JAVA SWING BASED – FARMER_CUSTOMER – SQL CONNECTIVITY USING JDBC

A Report
Submitted in partial fulfillment of the Requirements
for the COURSE

DATABASE MANAGEMENT SYSTEMS

By

KOTHA MANISAIGANESH **<1602-21-737-032>**
Under the guidance of Ms B. Leelavathy

Department of Information Technology Vasavi College of Engineering (Autonomous)
(Affiliated to Osmania University)
Ibrahimbagh, Hyderabad-31
2022-2023

BONAFIDE CERTIFICATE

This is to certify that this project report titled

'Online Registration and Processing of Dairy Farmers Producing Milk and Other Livestock Products'

is a project work of ***Kotha Manisaiganesh*** bearing roll no. 1602-21-737-032 who carried out this project under my supervision in the IV semester for the academic year 2022- 2023

3

Signature
External Examiner

Signature
Internal Examiner

ABSTRACT

The Online Registration and Processing of Dairy Farmers Producing Milk and Other Livestock Products project is designed to facilitate the registration and management of dairy farmers, their livestock, and the products they produce. The project provides a web-based platform where farmers can register and manage their farm and livestock details, and also sell their products online. The project is built using a MySQL database to store information about farms, farmers, livestock, products and orders. The system allows farmers to register their farms and provide details about their livestock, such as their type, date of birth, and farm location. The farmers can also add products that they produce, such as milk, cheese, and butter, and set prices for them.Customers can browse the products available for sale and place orders online. The system allows customers to pay for their orders securely. The system also provides a dashboard for farmers to manage their orders and deliveries.
Overall, the Online Registration and Processing of Dairy Farmers Producing Milk and Other Livestock Products project aims to streamline the registration and management process for dairy farmers, while also providing customers with an easy and convenient way to purchase high-quality dairy products.

Requirement Analysis

List of Tables:

- Farmer
- Farm
- Livestock
- Product
- Customer
- Orders

List of Attributes with their Domain Types:

Farmer

FARMER_ID	NUMBER(3)	PRIMARY KEY
NAME	VARCHAR2(20)	NOT NULL
EMAIL	VARCHAR2(40)	
PHONE	VARCHAR2(10)	

Farm

FARM_ID	NUMBER(3)	PRIMARY KEY
FARMER_ID	NUMBER(3)	FOREIGN KEY(FARMER)
FARMER_NAME	VARCHAR2(30)	
ADDRESS	VARCHAR2(30)	
CITY	VARCHAR2(10)	
STATE	VARCHAR2(10)	
ZIPCODE	VARCHAR2(6)	

Livestock

LIVESTOCK_ID	NUMBER(3)	PRIMARY KEY
FARM_ID	NUMBER(3)	FOREIGN KEY(FARM)
TYPE	VARCHAR2(20)	
GENDER	VARCHAR2(7)	
DOB	DATE	
BREED	VARCHAR2(20)	
DATE_ADDED	DATE	

Product

PRODUCT_ID	NUMBER(3)	PRIMARY KEY
LIVESTOCK_ID	NUMBER(3)	FOREIGN KEY(LIVESTOCK)
TYPE	VARCHAR2(10)	
QUANTITY	NUMBER(4)	
UNIT_PRICE	NUMBER(4)	
DATE_PRODUCED	DATE	

Customer

CUSTOMER_ID	NUMBER(3)	PRIMARY KEY
NAME	VARCHAR2(20)	
PHONE	VARCHAR2(10)	
EMAIL	VARCHAR2(20)	
ADDRESS	VARCHAR2(30)	
CITY	VARCHAR2(30)	
STATE	VARCHAR2(10)	
ZIPCODE	VARCHAR2(6)	

Orders

ORDER_ID	NUMBER(3)	PRIMARY KEY
CUSTOMER_ID	NUMBER(3)	FOREIGN KEY(CUSTOMER)
PRODUCT_ID	NUMBER(3)	FOREIGN KEY(PRODUCT)
ORDER_DATE	DATE	

AIM AND PRIORITY OF THE PROJECT

To create a **Java GUI-based** desktop application that connects students looking for career choices with skills and Interest. It takes values like student name, username, Age, Skills, etc through forms which are then updated in the database using JDBC connectivity.

