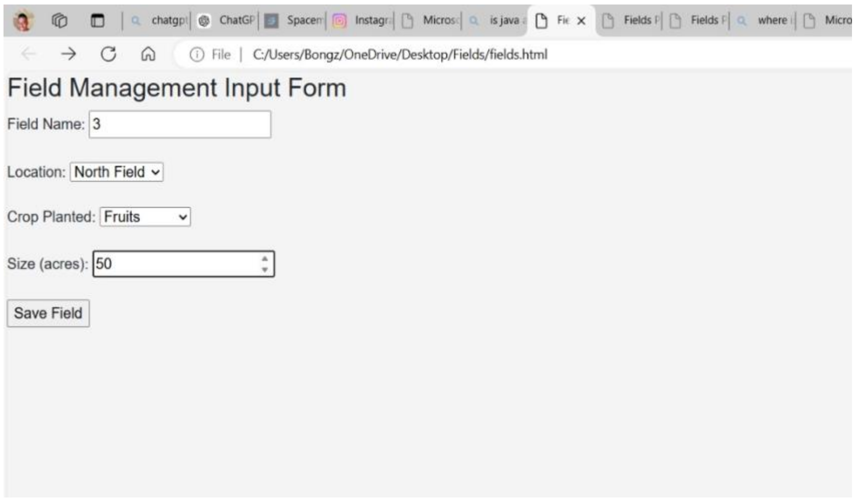
Field Name (text input) o Location

(dropdown for area/coordinates) o Crop

Planted (dropdown for crop) o Size

(numeric input for acreage/hectares)

o [Save Field] Button



C:/Users/Bongz/OneDrive/Desktop/Fields/fields.html

Field Management Input Form

Field Name: 3

Location: North Field

Crop Planted: Fruits

Size (acres): 50

Save Field

4. **Labor & Inventory Input Form** o

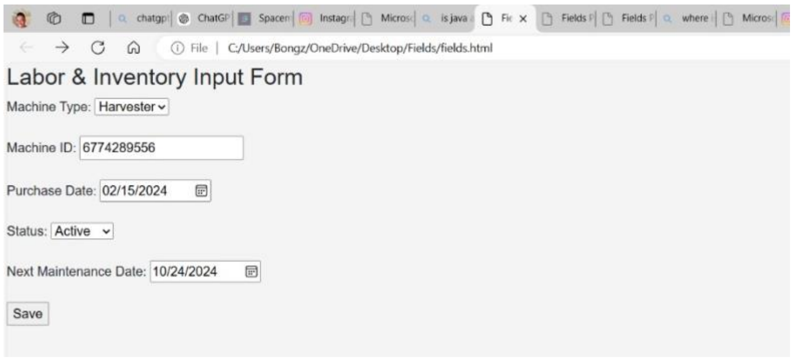
Machine Type (dropdown) o Machine ID

(text) o Purchase Date (date picker) o

Status (dropdown for Active/Inactive) o

Next Maintenance Date (date picker) o

[Save] Button



C:/Users/Bongz/OneDrive/Desktop/Fields/fields.html

Labor & Inventory Input Form

Machine Type: Harvester

Machine ID: 6774289556

Purchase Date: 02/15/2024

Status: Active

Next Maintenance Date: 10/24/2024

Save

Key considerations:

- **Dropdown menus** for predefined selections like animal types, fields, or health statuses.
- **Text fields** for customizable inputs like IDs or breed types.
- **Numeric inputs** for weight, cost, or size entries.

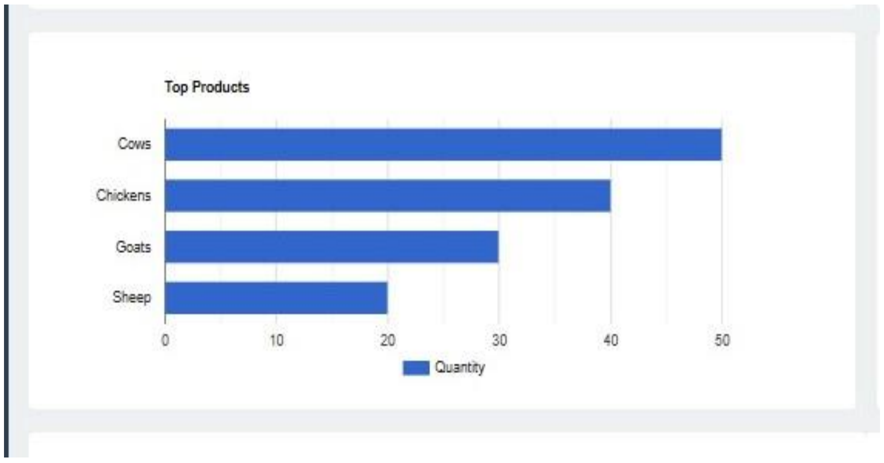
3. Output Design

The **Output Design** focuses on displaying farm data in a structured and easy-to-read format. This can include tables, charts, reports, and summaries. The output design must allow users to **view, filter, and export data**.

