Project Report On

FARMER TRADE & CROP MANAGEMENT

Batch No.: **PGDEA 103**

Submitted by:

1. Shweta

2. Ashwini S A

3. Naveen RD

Under the guidance

SANDEEP PADESUR

TABLE OF CONTENTS

INTRODUCTION

Agriculture is a key sector in India, yet traditional crop trading often lacks transparency and efficiency. Farmers depend on intermediaries, which reduces their profit and limits direct communication with buyers.

The **Farmer Trade & Crop Management System** is a simple C-based program that allows farmers to add and manage crops, buyers to view and purchase produce, and the system to automatically record transactions and calculate revenue. This project demonstrates how basic programming concepts like file handling and data structures can solve real-world problems in agriculture.

OBJECTIVE:

To create a digital platform that facilitates direct trade between farmers and buyers, enabling efficient crop management and transaction processing while maintaining proper records and revenue tracking.

PROBLEM STATEMENT:

Traditional agricultural trading systems often lack:

- **Transparency** in pricing and inventory management
- **Direct communication** between farmers and buyers (eliminating middlemen)
- **Digital record-keeping** for transactions and revenue
- **Real-time inventory tracking** for available crops
- **Secure access control** for different user types
- **Persistent data storage** for business continuity

KEY FEATURES

Management

- **Login system for two roles** – Farmers and Buyers have separate logins with their own menus. Users log in using a username and password.
- **User data saved in files** – Some default accounts are created automatically, and all usernames/passwords are stored in files so they remain even after the program closes.

Crop Management (Farmer Features)

- **Add crops**: Input crop name, quantity, and price
- **View inventory**: Display all available crops with details
- **Revenue tracking**: Monitor total earnings from sales
- **Inventory limits**: Maximum crop capacity management

Trading System (Buyer Features)

- **Browse crops**: View available crops with prices and quantities
- **Purchase crops**: Buy specified quantities if stock is available
- **Stock validation**: Prevents overselling
- **Cost calculation**: Automatic price computation

Transaction Management

- **Transaction recording**: Linked list-based transaction history
- **Purchase tracking**: Records crop name, quantity, and total cost
- **Transaction history**: View all past purchases
- **Revenue updates**: Automatic farmer revenue calculation

This system essentially digitizes the farmer-buyer relationship, providing a simple platform for agricultural trade management with proper record-keeping and user access control.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 100
#define MAX_USERS 100

// ================= Structures =================
struct Crop
{
   char name[50];
   int qty;
   float price;
};

struct User
 {
   char username[50];
   char password[50];
};

struct Transaction
 {
   char cropName[50];
   int qtyBought;
   float totalCost;
   struct Transaction *next;
};
```

```
// ================= Global Variables =================
struct Crop crops[MAX];
int cropCount = 0;
float farmerRevenue = 0;
struct Transaction *transactionHead = NULL;

struct User farmers[MAX_USERS];
int farmerCount = 0;
struct User buyers[MAX_USERS];
int buyerCount = 0;

// File paths
const char *cropFile = "crops.txt";
const char *revenueFile = "revenue.txt";
const char *transactionFile = "transactions.txt";
const char *farmerFile = "farmers.txt";
const char *buyerFile = "buyers.txt";

// ================= Function Prototypes =================
void loadData();
void saveData();
void loadTransactions();
void saveTransactions();
void loadUsers();
void saveUsers();
void createDefaultUsers();
int farmerLogin();
int buyerLogin();
void farmerMenu();
void buyerMenu();
void addCrop();
void viewCrops();
void checkRevenue();
void viewTransactions();
void buyCrop();
void addTransaction(char *name, int qty, float cost);
```

```
// ================= Main =================
int main()
{
   int choice;
   loadData();


 while (1)
{
       printf("\n=== Farmer Trade & Crop Management ===\n");
       printf("1. Farmer Login\n");
       printf("2. Buyer Login\n");
       printf("3. Exit\n");
       printf("Enter choice: ");
       scanf("%d", &choice);

       switch (choice)
{
          case 1:
             if (farmerLogin())
                farmerMenu();
             else
                printf("\nInvalid Farmer Credentials!\n");
             break;

          case 2:
             if (buyerLogin())
                buyerMenu();
             else
                printf("\nInvalid Buyer Credentials!\n");
             break;



  case 3:
```

```
            saveData();
            printf("\nExiting... Data Saved!\n");
            exit(0);

        default:
            printf("Invalid choice!\n");
        }
    }
    return 0;
}

// ================ File Handling ================
void loadData()
{
    FILE *f = fopen(cropFile, "r");
    if (f)
 {
        while (fscanf(f, "%s %d %f",
                crops[cropCount].name,
                &crops[cropCount].qty,
                &crops[cropCount].price) == 3)
            cropCount++;
        fclose(f);
    }

    FILE *r = fopen(revenueFile, "r");
    if (r) {
        fscanf(r, "%f", &farmerRevenue);
        fclose(r);
    }

    loadTransactions();
    loadUsers();
}

void saveData()
```

```
{
    FILE *f = fopen(cropFile, "w");
    for (int i = 0; i < cropCount; i++)
        fprintf(f, "%s %d %.2f\n",
                crops[i].name,
                crops[i].qty,
                crops[i].price);
    fclose(f);