ARCHITECTURE AND TECHNOLOGY

Software used:

Java, Oracle 11g Database, Java SE version 14, Run SQL.

Java SWING:

Java SWING is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier AWT. Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

SQL:

Structure Query Language(SQL) is a database query language used for storing and managing data in **Relational** DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

DESIGN

Entity Relationship Diagram

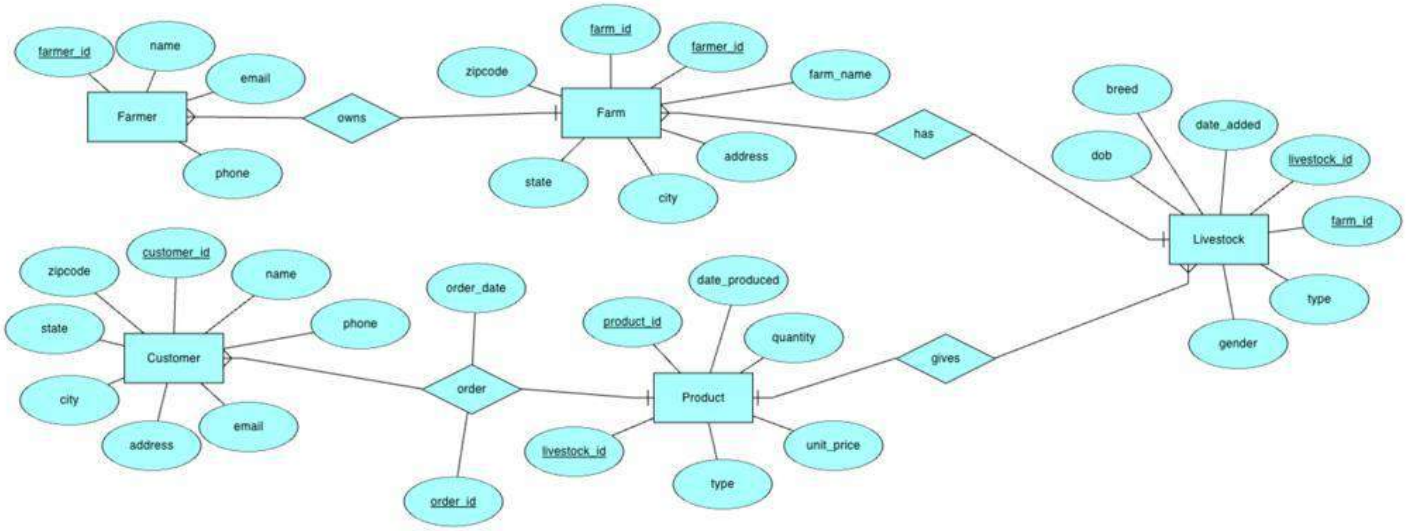


TABLE CREATED IN SQL:

1.Farmer Table

```
mysql> create table Farmer(farmer_id INTEGER(3), name VARCHAR(20), email VARCHAR(20), phone INTEGER(10), PRI
MARY KEY(farmer_id));
Query OK, 0 rows affected, 2 warnings (0.04 sec)

mysql> desc Farmer;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| farmer_id | int         | NO   | PRI | NULL    |       |
| name      | varchar(20) | YES  |     | NULL    |       |
| email     | varchar(20) | YES  |     | NULL    |       |
| phone     | int         | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.02 sec)
```

2.Farm Table

```
mysql> create table Farm(farm_id INTEGER(3), farmer_id INTEGER(3), farm_name VARCHAR(10), address VARCHAR(10
), city VARCHAR(10), state VARCHAR(10), zipcode INTEGER(6), PRIMARY KEY(farm_id,farmer_id), FOREIGN KEY(farm
er_id) REFERENCES Farmer(farmer_id));
Query OK, 0 rows affected, 3 warnings (0.02 sec)

mysql> desc Farm;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| farm_id   | int         | NO   | PRI | NULL    |       |
| farmer_id | int         | NO   | PRI | NULL    |       |
| farm_name | varchar(10) | YES  |     | NULL    |       |
| address   | varchar(10) | YES  |     | NULL    |       |
| city      | varchar(10) | YES  |     | NULL    |       |
| state     | varchar(10) | YES  |     | NULL    |       |
| zipcode   | int         | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

3.Livestock Table

