# 1. Introduction:

"**Farmigos**" is a web-based platform for agriculture related requirements. In this application admin will manage both farmers and customers. Farmers can list out the products which are available and the customer can order the products according to the requirements. Payments can be made both online and offline and feedback can be made online.

## 1.1 Title of the project:

**FARMIGOS**

## 1.3 Objective of the project:

- The main objective of this system is to provide virtual platform for the farmers, customers.
- Both the farmer and the customer can avoid the middleman and can sell and buy the products through online.
- Farmers can sell their products through online with affordable price.
- Customer can buy the products according to their needs.

## 1.4 Project Category:

Web-Based Application Using RDBMS.

## 1.5 Languages to be used:

- Front-end: HTML, CSS, JavaScript
- Back-end: MySQL, PHP

## 1.6 Hardware Interface:

| | | |
|---|---|---|
| 1.5.1 | Processors | : Intel core i3 or above |
| 1.5.2 | RAM | : 2GB and above |
| 1.5.3 | Hard disk Utilization | : 40GB and above |
| 1.5.4 | Input Devices | : Mouse, Keyboard |

## 1.7 Software Interface:

| | | |
|---|---|---|
| 1.5.5 | Browser | : Internet Explorer, Google Chrome, MozillFirefox |
| 1.5.6 | Server | : Apache |

1.5.7 IDE : visual code

## 1.8 <u>Module description:</u>

### 1.8.1 <u>Admin:</u>

- **Login**: Admin will enter the website using username and password.
- **Manage Farmers**: In this module, the admin can manage the farmers who are registered.
- **Manage Customers**: In this module, the admin can view all the customers who are registered.
- **View Payment**: Here the admin can view the payments made by the customers.
- **View Feedback**: The admin can view the feedback sent by the Customers to the products.
- **View Contacts**: Admin can view the feedback sent to him by the Customer.
- **Add blog:** Admin can add different blogs which is displayed on customer page.
- **Report:** Admin can view the report of the farmer.
- **Add Location**: Admin can provide the location details to farmers.
- **Manage category:** Admin will manage the product based on their details.

### 1.8.2 <u>Farmer:</u>

- **Register:** Here the farmer will register in to the website.
- **Login:** The farmer can log in using his/her username and password.
- **View Profile:** The farmer can manage his/her personal details.
- **Add product**: Farmer will add the list of items in to the website.
- **Manage order:** In this module, the farmer will manage the order placed by customer.
- **View payment:** The farmer can view the payment details made by the customer.
- **View Feedback:** The farmer can view feedback given by their customer.
- **Report**: in this module, Farmer will generate necessary reports on their product.

### 1.8.3 Customer:

- **Registration:** The customer can give his/her basic details.
- **Login:** The customer can log in using his/her username and password.
- **Manage Profile:** The customer can manage his/her personal details.
- **View Blog:** The customer can view the blog which is written by admin.
- **Filter by location:** In this module, customer can filter the list of farmers near to their location.
- **Filter by category:** In this module customer can filter the products based on their category.
- **View products:** The customer can view the list of products.
- **Add to cart:** Here the customer can choose the product to purchase without completing the payment.
- **Place order:** Here the customer can place their order.
- **Track Order:** In this module customer can track the order.
- **Payment:** Here, the customer can make payment for purchasing their wished product.
- **Contact:** Customer can Contact the admin for various reasons.
- **Feedback:** The customer can send feedback about the particular product.

## 1.9 Limitations:

- It requires reliable internet access.
- It will be difficult for some users because lack of computer and internet knowledge.
- Limited participation: Some farmers may be reluctant to participate in the agro market system due to various reasons, such as lack of trust, lack of awareness, and fear of losing control over their produce.

## 1.10 Future Scope:

- In future, it can also be converted to an mobile application.
- Establishing linkages with international markets to increase exports and generate foreign exchange

# 2. SOFTWARE REQUIREMENT SPECIFICATION

## 2.1 Introduction

This Software Requirement Specification document provides a complete description of all the functionalities and the specifications of the "**Farmigos**" system. The following section provides an overview of the derived Software Requirements Specification (SRS) for the application. To begin with, the purpose of the document is presented**,** and its intended audience is outlined. Subsequently, the scope of the project specified by the document is given with a particular focus on what the resultant software will do and the relevant benefits associated with it. The nomenclature used throughout the SRS is also offered. To conclude, a complete document overview is provided to facilitate increased reader comprehension and navigation.

### 2.1.1 Purpose

This document is developed for the project "**Farmigos**" is based on buying and selling agriculture products through online. It is helpful to the farmer to sell their products and customer can buy the products with affordable price. The purpose of this document is to describe the external requirement of the project. The main purpose is to translate the ideas in the mind of the client into a formal document. A software requirements specification (SRS) minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs, and human users in a wide variety of real-world situations. The SRS provides critical information about development operations, quality assurance, and maintenance to the developers.

### 2.1.2 Scope

The primary scope of the "**Farmigos**" is to create a marketplace that connects farmers and buyers. This system aims to provide farmers with a platform to sell their products directly to the buyers without any intermediaries. Additionally, it aims to provide buyers with a wider range of agricultural products to choose from and purchase at a fair price. Today is the era of computers. This software project solves all the problems experienced in the present system. The proposed system provides the following features:

- It creates an online marketplace for farmers to sell their products directly to buyers without any intermediaries.
- It provides a search and filter module that allows buyers to search for products based on their location, product category, and price range.
- The admin can keep track both farmers and customers details.

### 2.1.3 Definition, Acronyms, Abbreviations

| | |
|---|---|
| GUI | - Graphical User Interface |
| DBMS | - Database Management System |
| RDBMS | - Relational Database Management System |
| SRS | - Software Requirement Specification |
| ADMIN | - The Administrator. |
| CPU | - Central processing unit |
| PHP | - Hypertext Preprocessor. |
| SQL | - Structured Query Language. |
| HTML | - Hyper Text Markup Language. |
| CSS | - Cascading style sheet |

### 2.1.4 References

i. www.w3schools.com
ii. www.tutorialsapoint.com
iii. www.geeksforgeeks.com
iv. www.javatpoint.com

### 2.1.5 Overview

The “**Farmigos**” is an online platform that aims to streamline the agricultural market by providing a common marketplace for farmers and buyers. This system eliminates the middleman and creates an efficient process for both parties. The system provides a user-friendly interface for farmers to upload their products and for buyers to browse and purchase products from various sellers.

## 2.2 Overall Description

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce its basic functionality of it. It will also describe what type of users will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented.

### 2.2.1 Product Perspective

The "**Farmigos**" is a fully independent product and not a part of any other system. The users of this system are categorized as admin, farmer and customer. The system gives online access to each separate user. This system provides an easy way for interact between farmer and customer that managed by an admin. In this system farmer and customer can register and login into the system. A customer can view profile and book a product from the comfort of their homes, using their computer, laptop or mobile and at any time. No matter where they are, they can contact farmer of their choice in any location. Customer can view the product details provided by the farmer. This Project is self-contained. It provides a simple database rather than complex ones for high requirements and it provides a good and easy graphical user interface (GUI) to both new as well as experienced users of the system.

### 2.2.2 Product Functions

- The project "**Farmigos**" provide an online platform for farmers to give the clear picture of the crop sown and also give the efficient manner for the crop.
- It manages all the activities online.
- All the details of the crop surveys details are stored in the admins database.
- It enables to maintain a clear picture of the crop sown.
- It enables the admin to get the information about the crop survey.
- It provides requirements for the farming.
- It enables the users to grow their farming through online.
- It enlarges the flexibilities in the existing system with a web-based user-interactive interface.

### 2.2.3 User Classes and Characteristics

The system has two user levels.

#### Admin:

The admin can log in to the system using his/her unique credential. Here admin can manage the details of farmer and customer.

- ➢ Login.

- Manage Farmers.
- View Customer.
- Market update.
- View Payment.
- View Feedback.
- View Contact
- Report.
- Add Location.
- Add blog
- Manage category.

### Farmers:

Farmers can register through username and password. Farmers will login through this website for selling their products. They can also view the payments.

- Registration.
- Login.
- View Profile.
- Add Product.
- View payment.
- View feedback.
- Report.

### Customers:

Customer can register by giving their basic details and login through username and password. Then customer can view farmers nearby location and their availability products on this web site.

- Registration.
- Login.
- Manage Profile.
- View farmer.
- Filter by location.
- Filter by category.
- View products.
- Add to cart.
- Place order.
- Track Order.
- Payment.

- Feedback
- View blog
- Contact.
- Report.

### 2.2.4 General Constraints:

General constraints include the following:

- This application requires an internet connection.
- Only Admin can manage this application.
- No one has the right to change the survey information in the application.
- The end system should also allow for seamless recovery, without data loss, from individual device failure

### 2.2.5 Assumptions and Dependencies:

- It is assumed that this system has two types of users, i.e., Admin and Farmers.
- The Farmers should be careful in approving the crop details.
- All the details related to crop survey should be maintained properly.
- All the data entered will be correct and up to date.
- It is assumed that the needed changes, to collect and store the data, will be made within the current application and database.

# 3. SPECIFIC REQUIREMENTS

## 3.1 External Interface Requirements:

### 3.1.1 User Interface

A user interface is a point of human interaction and communication with the system.

We have taken the following requirements during design,

- Text boxes to enter details.
- Buttons to add, delete, update and search.
- Labels to display the information.
- Check boxes.
- Combo boxes and list boxes.
- Grid box to display the information.

### 3.1.2 Software Requirements

- Operating system: Windows.
- Text editor: Visual code.
- Language: PHP
- Server: Apache
- User interface: HTML, CSS, JavaScript
- Database: MySQL
- Browser: Chrome, Mozilla Firefox, or any other browsing application.

### 3.1.3 Hardware Requirements

- Processor: Intel Pentium dual-core or above
- Processor Speed: 2GHz
- RAM – 1GB
- Hard Disk – Minimum 40 GB

### 3.1.4 Communication Interface

The communications function required by this product is HTTP protocol, and internet communication is through TCP/IP protocol.

## 3.2 Functional Requirements

### 3.2.1 Functional Requirements

**Admin:**

- **Login**: Admin will enter the website using username and password.
- **Manage Farmers**: In this module, the admin can manage the farmers who are registered.
- **View Customers**: In this module, the admin can view all the customers who are registered.
- **View Payment**: Here the admin can view the payments made by the customers.
- **View Feedback**: The admin can view the feedback made by the Customer.
- **View Contacts**: Admin can view the feedback sent to him by the Customer.

- **Add blog:** Admin can add different blogs which is displayed on customer page.
- **Report:** Admin can view the report of the farmer.
- **Add Location**: Admin can provide the location details to farmers.
- **Manage category:** Admin will manage the product based on their details.

## Farmer:

- **Register:** Here the farmer will register in to the website.
- **Login:** The farmer can log in using his/her username and password.
- **View Profile:** The farmer can manage his/her personal details.
- **Add product**: Farmer will add the list of items in to the website.
- **Manage order**: In this module, the farmer will manage the order placed by customer
- **View Payment:** The farmer can view the payment details made by the customer.
- **View Feedback:** The farmer can view feedback given by their customer.
- **Report**: In this module, Farmer will generate necessary reports on their product.

## Customer:

- **Registration:** The user can register by using his/her basic details
- **Login:** The patient can log in using their username and password
- **Manage Profile:** The patient can change their profile details.
- **View Farmers:** The customer can view the farmer by their location.
- **Filter by location:** In this module, customer can filter the list of farmers near to their location.
- **Filter by category:** in this module customer can filter the products based on their category
- **View products:** The customer can view the list of products.
- **Add to cart:** Here the customer can choose the product to purchase without completing the payment.
- **Place order:** Here the customer can place their order.
- **Track Order:** In this module customer can track the order.

- **Payment:** Here, the customer can make payment for purchasing their wished product.
- **Feedback:** The customer can send feedback about the particular
- **Contact:** Customer can Contact the admin for various reasons.
- **View Blog**: The customer can view the blog which is written by admin

## 3.3 Performance Requirements:

- The server shall be capable of supporting the survey details and maintain the feedback to the admin and no request and requirements shall be lost under any circumstances.
- Page load time should be less than 40 sec.
- Should have a good memory space.
- The server shall be capable of store the survey details.
- Should be error-free.
- 1MB file should get uploaded in 60 sec.

## 3.4 Design Constraints:

- While user register to the system, mandatory fields must be checked for validation whether the user cannot be open application. If not, a proper error message should be displayed, or else the data is to be stored in a database for later retrieval.
- All mandatory fields should be filled by the user while registering the basic details.
- The system must be designed in such a way that will be easy to use and visible on most browsers.

## 3.5 System Attributes

The Quality of the database is maintained in such a way that it can be very user-friendly to users of the database.

- **Performance:** The system should be able to handle all the users at a time using any of the web browsers.
- **Reliability:** good validation of user inputs will be done to avoid incorrect storage of records.
- **User-friendly Interfaces:** Self-explanatory or easy to use screen layouts are expected to make better quality in terms of usage.

- **Maintainability:** Maintainability is the ability of the system to change with a degree of ease.
- **Portability:** This system shall be portable, and we can switch the server very easily.
- **Flexibility:** The system will keep on updating the data according to the transactions that take place.
- **Timeliness:** The system shall carry out all the operations with consumptions of very less time.

## 3.6 Other Requirements

### 3.6.1 Safety Requirements:

- There are three user levels in “**Farmigos**” Access to the various subsystems will be protected by a user log-in screen that requires a username and password. This gives different views and accessible functions of user levels through the system.
- Email ID once registered to the system cannot be changed to make every user unique and easily identifiable
- Maintaining backups ensures the system database security. The system can be restored in any case of an emergency.

### 3.6.2 Security Requirements:

- The server on which the “**Farmigos**” resides will have its security to prevent unauthorized write/delete access. There is no restriction on reading access.
- The proposed website will be secure. There are different categories of users they are Admin, Farmer and Customer.
- Depending upon the category of using the access rights are decided.
- Admin has the maximum privilege to all subsystems.

# 3. SYSTEM DESIGN

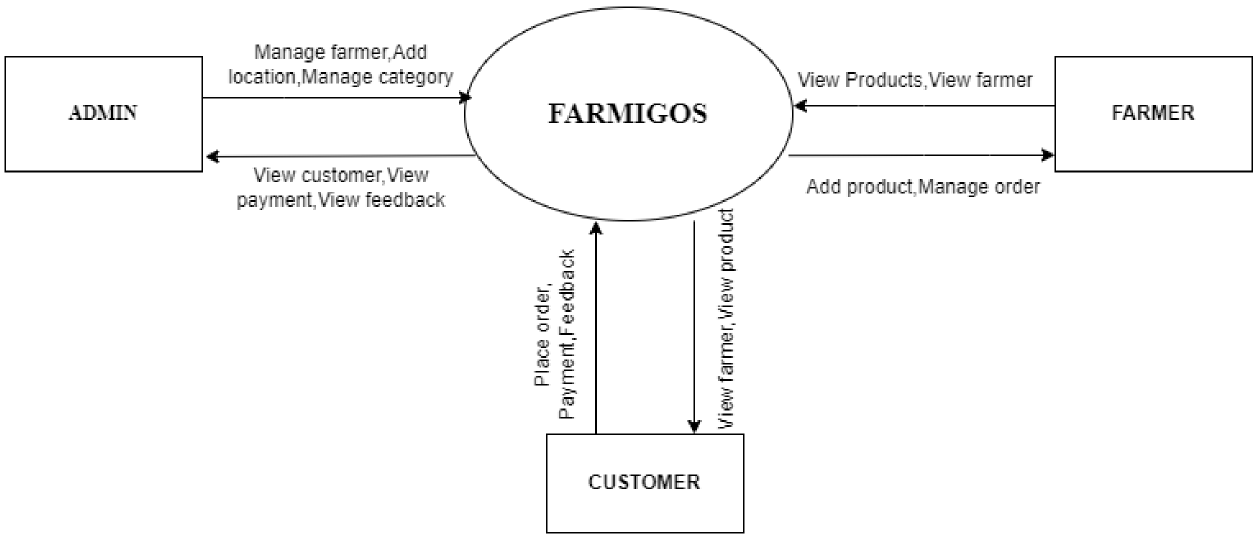
## 3.1 Introduction:

The goal of design process is to produce a model or representation of a system, which can be used later to build the system. Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. The focus of system design is on deciding which modules are needed for system, the specifications of these modules and how the modules should be interconnected. Systems design could be seen as the application of systems theory to product development. Systems design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way the process is systematic wherein it takes into account all related variables of the system that needs to be created. The system design controls the major structural characteristics of the system. It has a major impact on the testability and modifiability of system. The output is the architectural design for the software system to be built.

## 3.2 Context Flow Diagram:

Context flow diagram is a top level data flow diagram. It only contains one process node that generalizes the function of the entire system in relationship to external entities. In context diagram the entire system is treated as a single process and all its inputs, outputs, sinks and sources are identified and shown.

## CFD:


ADMIN
Manage farmer,Add location,Manage category
View customer,View payment,View feedback
FARMIGOS
View Products,View farmer
Add product,Manage order
FARMER
Place order, Payment,Feedback
View farmer,View product
CUSTOMER

## 3.3 Data flow diagram:

Data Flow Diagram is a graphical representation of a system or a portion of the system. It consists of data flows, process, sources and sink and stores all the description through the use of easily understandable symbols.

DFD is one of the most important modelling tools. It is used to model the system, components that interact with the system, uses the data and information flows in the system.

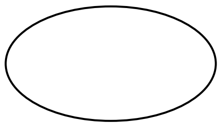
DFD shows the information moves through the and how it is modified by a series of transformations. It is a graphical technique that depicts information moves from input or output.

DFD is also knows as bubble chart or Data Flow Graphs. DFD may be used to represent the system at any level of abstraction. DFD's may partition into a level that represents increasing information flows and functional details.

## 3.4 Rules Regarding DFD Construction:

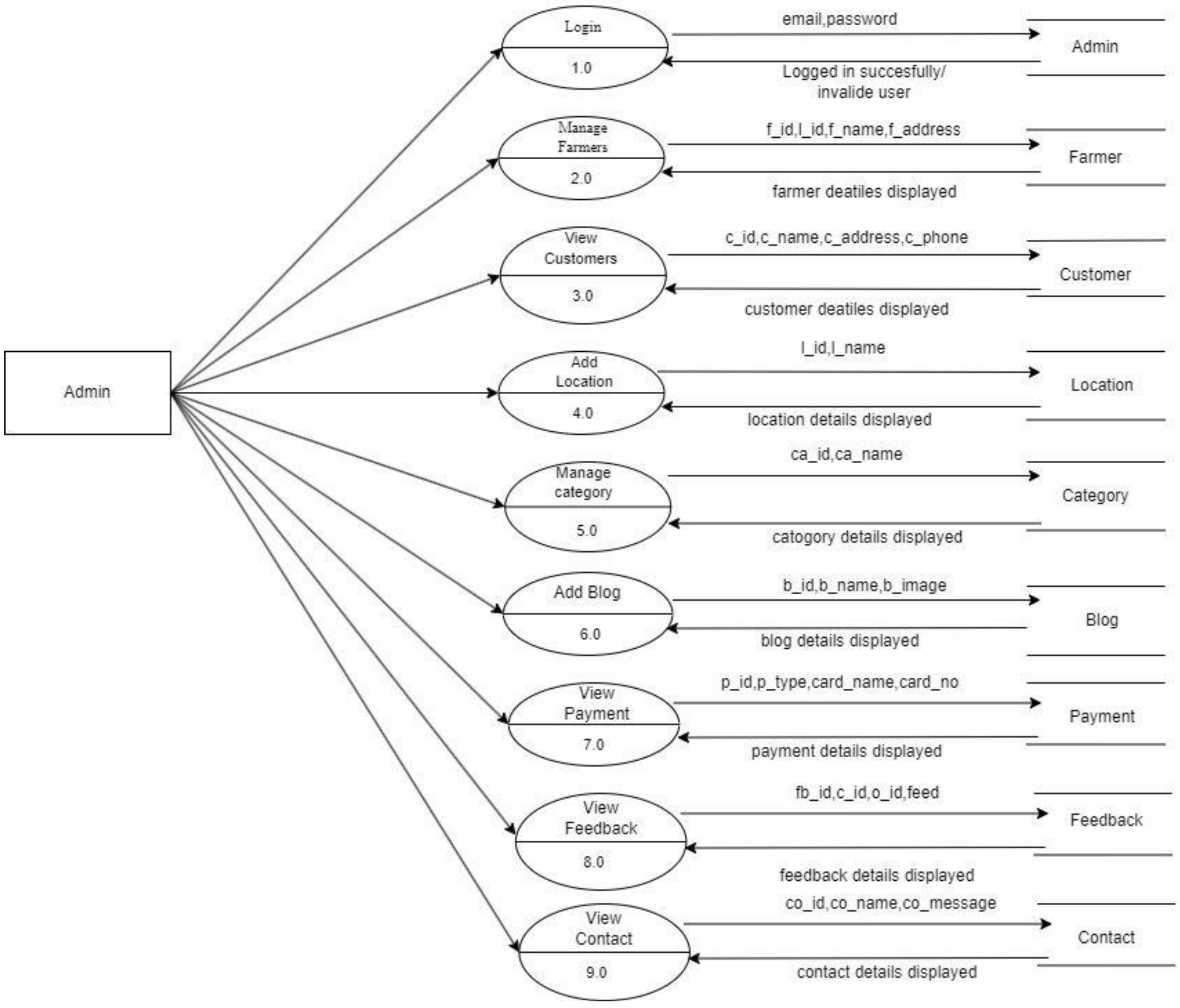
- A process cannot have only inputs.
- The inputs to a process must be sufficient to produce the outputs from the process.
- All data stores must be connected to at least one process.
- All data stores must be connected to a source or sink.
- A data flow can have only one direction of flow. Multiple data flows to and/or from the same process and data store must be shown by separate arrows.
- If the exact same data flows to two separate arrows, it should be represented by a forked arrow.
- Data cannot flow directly back into the process it has just left. All data flows must be named using a noun phrase.