Key Output Features:

1. **Livestock List (Bar Graph)** o Columns: Animal ID, Type, Breed, Age, Health Status, Location, Weight. o Options to **filter** by type, health status, or age. o

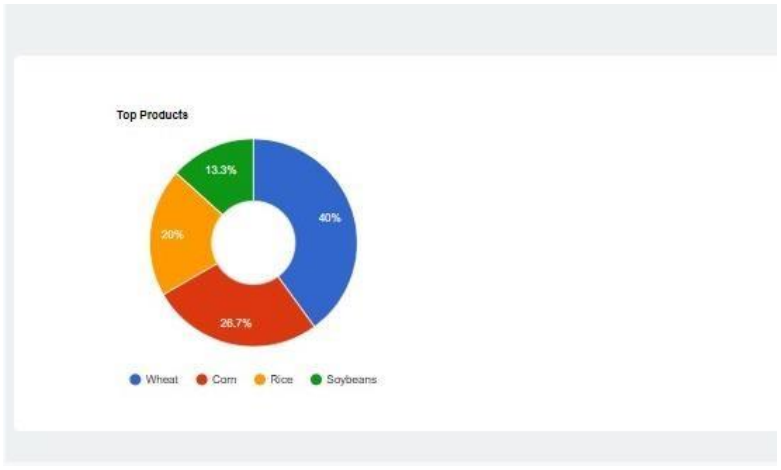
   Buttons for viewing or editing individual animal records.



2. **Crop Summary (Pie Chart)**

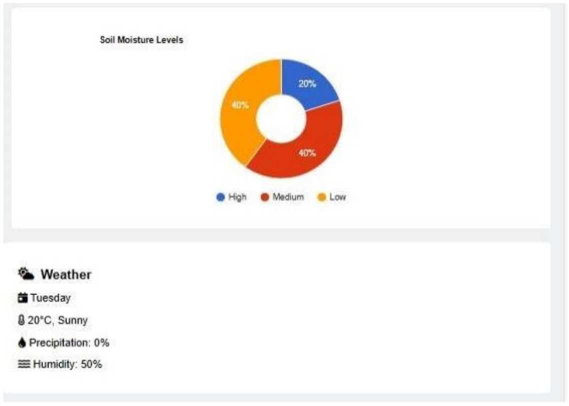
   o List of current crops in fields.

   o Filter options by crop type or field. o Harvest date projections.

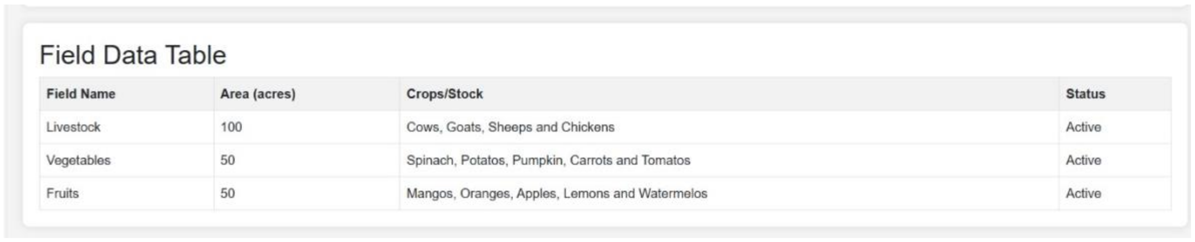


3. **Field Management Output**

   o Table listing fields with columns for Field Name, Size, Crop Planted, Status (Planted, Empty). o Field utilization charts (e.g., pie chart showing soil moisture level). o Field Data table



Weather
Tuesday
20°C, Sunny
Precipitation: 0%
Humidity: 50%



Field Data Table

Field Name	Area (acres)	Crops/Stock	Status
Livestock	100	Cows, Goats, Sheeps and Chickens	Active
Vegetables	50	Spinach, Potatos, Pumpkin, Carrots and Tomatos	Active
Fruits	50	Mangos, Oranges, Apples, Lemons and Watermelos	Active

4. **Labor & Inventory List**

   o Table with columns for Machine ID, Type, Status, Last Maintenance, Next Maintenance Date.

   o Sort or filter by machine type or status.

