A picture containing text, plant, screenshot, grass

Description automatically generated

Hands touching green leaves in a garden

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

A black text on a white background

Description automatically generated with low confidence

Installation:

The design and build process for a system that connects farmers and potential customers can be broken down into several phases, including:

**1. Pre-development requirements:** Before starting the design and build process, determine the system's hardware, software, and network needs as well as its functional and non-functional requirements.

**2. Development phases:**

a. Software installation: Install the software components that are required to build the system, such as web servers, database management systems, and development frameworks.

b. Front-end and back-end development: Develop the user interface and server-side components of the system, including the logic for processing orders, managing farmer and customer data, and handling payments.

c. Testing and validation: Test the system thoroughly to ensure that it meets the requirements and functions as expected. This includes both unit testing and integration testing, as well as user testing to ensure that the system is user-friendly and intuitive.

**3. Post-Development Tasks:**

a. Deployment: Deploy the system to a production environment where farmers and customers can access it. Monitor the system to ensure that it continues to function properly and make updates as necessary.

b. post-installation configuration: Configure the platform settings, such as security options, user access, and system integrations, after the program has been deployed.

c. Training and support: Provide training and assistance to help end users use the platform and resolve any issues that might come up. This may include developing user guides, hosting training sessions, and offering customer support.

d. Maintenance and support: Provide ongoing maintenance and support to ensure that the system continues to function properly and meets the needs of its users. This may include fixing bugs, making updates to the system, and providing customer support to users who have questions or issues.

# Transition:

The transition phase is crucial for helping an organization resume operations after a security event including data breaches and security loss. The most critical transition is from the incident response phase to the recovery phase. During the transition period, a number of important actions are carried out, such as:

* Setting revenue objectives and customer acquisition goals that are consistent with the company's overall vision and strategy can help an app develop clear goals and metrics to monitor progress and success.
* As the company expands, the team may also do so by adding new personnel, elevating team members to senior positions, or contracting out some duties to outside service providers.
* The app may require further financing as it expands in order to support new developments. This might entail obtaining funding from angel investors, venture capitalists, or other kinds of funding.
* Testing of remedial measures: The business assesses the effectiveness of the corrective measures to ensure that they completely solve the discovered security weaknesses and do not create new vulnerabilities.
* Analysis of the damage The first stage of the transition phase is evaluating the damage brought on by the security event. Finding out what data was stolen, how much damage was done, and whether any new vulnerabilities had been made public are all necessary for this.
* Once the harm has been assessed, the organization determines any gaps or weak points in its security measures that were exploited by the attacker. To improve its security posture, the company makes use of this information.in order to stop upcoming violations

In the end, the transition stage for a company is a key time of development and change that demands careful planning, execution, and continuing modification as the organization develops.

# Training:

Training must be a part of the implementation of any new system. End-user training is essential to ensuring that users can make the most use of the system and a successful adoption of the system. Especially in a system where illiterates or people who are not familiar with technology or who feel uneasiness to use mobiles or apps, in this case farmers are one example. Training has a direct impact on an organization's productivity and performance in general as the product flows and features will be understood in a better way. The following are the primary components of the training curriculum:

**Training Needs Assessment:** Identifying the end-users' training needs is the initial step. To determine the degree of knowledge and skills necessary to operate the new system, questionnaires, interviews, and focus groups may be used.

**Analyze & Understand the Training Needs:** The first step is to determine the training requirements of the end users. Who are our end users? (Admin, Customers, Farmers). Questionnaires, interviews, and focus groups can be used to determine the level of expertise and knowledge required to operate the new system.

**Preparing Content for Training:** In view of the assessment of the preparation needs, preparing materials will be made. The information will be created in a variety of formats, such as videos, tutorials, webinars, and manuals, to accommodate a wide range of learning styles. And for farmers they wil be provided with manual guides in their native language to make them understand the application or the flows in an application.

**Delivery of training:** The training will be provided in a variety of formats, product videos and videos showing how to use the application and its features, and a person will help farmers understand how to use the app for inventory management and other functionalities. Delivery of training would be done in video formats, product manuals, in-person explanations on fields.

**Schedule for training:** A training program will be created to ensure that all end users are properly trained before the system is launched. For end users to make plans that work with their schedules, the schedule will be communicated to them well in advance.