## 3.5 DFD Symbols:

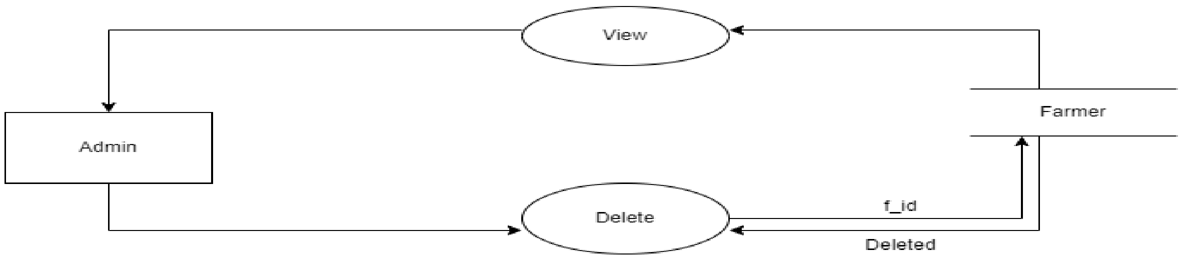
| Name | Notation | Description |
|---|---|---|
| **Process** |  | A process transforms incoming data flow into outgoing data flow. The processes are shown by named circles. |

| | | |
|---|---|---|
| **Datastore** | ═ | Data stores are repositories of data in the system. They are sometimes also referred to as files. |
| **Dataflows** | ← → | Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it. |
| **External Entity** | ▭ | External entities are objects outside the system with which the system communicates. External Entities are sources and destinations of the system's inputs and outputs |

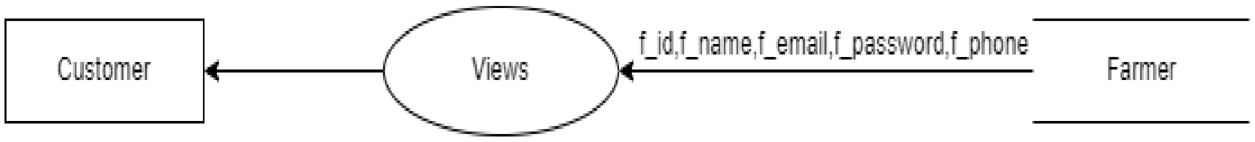
## 3.6 DFD level 1(Admin)



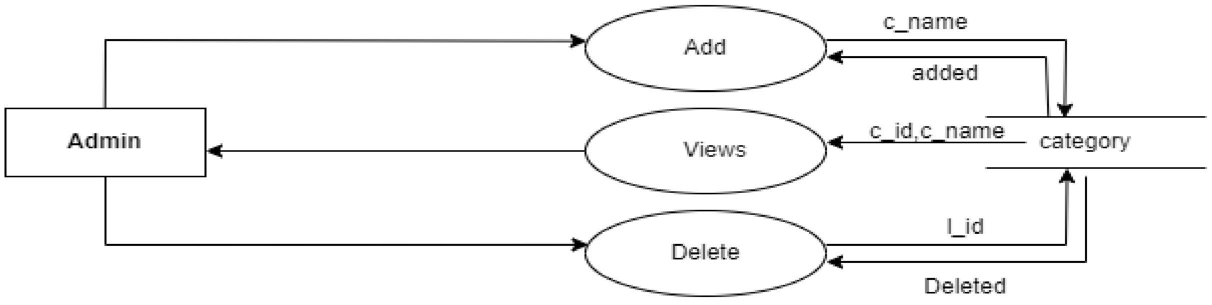
### 3.6.1 DFD level 2 (Manage Farmers)



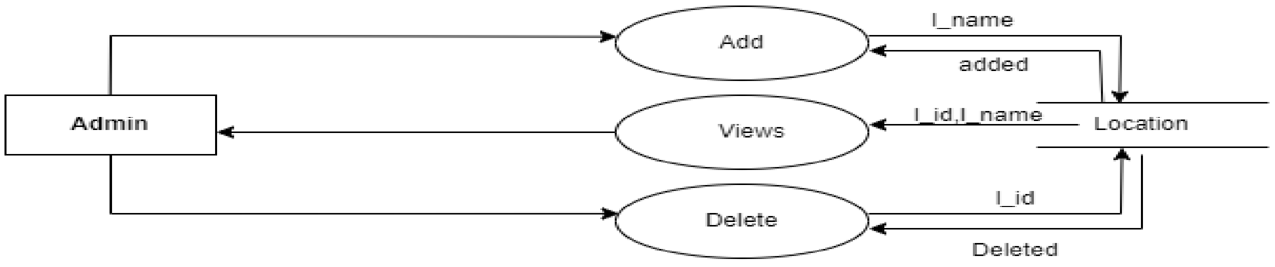
### 3.6.2 DFD level 2 (Manage customer)



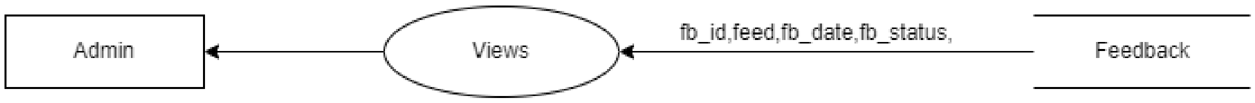
### 3.6.3 DFD level 2 (Manage category)



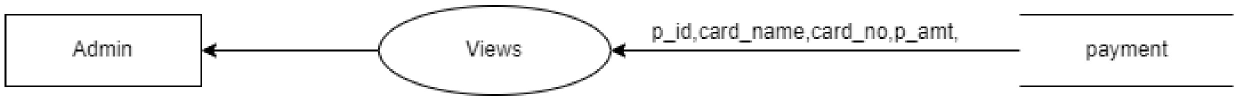
### 3.6.4 DFD level 2 (Manage location)



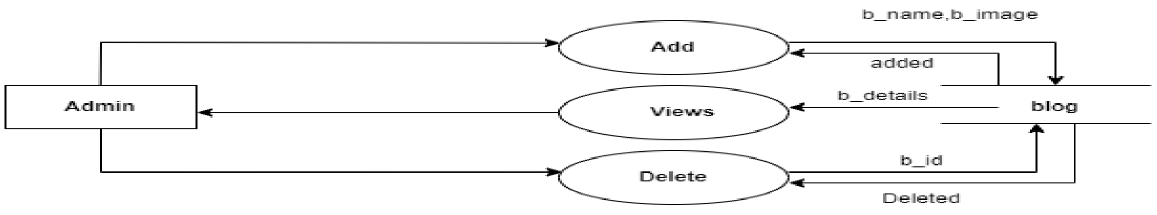
### 3.6.5 DFD level 2 ((feedback)



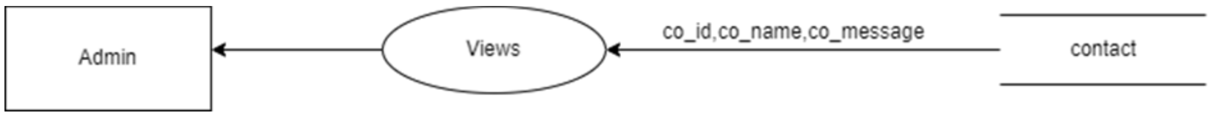
### 3.6.6 DFD level 2 (payment)



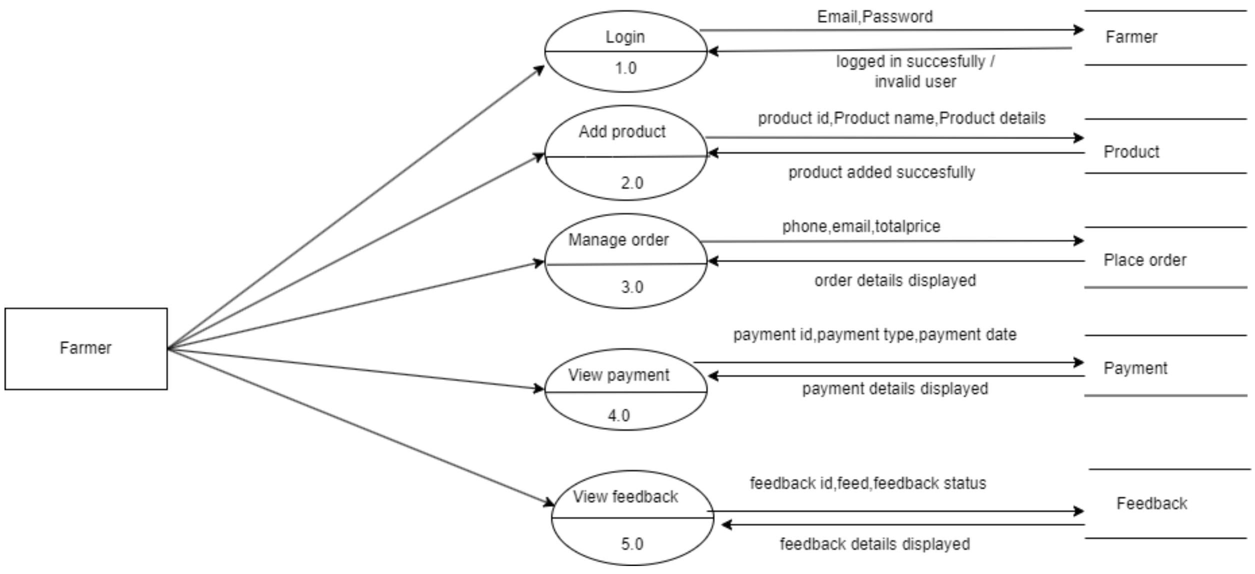
### 3.6.7 DFD level 2 ( Blog)



### 3.6.8 DFD level 2 (View Contact)

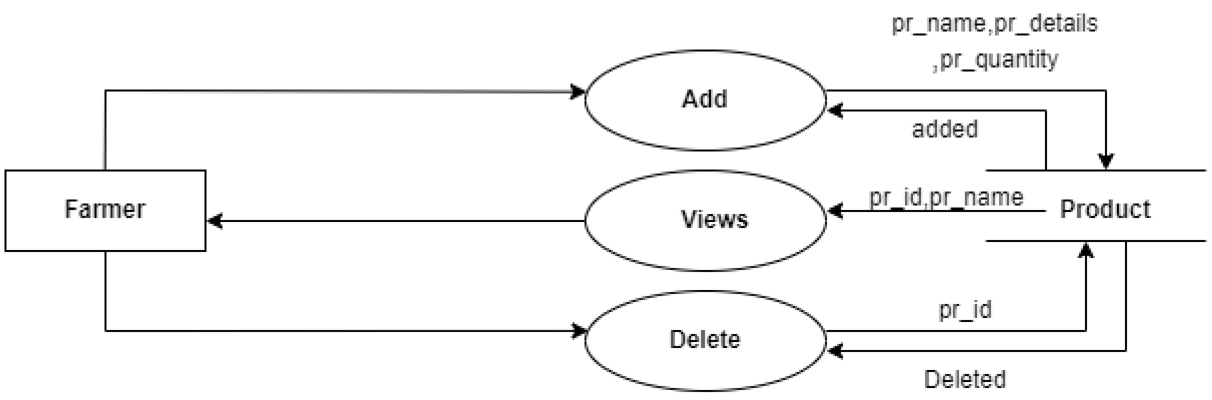


## 3.7 DFD level 1 (Farmer)

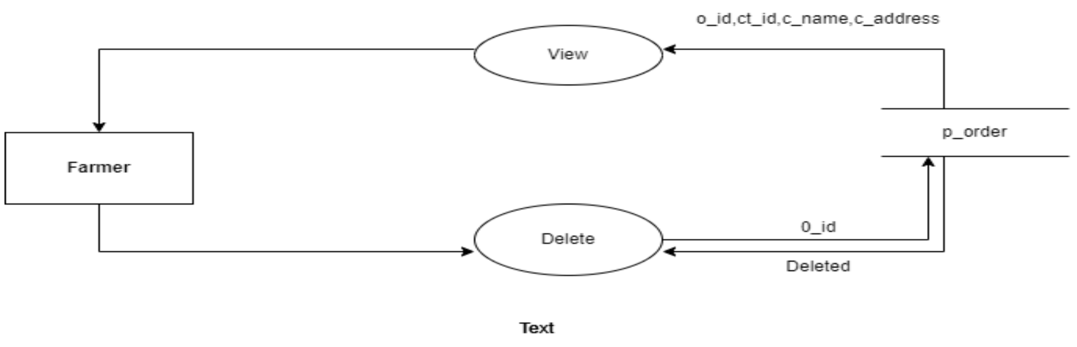


Farmer
Login
1.0
Email,Password
logged in succesfully / invalid user
Farmer
Add product
2.0
product id,Product name,Product details
product added succesfully
Product
Manage order
3.0
phone,email,totalprice
order details displayed
Place order
View payment
4.0
payment id,payment type,payment date
payment details displayed
Payment
View feedback
5.0
feedback id,feed,feedback status
feedback details displayed
Feedback

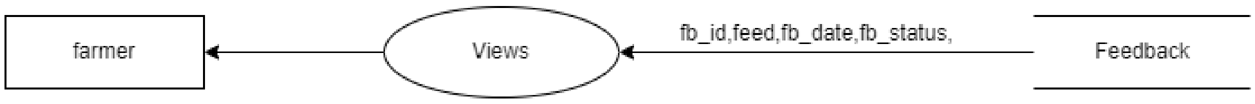
### 3.7.1 DFD level 2 (Add product)



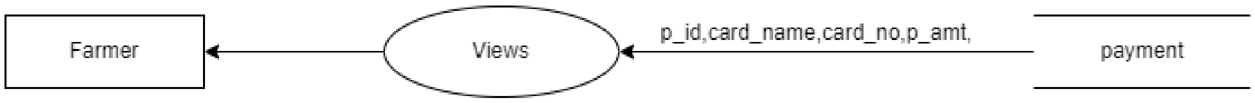
### 3.7.2 DFD level 2 (Manage orders)



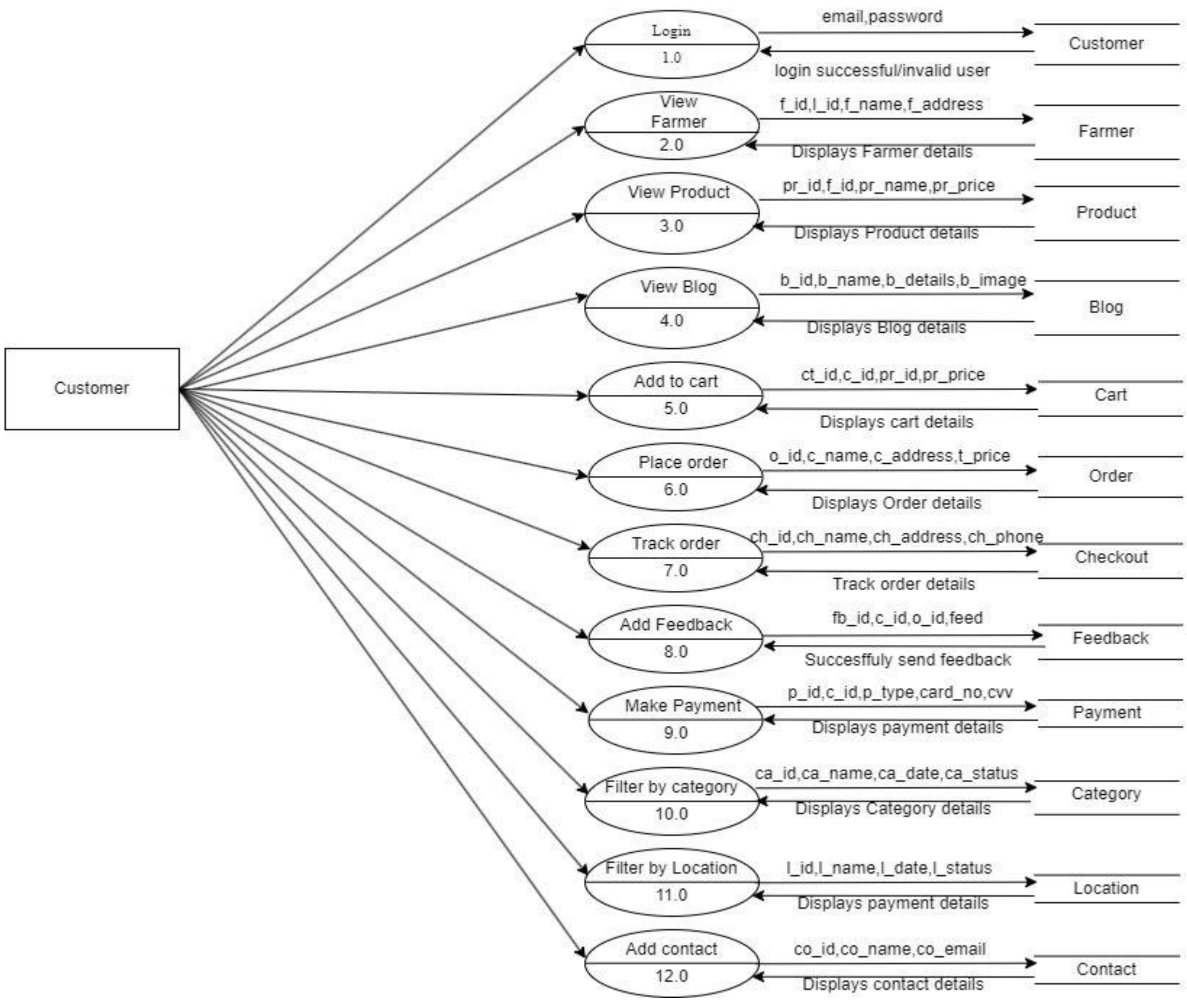
### 3.7.3 <u>DFD level 2 (Feedback)</u>



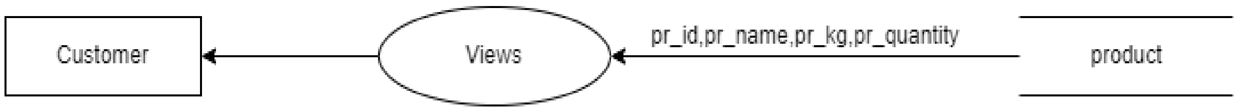
### 3.7.4 <u>DFD level 2 (Payment)</u>



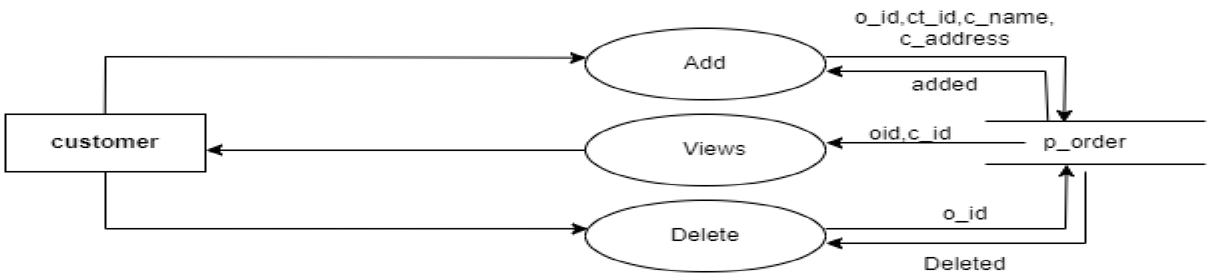
## 3.8 <u>DFD level 1 (Customer)</u>



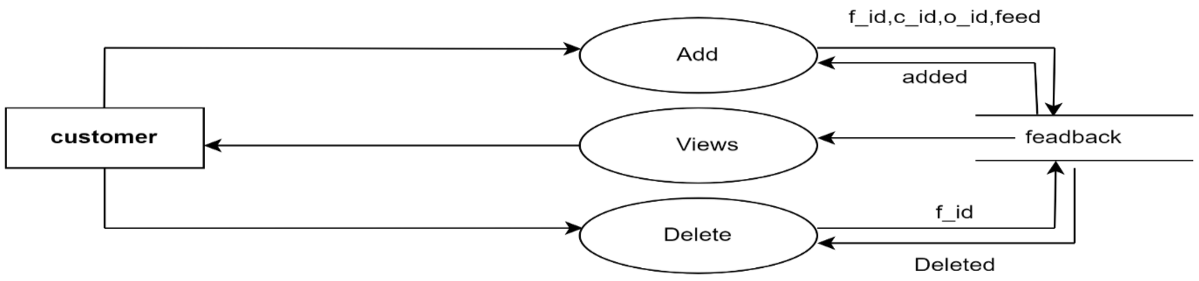
### 3.8.1 DFD level 2 (Views product)



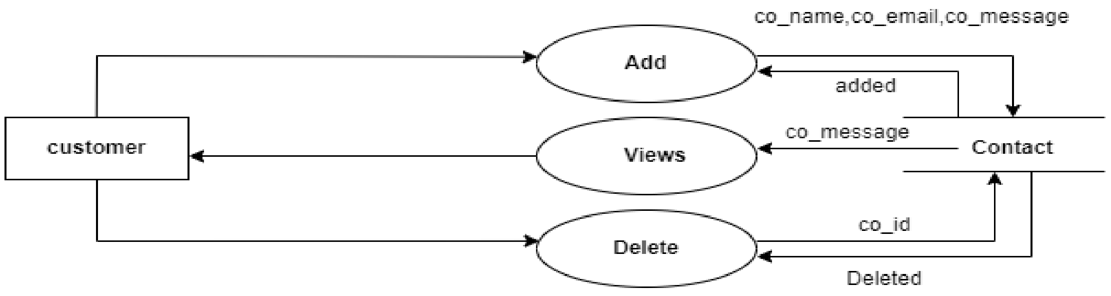
### 3.8.2 DFD level 2 (Manage order)



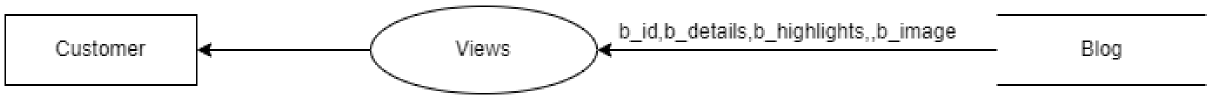
### 3.8.3 DFD level (feedback)