5. **Financial Reports** o Monthly or yearly report summarizing farm expenses and income.

   o Filters to generate reports by livestock sales, crop revenue, or equipment costs.

Graphical Output (Dashboard):

- **Pie Chart** for field usage (planted vs. free fields).
- **Bar Graph** for crop yield over seasons.
- **Line Graph** for tracking machinery usage over time.

Key considerations for output:

- **Tables** to provide detailed data views.
- **Charts** (bar, pie, line) to visualize data trends.
- **Export options** (PDF, CSV) for reporting and record-keeping.

Summary:

1. **Menu Interface**: A navigational component with sections for livestock, crops, fields, Inventory, and reports.

2. **Input Design**: Form-based inputs for managing data on livestock, crops, fields, and machinery. Should be intuitive and structured with dropdowns, text fields, and date pickers.

3. **Output Design**: Structured data display using tables, charts, and reports. Output should be filterable and exportable.

This structure ensures that the farm management system will be easy to use, with streamlined data entry and comprehensive reporting capabilities.

4.8 Security Backup Design

This security backup design outlines the strategies and procedures to protect the Farming Management System (FMS) data, ensure business continuity, and safeguard against potential data loss or security breaches.

1. **Backup Strategy**

1.1 Data Classification

Classify FMS data into categories based on criticality:

- Critical: User accounts, financial records, crop/livestock data
- Important: Historical yield data, weather records
- Non-critical: Temporary logs, cache files

1.2 Backup Frequency

- Critical data: Daily incremental backups, weekly full backups
- Important data: Weekly full backups
- Non-critical data: Monthly full backups

1.3 Backup Types

1. Full Back up: Complete copy of all data
2. Incremental Backup: Only changes since the last backup
3. Differential Backup: All changes since the last full backup

1.4 Retention Policy

- Daily backups: Retain for 30 days
- Weekly backups: Retain for 3 months
- Monthly backups: Retain for 1 year
- Yearly backups: Retain for 7 years

2. Backup Infrastructure

2.1 On-premises Backup

- Primary backup server with RAID 6 configuration
- Network-attached storage (NAS) for local backups

2.2 Cloud Backup

    - Utilize a reputable cloud service provider (e.g., AWS S3, Azure Blob Storage, Google Cloud Storage)
    - Implement multi-region replication for disaster recovery

2.3 <u>Offline Backup</u>

- Weekly offline backups stored in a secure, off-site location

3. Backup Process

3.1. Initiate automated backup process during off-peak hours
3.2. Compress and encrypt backups before transmission or storage
3.3. Verify backup integrity through automated checksum validation
3.4. Log all backup activities and results

4. Security Measures

4.1 <u>Encryption</u>

- Encrypt data at rest using AES-256 encryption
- Encrypt data in transit using TLS 1.3

4.2 <u>Access Control</u>

- Implement role-based access control (RBAC) for backup systems
- Use multi-factor authentication (MFA) for admin access
- Regularly audit and review access logs

4.3 <u>Network Security</u>

- Isolate backup infrastructure on a separate VLAN
- Use firewalls to restrict access to backup systems
- Implement intrusion detection and prevention systems (IDS/IPS)

5. Testing and Verification

5.1 <u>Regular Testing</u>

- Conduct monthly restore tests for critical data
- Perform quarterly full system restore tests
- Annual disaster recovery simulation

5.2 <u>Verification Procedures</u>

- Automated integrity checks after each backup
- Manual spot checks of restored data

    - Document and review all test results

6. Disaster Recovery

6.1 Recovery Time Objective (RTO)

- Critical systems: 4 hours
- Non-critical systems: 24 hours

6.2 Recovery Point Objective (RPO)

- Critical data: 1 hour
- Important data: 24 hours
- Non-critical data: 1 week

6.3 Disaster Recovery Plan

1. Assess the extent of data loss or system failure
2. Initiate the recovery process based on the disaster severity
3. Restore critical systems and data first
4. Verify data integrity post-restoration
5. Gradually restore non-critical systems and data
6. Conduct a post-incident review and update the DR plan if necessary

7. Documentation and Training

- Maintain detailed documentation of all backup and recovery procedures
- Conduct regular training sessions for IT staff on backup and recovery processes
- Keep an up-to-date contact list for key personnel and vendors

8. Compliance and Auditing

- Ensure backup processes comply with relevant data protection regulations (e.g., GDPR, CCPA)
- Conduct annual internal audits of the backup system
- Perform biennial external audits by a certified third party

9. Continuous Improvement

- Regularly review and update the backup strategy based on changing business needs and technological advancements
- Stay informed about new threats and adjust security measures accordingly
- Solicit feedback from stakeholders and incorporate improvements into the backup process