**Evaluation of Training Process:** A review of the training will be conducted to assess the efficacy of the process carried out to make all employees/farmers/admin understand the new system. End users input will be assembled to find out/understand where we can improve the conveyance or make any betterments to the system before it’s launched.

# Maintenance:

Here are some key features that could be included in such a system:

**Farmer profiles**: Each farmer would have their own profile on the platform, which would include information about their farm, the products they sell, their agricultural procedures, and contact information.

**Product listings**: Farmers would be able to create entries for their available products, including images and descriptions. Buyers could then search these ads for products of interest.

**Communication tools**: The platform would include communication tools like messaging or chat services, allowing farmers and buyers to engage directly with one another.

**Ordering and payment**: The system would enable buyers to place product orders and pay directly through the platform.

**Delivery and logistics**: The platform might also provide tools to assist farmers with logistics management, such as scheduling.

**Reviews and ratings**: To help customers establish trust and reputation, the platform might incorporate a review and rating system where they can offer feedback on their experience with a certain farmer or product.

# Support:

**Support:** To ensure the system's smooth functioning, a comprehensive support strategy must be established to address any potential issues or concerns that users may have. The support plan should include the following components:

**Help Desk:** A dedicated help desk or support team should be available to handle user inquiries and concerns. The response time for the help desk should be included in the support plan, and users should be able to contact the help desk via phone, email, or chat.

**Knowledge Base:** A knowledge base containing relevant system information, such as frequently asked questions, how-to guides, and best practices, should be available. Users should be able to access the knowledge base through the help desk or a self-service portal.

**Software Updates:** Regular software updates should be provided to users to ensure that the system remains up-to-date, and any known issues or vulnerabilities are addressed. The support plan should include details on the upgrade process, including any necessary downtime.

**User Training:** To ensure that they are familiar with the features and capabilities of the system, users should undergo training. The training structure and schedule, which may include in-person, online, or self-paced choices, should be described in the support plan.

**Escalation Procedure:** An escalation process should be established for addressing any issues that the help desk is unable to resolve. The escalation procedure should include the chain of command and the response time for each level of escalation.

**Service Level Agreement (SLA):** A service level agreement should be developed between the vendor and the customer, outlining the level of service to be provided, as well as response times, uptime guarantees, and other relevant information.

By implementing a comprehensive support plan, the system can be effectively managed and users can receive timely assistance and support when needed.

# Version and Update Rollout Plan:

Version and Update Rollout plan (ALM): An essential element in the Application Lifecycle Management (ALM) approach is the version and update distribution strategy. It defines the steps to follow as new upgrades or versions of the system are released.

The following is an example version and update deployment plan:

prepare and plan: The first and most essential element of the rollout strategy is to identify the modifications you intend to make in the updated version of the application. You must decide what features or functionality to add or enhance, as well as what challenges to tackle. Once the changes have been identified, you must determine the target audience for the update and set a timeline for the release.

Development Phase: During the development phase, the team produces the updated version or update of the system. They do unit, integration, and system testing to ensure that the updated version or upgrade works as expected.

Internal testing: During this phase, your development team should do extensive internal testing of the updated version of the program to confirm that it is running properly and that all new features are working as planned. This stage is crucial for identifying any serious flaws before the update is published to beta testers or users.

Alpha testing During this phase, the updated version of the application is distributed to a small group of users known as alpha testers. Alpha testing aids in identifying lingering flaws or bugs and provides feedback on the user experience. Before the update is disseminated to a larger audience, this group of users can provide valuable feedback.

Beta testing: The updated version of the application is released to a bigger number of users known as beta testers during this phase. Beta testing helps identify remaining flaws and provides feedback on the user experience. Before the update is given to all users, this group of users can provide valuable feedback.

Release to a limited audience: Following the beta testing phase's completion, the updated version of the application can be released to a limited audience, such as a specific region or user group. This enables the development team to monitor the application's performance in a controlled environment and discover any issues that must be addressed before the update is made available to all users.

Release to all users: After the application has been tested and refined, the updated version can be made available to all users. It is critical to convey the changes to users and give support if any problems develop. This phase entails distributing the update to all users, tracking their feedback, and any issues that may develop.