### 3.8.4 DFD level (Adds Contact)



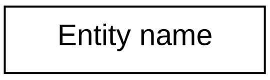
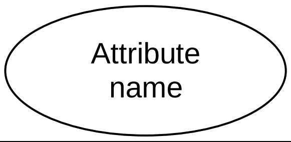
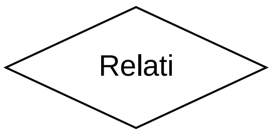
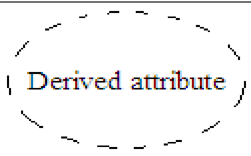
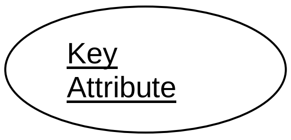
### 3.8.5 DFD level (blog)

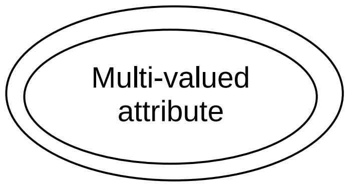


## 3.9 Entity-Relationship Diagram:

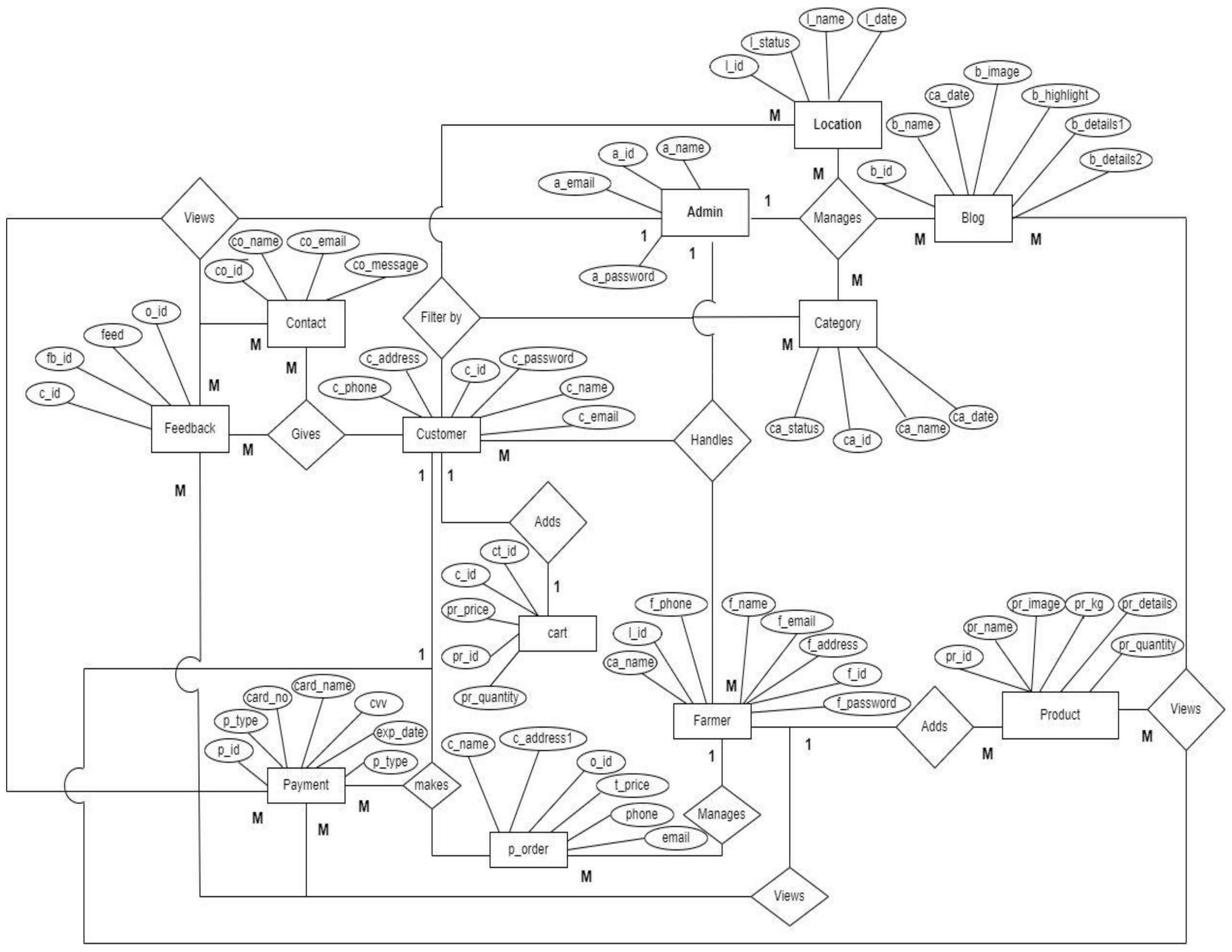
The basic objective of the ER model representation is an entity which is a "thing" in a real world with an independent existence. Entities are physical items or aggregations of data items that are important to the business we analyze or to the system; we intend to build. An entity represents an object defined within the information system about which you want to store information. Entities are named as singular nouns and are shown in rectangles in an ER-Diagram. Each entity is described by several attributes; individual instances of an entity will have different attribute values.

### 3.9.1 ER-Diagram Symbols:

| Name | Notation | Description |
|---|---|---|
| **Entity** |  | It may be an object with the physical existence or conceptual existence. It is represented by a Rectangle. |
| **Attribute** |  | The properties of the entity can be a attribute. It is represented by a Ellipse. |
| **Relationship** |  | Whenever an attribute of one entity refers to another entity, some relationship exists. It is represented by a Diamond. |
| **Link** | | Lines link attributes to entity sets and entity sets to relation. |
| **Derived Attribute** |  | Dashed ellipse denotes derived attributes. |
| **Key Attribute** |  | An entity type usually has an attribute whose values are distinct for each individual entry in the entity set. It is represented by a Underlined word in ellipse. |

| | | |
|---|---|---|
| **Multivalued Attribute** |  | Attributes that have different numbers of values for a particular attribute. It is represented by a Double ellipse represents multi-valued attributes. |
| **Cardinality Ratio** | 1) 1:1<br>2) 1:M<br>3) M:1<br>4) M:M | It specifies the maximum number of relationships instances that an entity can participate in. There are four cardinality ratios. |

## 3.9.2 <u>ER Diagram:</u>



# 4. Database Design

## 4.1 Introduction:

- **Database**: A Database is collection of related data, which can be of any size and complexity. By using the concept of Database, we can easily store and retrieve the data. The major purpose of a database is to provide the information, which utilizes it with the information's that the system needs according to its own requirements.

- **Database Design:** Database design is done before building it to meet needs of end users within a given information-system that the database is intended to support. The database design defines the needed data and data structures that such a database comprises the database is physically implemented using MySQL.

## 4.2 The database for "Farmigos" is organized into 11 tables:

- Admin
- blog
- customer
- category
- cart
- checkout
- contact
- farmer
- feedback
- location
- payment
- product
- p_order

Each entity can be described as follows along with its attributes:

## 4.3 Database Table Structure

### 4.3.1 Structure of Table "admin":

| Field Name | Field Type | Size | Constraints | Description |
|---|---|---|---|---|
| a_id | int | 11 | PRIMARY KEY | Admin ID |
| a_name | varchar | 50 | NOT NULL | Admin Name |
| a_email | varchar | 30 | NOT NULL | Login ID |
| a_password | varchar | 10 | NOT NULL | Password |

### 4.3.2 Structure of Table "blog":

| Field Name | Field Type | Size | Constraints | Description |
|---|---|---|---|---|
| b_id | int | 11 | Primary key | Blog id |
| b_name | varchar | 200 | Not null | Blog name |
| b_image | varchar | 100 | Not null | Blog image |
| b_details1 | Longtext | - | Not null | Blog primary details |
| b_details2 | Longtext | - | Not null | Blog secondary details |
| b_highlight | Longtext | - | Not null | Blog highlight |
| b_date | Datetime | 6 | Not null | Blog date |

### 4.3.3 Structure of Table "cart":

| Field Name | Field Type | Size | Constraints | Description |
|---|---|---|---|---|
| ct_id | int | 11 | PRIMARY KEY | Cart ID |
| c_id | int | 11 | FOREIGN KEY | Customer id |
| pr_id | int | 11 | FOREIGN KEY | Product id |
| ct_quantity | int | 5 | NOT NULL | Cart quantity |
| pr_price | Big int | 10 | NOT NULL | Cart price |
| ct_date | date | 6 | NOT NULL | Date |
| ct_status | varchar | 10 | NOT NULL | status |

### 4.3.4 Structure of Table "catogory":

| Field Name | Field Type | Size | Constraints | Description |
|---|---|---|---|---|
| ca_id | int | 11 | PRIMARY KEY | Category ID |
| ca_name | varchar | 50 | NOT NULL | Category Name |
| ca_date | date | 6 | NOT NULL | Date |
| ca_status | varchar | 10 | NOT NULL | Status |

### 4.3.5 Structure of Table "checkout":

| Field Name | Field Type | Size | Constraints | Description |
|---|---|---|---|---|
| ch_id | int | 11 | PRIMARY KEY | checkout ID |
| o_id | int | 11 | FOREIGN KEY | Order ID |
| c_id | int | 11 | FOREIGN KEY | Customer ID |
| f_name | varchar | 20 | FOREIGN KEY | Farmer Name |
| l_name | varchar | 20 | FOREIGN KEY | Location Name |
| ch_address | varchar | 40 | NOT NULL | Address |
| ch_email | varchar | 20 | NOT NULL | Email ID |
| ch_country | varchar | 20 | NOT NULL | Country name |
| ch_city | varchar | 20 | NOT NULL | City Name |
| ch_state | varchar | 20 | NOT NULL | State Name |
| ch_zip | int | 11 | NOT NULL | Zip code |
| ch_phone | bigint | 10 | NOT NULL | Phone No |
| ch_date | date | 6 | NOT NULL | Date |
| ch_status | varchar | 20 | NOT NULL | Status |

### 4.3.6 Structure of Table "Contact":

| Field Name | Field Type | Size | Constraints | Description |
|---|---|---|---|---|
| co_id | int | 11 | Primarykey | Contact id |
| co_name | varchar | 30 | Not null | Contact name |
| co_email | varchar | 30 | Not null | Contact email |
| co_message | longtext | _ | Not null | Contact message |

### 4.3.7 Structure of Table "customer":

| Field Name | Field Type | Size | Constraints | Description |
|---|---|---|---|---|
| c_id | int | 11 | Primary key | Customer ID |
| c_name | varchar | 20 | Not null | Customer Name |
| c_address | varchar | 50 | Not null | Address |
| c_email | varchar | 20 | Not null | Email ID |

| | | | | |
|---|---|---|---|---|
| c_password | Big int | 20 | Not null | Password |
| c_phone | Big int | 10 | Not null | Phone number |
| b_date | Datetime | 6 | Not null | Blog date |

## 4.3.8 Structure of Table “farmer”:

| Field Name | Field Type | Size | Constraints | Description |
|---|---|---|---|---|
| f_id | int | 11 | Primary key | Farmer id |
| l_id | int | 11 | Forign key | Location id |
| f_name | varchar | 20 | Not null | Farmer name |
| f_email | varchar | 50 | Not null | Email ID |
| f_address | varchar | 10 | Not null | Address |
| f_phone | bigint | 10 | Not null | Phone number |

## 4.3.9 Structure of Table “feedback”:

| Field Name | Field Type | Size | Constraints | Description |
|---|---|---|---|---|
| fb_id | int | 11 | PRIMARYKEY | Feedback id |
| c_id | int | 11 | FOREIGN KEY | Customer id |
| o_id | int | 11 | FOREIGN KEY | Order id |
| fb_name | varchar | 30 | NOT NULL | Feedback Name |
| fb_email | varchar | 10 | NOT NULL | Feedback Email |
| fb_details | varchar | 100 | NOT NULL | Feedback details |
| fb_image | datetime | 6 | NOT NULL | Feedback Image |
| fb_date | varchar | 10 | NOT NULL | Feedback Date |

## 4.3.10 Structure of Table “location”:

| Field Name | Field Type | Size | Constraints | Description |
|---|---|---|---|---|
| l_id | int | 11 | Primary key | Location ID |
| l_name | varchar | 50 | Not null | Location name |

| | | | | |
|---|---|---|---|---|
| l_date | datetime | 6 | Not null | Location date |
| l_status | varchar | 10 | Not null | Location status |

### 4.3.11 Structure of Table “payment”:

| Field Name | Field Type | Size | Constraints | Description |
|---|---|---|---|---|
| p_id | int | 11 | PRIMARY KEY | Payment id |
| o_id | int | 11 | FOREIGN KEY | Order id |
| c_id | int | 11 | FOREIGN KEY | Customer id |
| f_id | int | 11 | FOREIGN KEY | Farmer id |
| p_type | varchar | 10 | NOT NULL | Payment type |
| card_name | varchar | 20 | NOT NULL | Card name |
| card_no | int | 16 | NOT NULL | Card number |
| cvv | int | 3 | NOT NULL | CVV number |
| exp_date | date | 11 | NOT NULL | Expiry date |
| transaction_no | int | 20 | NOT NULL | Transaction number |
| p_date | datetime | 6 | NOT NULL | Payment date |
| p_status | varchar | 10 | NOT NULL | Payment status |
| p_amt | bigint | 10 | NOT NULL | Payment amount |

### 4.3.12 Structure of Table “Product”:

| Field Name | Field Type | Size | Constraints | Description |
|---|---|---|---|---|
| pr_id | int | 11 | PRIMARYKEY | Product id |
| f_id | int | 11 | FOREIGN KEY | Farmer id |
| ca_id | int | 11 | FOREIGN KEY | Category id |
| pr_name | varchar | 30 | NOT NULL | Product Name |
| pr_kg | varchar | 10 | NOT NULL | Price per kg |
| pr_details | varchar | 100 | NOT NULL | Product details |
| pr_date | datetime | 6 | NOT NULL | Product Date |
| pr_status | varchar | 10 | NOT NULL | Product Status |
| pr_quantity | int | 5 | NOT NULL | Product quantity |
| pr_image | varchar | 300 | NOT NULL | Product image |

### 4.3.13 Structure of Table “p_order”:

| **Field Name** | **Field Type** | **Size** | **Constraints** | **Description** |
|---|---|---|---|---|
| o_id | int | 11 | PRIMARYKEY | Order id |
| c_id | int | 11 | FOREIGN KEY | Customer id |
| pr_id | int | 11 | FOREIGN KEY | Product id |
| qty | int | 11 | NOT NULL | Quantity |
| t_price | int | 10 | NOT NULL | Total Price |
| o_status | varchar | 10 | NOT NULL | Order status |
| o_date | datetime | 6 | NOT NULL | Order date |

# 5. Detailed Design

## 5.1 Introduction

The purpose of preparing this document is to explain complete design details of Classic Chicken farm. This detailed design report will mainly contain the general definition and features of the project, design constraints,the overall system architecture and data architecture. Additionally, a brief explanation about our current progress and schedule of the project will be provided in related sections. Design of the system and subsystems/modules will be explained both verbally and visually by means of diagrams in order to help the programmer to understand all information stated in this document correctly and easily.

## 5.2 Applicable documents

The documents used during detailed design are:

- System Requirements Document
- System Design
- Database Design

## 5.3 Structure of the software package

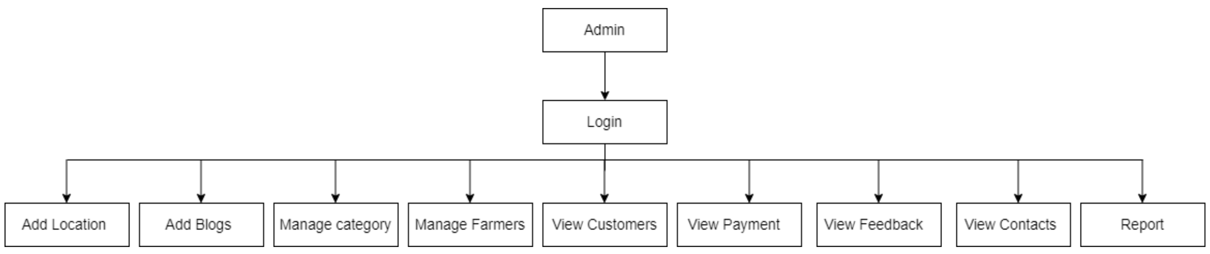
The Components are:

- Registration component for Customers, Farmers.
- Login component for Customers, Farmer, Admin.
- Product component for customers, Farmers, Admin.
- Purchase component for Customers.
- Category selection components for Customers, Farmers.

## 5.4 Modular decomposition of components

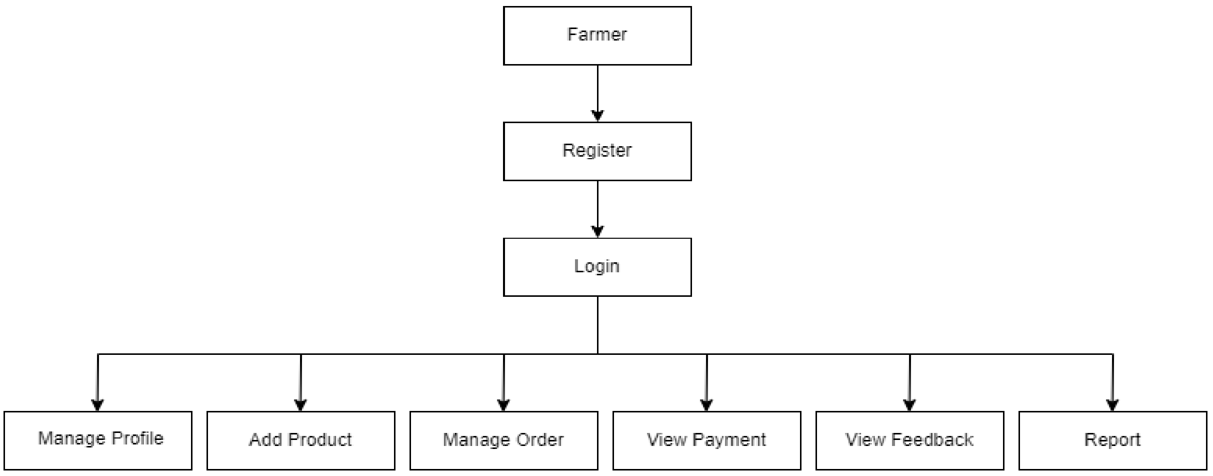
### 5.4.1 Admin component

#### 5.4.1.1 Structure chart for Admin:



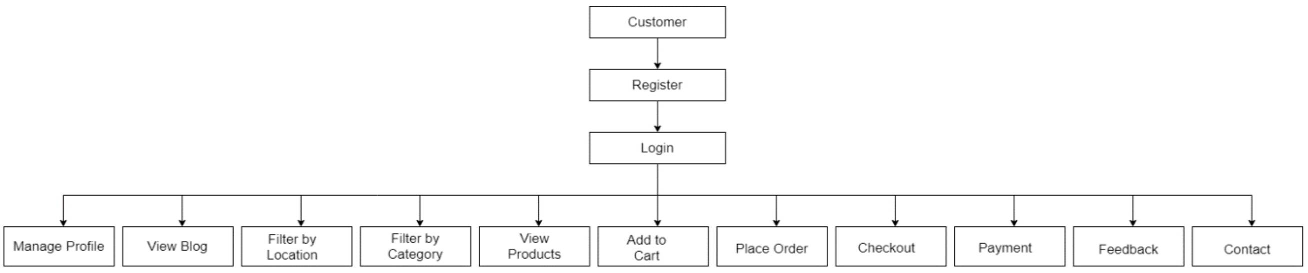
## 5.4.2 Farmer component

### 5.4.2.1 Structure chart for Farmer:



## 5.4.3 Customer component

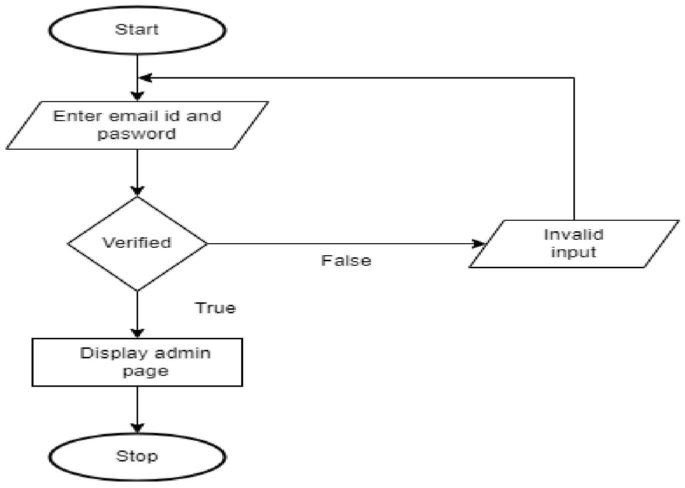
### 5.4.3.1 Structure chart for Customer:



## 5.4.4 Procedural details of Admin

### 5.4.4.1 LOGIN:

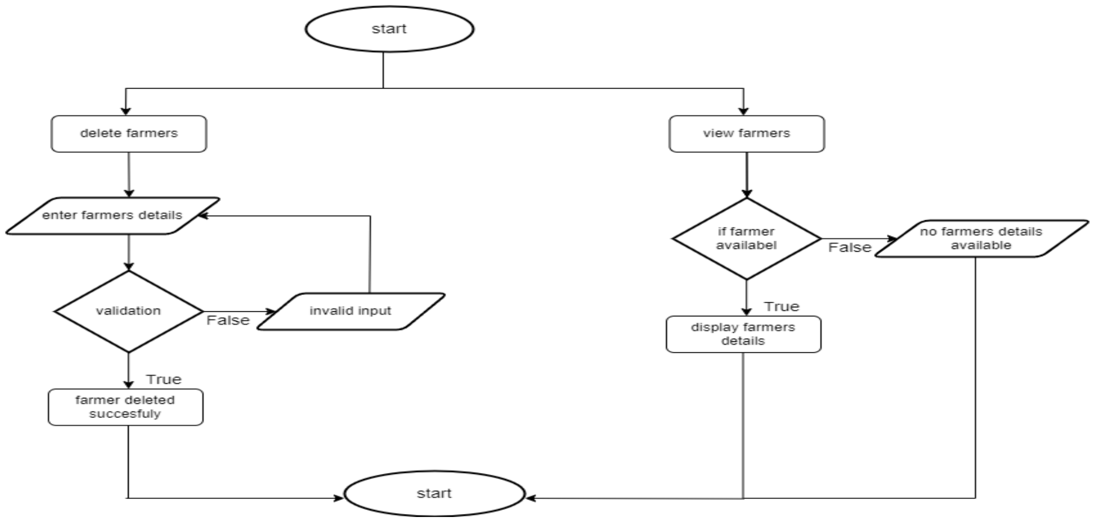
**Input:** Email_id,password
Procedural details:



File I/O interface: Admin table
Output: Email id and password will be checked for validity. If it is valid admin will be directed to adminpage.