```
mysql> create table Livestock(livestock_id INTEGER(3),
    -> farm_id INTEGER(3),
    -> type VARCHAR(20),
    -> gender VARCHAR(7),
    -> dob DATE,
    -> breed VARCHAR(20),
    -> date_added DATE,
    -> PRIMARY KEY(livestock_id,farm_id),
    -> FOREIGN KEY(farm_id) REFERENCES Farm(farm_id));
Query OK, 0 rows affected, 2 warnings (0.02 sec)

mysql> desc Livestock;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| livestock_id | int         | NO   | PRI | NULL    |       |
| farm_id      | int         | NO   | PRI | NULL    |       |
| type         | varchar(20) | YES  |     | NULL    |       |
| gender       | varchar(7)  | YES  |     | NULL    |       |
| dob          | date        | YES  |     | NULL    |       |
| breed        | varchar(20) | YES  |     | NULL    |       |
| date_added   | date        | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

4. Product Table

```
mysql> create table Product(product_id INTEGER(3),
    -> livestock_id INTEGER(3),
    -> type VARCHAR(10),
    -> quantity INTEGER(4),
    -> unit_price INTEGER(4),
    -> date_produced DATE,
    -> PRIMARY KEY(product_id),
    -> FOREIGN KEY(livestock_id) REFERENCES Livestock(livestock_id));
Query OK, 0 rows affected, 4 warnings (0.02 sec)

mysql> desc Product;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| product_id    | int         | NO   | PRI | NULL    |       |
| livestock_id  | int         | YES  | MUL | NULL    |       |
| type          | varchar(10) | YES  |     | NULL    |       |
| quantity      | int         | YES  |     | NULL    |       |
| unit_price    | int         | YES  |     | NULL    |       |
| date_produced | date        | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

5. Customer table

```
mysql> create table Customer(customer_id INTEGER(3), name VARCHAR(20), phone INTEGER(10), email VARCHAR(20),
 address VARCHAR(10), city VARCHAR(10), state VARCHAR(10), zipcode INTEGER(6), PRIMARY KEY(customer_id));
Query OK, 0 rows affected, 3 warnings (0.01 sec)

mysql> desc Customer;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| customer_id | int         | NO   | PRI | NULL    |       |
| name        | varchar(20) | YES  |     | NULL    |       |
| phone       | int         | YES  |     | NULL    |       |
| email       | varchar(20) | YES  |     | NULL    |       |
| address     | varchar(10) | YES  |     | NULL    |       |
| city        | varchar(10) | YES  |     | NULL    |       |
| state       | varchar(10) | YES  |     | NULL    |       |
| zipcode     | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

6. Orders table

```
mysql> create table Orders(order_id INTEGER(3), customer_id INTEGER(3), product_id INTEGER(3), order_date DA
TE, PRIMARY KEY(order_id,customer_id,product_id), FOREIGN KEY(customer_id) REFERENCES Customer(customer_id),
 FOREIGN KEY(product_id) REFERENCES Product(product_id));
Query OK, 0 rows affected, 3 warnings (0.01 sec)

mysql> desc Orders;
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| order_id    | int  | NO   | PRI | NULL    |       |
| customer_id | int  | NO   | PRI | NULL    |       |
| product_id  | int  | NO   | PRI | NULL    |       |
| order_date  | date | YES  |     | NULL    |       |
+-------------+------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

DATABASE DESIGN:

```
SQL> select * from tab;


TNAME                          TABTYPE  CLUSTERID

------------------------------ ------- ----------

CUSTOMER                        TABLE

FARM                           TABLE

FARMER                          TABLE

LIVESTOCK                       TABLE

ORDERS                         TABLE

PRODUCT                         TABLE


6 rows selected.