    FILE *r = fopen(revenueFile, "w");
    fprintf(r, "%.2f", farmerRevenue);
    fclose(r);

    saveTransactions();
    saveUsers();
}

void loadTransactions()
 {
    FILE *t = fopen(transactionFile, "r");
    if (!t)
        return;

    char name[50];
    int qty;
    float cost;
    while (fscanf(t, "%s %d %f", name, &qty, &cost) == 3)
 {
        addTransaction(name, qty, cost);
    }
    fclose(t);
}

void saveTransactions()
{
    FILE *t = fopen(transactionFile, "w");
```

```
    struct Transaction *temp = transactionHead;
    while (temp) {
        fprintf(t, "%s %d %.2f\n",
                temp->cropName,
                temp->qtyBought,
                temp->totalCost);
        temp = temp->next;
    }
    fclose(t);
}

void createDefaultUsers()
{
    FILE *f = fopen(farmerFile, "r");
    if (!f) {
        f = fopen(farmerFile, "w");
        fprintf(f, "farmer1 pass123\nfarmer2 farmerpass\n");
        fclose(f);

   }
  Else
  {
        fclose(f);
    }

    FILE *b = fopen(buyerFile, "r");
    if (!b)
  {
        b = fopen(buyerFile, "w");
        fprintf(b, "buyer1 pass123\nbuyer2 buyerpass\n");
        fclose(b);
    }
  else {
        fclose(b);
    }
}
```

```
void loadUsers()
 {
   createDefaultUsers();

   FILE *f = fopen(farmerFile, "r");
   if (f)
 {
      while (fscanf(f, "%s %s",
                farmers[farmerCount].username,
                farmers[farmerCount].password) == 2)
         farmerCount++;
      fclose(f);
   }




   FILE *b = fopen(buyerFile, "r");
   if (b)
{
      while (fscanf(b, "%s %s",
                buyers[buyerCount].username,
                buyers[buyerCount].password) == 2)
         buyerCount++;
      fclose(b);
   }
}

void saveUsers() {
   FILE *f = fopen(farmerFile, "w");
   for (int i = 0; i < farmerCount; i++)
      fprintf(f, "%s %s\n",
            farmers[i].username,
            farmers[i].password);
   fclose(f);
```

```
    FILE *b = fopen(buyerFile, "w");
    for (int i = 0; i < buyerCount; i++)
        fprintf(b, "%s %s\n",
                buyers[i].username,
                buyers[i].password);
    fclose(b);
}

// ================= Login =================
int farmerLogin()
{
    char u[50], p[50];
    printf("Enter Farmer Username: ");
    scanf("%s", u);
    printf("Enter Password: ");
    scanf("%s", p);
    for (int i = 0; i < farmerCount; i++)
{
        if (strcmp(u, farmers[i].username) == 0 &&
            strcmp(p, farmers[i].password) == 0)
 {
            return 1;
        }
    }
    return 0;
}


int buyerLogin()
 {
    char u[50], p[50];
    printf("Enter Buyer Username: ");
    scanf("%s", u);
    printf("Enter Password: ");
    scanf("%s", p);
```

```
    for (int i = 0; i < buyerCount; i++)
 {
        if (strcmp(u, buyers[i].username) == 0 &&
            strcmp(p, buyers[i].password) == 0)
 {
            return 1;
        }
    }
    return 0;
}

// ================= Farmer Menu ================
void farmerMenu()
 {
    int choice;
    while (1)
{
        printf("\n--- Farmer Menu ---\n");
        printf("1. Add Crop\n");
        printf("2. View All Crops\n");
        printf("3. Check Revenue\n");
        printf("4. Back\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice)
 {
            case 1: addCrop(); break;
            case 2: viewCrops(); break;
            case 3: checkRevenue(); break;
            case 4: return;
            default: printf("Invalid choice!\n");
        }
    }
}
```

```
void addCrop()
{
   if (cropCount >= MAX)
 {
       printf("Cannot add more crops!\n");
       return;
   }
   printf("Enter Crop Name: ");
   scanf("%s", crops[cropCount].name);
   printf("Enter Quantity: ");
   scanf("%d", &crops[cropCount].qty);
   printf("Enter Price: ");
   scanf("%f", &crops[cropCount].price);

   cropCount++;
   printf("Crop Added Successfully!\n");
}

void viewCrops()
 {
   if (cropCount == 0)
 {
       printf("No crops available!\n");
       return;
   }
   printf("\nAvailable Crops:\n");
   for (int i = 0; i < cropCount; i++)
   printf("%d. %s | Qty: %d | Price: ₹%.2f\n", i + 1,crops[i].name, crops[i].qty, crops[i].price);
}

void checkRevenue()
{
   printf("\nTotal Revenue: ₹%.2f\n", farmerRevenue);
}
```