### 5.4.4.2 <u>Manage Farmers:</u>

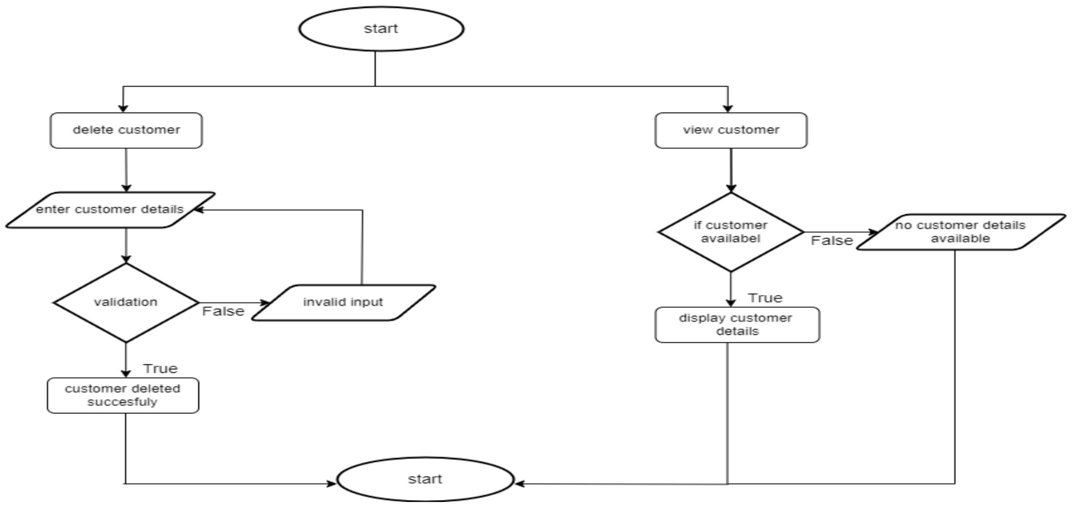
**Input:** f_id
Procedural details:



File I/O interface: Farmer table
Output: Admin can view Farmers details and admin can delete Farmers from the list.

### 5.4.4.3 <u>Manage Customer:</u>

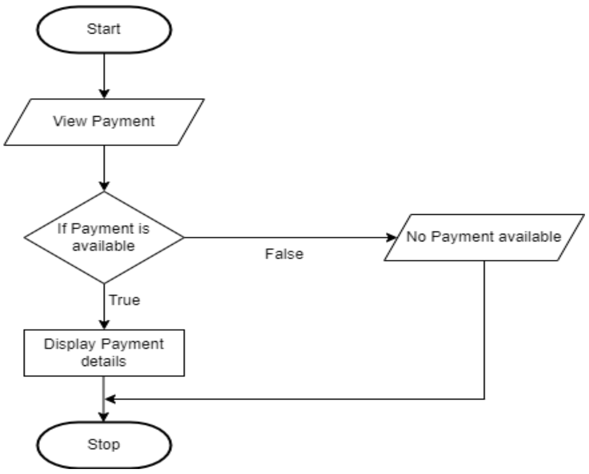
**Input:** c_id
Procedural details:



File I/O interface: Customer table
Output: Admin can view customer details and admin can delete the Customer from the list.

### 5.4.4.4 <u>View Payment:</u>

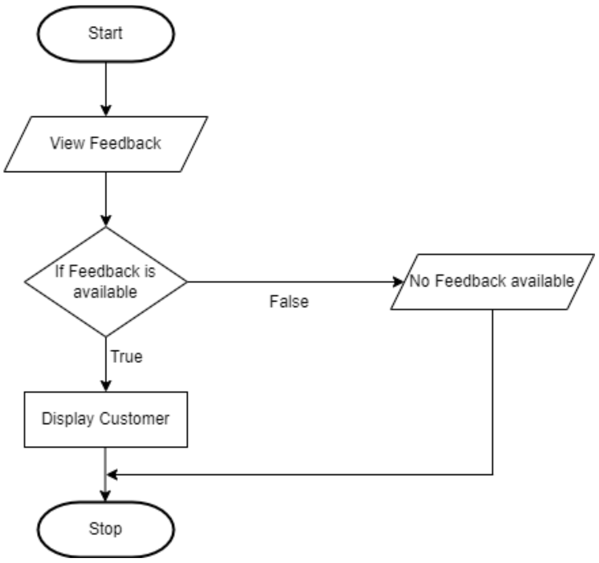
**Input:** p_id
**Procedural details:**



File I/O interface: Payment table
Output: Admin can view the payment details here.

### 5.4.4.5 View Feedback:

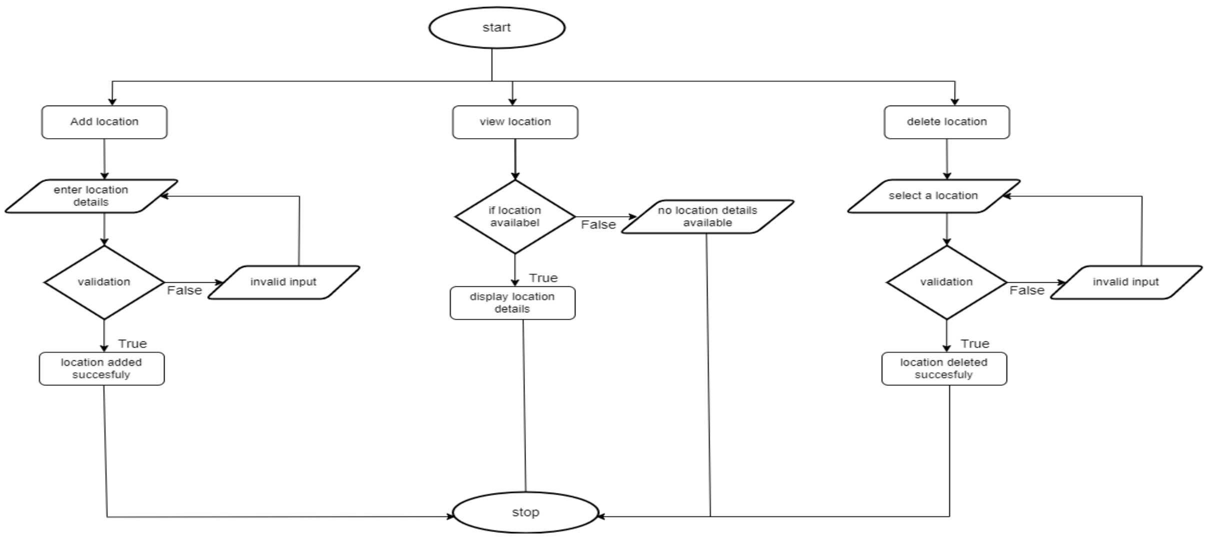
**Input:** feed_id
**Procedural details:**



File I/O interface: Feedback table
Output: Admin can view the feedback details here.

### 5.4.4.6 Add location:

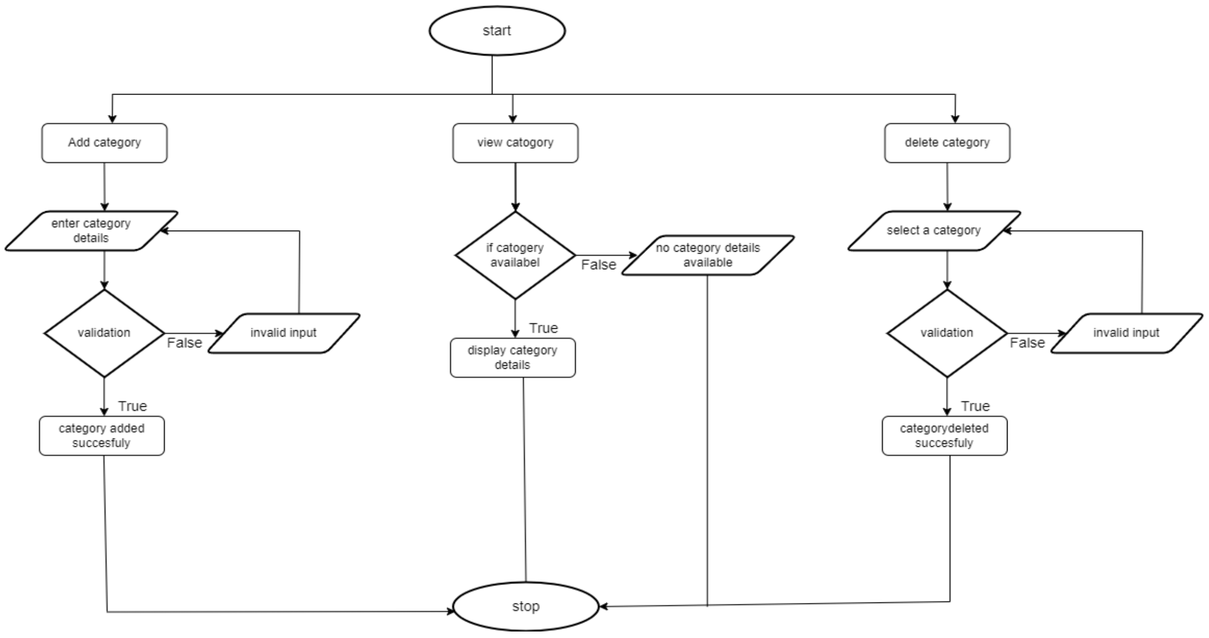
**Input:** l-id, l-name
**Procedural details:**



File I/O interface: Location table
Output: Admin can Add the Location or delete the location here.

## 5.4.4.7 Manage category:

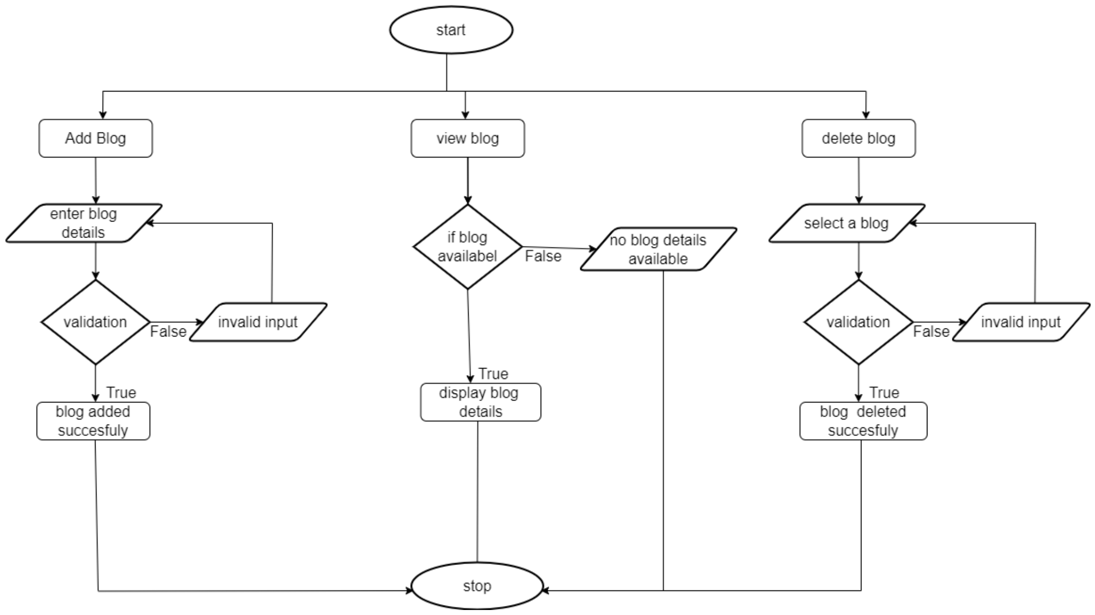
**Input:** ca_id
**Procedural details:**



File I/O interface: Category table
Output: Admin can Add the category or delete the category here.

## 5.4.4.8 Add Blogs:

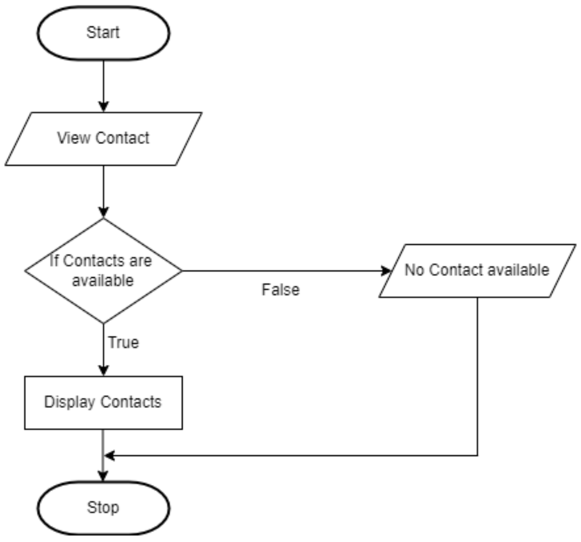
**Input:** b-id, b_name
**Procedural details:**



File I/O interface: blog table
Output: Admin can Add the blog or delete the blog here.

#### 5.4.4.9 View contacts:

**Input:** co_id
**Procedural details:**

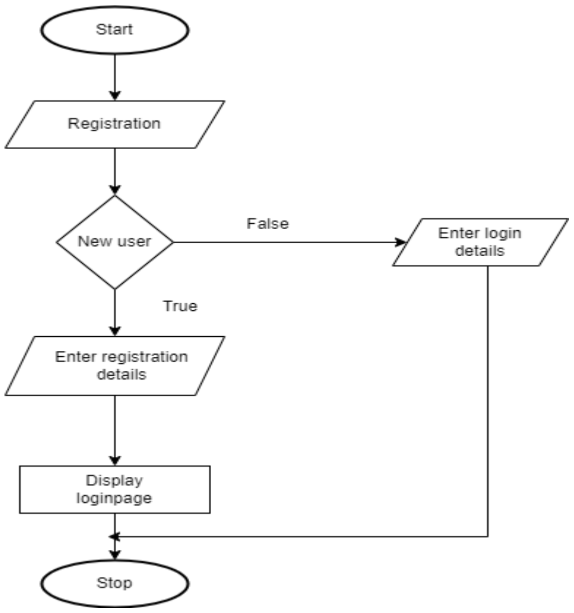


File I/O interface: contact table
Output: Admin can view contacts made by customers here.

### 5.4.6 Procedural details of Farmer

#### 5.4.6.1 Registration:

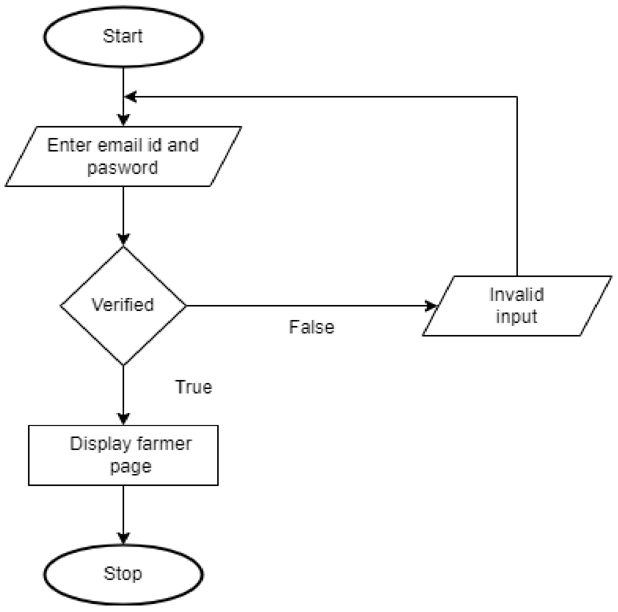
**Input:** Name,phone,address, email id, password
**Procedural details**:



File I/O interface: Farmer table
Output: here Farmer can give his required details and register.

### 5.4.6.2 LOGIN:

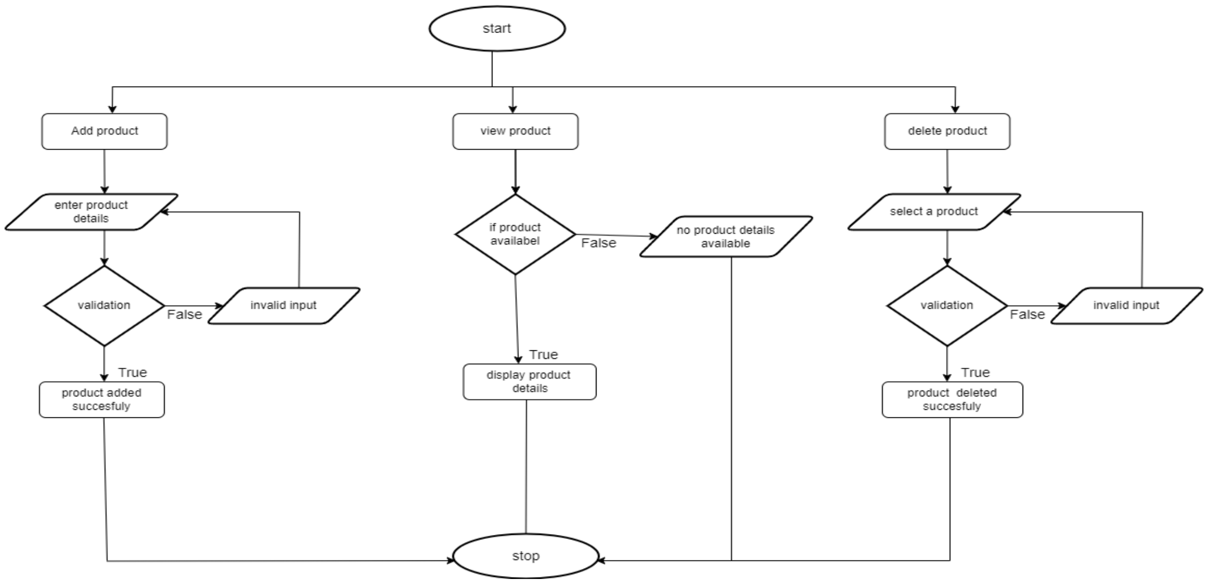
**Input:** email id, password
Procedural details:



File I/O interface: farmer table
Output: Email id and password will be checked for validity. If it is valid farmer will be directed to farmerpage.

### 5.4.6.3 Add Product

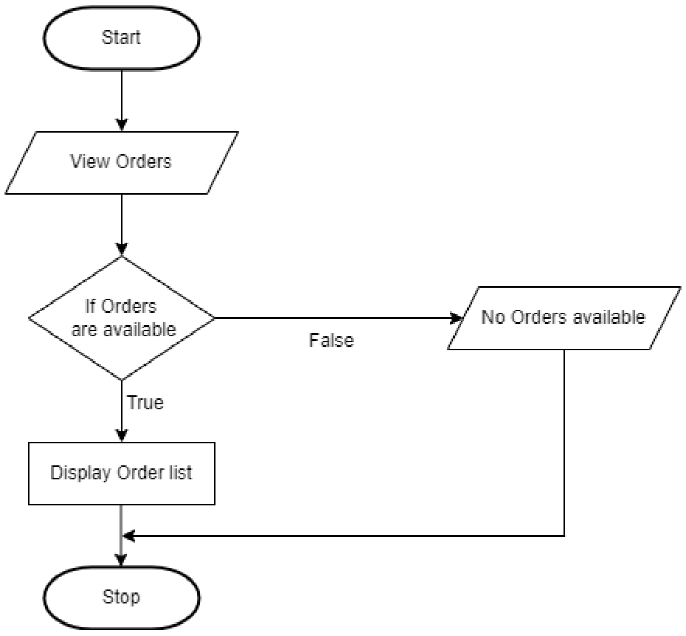
**Input:** pr_id,pr_name,pr_quantity,pr_kg,pr_details,pr_image
Procedural details:



File I/O interface: product table
Output: Farmer can Add the product or delete the product here.

### 5.4.6.4 Manage Order

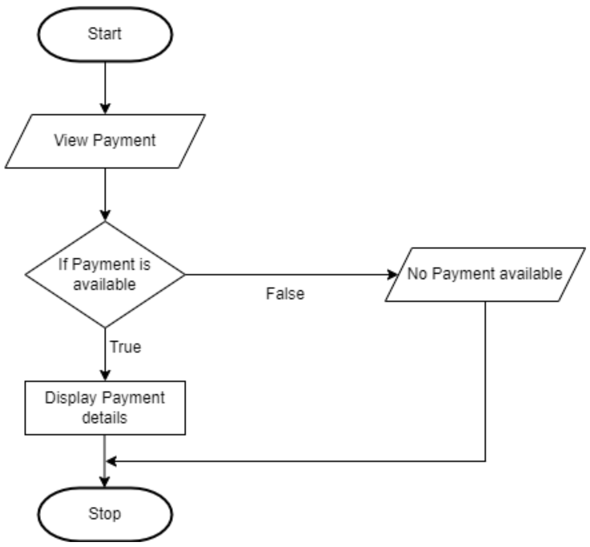
**Input:** pr_id
Procedural details:



File I/O interface: p_order table
Output: Farmer can manage the orders here.