SQL> desc Farmer;

 Name                                                                    Null?    Type

 ---------------------------------------------------------------------------------------------------------------------------------- -------- ------------

-------------------------------------------------------------------

 FARMER_ID                                                                NOT NULL NUMBER(3)

 NAME                                                                    NOT NULL VARCHAR2(30)
```

```
EMAIL                                                     VARCHAR2(40)

PHONE                                                     VARCHAR2(10)


SQL> desc Farm;

Name                                                      Null?    Type

---------------------------------------------------------------------------------------------------------------------- -------- ------------

-------------------------------------------------------------

FARM_ID                                                   NOT NULL NUMBER(3)

FARMER_ID                                                 NUMBER(3)

FARM_NAME                                                 VARCHAR2(30)

ADDRESS                                                   VARCHAR2(30)

CITY                                                      VARCHAR2(10)

STATE                                                     VARCHAR2(10)

ZIPCODE                                                   VARCHAR2(6)


SQL> desc Livestock;

Name                                                      Null?    Type

---------------------------------------------------------------------------------------------------------------------- -------- ------------

-------------------------------------------------------------

LIVESTOCK_ID                                              NOT NULL NUMBER(3)

FARM_ID                                                   NUMBER(3)

TYPE                                                      VARCHAR2(20)
```

```
GENDER                                                    VARCHAR2(7)

DOB                                                       DATE

BREED                                                     VARCHAR2(20)

DATE_ADDED                                                DATE


SQL> desc Product;

Name                                                      Null?    Type

----------------------------------------------------------------------------------------------------------------------------------- -------- ------------

-------------------------------------------------------------

PRODUCT_ID                                                NOT NULL NUMBER(3)

LIVESTOCK_ID                                              NUMBER(3)

TYPE                                                      VARCHAR2(10)

QUANTITY                                                  NUMBER(4)

UNIT_PRICE                                                NUMBER(4)

DATE_PRODUCED                                             DATE


SQL> desc Customer;

Name                                                      Null?    Type

----------------------------------------------------------------------------------------------------------------------------------- -------- ------------

-------------------------------------------------------------

CUSTOMER_ID                                               NOT NULL NUMBER(3)

NAME                                                      VARCHAR2(30)
```

```
PHONE                                                    VARCHAR2(10)

EMAIL                                                    VARCHAR2(40)

ADDRESS                                                  VARCHAR2(30)

CITY                                                     VARCHAR2(30)

STATE                                                    VARCHAR2(20)

ZIPCODE                                                  VARCHAR2(6)


SQL> desc Orders;

Name                                                     Null?    Type

---------------------------------------------------------------------------------------------------------------------- -------- ----------

-------------------------------------------------------------

ORDER_ID                                                 NOT NULL NUMBER(3)

CUSTOMER_ID                                              NUMBER(3)

PRODUCT_ID                                               NUMBER(3)

ORDER_DATE                                               DATE
```

DML Operations

1. INSERTING VALUES INTO FAMER TABLE:

```
mysql> insert into Farmer values(2,'Sundharam','sundharam16@gmail.com','7329746295');
Query OK, 1 row affected (0.01 sec)

mysql> insert into Farmer values(3,'Ramesh','ramesh89@gmail.com','9763412794');
Query OK, 1 row affected (0.00 sec)

mysql> select * from Farmer;
+-----------+-----------+-----------------------+------------+
| farmer_id | name      | email                 | phone      |
+-----------+-----------+-----------------------+------------+
|         1 | Rangaiah  | rangaiah12@gmail.com  | 7014321874 |
|         2 | Sundharam | sundharam16@gmail.com | 7329746295 |
|         3 | Ramesh    | ramesh89@gmail.com    | 9763412794 |
+-----------+-----------+-----------------------+------------+
3 rows in set (0.00 sec)
```

2. INSERTING VALUES INTO FARM TABLE:

```
mysql> insert into Farm values(501,1,'Happy Valley','Madhulapalli','Jagtial','Telangana','505452');
Query OK, 1 row affected (0.00 sec)

mysql> insert into Farm values(502,2,'Sundharam Acres','Chintakunta','Karimnagar','Telangana','518348'
);
Query OK, 1 row affected (0.00 sec)

mysql> insert into Farm values(503,3,'Ramesh Fields','Cherlapalli','Jagtial','Telangana','505454');
Query OK, 1 row affected (0.00 sec)

mysql> select * from Farm;
+---------+-----------+-----------------+--------------+------------+-----------+---------+
| farm_id | farmer_id | farm_name       | address      | city       | state     | zipcode |
+---------+-----------+-----------------+--------------+------------+-----------+---------+
|     501 |         1 | Happy Valley    | Madhulapalli | Jagtial    | Telangana | 505452  |
|     502 |         2 | Sundharam Acres | Chintakunta  | Karimnagar | Telangana | 518348  |
|     503 |         3 | Ramesh Fields   | Cherlapalli  | Jagtial    | Telangana | 505454  |
+---------+-----------+-----------------+--------------+------------+-----------+---------+
3 rows in set (0.00 sec)
```

3. INSERTING VALUES INTO LIVESTOCK TABLE:

```
mysql> insert into Livestock values(601,501,'Cattle','Female','2019-04-20','Angus','2019-05-05');
Query OK, 1 row affected (0.01 sec)

mysql> insert into Livestock values(602,502,'Sheep','Female','2021-02-10','Dorper','2021-03-01');
Query OK, 1 row affected (0.00 sec)

mysql> insert into Livestock values(603,503,'Goat','Male','2018-09-17','Boer','2018-10-01');
Query OK, 1 row affected (0.00 sec)

mysql> select * from Livestock;
+--------------+---------+--------+--------+------------+--------+------------+
| livestock_id | farm_id | type   | gender | dob        | breed  | date_added |
+--------------+---------+--------+--------+------------+--------+------------+
|          601 |     501 | Cattle | Female | 2019-04-20 | Angus  | 2019-05-05 |
|          602 |     502 | Sheep  | Female | 2021-02-10 | Dorper | 2021-03-01 |
|          603 |     503 | Goat   | Male   | 2018-09-17 | Boer   | 2018-10-01 |
+--------------+---------+--------+--------+------------+--------+------------+
3 rows in set (0.01 sec)
```

4. INSERTING VALUES INTO PRODUCT TABLE:

```
mysql> insert into Product values(201,501,'Milk',800,3,'2022-02-15');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`msgdb`.`product
`, CONSTRAINT `product_ibfk_1` FOREIGN KEY (`livestock_id`) REFERENCES `livestock` (`livestock_id`))
mysql> insert into Product values(201,601,'Milk',800,3,'2022-02-15');
Query OK, 1 row affected (0.01 sec)

mysql> insert into Product values(202,602,'Meat',50,20,'2022-03-01');
Query OK, 1 row affected (0.00 sec)

mysql> insert into Product values(202,602,'Goat',45,12,'2019-03-18');
ERROR 1062 (23000): Duplicate entry '202' for key 'product.PRIMARY'
mysql> insert into Product values(203,603,'Goat',45,12,'2019-03-18');
Query OK, 1 row affected (0.00 sec)

mysql> select * from Product;
+------------+--------------+------+----------+------------+---------------+
| product_id | livestock_id | type | quantity | unit_price | date_produced |
+------------+--------------+------+----------+------------+---------------+
|        201 |          601 | Milk |      800 |          3 | 2022-02-15    |
|        202 |          602 | Meat |       50 |         20 | 2022-03-01    |
|        203 |          603 | Goat |       45 |         12 | 2019-03-18    |
+------------+--------------+------+----------+------------+---------------+
3 rows in set (0.00 sec)
```

5. INSERTING VALUES INTO CUSTOMER TABLE:

```
mysql> insert into Customer values(201,'Rahul','8765432109','rahul576@gmail.com','Second Cross Road','Chennai'
,'TN','600001');
Query OK, 1 row affected (0.01 sec)

mysql> insert into Customer values(202,'Rani','9876543210','rani524@gmail.com','Gachibowli','Hyderabad','TS','
500032');
Query OK, 1 row affected (0.00 sec)

mysql> insert into Customer values(203,'Vikram','7654321098','vikram298@gmail.com','Durgam Cheruvu','Hyderabad
','TS','500081');
Query OK, 1 row affected (0.01 sec)

mysql> select * from Customer;
+-------------+--------+------------+--------------------+-------------------+-----------+-------+---------+
| customer_id | name   | phone      | email              | address           | city      | state | zipcode |
+-------------+--------+------------+--------------------+-------------------+-----------+-------+---------+
|         201 | Rahul  | 8765432109 | rahul576@gmail.com | Second Cross Road | Chennai   | TN    | 600001  |
|         202 | Rani   | 9876543210 | rani524@gmail.com  | Gachibowli        | Hyderabad | TS    | 500032  |
|         203 | Vikram | 7654321098 | vikram298@gmail.com | Durgam Cheruvu    | Hyderabad | TS    | 500081  |
+-------------+--------+------------+--------------------+-------------------+-----------+-------+---------+
3 rows in set (0.00 sec)
```

6.INSERTING VALUES INTO ORDERS TABLE:

```
mysql> insert into Orders values(101,201,202,'2022-03-04');
Query OK, 1 row affected (0.00 sec)

mysql> insert into Orders values(102,202,203,'2019-04-04');
Query OK, 1 row affected (0.00 sec)

mysql> insert into Orders values(103,203,201,'2022-03-01');
Query OK, 1 row affected (0.00 sec)

mysql> select * from Orders;
+----------+-------------+------------+------------+
| order_id | customer_id | product_id | order_date |
+----------+-------------+------------+------------+
|      101 |         201 |        202 | 2022-03-04 |
|      102 |         202 |        203 | 2019-04-04 |
|      103 |         203 |        201 | 2022-03-01 |
+----------+-------------+------------+------------+
3 rows in set (0.00 sec)
```

IMPLEMENTATION

JAVA-SQL Connectivity using JDBC:

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

The connection to the database can be performed using Java programming (JDBC API) as:

```
{

        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

         // Connect to Oracle Database

         Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE"
,"manisaiganesh","vasavi");

         Statement = con.createStatement()

         String query = “INSERT INTO FARMER VALUES(?,?,?,?)”;

         ResultSet rs = statement.executeQuery(query);

         JOptionPane.showMessageDialog(new JFrame(), "Upadated Successfully", "INFORMATION",
JOptionPane.INFORMATION_MESSAGE);

         rs.close();

         statement.close();