```
// ================= Buyer Menu =================
void buyerMenu()
 {
   int choice;
   while (1)
{
       printf("\n--- Buyer Menu ---\n");
       printf("1. View Crops\n");
       printf("2. Buy Crop\n");
       printf("3. View Transactions\n");
       printf("4. Back\n");
       printf("Enter choice: ");
       scanf("%d", &choice);

       switch (choice)
 {
           case 1: viewCrops(); break;
           case 2: buyCrop(); break;
           case 3: viewTransactions(); break;
           case 4: return;
           default: printf("Invalid choice!\n");
       }
   }
}

void buyCrop()
 {
   char name[50];
   int qty;

   viewCrops();
   printf("Enter Crop Name to Buy: ");
   scanf("%s", name);
   printf("Enter Quantity: ");
   scanf("%d", &qty);
```

```
for (int i = 0; i < cropCount; i++)
 {
      if (strcmp(name, crops[i].name) == 0)
 {
         if (qty > crops[i].qty)
 {
            printf("Not enough stock!\n");

         return;
         }
         float cost = qty * crops[i].price;
         crops[i].qty -= qty;
         farmerRevenue += cost;
         addTransaction(name, qty, cost);
         printf("Purchase Successful! Total Cost: ₹%.2f\n", cost);
         return;
      }
   }
   printf("Crop not found!\n");
}

// ================= Transactions ================
void addTransaction(char *name, int qty, float cost)
{
   struct Transaction *newNode =
      (struct Transaction *)malloc(sizeof(struct Transaction));
   strcpy(newNode->cropName, name);
   newNode->qtyBought = qty;
   newNode->totalCost = cost;
   newNode->next = transactionHead;
   transactionHead = newNode;
}
```

```
void viewTransactions()
 {
   struct Transaction *temp = transactionHead;
   if (!temp)
 {
      printf("No transactions yet!\n");
      return;
   }
   printf("\n--- Transaction History ---\n");
   while (temp)
 {
      printf("Bought: %s | Qty: %d | Cost: ₹%.2f\n",
            temp->cropName,
            temp->qtyBought,
            temp->totalCost);
      temp = temp->next;
   }
}
```

DEMONSTRATION OF PROGRAM OUTPUT

=== Farmer Trade & Crop Management ===
1. Farmer Login
2. Buyer Login
3. Exit
Enter choice: 1

Enter Farmer Username: farmer1
Enter Password: pass123

--- Farmer Menu ---
1. Add Crop
2. View All Crops
3. Check Revenue
4. Back
Enter choice: 1
Enter Crop Name: Wheat
Enter Quantity: 50
Enter Price: 20
Crop Added Successfully!

--- Farmer Menu ---
1. Add Crop
2. View All Crops
3. Check Revenue
4. Back
Enter choice: 2

Available Crops:
1. Wheat | Qty: 50 | Price: ₹20.00

Enter choice: 4

=== Farmer Trade & Crop Management ===
1. Farmer Login
2. Buyer Login
3. Exit
Enter choice: 2

Enter Buyer Username: buyer1
Enter Password: pass123

--- Buyer Menu ---
1. View Crops
2. Buy Crop
3. View Transactions
4. Back
Enter choice: 1

Available Crops:
1. Wheat | Qty: 50 | Price: ₹20.00

--- Buyer Menu ---
1. View Crops
2. Buy Crop
3. View Transactions
4. Back
Enter choice: 2
Enter Crop Name to Buy: Wheat
Enter Quantity: 10
Purchase Successful! Total Cost: ₹200.00

--- Buyer Menu ---
1. View Crops
2. Buy Crop
3. View Transactions
4. Back
Enter choice: 3

--- Transaction History ---
Bought: Wheat | Qty: 10 | Cost: ₹200.00

Enter choice: 4

=== Farmer Trade & Crop Management ===
1. Farmer Login
2. Buyer Login
3. Exit
Enter choice: 1

Enter Farmer Username: farmer1
Enter Password: pass123

--- Farmer Menu ---
1. Add Crop
2. View All Crops
3. Check Revenue
4. Back
Enter choice: 3

Total Revenue: ₹200.00

FUTURE SCOPE:

- Integrate an **online payment gateway** so buyers can pay farmers directly through the platform.
- Replace text files with a **database system** (MySQL, SQLite) for better scalability and security.
- Add a **graphical user interface (GUI)** or mobile app to make the system easier to use.
- Provide **real-time notifications** for new crops or price changes.
- Support **multi-language interfaces** to help farmers from different regions.
- Enable **analytics and reports** to show farmers sales trends over time.

CONCLUSION:

The *Farmer Trade & Crop Management System* successfully demonstrates how a simple C-based program can digitise agricultural trading. It enables farmers to add and manage crops, buyers to purchase produce directly, and the system to automatically record transactions and track revenue. By using file handling and data structures like arrays and linked lists, the program provides a low-cost, easily deployable solution for small farmers and buyers. This project shows how programming concepts can be applied to solve real-world problems in a transparent and efficient way.

.

REFERENCES:

- Class notes and teaching material from Cranes varsity.
- “Let Us C” by Yashavant Kanetkar (for C programming concepts).
- GeeksforGeeks – C Programming, File Handling, Linked List tutorials.
- TutorialsPoint – C Language Basics and Examples.