### 5.4.6.2 View Payment

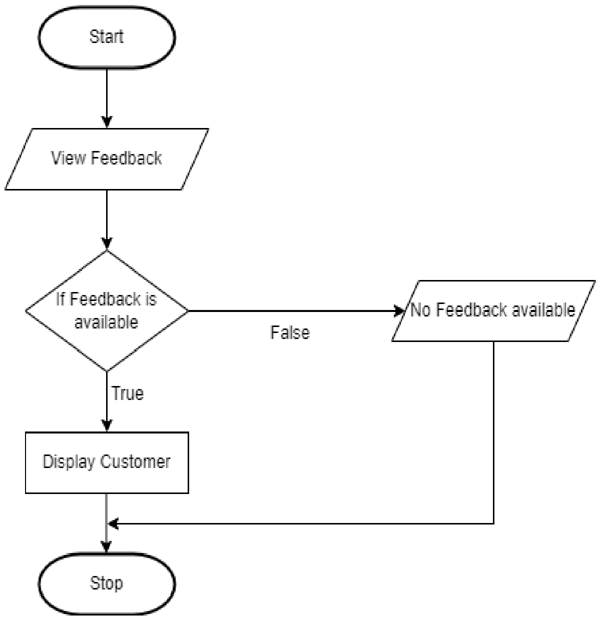
Input: p_id
Procedural details:



File I/O interface: Payment table
Output: Farmer can view the payment details here.

### 5.4.6.6 <u>View Feedback:</u>

**Input:** feed_id
**Procedural details:**

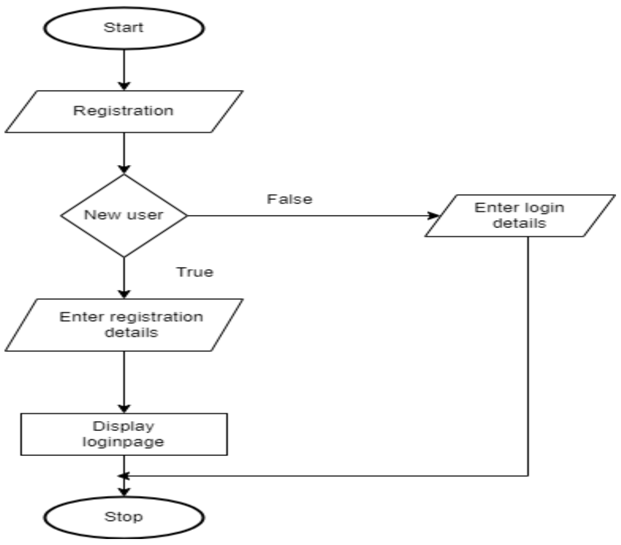


File I/O interface: Feedback table
Output: Farmer can view the feedback details here.

## 5.4.7 <u>Procedural details of Customer</u>

### 5.4.7.1 <u>Registration:</u>

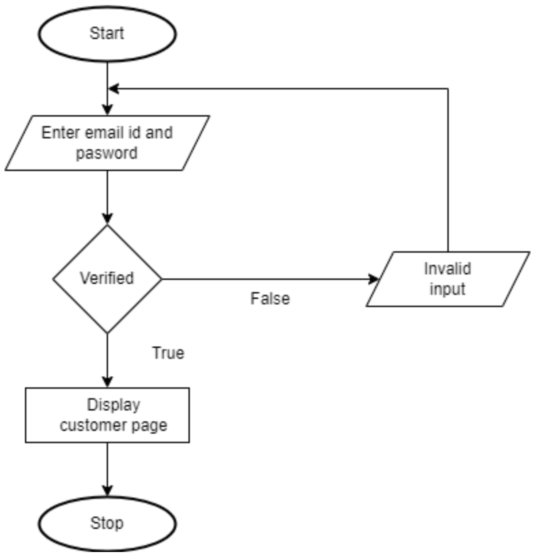
**Input:** Name, phone, address, email id, password
**Procedural details**:



File I/O interface: Customer table
Output: here customer can give his required details and register.

### 5.4.7.2 Login:

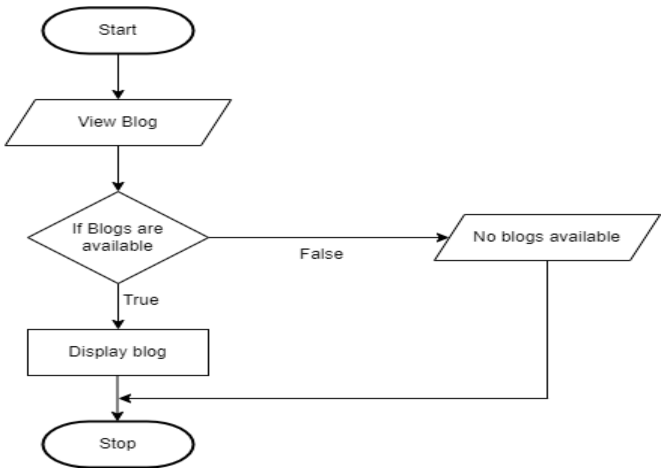
**Input:** email id, password
Procedural details:



File I/O interface: customer table
Output: Email id and password will be checked for validity. If it is valid customer will be directed to customerpage.

### 5.4.7.3 View blog

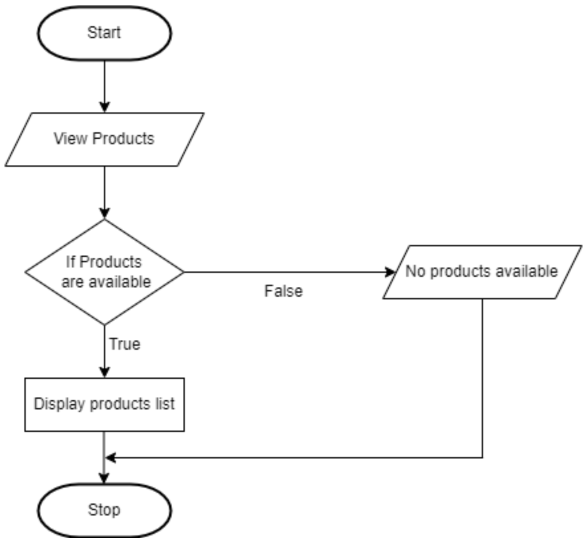
**Input:** b_id,b_details
Procedural details:



File I/O interface: blog table
Output: Customer can view the blogs here.

### 5.4.7.4 View products

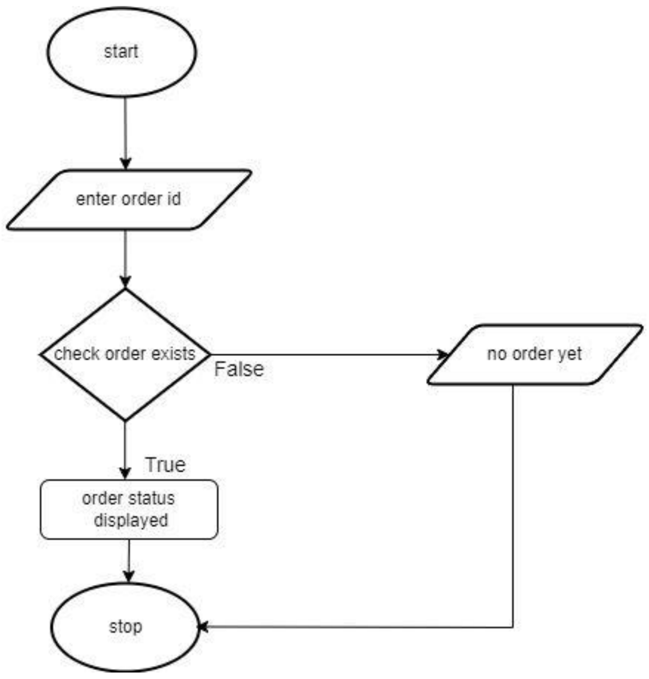
**Input:** pr_id
Procedural details:



File I/O interface: Product table
Output: Customer can view the Product details here.

### 5.4.7.5 Track Order

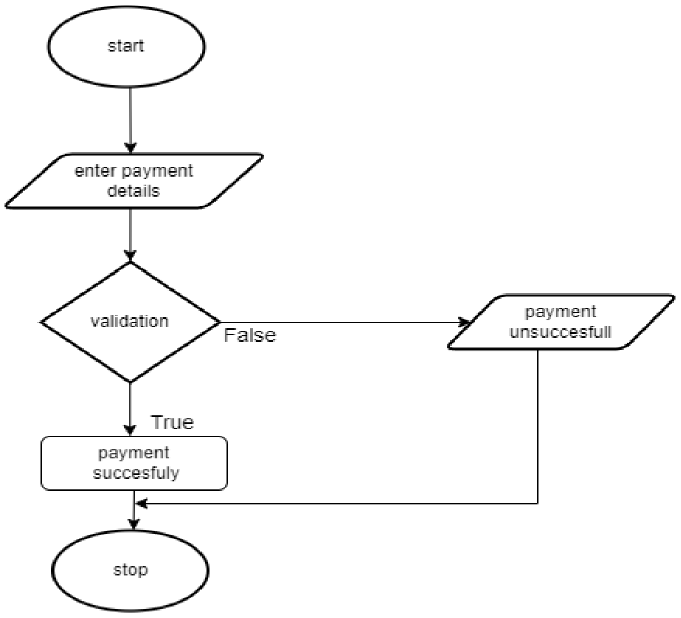
**Input:** ch_id,
Procedural details:



File I/O interface: Checkout table
Output: Customer can Track order details here.

### 5.4.7.6 Make Payment

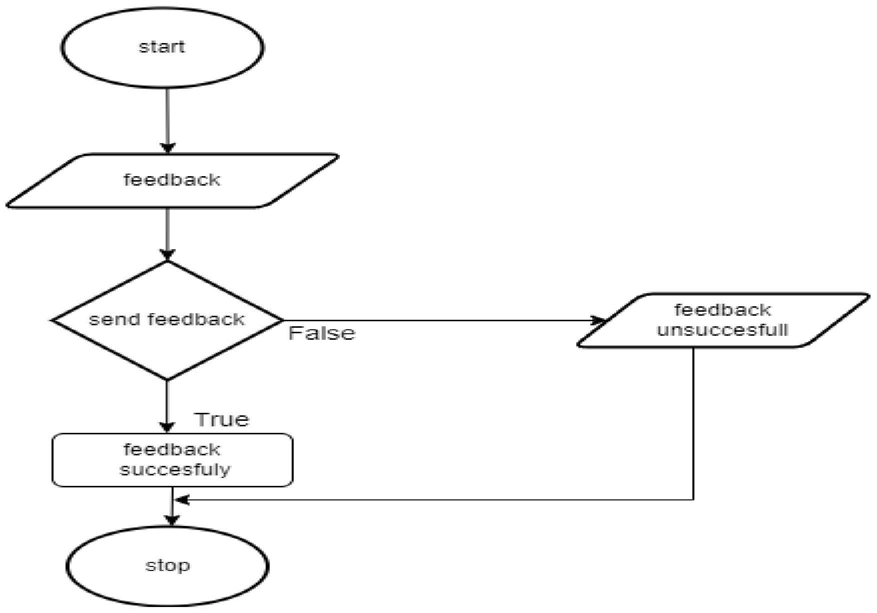
**Input:** p_id,
Procedural details:



File I/O interface: Payment table
Output: Customer can make payment here.

### 5.4.7.7 Give Feedback

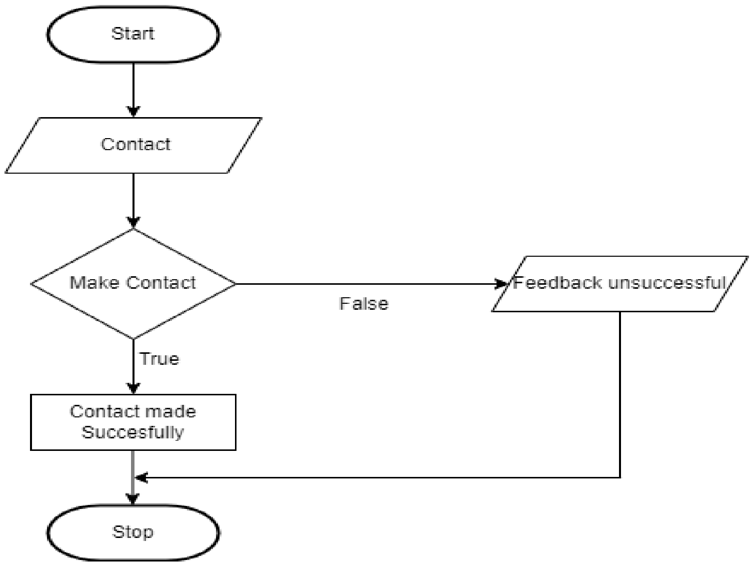
**Input:** feed_id,
Procedural details:



File I/O interface: Feedback table
Output: Customer can give feedback here.

### 5.4.7.8 <u>Contact</u>

**Input:** co_id,co_name , co_message
Procedural details:



File I/O interface: Contact table
Output: Customer can make contact here.

# 6. PROGRAM CODE LISTING

## 6.1 Database connection

The function to connect to MYSQL is called MYSQL connect. This function returns resource which is a pointer to the database connection.

```
 <?php
ob_start();
session_start();
// database connection variables
define('DB_SERVER', "localhost:3306"); // database host name
define('DB_USER', "root"); // database user name eg. root
define('DB_DATABASE', "farmigo"); //database name
define('DB_PASSWORD', ""); //database user password
define('DB_TYPE', 'mysql'); //database drive eg. mysql, pgsql, mongodb etc

// site details described here
define('SITE_TITLE', 'demo.com');
define('SITE_TAG_LINE', 'give your tag line of your project here');
define('SITE_CONTACT', 'your number'); //contact information
define('SITE_EMAIL_INFO', 'your mail id'); //email information
define('BASE_URL', 'http://localhost/cud opertaion/'); //url information

// included main class
require_once 'app/Main.php';
require_once 'app/Controller.php';
require_once 'app/Admin.php';
```

## 6.2 Admin

### 6.2.1 Login:

```
<?php include '../../config.php';
$admin= new Admin();
if(isset($_POST['login']))
{
   $email=$_POST['email'];
   $password=$_POST['password'];
   $stmt=$admin->ret("SELECT * FROM `admin` WHERE `a_email`='$email' AND
`a_password`='$password' ");
   $row=$stmt->fetch(PDO::FETCH_ASSOC);
   $num=$stmt->rowCount();
   if($num>0){
      $id=$row['a_id'];
      $_SESSION['a_id']=$id;
      echo "<script> alert('Login success');window.location.href='../index.php';</script>";

   }else{
      echo"<script> alert('Invalid User');window.location.href='../login.php';</script>";
   }
}
?>
```

### 6.2.2 Add Blog

```
<?php
include '../../config.php';
$admin=new Admin();

if(isset($_POST['add'])){
   $title=$_POST['title'];
   $about1=$_POST['about1'];
   $about2=$_POST['about2'];
   $highlight=$_POST['highlight'];

   $target="upload/";
   $image=$target.basename($_FILES['img']['name']);
   move_uploaded_file($_FILES['img']['tmp_name'],$image);


   $stmt=$admin->cud("INSERT INTO
`blog`(`b_name`,`b_image`,`b_details1`,`b_details2`,`b_highlight`,`b_date`)VALUES('$title','
$image','$about1','$about2','$highlight',now())","saved");
    echo "<script>window.location='../viewblog.php';</script>";
}
?>
```

### 6.2.3 Update Blog

```
<?php
include '../../config.php';
$admin= new Admin();
if(isset($_POST['add'])){
   $name=$_POST['title'];
   $high=$_POST['highlight'];
   $details1=$_POST['about1'];
   $details2=$_POST['about2'];
   $id=$_POST['bid'];
   $target="upload/";
   $image=$target.basename($_FILES['img']['name']);
   move_uploaded_file($_FILES['img']['tmp_name'],$image);
   $stmt=$admin->cud("UPDATE `blog` SET
`b_id`='$id',`b_name`='$name',`b_image`='$image',`b_highlight`='$high',`b_details1`='$detai
ls1',`b_details2`='$details2' WHERE `b_id`='$id' ","Update");
      echo"<script>alert('Data Updated
succesfully');window.location.href='../viewblog.php';</script>";

}
?>
```

### 6.2.4 Delete Blog

```
<?php include '../../config.php';
   $admin=new Admin();
   $caid=$_GET['bid'];
   $stmt=$admin->cud("DELETE FROM `blog` WHERE `b_id`='$caid'","delete");
   echo "<script>alert('Data deleted
successfully');window.location.href='../viewblog.php';</script>";
   ?>
```

### 6.2.5 Insert Category

```
<?php
include '../../config.php';
$admin= new Admin();
if(isset($_POST['add'])){
   $name=$_POST['ca_name'];
   $stmt=$admin->cud("INSERT INTO
`category`(`ca_name`)VALUES('$name')","Inserted");
    echo"<script>alert('Data inserted
succesfully');window.location.href='../category.php';</script>";
}
?>
```

### 6.2.6 Update Category

```
<?php
include '../../config.php';
$admin= new Admin();
if(isset($_POST['add'])){
   $name=$_POST['ca_name'];
   $id=$_POST['caname'];

   $stmt=$admin->cud("UPDATE `category` SET `ca_id`='$id',`ca_name`='$name' WHERE
`ca_id`='$id' ","Update");
     echo"<script>alert('Data Updated
succesfully');window.location.href='../category.php';</script>";

}
?>
```

### 6.2.7 Delete Category

```
  <?php include '../../config.php';
   $admin=new Admin();
   $caid=$_GET['caid'];
   $stmt=$admin->cud("DELETE FROM `category` WHERE `ca_id`='$caid'","delete");
   echo "<script>alert('Data deleted
successfully');window.location.href='../category.php';</script>";
   ?>
```

### 6.2.8 Insert Location

```
   <?php include '../../config.php';
   $admin=new Admin();
   $caid=$_GET['caid'];
   $stmt=$admin->cud("DELETE FROM `category` WHERE `ca_id`='$caid'","delete");
   echo "<script>alert('Data deleted
successfully');window.location.href='../category.php';</script>";
   ?>
```

### 6.2.9 Update Location

```
<?php
include '../../config.php';
$admin= new Admin();
if(isset($_POST['add'])){
   $name=$_POST['l_name'];
   $id=$_POST['laname'];

   $stmt=$admin->cud("UPDATE `location` SET `l_id`='$id',`l_name`='$name' WHERE
`l_id`='$id' ","Update");
     echo"<script>alert('Data Updated
succesfully');window.location.href='../location.php';</script>";
}
?>
```

### 6.2.10 Delete Location

```
 <?php include '../../config.php';
$admin=new Admin();
$lid=$_GET['lid'];
$stmt=$admin->cud("DELETE FROM `location` WHERE `l_id`='$lid'","delete");
 echo "<script>alert('Data deleted
successfully');window.location.href='../location.php';</script>";
```

### 6.2.11 Delete Farmer

```
<?php include '../../config.php';
$admin=new Admin();
$fid=$_GET['fid'];
$stmt=$admin->cud("DELETE FROM `farmer` WHERE `f_id`='$fid'","delete");
 echo "<script>alert('Data deleted
successfully');window.location.href='../farmer.php';</script>";
?>
```

### 6.2.10 Delete Customer

```
<?php include '../../config.php';
$admin=new Admin();
$cid=$_GET['cid'];
$stmt=$admin->cud("DELETE FROM `customer` WHERE `c_id`='$cid'","delete");
 echo "<script>alert('Data deleted
successfully');window.location.href='../customer.php';</script>";
?>
```

### 6.2.12 Logout

```
<?php
session_start();
session_destroy();
header('Location:../login.php');
?>
```

## 6.3 Farmer

### 6.3.1 Register:

```
 <?php
include '../../config.php';
$admin= new Admin();
if(isset($_POST['submit'])){

   $email=$_POST['f_email'];
   $uname=$_POST['u_name'];
   $password=$_POST['f_password'];
   $name=$_POST['f_name'];
   $gender=$_POST['f_gender'];
   $city=$_POST['city'];
   $lid=$_POST['loc'];
   $state=$_POST['f_state'];
   $phone=$_POST['f_phone'];
   $stmt=$admin->cud("INSERT INTO
`farmer`(`f_email`,`u_name`,`f_password`,`f_name`,`f_gender`,`f_city`,`l_id`,`f_state`,`f_pho
ne`)VALUES('$email','$uname','$password','$name','$gender','$city','$lid','$state','$phone')","
Inserted");
   echo"<script>alert('Registration
Sucessfull');window.location.href='../index.php';</script>";
}
?>
```

### 6.3.2 Login:

```
<?php include '../../config.php';
$admin= new Admin();
if(isset($_POST['login']))
{
   $uname=$_POST['u_name'];
   $password=$_POST['f_password'];
   $stmt=$admin->ret("SELECT * FROM `farmer` WHERE `u_name`='$uname' AND
`f_password`='$password' ");
   $row=$stmt->fetch(PDO::FETCH_ASSOC);
   $num=$stmt->rowCount();
   if($num>0){
      $id=$row['f_id'];
      $_SESSION['F_id']=$id;
      echo "<script> alert('Login success');window.location.href='../index.php';</script>";
   }
   else{
      echo"<script> alert('Invalid User');window.location.href='../login.php';</script>";
   }
}
?>
```

### 6.3.3 Insert Product

```
<?php
include '../../config.php';
$admin= new Admin();
```

```
if(isset($_POST['add'])){
    $pc_name=$_POST['ca_id'];
    $pr_name=$_POST['pr_name'];
    $pr_kg=$_POST['pr_kg'];
    $fid = $_POST['fid'];
    $pr_details=$_POST['pr_details'];
    $pr_quantity=$_POST['pr_quantity'];
    $image_path="uploader/".basename($_FILES['image']['name']);
    move_uploaded_file($_FILES['image']['tmp_name'],$image_path);

    $stmt=$admin->cud("INSERT INTO
`product`(`ca_id`,`f_id`,`pr_name`,`pr_kg`,`pr_details`,`pr_quantity`,`pr_image`)VALUES('$
pc_name','$fid','$pr_name','$pr_kg','$pr_details','$pr_quantity','$image_path')","Inserted");
      echo"<script>alert('Data inserted
succesfully');window.location.href='../product.php';</script>";
}
?>
```