Monitor user feedback: Once the update has been launched, it is critical to monitor user response to detect any issues or bugs that may have been missed during testing. This will assist you in identifying problems and making necessary fixes.

Future system enhancements: Based on user feedback, the development team should continue to iterate on the program, improving and adding new features as needed. In this phase, user feedback is incorporated into future updates and releases to ensure that the application remains current and meets user expectations.

A successful version and update rollout plan for any program entails careful planning and preparation, extensive testing, releasing the update to a small group of users before releasing it to all users, and continual iteration depending on user input.

System Specification Review (SSR)

**Physical Characteristics of the system:**

1. Performance Requirements
   * 1. Application server requires at least 16GB RAM and 2 Cores
     2. The system must be able to handle up to 1000 concurrent users.
     3. The system must be able to store up to 1 million products.
     4. The system should have failover capability to ensure high availability.
2. Interface Requirements with other systems
   * 1. Secure payment gateway for online transactions
     2. SMS or email notifications for order status updates
     3. Integration with Google Maps API to display farmers' locations.
3. Internal Requirements
   * 1. Input: Farmers can input details about their products and prices, and update product availability.
     2. Processing: The system should display available products to customers based on their location and preferences.
     3. Output: Customers can place orders and receive order confirmation and delivery

details.

**Software:**

* 1. Application Server: Microsoft Azure server
  2. Web Client: Proprietary JavaScript/TypeScript, HTML, CSS, Java code base
  3. Database Server: MS SQL Database Instance + Failover instance
  4. Backend: Python

**Personnel:**

* 1. Application Manager ($100,000)
     + - Manage the presentation layer and application server software.
       - Patch and update versions as needed.
       - Install required plugins.
  2. Product Manager ($100,000)
     + - Responsible for product roadmap and client experience.
       - Determines which feature additions are to be prioritized.
  3. Senior Full Stack Developers ($110,000)
     + - Scrum team to build out the application and deliver milestones.
       - Update features based on priority.
  4. Junior Full Stack Developers ($80,000)
     + - Scrum team working under seniors developing the application.
       - Responsible for bug remediation.
  5. Scrum Master ($90,000)

• Responsible for managing the Scrum team backlog and addressing roadblocks.

* 1. Database Administrator ($110,000)
     + - Responsible for originating the schema, tables, and stored procedures related to

backend logic for the application.

* + - * Responsible for failover and security.
      * Responsible for performance related to backend integration.
  1. Tech Support Team ($15/hr x 10)
  + Rotating shifts to provide 24/7 support.

**Facilities:**

* + Office space with infracture (tables, chairs, monitors, desktops, laptops)
  + Meeting rooms with audio and video conferencing capabilities
  + Internet/Wifi at work station
  + Cafeteria

**Licenses:**

* 1. MS SQL Server as a Service.
  2. Integration with Google Maps API will require a license.
  3. A secure payment gateway license is required.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Description** | **Days Required** | **Cost per day**  **(per developer)** | **No. Of Developers** | **Total** |
| Testing | 5 | $400 | 8 | $16,000 |
| Bug Fixes | 11 | $400 | 8 | $33,600 |
| Development | 50 | $400 | 8 | $160,000 |
| Total | 66 | - | - | $209,600 |

**Cost Models:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Feature** | **Function** | **Cost** |
| User-friendly  interface | Interactive and easy-to-use interface | Allows users to quickly and easily  access various services | 500 dev hours at $50/hr = $25,000 |
| Scalable and modular architecture | Scalable platform | Accommodates future growth and new  services  Eventual migration to AWS full cloud architecture. | $10k/month |

|  |  |  |  |
| --- | --- | --- | --- |
| High  availability and reliability | High availability and reliability | Ensures seamless user experience | 100 dev hours at $50/hr =  $5,000 |
| Robust security measures | Robust security measures | Protects user data and transactions | 100 dev hours at $50/hr =  $5,000 |
| Integrations with service providers | Integration with partner APIs | Allows access to partner services through a single interface | 500 dev hours at $50/hr = $25,000 |
| Direct buying  from farmers | Online marketplace for buying fresh produce directly from farmers | Facilitates direct transaction between  farmers and customers | 1000 dev  hours at  $50/hr =  $50,000 |
| Delivery | Integration with delivery providers | Allows customers to choose from multiple delivery options | 500 dev hours at $50/hr = $25,000 |
| Payment processing | Integration with  payment gateways | Allows customers to securely pay for their orders | 500 dev hours at $50/hr = $25,000 |
| Customer support | 24/7 customer support | Provides in-app chat, email, or phone support for customers | 10 agents @  $15/hr each, 24 hours per day =  $3,600/day or $108,000/mo  nth |