### 6.3.4 Update Product

```
<?php
include '../../config.php';
$admin= new Admin();
if(isset($_POST['add'])){
  $id=$_POST['prid'];
    $caname=$_POST['ca_name'];
    $name=$_POST['pr_name'];
    $prkg=$_POST['pr_kg'];
    $prdetails=$_POST['pr_details'];
    $prquantity=$_POST['pr_quantity'];
    $image_path="uploader/".basename($_FILES['image']['name']);
    move_uploaded_file($_FILES['image']['tmp_name'],$image_path);

    $stmt=$admin->cud("UPDATE `product` SET
ca_id='$caname',`pr_name`='$name',`pr_kg`='$prkg',`pr_details`='$prdetails',`pr_quantity`='
$prquantity',`pr_image`='$image_path' WHERE `pr_id`='$id' ","Update");
        echo"<script>alert('Data Updated
succesfully');window.location.href='../productview.php';</script>";

}
```

### 6.3.5 Delete Product

```
<?php include '../../config.php';
$admin=new Admin();
$prid=$_GET['prid'];
$stmt=$admin->cud("DELETE FROM `product` WHERE `pr_id`='$prid'","delete");
 echo "<script>alert('Data deleted
successfully');window.location.href='../product.php';</script>";
```

### 6.3.6 Change Order Status

```
<?php
```

```
    include '../../config.php';
    $admin = new Admin();

    if(isset($_GET['orderstatus']) ){
        $status = $_GET['orderstatus'];
        $odid = $_GET['odid'];
          $stmt = $admin -> cud("UPDATE `p_order` SET `o_status` = '$status' WHERE `o_id`
= '$odid'  ","updated");
          echo "<script>alert('Status changed
successfully.');window.location='../booking.php';</script>";

    }
    ?>
```

### 6.3.7 Logout

```
<?php
session_start();
session_destroy();
header('Location:../login.php');
?>
```

## 6.4 Customer

### 6.4.1 Register:

```
<?php
include '../../config.php';
$admin= new Admin();
if(isset($_POST['submit'])){
    $name=$_POST['c_name'];
    $email=$_POST['c_email'];
    $password=$_POST['c_password'];
    $address=$_POST['c_address'];
    $phone=$_POST['c_phone'];
    $stmt=$admin->cud("INSERT INTO
`customer`(`c_name`,`c_email`,`c_password`,`c_address`,`c_phone`)VALUES('$name','$ema
il','$password','$address','$phone')","Inserted");
     echo"<script>alert('Registration
Sucessfull');window.location.href='../index.php';</script>";
}
?>
```

### 6.4.2 Login:

```
<?php include '../../config.php';
$admin= new Admin();
if(isset($_POST['login']))
{
    $email=$_POST['email'];
    $password=$_POST['password'];
    $stmt=$admin->ret("SELECT * FROM `customer` WHERE `c_email`='$email' AND
`c_password`='$password' ");
    $row=$stmt->fetch(PDO::FETCH_ASSOC);
```

```
   $num=$stmt->rowCount();
   if($num>0){
      $id=$row['c_id'];
      $_SESSION['c_id']=$id;
      echo "<script> alert('Login success');window.location.href='../index.php';</script>";
   }else{
      echo"<script> alert('Invalid User');window.location.href='../login.php';</script>";
   }
}
?>
```

### 6.4.3 Cart

```
<?php include '../../config.php';
$admin=new Admin();

$quantity=$_POST['quantity'];
$pid=$_POST['pid'];
$cid=$_SESSION['c_id'];
$stmt=$admin->ret("SELECT * FROM `cart` WHERE `pr_id`='$pid'AND `c_id`='$cid'");
$row=$stmt->fetch(PDO::FETCH_ASSOC);
   $num=$stmt->rowCount();
   if($num>0){
      $updatedquant=0;
      $dbqty=$row['ct_quantity'];
      $updatedquant=$quantity+$dbqty;
      $stmt1=$admin->cud("UPDATE `cart` SET `ct_quantity`='$updatedquant'WHERE
`pr_id`='$pid'AND`c_id`='$cid'","updated");
       echo"<script>window.location.href='../cart.php';</script>";
   }else{
      $stmt2=$admin->cud("INSERT
INTO`cart`(`pr_id`,`c_id`,`ct_quantity`,`ct_date`)VALUES('$pid','$cid','$quantity',now())","s
aved");
       echo"<script>window.location.href='../cart.php';</script>";
   }
?>
```

### 6.4.4 Update Cart

```
<?php
include '../../config.php';
$admin=new Admin();

$cid=$_SESSION['c_id'];
$cartid=$_GET['cartid'];
$qty=$_GET['qty'];
$stmt=$admin->cud("UPDATE `cart` SET `ct_quantity`='$qty' WHERE `ct_id`=
'$cartid'","saved");
?>
<div class="card" id="tablecart">
            <div class="card-header">
               <h2>Shopping Cart</h2>
            </div>
            <?php
```

```
        $stmt4=$admin->ret("SELECT * FROM `cart` WHERE `c_id`='$cid'");
        $num=$stmt4->rowCount();
        if($num==0){
        ?>
           <h5 style="color:red;margin-left:500px">Your cart is empty!!</h5>
        <?php } else { ?>
           <div class="card-body">
              <div class="table-responsive" style="display: flex;flex-direction:column-reverse">
                 <table class="table table-bordered m-0">
                    <thead>
                       <tr>
                          <th class="text-center py-3 px-4" style="min-width: 400px;">Product
Name &amp; Details</th>
                          <th class="text-right py-3 px-4" style="width: 100px;">Price</th>
                          <th class="text-center py-3 px-4" style="width: 120px;">Quantity</th>
                          <th class="text-right py-3 px-4" style="width: 100px;">Total</th>
                          <th class="text-center align-middle py-3 px-0" style="width: 40px;"><a
href="#" class="shop-tooltip float-none text-light" title="" data-original-title="Clear cart"><i
class="ino ion-md-trash"></i></a></th>
                       </tr>
                    </thead>
                    <tbody>
                       <?php
                       $grandtotal=0;
                       $total=0;
                       $stmt = $admin->ret("SELECT * FROM `cart` INNER JOIN `product` ON
product.pr_id=cart.pr_id WHERE `c_id`='$cid'");
                       while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
                          $qty = $row['ct_quantity'];
                          $price = $row['pr_kg'];
                          $total = $qty * $price;
                          $grandtotal=$grandtotal+$total;
                       ?>
                          <tr>
                             <td class="p-4">
                                <div class="media" style="display: flex;flex-direction:column">
<img src="../Farmer/controller/<?php echo $row['pr_image'] ?>" style="width: 140px;" >
                                   <div class="media-body">
                                  <h6 style="margin-top: 5px;"><?php echo $row['pr_name'] ?></h6>
                                      <small>
  <span class="text-muted">Quantity: </span> <?php echo $row['ct_quantity'] ?> &nbsp;
                                      </small>
                                   </div>
                                </div>
                             </td>
    <td class="text-right font-weight-semibold align-middle p-4">₹<?php echo $row['pr_kg']
?></td>
  <td class="align-middle p-4">

                                <div class="col" style="display: flex;">
```

```
                <button onclick="decrement(<?php echo $row['ct_id'] ?>)">-</button>
              <input type="text" id="<?php echo $row['ct_id'] ?>" value="<?php echo
$row['ct_quantity'] ?>" name="quantity" readonly style="width: 50px;">
 <button onclick="increment(<?php echo $row['pr_quantity'] ?>,<?php echo $row['ct_id']
?>)">+</button>
      </div>
       </td>
   <td class="text-right font-weight-semibold align-middle p-4">₹<?php echo $total ?></td>
      <td class="text-center align-middle px-0"><a href="deletecart.php?cartid=<?php echo
$row['ct_id'] ?>" class="shop-tooltip close float-none text-danger" title="" data-original-
title="Remove">×</a></td>
                       </tr>
                    <?php } ?>
                 </tbody>
                 <div style="display: flex;flex-direction:column">
                <div class="d-flex flex-wrap justify-content-between align-items-center pb-4">
                    <div class="d-flex">
                       <div class="text-right mt-4" style="margin-left:960px">
                         <label class="text-muted font-weight-normal m-0">Total price</label>
                  <div class="text-large"><strong>₹<?php echo $grandtotal ?></strong></div>
                       </div>
                    </div>
                 </div>
                 <div class="float-right" style="margin-left: 700px;">
                <a href="index.php" class="btn btn-lg btn-default md-btn-flat mt-2 mr-3">Back
to shopping</a>
                   <a href="checkout.php" class="btn btn-lg btn-primary mt-2">Checkout</a>
                 </div>
                 </div>
              </table>
            </div>
         </div>
         <?php } ?>
       </div>
```

## 6.4.5 Delete Cart

```
<?php include '../../config.php';
$admin=new Admin();
$cartid=$_GET['cartid'];
$stmt=$admin->cud("DELETE FROM `cart` WHERE `ct_id`='$cartid'","Deleted");
echo"<script>alert('Item removed from cart
succesfully');window.location.href='../cart.php';</script>";
?>
```

## 6.4.6 Checkout

```
<?php include '../../config.php';
$admin=new Admin();
$cid=$_SESSION['c_id'];

if(isset($_POST['checkout'])){
   $fname=$_POST['fname'];
```

```
    $lname=$_POST['lname'];
    $email=$_POST['email'];
     $phone=$_POST['phone'];
    $address=$_POST['address'];
    $country=$_POST['country'];
     $city=$_POST['city'];
    $state=$_POST['state'];
    $zip=$_POST['zip'];
    $paymethod=$_POST['payment_method'];
    $transaction=$_POST['transaction'];
    $cardname=$_POST['cardname'];
    $cardnumber=$_POST['cardnumber'];
    $expiry=$_POST['expiry'];
    $cvv=$_POST['cvv'];
    $stmt=$admin->ret("SELECT * FROM `cart` INNER JOIN `product` ON
product.pr_id=cart.pr_id WHERE `c_id`='$cid'");
    while($row=$stmt->fetch(PDO::FETCH_ASSOC)){
        $pdid=$row['pr_id'];
        $cqty=$row['ct_quantity'];
        $price=$row['pr_kg'];
        $fid= $row['f_id'];
        $total=$cqty*$price;
    $stmt2=$admin->Rcud("INSERT INTO
`p_order`(`c_id`,`pr_id`,`qty`,`t_price`,`o_status`,`o_date`)VALUES('$cid','$pdid','$cqty','$tot
al','ordered',now())");
    $stmt3=$admin->cud("INSERT INTO
`payment`(`o_id`,`f_id`,`c_id`,`p_type`,`card_name`,`card_no`,`cvv`,`exp_date`,`transaction_
no`,`p_date`,`p_amt`)VALUES('$stmt2','$fid','$cid','$paymethod','$cardname','$cardnumber','
$cvv','$expiry','$transaction',now(),'$total')","success");
    $stmt4=$admin->cud("INSERT INTO
`checkout`(`o_id`,`c_id`,`f_name`,`l_name`,`ch_address`,`ch_email`,`ch_country`,`ch_city`,`
ch_state`,`ch_zip`,`ch_phone`,`ch_date`)VALUES('$stmt2','$cid','$fname','$lname','$address',
'$email','$country','$city','$state','$zip','$phone',now())","saved");
    $stmt5=$admin->cud("DELETE FROM `cart` WHERE `c_id`='$cid'","deleted");
    echo "<script>window.location='../thankyoupage.php';</script>";
    }
}
?>
```

### 6.4.7 Cancel Order

```
<?php include '../../config.php';
$admin=new Admin();
$oid=$_GET['odid'];
$stmt=$admin->cud("UPDATE `p_order` SET `o_status`='cancelled' WHERE
`o_id`='$oid'","Updated");
echo"<script>alert('Order cancelled
succesfully');window.location.href='../status.php';</script>";
?>
```

### 6.4.8 Contact

```
<?php
include '../../config.php';
$admin= new Admin();
if(isset($_POST['sub'])){
   $name=$_POST['name'];
   $email=$_POST['email'];
   $msg=$_POST['msg'];
   $stmt=$admin->cud("INSERT INTO
contact`(`co_name`,`co_email`,`co_message`)VALUES('$name','$email','$msg')","Inserted");
     echo"<script>alert('Message Sent
Sucessfull');window.location.href='../index.php';</script>";
}
?>
```

### 6.4.9 Feedback

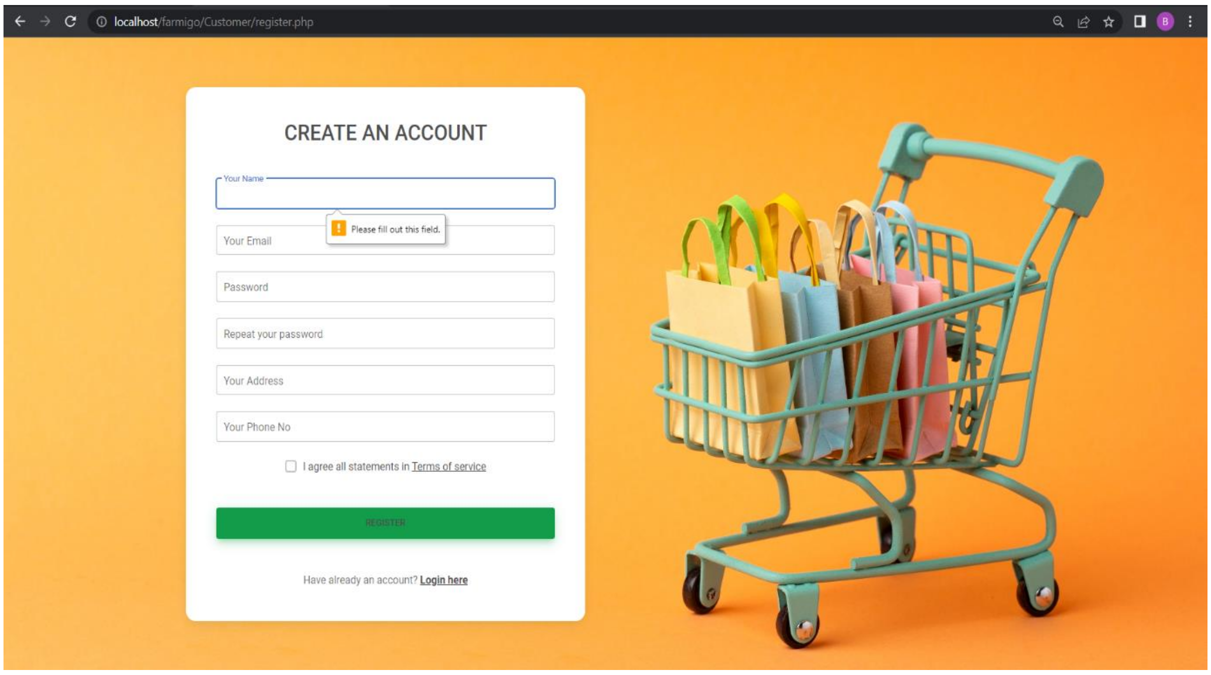
```
<?php
include '../../config.php';
$admin= new Admin();
$cid=$_SESSION['c_id'];

if(isset($_POST['send'])){
   $name=$_POST['name'];
   $email=$_POST['email'];
   $pname=$_POST['sub'];
   $oid =$_POST['oid'];
    $image_path="uploader/".basename($_FILES['image']['name']);
    move_uploaded_file($_FILES['image']['tmp_name'],$image_path);
    $details=$_POST['details'];
   $stmt2=$admin->cud("INSERT INTO
`feedback`(`c_id`,`o_id`,`fb_name`,`fb_email`,`fb_sub`,`fb_details`,`fb_image`)VALUES('$c
id','$oid','$name',' $email','$pname','$details','$image_path')","Inserted");
      echo"<script>alert('Feedback Sent
Succesfully');window.location.href='../status.php';</script>";
}
?>
```

### 6.4.7 Logout

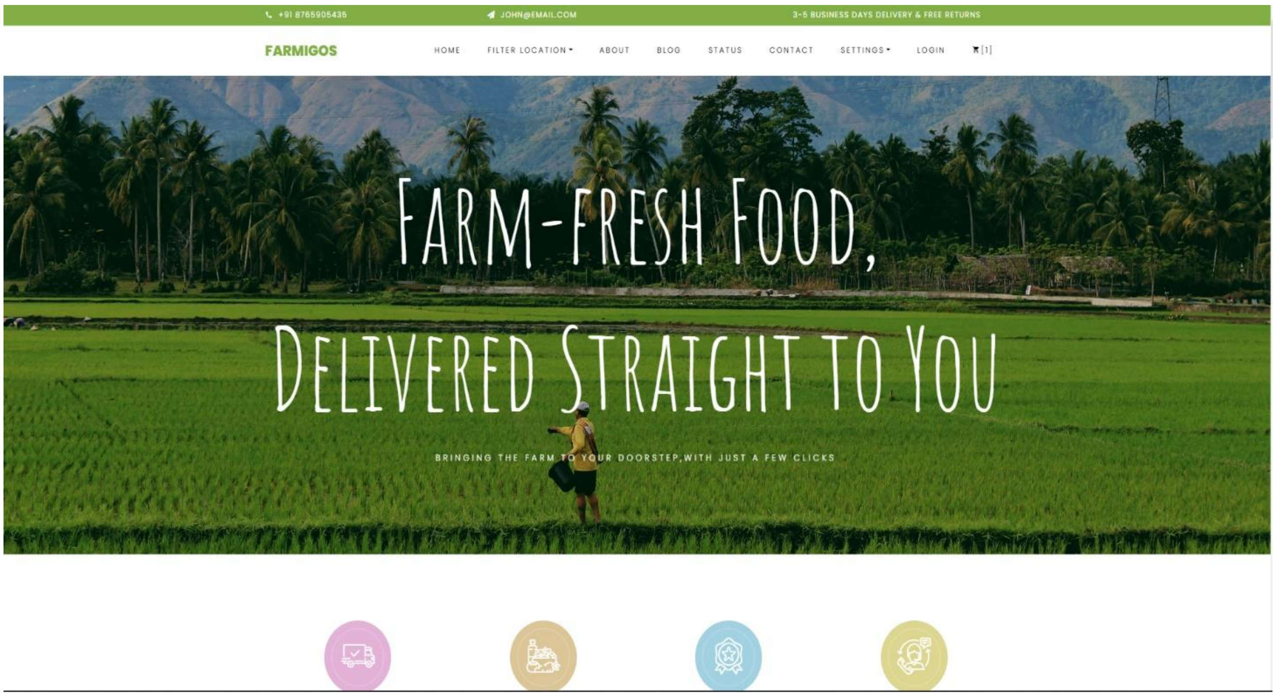
```
<?php
session_start();
session_destroy();
header('Location:../login.php');
?>
```

# 7. User Interface

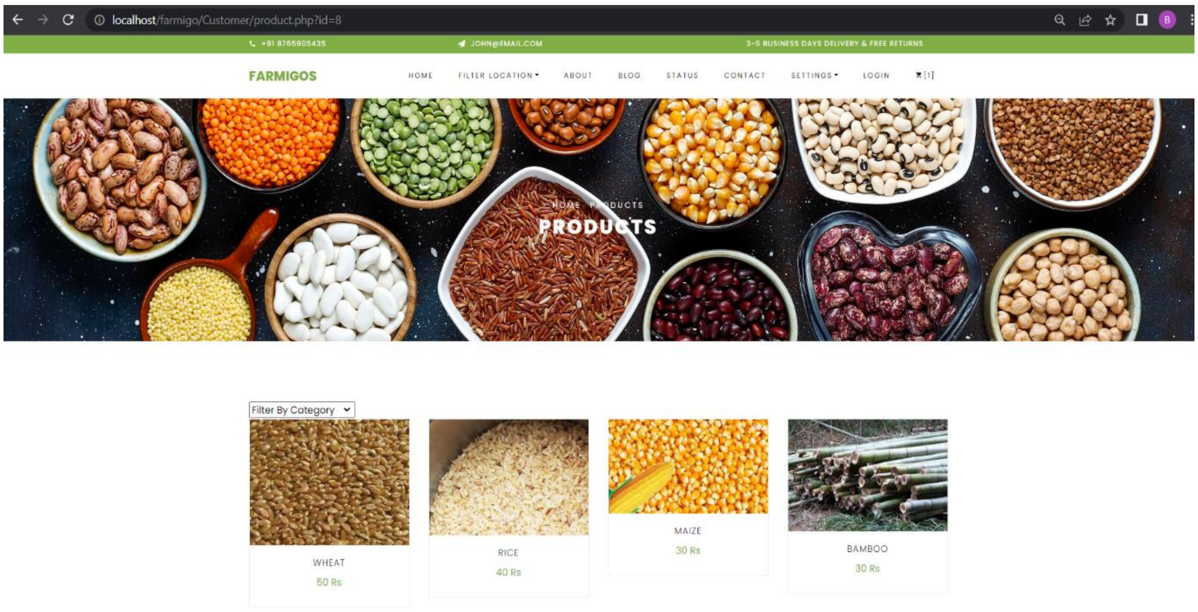
## Customer Register Page


localhost/farmigo/Customer/register.php
CREATE AN ACCOUNT
Your Name
Please fill out this field.
Your Email
Password
Repeat your password
Your Address
Your Phone No
I agree all statements in Terms of service
REGISTER
Have already an account? Login here

## Customer Home Page


+91 8765905435
JOHN@EMAIL.COM
3-5 BUSINESS DAYS DELIVERY & FREE RETURNS
FARMIGOS
HOME
FILTER LOCATION
ABOUT
BLOG
STATUS
CONTACT
SETTINGS
LOGIN
[1]
FARM-FRESH FOOD,
DELIVERED STRAIGHT TO YOU
BRINGING THE FARM TO YOUR DOORSTEP,WITH JUST A FEW CLICKS

## View Products


localhost/farmigo/Customer/product.php?id=8

+91 8765905435 | JOHN@EMAIL.COM | 3-5 BUSINESS DAYS DELIVERY & FREE RETURNS

FARMIGOS | HOME | FILTER LOCATION | ABOUT | BLOG | STATUS | CONTACT | SETTINGS | LOGIN | [1]

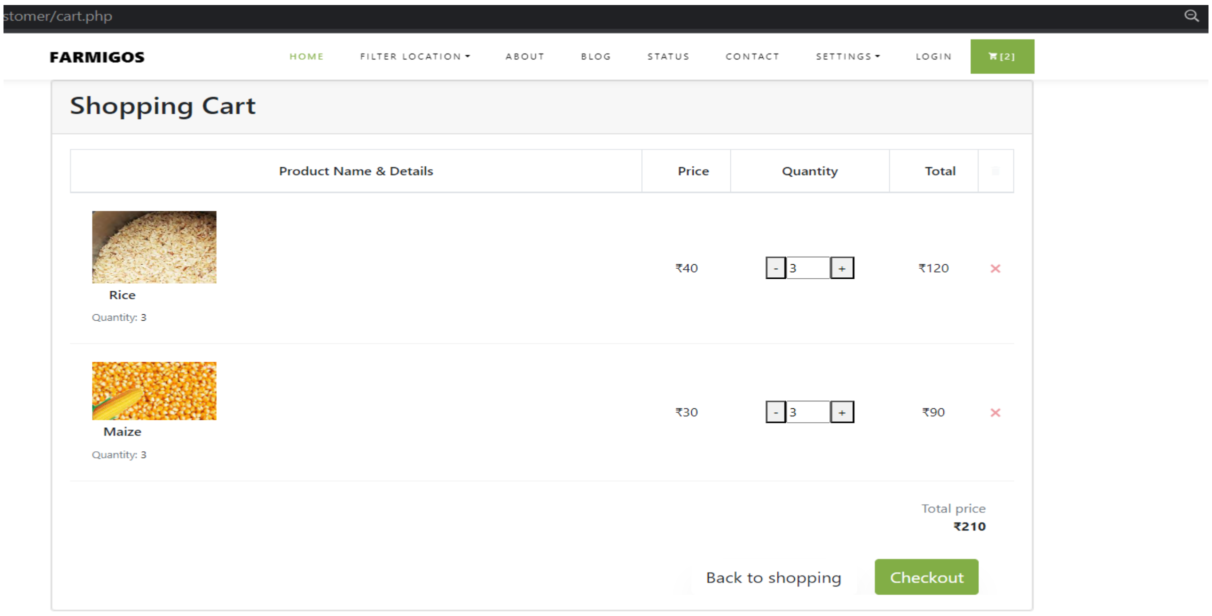
HOME / PRODUCTS

PRODUCTS

Filter By Category

| WHEAT | RICE | MAIZE | BAMBOO |
|---|---|---|---|
| 50 Rs | 40 Rs | 30 Rs | 30 Rs |

## Cart Page:


stomer/cart.php

FARMIGOS | HOME | FILTER LOCATION | ABOUT | BLOG | STATUS | CONTACT | SETTINGS | LOGIN | [2]

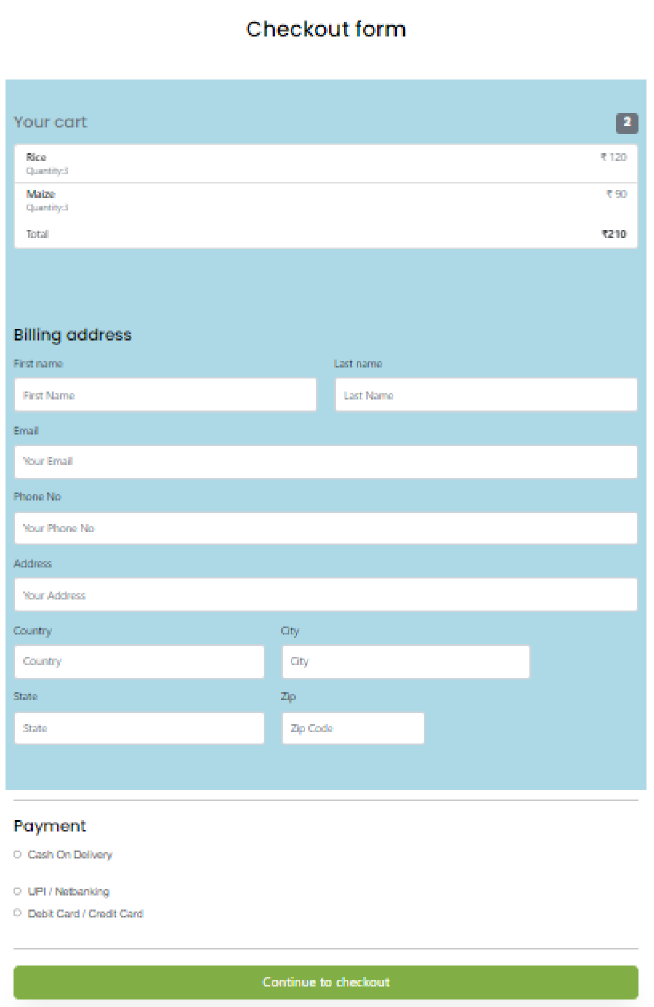
Shopping Cart

| Product Name & Details | Price | Quantity | Total | |
|---|---|---|---|---|
| Rice<br>Quantity: 3 | ₹40 | 3 | ₹120 | × |
| Maize<br>Quantity: 3 | ₹30 | 3 | ₹90 | × |

Total price
₹210

Back to shopping | Checkout

## Checkout Page:


Checkout form

Your cart 2

| | |
|---|---|
| Rice<br>Quantity:3 | ₹ 120 |
| Maize<br>Quantity:3 | ₹ 90 |
| Total | ₹210 |

Billing address

| | |
|---|---|
| First name: First Name | Last name: Last Name |
| Email: Your Email | |
| Phone No: Your Phone No | |
| Address: Your Address | |
| Country: Country | City: City |
| State: State | Zip: Zip Code |

Payment

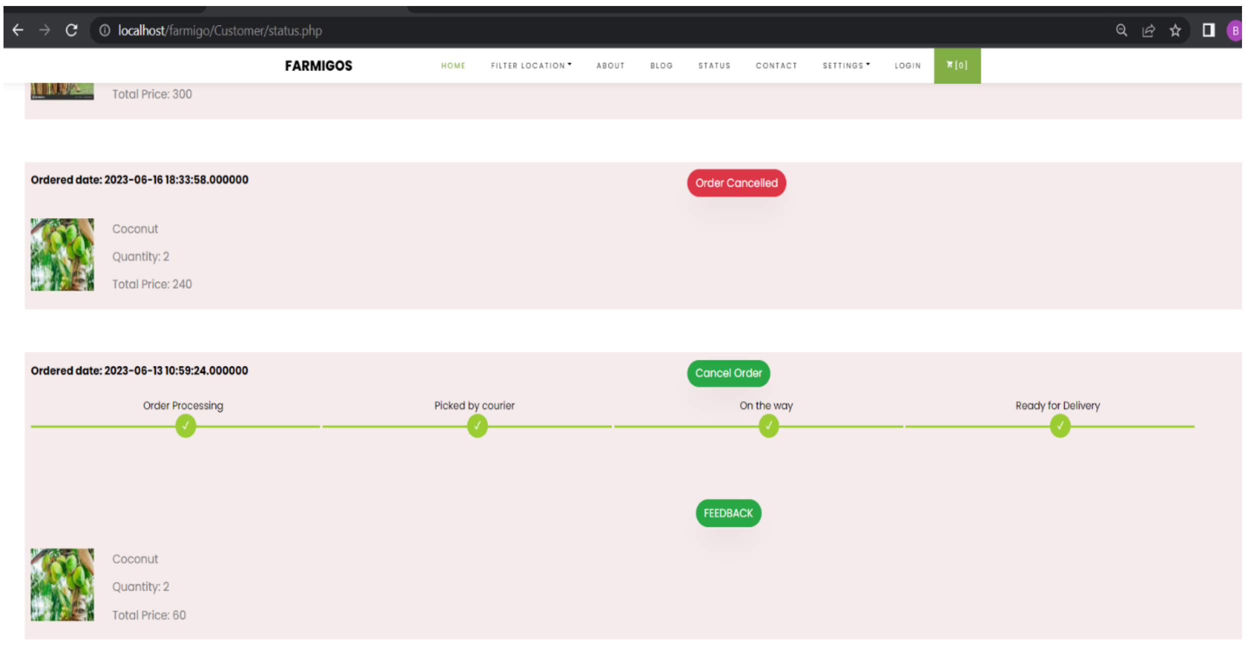
○ Cash On Delivery

○ UPI / Netbanking

○ Debit Card / Credit Card

Continue to checkout

## Track Order Page:



Total Price: 300

Ordered date: 2023-06-16 18:33:58.000000

Order Cancelled

Coconut
Quantity: 2
Total Price: 240

Ordered date: 2023-06-13 10:59:24.000000

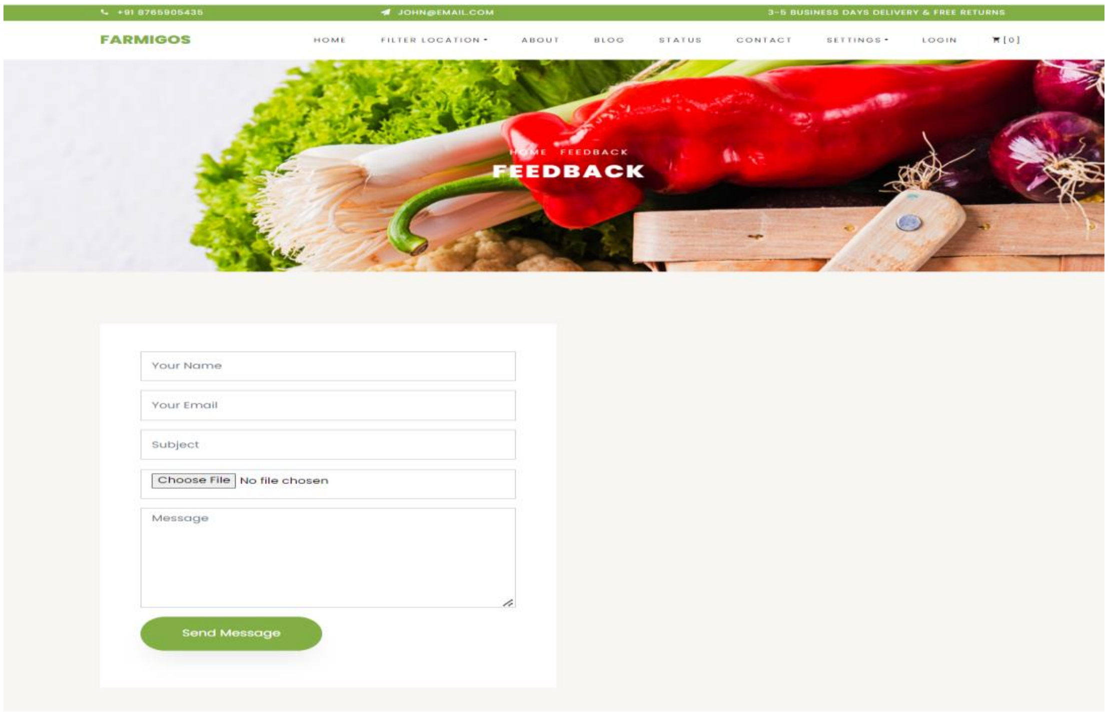
Cancel Order

Order Processing ✓ — Picked by courier ✓ — On the way ✓ — Ready for Delivery ✓

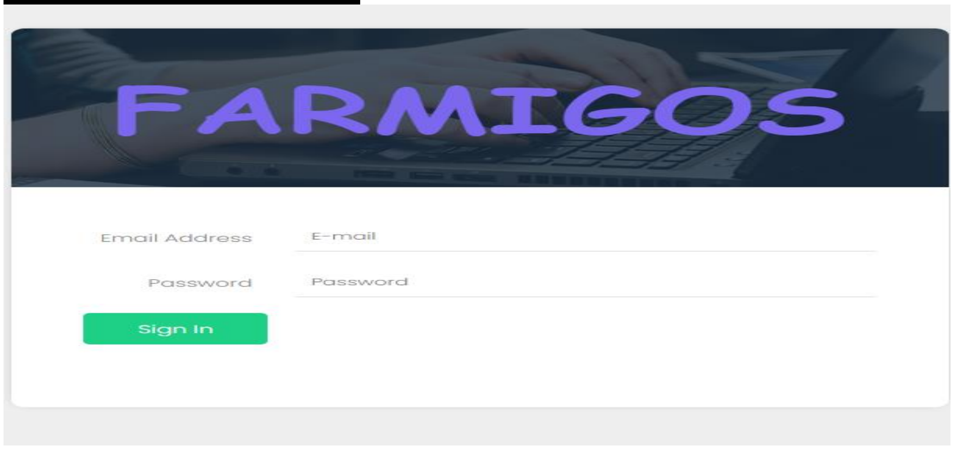
FEEDBACK

Coconut
Quantity: 2
Total Price: 60

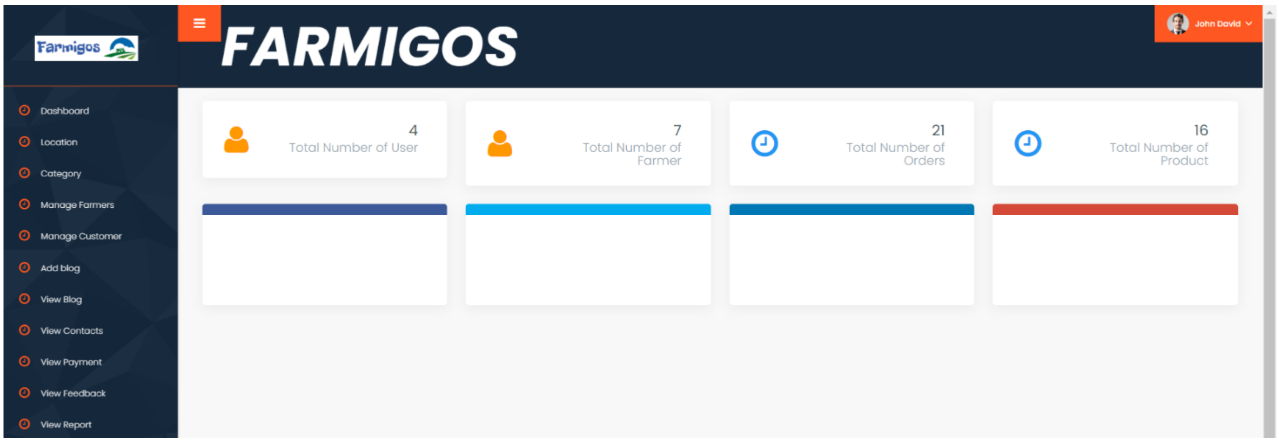
## Feedback Page:



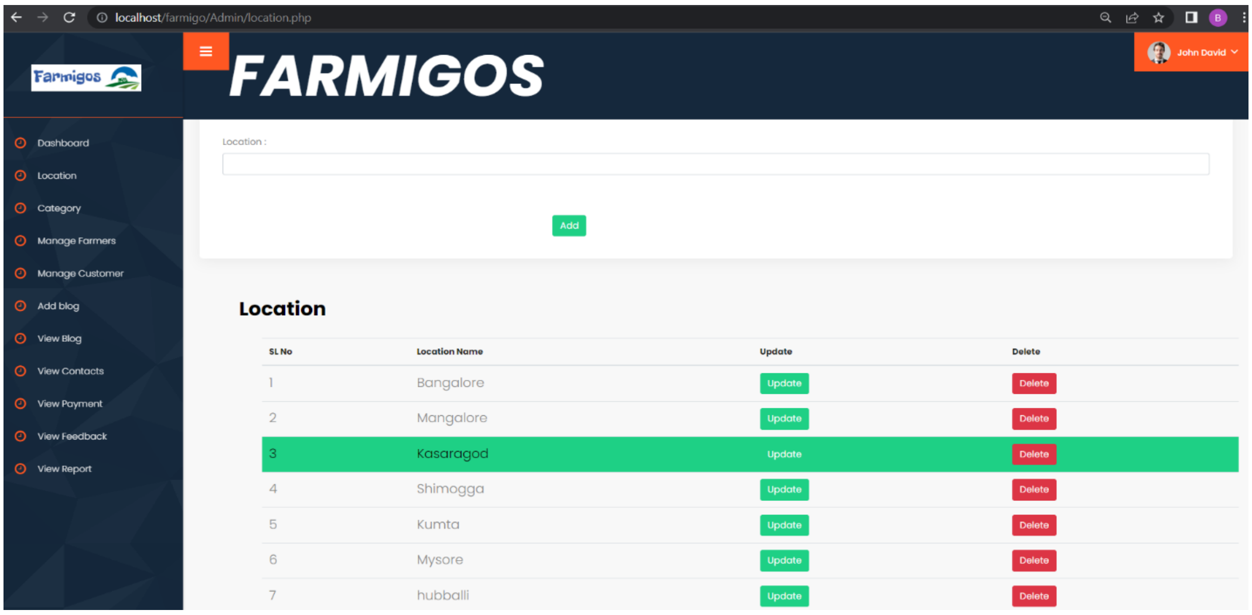
## Admin Login Page:



## Admin Home Page



## Add Location

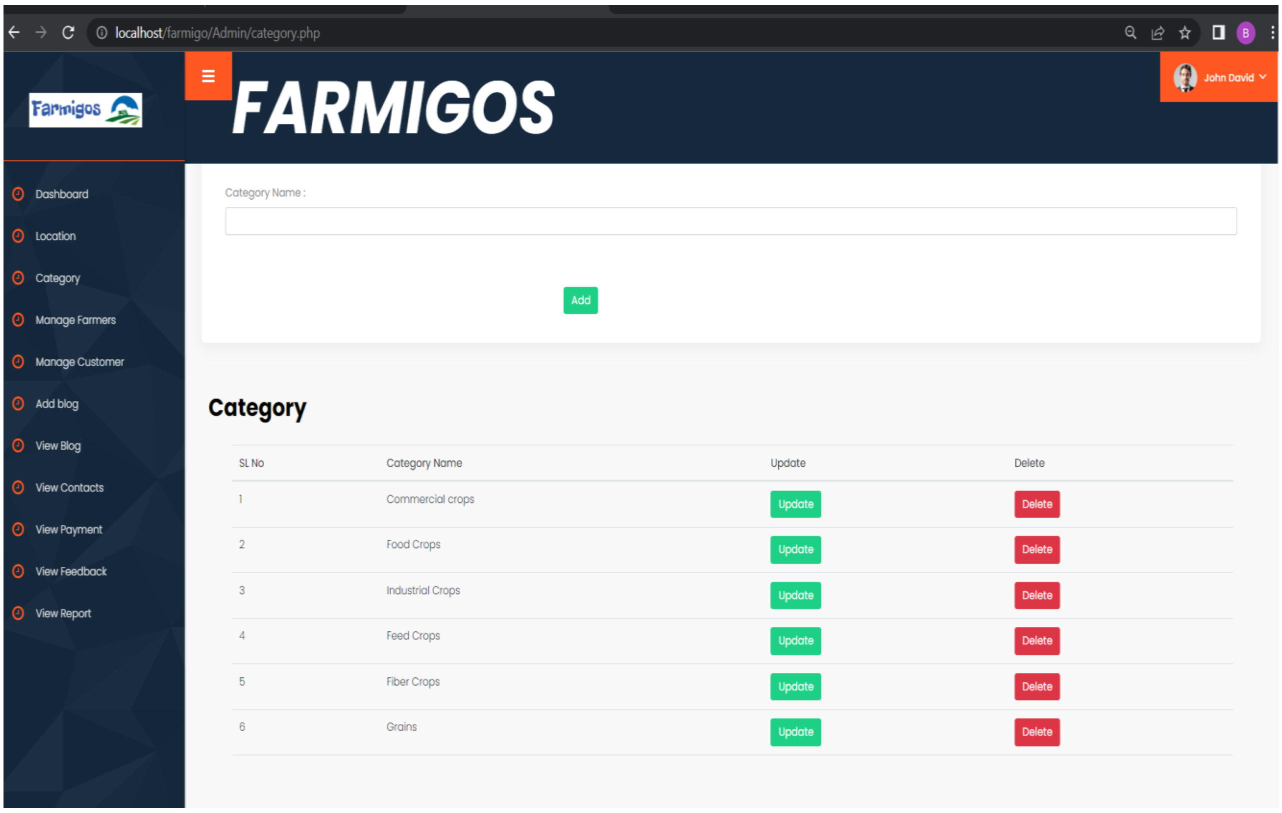


Location :

**Location**

| SL No | Location Name | Update | Delete |
|---|---|---|---|
| 1 | Bangalore | Update | Delete |
| 2 | Mangalore | Update | Delete |
| 3 | Kasaragod | Update | Delete |
| 4 | Shimogga | Update | Delete |
| 5 | Kumta | Update | Delete |
| 6 | Mysore | Update | Delete |
| 7 | hubballi | Update | Delete |