|  |  |
| --- | --- |
| **Cost Summary** |  |
| Initial Investment | $500,000 |
| Development | $210,000 |
| Linux Machines x3 | $10,000 |
| Monthly |  |
| AWS Hosting | $10,000 |
| Customer Support | $108,000 |
| Salaries | $49,000 |
| MS SQL Server License | $4,000 |
| Microsoft Azure server | TBD |
| Additional monthly subscription to fix bugs | $20,000 |
| Total | $220,000 + $191,000 monthly |

**Topology of Hardware, Software, Network:**

**Hardware:**

* + Application Server: 16GB RAM and 2 Cores virtual machine
  + Web Client: 8GB RAM 2 Core virtual machine to host the presentation layer.
  + Database Server: 2x MS SQL Server DB instances to hold application data with failover environment.

**Servers:**

To ensure high availability and scalability, consider utilizing load balancers to split incoming traffic over different servers. Backups and disaster recovery solutions should also be considered to protect against data loss and service interruptions.

**Storage:**

We may need to think about the storage system's performance and reliability needs. To ensure speedier data access times, you may choose to employ solid-state drives (SSDs) instead of hard disk drives (HDDs). You should also consider deploying backup and recovery solutions to protect against data loss.

**Mobile Devices:**

Considering mobile device compatibility needs, such as operating system versions and hardware specifications. To prevent illegal access, you should also consider installing security measures such as data encryption and two-factor authentication.

**Software:**

* Application Server: Microsoft Azure server
* Web Client: Proprietary JavaScript/TypeScript, HTML, CSS, Java code base
* Database Server: MS SQL Database Instance + Failover instance
* Backend: Python
* Middleware: We might want to think about employing caching technologies like Redis to increase the application's performance. We could also think about employing message queue systems like RabbitMQ to provide consistent message delivery between application components.

**Network:**

**Internet:**

We may wish to investigate implementing content delivery networks (CDNs) to increase the application's performance and availability across multiple geographical locations. To protect against DDoS and XSS assaults, we should also consider adding security measures such as SSL/TLS encryption and web application firewalls (WAFs).

**Local Network:**

To measure network usage and detect problems, we may wish to consider deploying network monitoring tools. To protect against unwanted access, we should also consider adopting intrusion detection and prevention systems (IDS/IPS).

**Security:**

To identify and address security issues, you may want to consider installing additional security measures such as vulnerability scanning and penetration testing. You should also ensure that the application conforms with applicable data protection and privacy laws,

such as GDPR and CCPA.

**Some potential supporting process models that could be useful include:**

1. **Supply Chain Management**: The flow of commodities and services from farmers to consumers might be managed using this process model. Procurement, shipping, storage, and distribution are examples of such tasks.
2. **Quality Control**: This process model might be used to ensure that the products offered via the application fulfil specific quality criteria. Testing, inspection, and certification are examples of such tasks.
3. **Client Relationship Management**: This process model could be used to manage client interactions such as ordering, delivery, and support. Customer service, sales, and marketing are examples of such activities.
4. **Data Analytics**: This process model could be used to analyse application data to uncover trends and patterns that can be utilized to inform business choices. Data mining, predictive analytics, and reporting are examples of such tasks.
5. **Financial Management**: This process model could be used to manage the application's financial components, such as billing, payment processing, and accounting. Invoicing, payment tracking, and financial reporting are examples of such activities.
6. **Marketing & Promotion Model**: Using multiple marketing channels such as social media, email campaigns, and advertising, this model can help promote the application and its products.

This can help farmers and other producers develop and succeed by attracting new clients and increasing brand awareness.

# Test Readiness Review (TRR)

**User Story mapped to each Requirement, Feature and Function.**

**Customer Interface user stories:**

* As a customer, I want to be able to sign up for an account on the website, so that I can easily access my account information and previous orders.
* As a customer, I want to be able to log in to my account quickly and easily, so that I can view my account information and order history.
* As a customer, I want to be able to search for specific farms or products, so that I can quickly find the items I'm looking for.
* As a customer, I want to be able to add items to my cart and remove items from my cart, so that I can easily adjust my order before checkout.
* As a customer, I want to be able to checkout and pay for my order quickly and easily, so that I can complete my purchase without any hassle.
* As a customer, I want to be able to view the status of my order and receive updates on the progress of my delivery, so that I can plan accordingly.
* As a customer, I want to be able to view my order history and reorder previous items, so that I can quickly purchase items I have ordered before.
* As a customer, I want to be able to receive notifications about new products, special deals or promotions, so that I can take advantage of discounts and save money on my purchases.
* As a customer, I want to be able to provide feedback on my purchases and experiences, so that I can help the website improve and other customers can benefit from my experiences.
* As a customer, I want to be able to contact customer service through the website, so that I can quickly and easily resolve any issues or concerns with my orders.

**Farmer Interface User stories:**

* Farmer should be able to sign up for an account and provide name, farm details, and farm name so that farmer can easily manage his/her account and products.
* Farmer should be able to login to his account so that he can access all of the features of the platform.
* Farmer should be able to search products and update their stock levels or modify the product details such as the description or pricing or quantity.
* Farmer should be able to upload stock updates through an Excel spreadsheet so that they can easily keep track of my inventory and save time.
* As a farmer, I want to be able to view my payment history and raise a complaint if I have not received payment for my products.
* As a farmer, I want to receive notifications when I receive new orders so that I can fulfill them in a timely manner.
* As a farmer, I want to be able to view my farm ratings and reviews so that I can improve my product offerings and customer satisfaction.

**Use Case mapped to each Requirement, Feature and Function.**

**The requirements for the app have been defined as follows:**

* The app must enable farmers to manage their inventory efficiently.
* The app must allow consumers to order fresh produce directly from farmers.
* The app must enable farmers to set prices for their products.
* The app must enable consumers to track the delivery of their orders.

**Features and Functions:**

* Inventory management system for farmers
* Online ordering system for consumers
* Price setting system for farmers
* Delivery tracking system for consumers

**Use Case Mapping:**

Each of the features and functions has been mapped to the following use cases:

* **Inventory management system for farmers:**
* Add new products to inventory
* Update product information
* Delete products from inventory
* Track inventory levels
* Set alerts for low inventory levels
* **Online ordering system for consumers:**
* Browse products by category
* Add products to cart
* View cart
* Place order
* Cancel order
* **Price setting system for farmers:**
* Set prices for products
* Update prices as needed
* View price history
* **Delivery tracking system for consumers:**
* Track order status
* Receive notifications on delivery updates
* Report delivery issues

**Test Case developed from each Use Case or User Story.**

**Requirement 1: Creating an account.**

|  |  |
| --- | --- |
| **Test Case Field** | **Description** |
| Account Creation | Creating an account successfully |
| Validation of data and appropriate fields | To avoid the creation of duplicate accounts, the user's input is verified by being compared to the data already present in the main database. |
| Purpose | The user needs to be able to provide details |
| Execution Steps | Web application/site à sign up screen à filling required details à submit |
| Expected Results | Able to register a new account |

**Requirement 2: Login Screen.**

|  |  |
| --- | --- |
| **Test Case Field** | **Description** |
| User identification and access management | Able to access profile after creating an account |
| Validation of data and appropriate fields | Testing the login page by making fresh accounts and using the same login information as when you made an earlier account |
| Purpose | Should be able to access the website by entering their login information. |
| Execution Steps | Web application/site à login page à enter valid credentialsà My Profile. |
| Expected Results | able to log in to the website and use more services than before. |