## Add Category

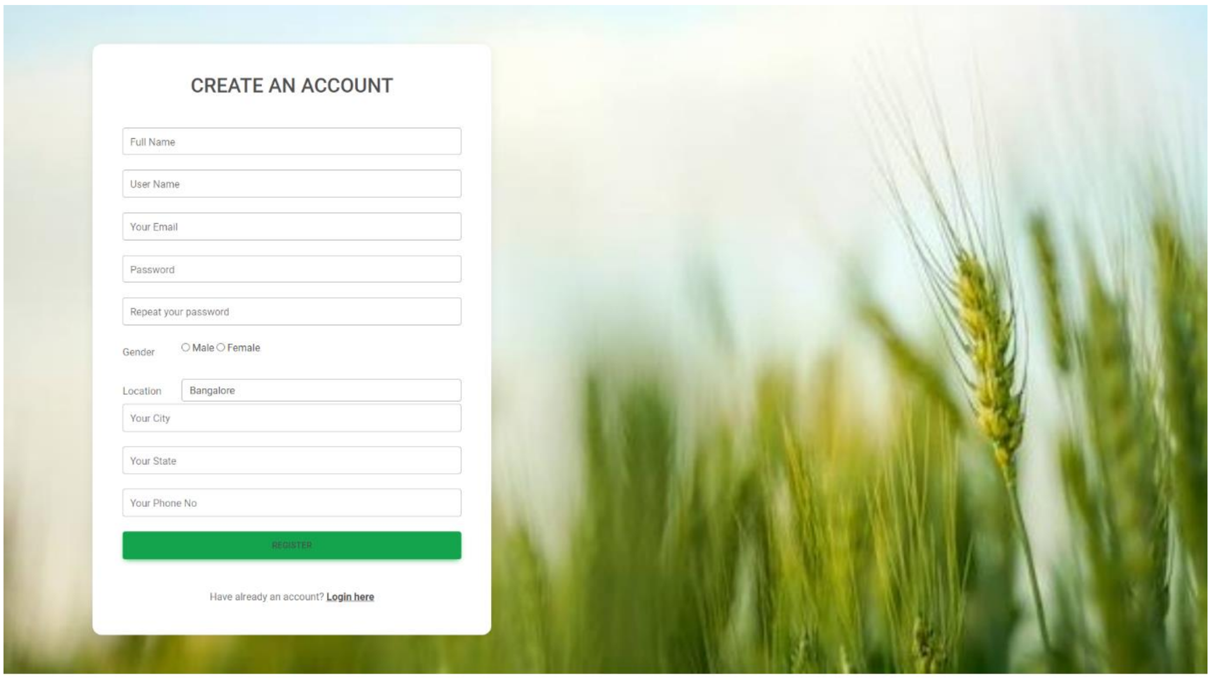


Category Name :

**Category**

| SL No | Category Name | Update | Delete |
|---|---|---|---|
| 1 | Commercial crops | Update | Delete |
| 2 | Food Crops | Update | Delete |
| 3 | Industrial Crops | Update | Delete |
| 4 | Feed Crops | Update | Delete |
| 5 | Fiber Crops | Update | Delete |
| 6 | Grains | Update | Delete |

## Farmer Register Page:


CREATE AN ACCOUNT

Full Name

User Name

Your Email

Password

Repeat your password

Gender ○ Male ○ Female

Location Bangalore

Your City

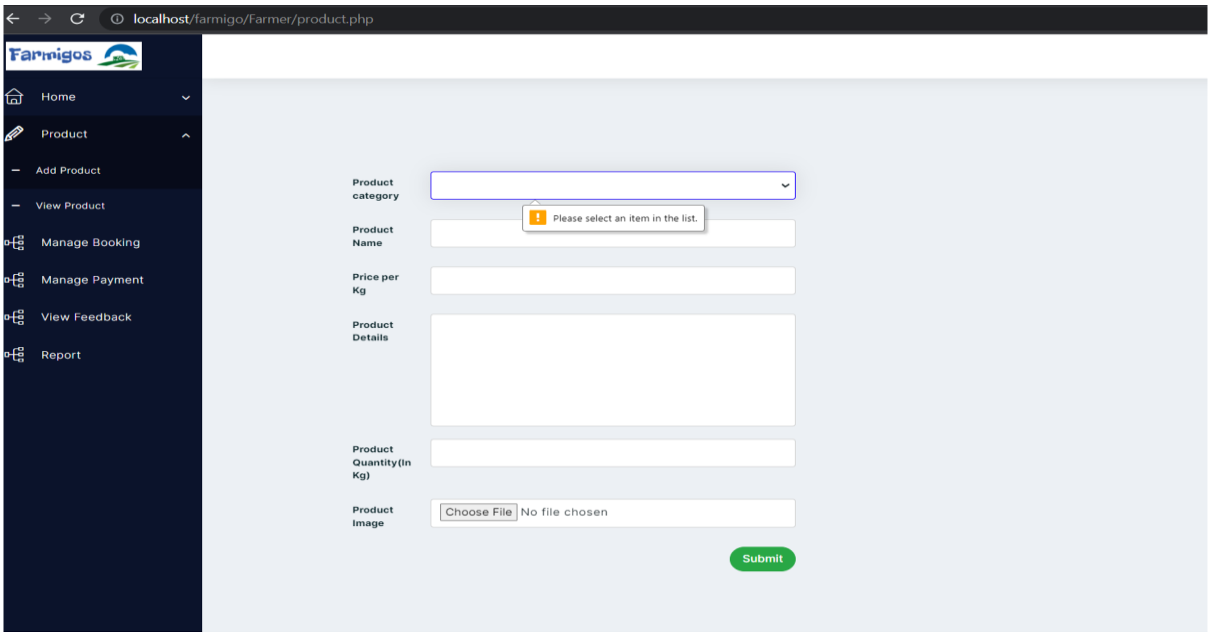
Your State

Your Phone No

REGISTER

Have already an account? Login here

## Add Product page


localhost/farmigo/Farmer/product.php

Product category

Please select an item in the list.

Product Name

Price per Kg

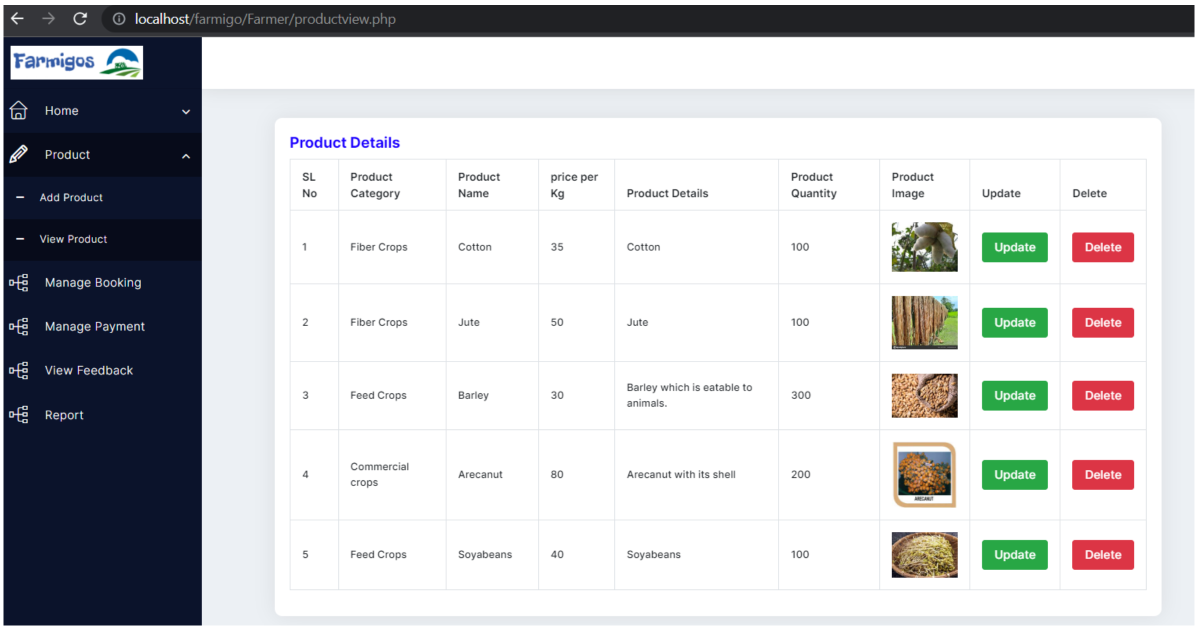
Product Details

Product Quantity(In Kg)

Product Image Choose File No file chosen

Submit

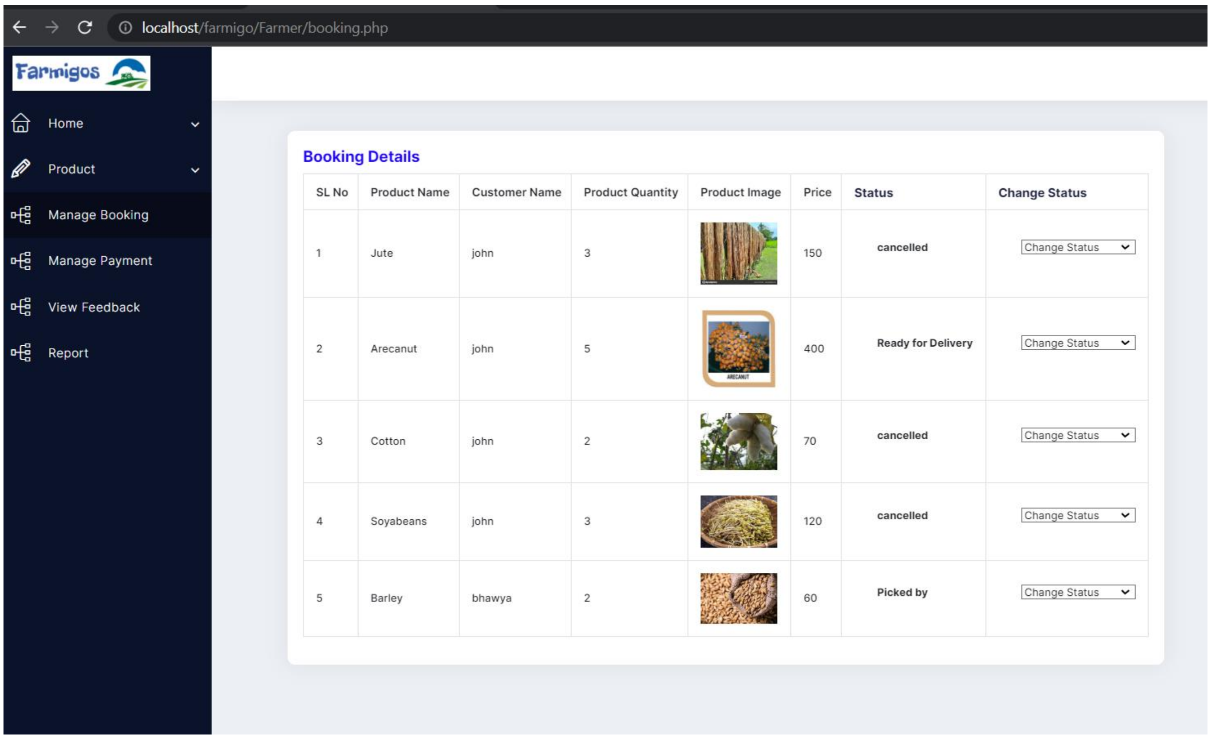
## View Product Page



**Product Details**

| SL No | Product Category | Product Name | price per Kg | Product Details | Product Quantity | Product Image | Update | Delete |
|---|---|---|---|---|---|---|---|---|
| 1 | Fiber Crops | Cotton | 35 | Cotton | 100 | | Update | Delete |
| 2 | Fiber Crops | Jute | 50 | Jute | 100 | | Update | Delete |
| 3 | Feed Crops | Barley | 30 | Barley which is eatable to animals. | 300 | | Update | Delete |
| 4 | Commercial crops | Arecanut | 80 | Arecanut with its shell | 200 | | Update | Delete |
| 5 | Feed Crops | Soyabeans | 40 | Soyabeans | 100 | | Update | Delete |

## Manage Booking Page:



**Booking Details**

| SL No | Product Name | Customer Name | Product Quantity | Product Image | Price | Status | Change Status |
|---|---|---|---|---|---|---|---|
| 1 | Jute | john | 3 | | 150 | cancelled | Change Status |
| 2 | Arecanut | john | 5 | | 400 | Ready for Delivery | Change Status |
| 3 | Cotton | john | 2 | | 70 | cancelled | Change Status |
| 4 | Soyabeans | john | 3 | | 120 | cancelled | Change Status |
| 5 | Barley | bhawya | 2 | | 60 | Picked by | Change Status |

# 7. TESTING

## 7.1 INTRODUCTION:

The process of analysing a software item to detect the differences between existing and required conditions (i.e., bugs) and to evaluate the features of the software items. During testing, the program to be tested is executed with a group of test cases and therefore the output of the program for the test cases is evaluated to work out if the program is performing needless to say. From this it's clear that testing is employed to seek out errors instead of to inform the precise nature of the error. Also, the success of the testing process clearly depends upon the test cases used. Testing may be a complex process. so as to form the method simpler, the testing activities are broken into smaller activities. Due to this, for a project, incremental testing is generally performed. In incremental testing process, the system is broken into set of subsystems and these subsystems are tested separately before integrating them to make the system for system testing.

## 7.2 TESTING OBJECTIVE

**Verification:** A outstanding goal of testing is verification, which allows testers to affirm that the software program meets the various commercial enterprise and technical requirements said by the client earlier than the inception of the whole venture. These requirements and specs guide the design and improvement of the software program, consequently are required to be observed rigorously. Moreover, compliance with these necessities and specs is important for the success of the task in addition to meet the client.

**Validation:** Confirms that the software performs as anticipated and as in line with the requirements of the clients. Validation involves checking the comparing the very last output with the predicted output and then making necessary changes if there is a difference among the two.

**Defects:** The most critical motive of checking out is to find extraordinary defects in the software to save you its failure or crash at some stage in implementation or go live of the project. Defects if left undetected or unattended can harm the functioning of the software and can lead to loss of resources, money, and recognition of the client. Therefore, software program trying out is executed frequently in the course of every level of software improvement to discover defects of numerous kinds.

**Compatibility:** It helps validate application’s compatibility with the implementation environment, various devices, Operating Systems, person requirements, among other things.

**Quality Analysis:** Testing enables improve the first-class of the software by constantly measuring and verifying its layout and coding. Additionally, various types of testing strategies are used by testers, which help them attain the preferred software program fine.

## 7.3 TEST CASES:

### 7.3.1 Unit Testing:

#### 7.3.1.1 Login page:

| Input | Test Condition | Test Output | Comments |
|---|---|---|---|
| Login | If the admin/customer/Farmer email_id is not entered | Please fill out this field. | Username is required |
|  | If the password is not entered | Please fill out this field. | Password is required |
|  | If the username and password are not valid | User not found. | Enter valid username and password |
|  | If the customer/Farmer email_id and password are valid. | Successfully logged in. | Appropriate password and email are entered and the customer/shop owner is now logged in. |
|  | If the Admin email_id and password valid. | Successfully logged in. | Appropriate password and email are entered and the Admin owner is now logged in to the admin page. |

#### 7.3.1.2 Customer/Farmer Registration Form:

| Input | Test Condition | Test Output | Comments |
|---|---|---|---|
| Register | If the name is not entered | Please fill out this field. | Customer/Farmer name is required |
|  | If the address is not entered | Please fill out this field. | Customer/Farmer address is required |
|  | If the email is not entered | Please fill out this field. | Customer/Farmer email address is required |
|  | If the phone no. is not entered | Please fill out this field. 10 numeric only | Customer/Farmer phone no is required |
|  | If the password is not entered | Please fill out this field. | Password required |
|  | If these above fields are valid | Registered successfully | Customer/Farmr registered |

### 7.3.1.3 Admin Home Page:

| Input | Test Condition | Test Output | Comments |
|---|---|---|---|
| location | If the location is not entered by the admin | Please fill out this field. | Location name is required |
| | If the location is entered by the admin | Data inserted successfully | Location name is entered. |
| category | If the category is not entered by the admin | Please fill out this field. | category name is required. |
| | If the category is entered by the admin. | Data inserted successfully. | category name is entered. |
| Blog | If the blog is not entered by the admin. | Please fill out this field. | Blog details is Required. |
| | If the blog is entered by the admin. | Data inserted successfully. | Blog details is entered. |

### 7.3.1.4 Farmer Home page:

| Input | Test Condition | Test Output | Comments |
|---|---|---|---|
| Product | If the category name is not selected. | Please select this field. | Category name is required. |
| | If the Product name is not entered. | Please fill out this field. | Product name is required. |
| | If the price is not entered. | Please fill out this field. | Price is required. |
| | If the product details is not entered. | Please fill out this field. | Product details is required. |
| | If the product quantity is not entered. | Please fill out this field. | Product quantity is required. |
| | If the product image is not selected. | Please select this field. | product image is required. |

**7.3.1.5 Customer Home page:**

| Input | Test Condition | Test Output | Comments |
|---|---|---|---|
| payment | If the customer does not select the payment method | Please select the type | Payment method is required |
| | If customer selected the method card and do not fill the form | Please fill out the field. | Enter transaction id |
| Track | If there are no orders | No orders yet! | Order is empty |
| | If there is order | Displays order details | Display order |
| Contact | If the contact details is not entered | Please fill out this field. | contact details is required. |
| | If the above field is filled | contact sent | contact is done |
| Feedback | If the feedback is not entered | Please fill out this field. | feedback is required |
| | If the above field is filled | Feedback sent | feedback is done |

## 7.3.2 Integration testing:

### 7.3.2.1 Admin Login:

| Test No. | Test Description | Expected Output | Expected Output |
|---|---|---|---|
| 1 | Login | The site is redirected to admin page | Success |

### 7.3.2.2 Customer/ Farmers register page and login:

| Test No. | Test Description | Test Output | Expected Output |
|---|---|---|---|
| 1. | Register | The entered data is stored in database and the site is redirected to login page | Success |
| 2. | Login | The site is redirected to user index page | Success |

### 7.3.2.3 Customer/Farmers Login:

| Test No. | Test Description | Expected Output | Expected Output |
|---|---|---|---|
| 1 | Login | The site is redirected to Customer/farmer index page | Success |

### 7.3.2.4 Admin Index Page:

| Test No. | Test Description | Expected Output | Expected Output |
|---|---|---|---|
| 1. | Manage Customer | The site is redirected to customer list page | Success |
| 2. | Manage Farmers | The site is redirected to Farmer list page | Success |
| 3. | Add location | The site is redirected To add location page | Success |
| 4. | Add category | The site is redirected to add category | Success |
| 5. | Add blog | The site is redirected to add blog page | Success |
| 6. | View contact | The site is redirected to view contact page | Success |
| 7. | View Payment | The site is redirected to View payment page | Success |
| 8. | View feedback | The site is redirected to view feedback page | Success |
| 9. | View Report | The site is redirected to view report page | Success |

**7.3.2.5 Farmer Index Page:**

| Test No. | Test Description | Expected Output | Expected Output |
|---|---|---|---|
| 1. | Add product | The site is redirected to add product page. | Success |
| 2. | View product | The site is redirected to view product page. | Success |
| 3. | Manage Booking | The site is redirected to booking page. | Success |
| 4. | Manage Payment | The site is redirected to payment page. | Success |
| 5. | View Feedback | The site is redirected to feedback page. | Success |
| 6. | Report | The site is redirected To report page. | Success |

**7.3.2.6 Customer Index Page:**

| Test No. | Test Description | Expected Output | Expected Output |
|---|---|---|---|
| 1. | Home page | The site is redirected to index page. | Success |
| 2. | Filter by location | The site is redirected to product page. | Success |
| 3. | About | The site is redirected to about page. | Success |
| 4. | Blog | The site is redirected to blog page. | Success |
| 5. | Status | The site is redirected to Status page. | Success |
| 6. | Profile | The site is redirected to Profile page. | Success |
| 7. | Feedback | The site is redirected to feedback page | Success |
| 8. | Contact | The site is redirected to contact page. | Success |
| 9. | Cart | The site is redirected to cart page. | |
| 10. | Checkout | The site is redirected to checkout page. | |
| 11. | Logout | The site is redirected to home page | Success |

## 7.3.3 System Testing:

System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer-based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

**7.3.3.1 System testing tables:**

| **Sl. No** | **Test condition** | **Test report** |
|---|---|---|
| 1. | System lading | Successful |
| 2. | System run procedure | Successful |
| 3. | File I/O operation | Successful |
| 4. | Database communication | Successful |
| 5. | Server/client interaction | Successful |
| 6. | Memory usage | Normal |
| 7. | System processor usage | Normal |
| 8. | Authentication/Authorization | Successful |

# 9. User Manual

## 9.1 Introduction

"**Farmigos**" is a web-based platform for agriculture related requirements. In this application admin will manage both farmers and customers. Farmers can list out the products which are available and the customer can order the products according to the requirements.

## 9.2 Hardware Requirement

- Processor: Intel core i3 or above
- Processor Speed: Minimum 2 GHz
- RAM: 4GB of RAM or Above
- Hard Disk: Minimum 40GB free space
- Input device: Mouse, Keyboard

## 9.3 Software requirement

- Web Technology: PHP 5.4
- Web Components: HTML5/CSS/JavaScript
- Software's: XAMPP, Visual code
- Database (Backend): MYSQL server
- Web Server: Apache

## 9.4 Limitations

- It requires reliable internet access.
- It will be difficult for some users because lack of computer and internet knowledge.
- Limited participation: Some farmers may be reluctant to participate in the agro market system due to various reasons, such as lack of trust, lack of awareness, and fear of losing control over their produce.

## 9.5 Future Scope

- In future, it can also be converted to a mobile application.
- Establishing linkages with international markets to increase exports and generate foreign exchange.

## 9.6 Bibliography

- **Book:**
    i. PHP- A Beginner's Guide by Vikram Vaswani
    ii. Leksono, D. Testing spatial data deliverance in SQL and NoSQL databaseusing NodeJS full stack web app.

- **Website:**
    **i.** www.w3schools.com
    **ii.** www.tutorialsapoint.com
    **iii.** www.geeksforgeeks.com
    **iv.** www.javatpoint.com

# 10. CONCLUSION

## Conclusion:

The project work titled “” has been designed using PHP and hasa user interactive application.

The project has been subjected to testing and has been successful.

The module has fulfilled all the objectives identified. The system isdeveloped in such a way that the user with common knowledge of computer canhandle it easily. The module has a user-friendly interface. The reports requested by the client have been generated and all documents required for operation and maintenance of all modules has been provided.

The future enhancement to the system can be made as technology improvesor changes.

# 11. PLAGIARISM