**Requirement 3: Online ordering system for consumers**

|  |  |
| --- | --- |
| **Test Case Field** | **Description** |
| User authentication | capable of completing the two-factor authentication |
| Agenda | This test case aims to persuade people to place orders for products to be delivered right to their door. Elderly people and those who can't get inventory on their own must have access to multifactor authentication in big numbers. |
| Purpose | Allowing consumers to add their inventory to the basket and make payments should be possible. |
| Execution Steps | Required products à Multifactor authentication à Adding to the cart à  Checkout à Payment gateway |
| Expected Results | capable of entering personal information, making payments via any practical method, and scheduling deliveries |

**Requirement 4: Payment system for farmers**

|  |  |
| --- | --- |
| **Test Case Field** | **Description** |
| User authentication | Payment will be processed. |
| Agenda | In this test scenario, clients will input their 16-digit pin number and CVV, and after they submit the right expiration date, they will be able to complete the transaction. |
| Purpose | Once the right payment information is provided, customers should be able to complete the transaction. |
| Execution Steps | Enter 16-digit card number à Enter CVV number à Enter expiration date  Make payment. |
| Expected Results | Users will submit their card information in the manner to proceed with the transaction. |

**Requirement 5: Order tracking** **and authentication**

|  |  |
| --- | --- |
| **Test Case Field** | **Description** |
| User authentication | The order will be examined, and the inventory care will authenticate it. |
| Agenda | Users can examine their orders for the items they've chosen and confirm their payments, but customer care will only finalize the transaction after checking the data with their inventory centre. The same notice will be displayed on the page. |
| Purpose | Payment is successful and the order can be seen. |
| Execution Steps | Payment successful à order review à confirming with the Inventory à  Order delivery. |
| Expected Results | The users will be receiving the order once their order is verified. |

**User Acceptance Tests**

**Registration process:** Test the registration process to make sure users can easily establish an account and that all necessary fields are present. Verify that users who register receive a confirmation email or message.

**Search function:** Test the search tool to make sure consumers can quickly find the things they are looking for. Verify that the search engine produces accurate and pertinent results.

**Ordering procedure:** Test the ordering procedure to make sure users can choose things, put them in their carts, and proceed to the payment page without any problems. Make sure customers can examine their orders before submitting them and that they get an order confirmation.

**Delivery procedure:** Run tests to make sure users can select a delivery time and address that works for them. Check that customers can track their deliveries in real-time and that they receive updates on their status.

**Payment procedure:** Test the payment procedure to make sure customers can pay for their orders quickly and conveniently using a range of payment options. Verify that users receive payment confirmation, and that payment information is kept secure.

**Admin Service/Issue Tickets:** tests ought to cover the creation of a ticket, the display of ticket information, updates to a ticket's status, the escalation process, communication channels, ticket tracking, the resolution process, and user feedback. You can confirm that the system is easy to use, effective, and efficient as well as that it gives your users a pleasant experience by carrying out these tests.

**Mobile compatibility:** To make sure the program runs well on both iOS and Android devices, test it for mobile compatibility. Check to see if the app is simple to use on a mobile device and if all functions are available.

**Performance:** Check the application's responsiveness, loading speed, and ability to avoid freezing or crashing while in use. Verify that the program can manage many users without becoming sluggish.

**Accessibility:** Test the application's accessibility to make sure all users, including those with disabilities, can use it. Check sure the program is accessible and that all content is compatible with screen readers and other assistive technology.

**User interface:** Check the user interface of the application to make sure it is simple to use and intuitive. Make sure the layout is clear and all buttons and links have distinct labels.

**Security:** To ensure that user data and payment details are kept secure, test the application's security. Check to see if all user data is encrypted and if security precautions have been taken to prevent unauthorized access.

**Performance underload:** Test the application's performance to ensure it can manage many orders and users without crashing or slowing down. Check to see if the application can expand to handle peak demand.

**Error handling:** Test the application's error handling to make sure that when something goes wrong, users receive informative and actionable error messages. Check that mistakes do not cause the application to crash and that users can recover from faults with ease.

# WORK BREAKDOWN STRUCTURE

A picture containing text, screenshot, diagram, number

Description automatically generated

A diagram of a farmer functional interface

Description automatically generated with low confidence

A picture containing text, screenshot, number, font

Description automatically generated

# IPO MODELS OF PROCESS WITH SCREENS

A picture containing text, screenshot, diagram, parallel

Description automatically generated

A picture containing text, screenshot, diagram, parallel

Description automatically generated