         con.close(); }
```

Front-end Programs:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.net.URL;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import java.util.Random;


public class FarmerCustomer extends JFrame implements ActionListener
{
   private JRadioButton farmerRadioButton, customerRadioButton;
   private JPanel imagePanel, radioPanel, insertPanel,updatePanel, farmerPanel, farmPanel, deletePanel,
livestockPanel, productPanel, customerPanel, productsPanel, orderPanel;
   private String farmer_id,customer_id;

   public FarmerCustomer()
   {
      setTitle("Farmer-Customer App");
      setSize(800, 600);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      mainPage();
      setLocationRelativeTo(null);
      setVisible(true);
   }

   private void mainPage() {
      farmerRadioButton = new JRadioButton("Farmer");
```

```
        customerRadioButton = new JRadioButton("Customer");

        ButtonGroup radioButtonGroup = new ButtonGroup();
        radioButtonGroup.add(farmerRadioButton);
        radioButtonGroup.add(customerRadioButton);

        radioPanel = new JPanel();
        radioPanel.setLayout(new FlowLayout());
        radioPanel.add(farmerRadioButton);
        radioPanel.add(customerRadioButton);
        try {
            URL imageURL = new URL("https://wallpapercave.com/wp/wp5520703.jpg");
            ImageIcon = new ImageIcon(imageURL);
            Image = imageIcon.getImage().getScaledInstance(getWidth(), getHeight(), Image.SCALE_SMOOTH);
            JLabel imageLabel = new JLabel(new ImageIcon(image));
            radioPanel.add(imageLabel, BorderLayout.SOUTH);
        } catch (IOException e) {
            e.printStackTrace();
        }

        add(radioPanel, BorderLayout.NORTH);

        farmerRadioButton.addActionListener(this);
        customerRadioButton.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == farmerRadioButton) {

            farmer_id = JOptionPane.showInputDialog(radioPanel, "Enter Farmer ID:", "Farmer ID",
JOptionPane.PLAIN_MESSAGE);

            final JMenuBar farmerMenuBar = new JMenuBar();
            final JMenu farmerMenu = new JMenu("Farmer");
            final JMenu FarmerItem = new JMenu("Farmer");
            final JMenuItem myInfoItem = new JMenuItem("My Profile");
            final JMenu myFarmsItem = new JMenu("My Farms");
            final JMenu myLivestockItem = new JMenu("My Livestock");
            final JMenu myProductsItem = new JMenu("My Products");
            final JMenuItem exitItem = new JMenuItem("Exit");

            FarmerItem.add(new JMenuItem("New Farmer")).addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
```

```
    {
        showInsertFarmerPanel();
    }
});
FarmerItem.add(new JMenuItem("Update Farmer")).addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        showUpdateFarmerPanel();
    }
});

myInfoItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        showFarmerInfoPanel();
    }
});

myFarmsItem.add(new JMenuItem("View Farms")).addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
            showFarmsInfoPanel();
    }
});

myFarmsItem.add(new JMenuItem("Add Farms")).addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
            showInsertFarmsPanel();
    }
});

myLivestockItem.add(new JMenuItem("View LiveStock")).addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
            showLivestockInfoPanel();
    }
});

myLivestockItem.add(new JMenuItem("Add Livestock")).addActionListener(new ActionListener()
{
```

```
    public void actionPerformed(ActionEvent e)
    {
            showInsertLivestockPanel();
    }
});


myProductsItem.add(new JMenuItem("View Products")).addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
            showProductInfoPanel();
    }
});

myProductsItem.add(new JMenuItem("Add Products")).addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
            showInsertProductPanel();
    }
});


exitItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});
imagePanel = new JPanel();
try {
    URL imageURL = new URL("https://wallpaperaccess.com/full/4293504.jpg");
    ImageIcon = new ImageIcon(imageURL);
    Image = imageIcon.getImage().getScaledInstance(getWidth(), 500, Image.SCALE_SMOOTH);
    JLabel imageLabel = new JLabel(new ImageIcon(image));
    imagePanel.add(imageLabel, BorderLayout.SOUTH);
} catch (IOException ex) {
    ex.printStackTrace();
}

JPanel bottomPanel = new JPanel(new GridBagLayout());
GridBagConstraints constraints = new GridBagConstraints();
constraints.gridx = 0;
constraints.gridy = 0;
```

```
        constraints.anchor = GridBagConstraints.CENTER;
        constraints.insets = new Insets(10, 0, 0, 0);

        JLabel highestOrdersLabel = new JLabel("Highest Orders By: ");
        highestOrdersLabel.setFont(new Font("Arial", Font.BOLD, 20)); // Increase the font size
        bottomPanel.add(highestOrdersLabel, constraints);

        // Execute the query to get the farmer with the highest orders
        try (Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
            String query = "SELECT * FROM (SELECT f.FARMER_ID, f.NAME, COUNT(o.ORDER_ID) AS
ORDER_COUNT FROM farmer f JOIN farm fa ON f.FARMER_ID = fa.FARMER_ID JOIN livestock l ON fa.FARM_ID
= l.FARM_ID JOIN product p ON l.LIVESTOCK_ID = p.LIVESTOCK_ID JOIN orders o ON p.PRODUCT_ID =
o.PRODUCT_ID GROUP BY f.FARMER_ID, f.NAME ORDER BY ORDER_COUNT DESC) WHERE ROWNUM = 1";
            PreparedStatement = connection.prepareStatement(query);
            ResultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                String farmerName = resultSet.getString("NAME");
                JLabel highestOrdersValueLabel = new JLabel(farmerName);
                highestOrdersValueLabel.setFont(new Font("Arial", Font.PLAIN, 18)); // Increase the font size

                constraints.gridx = 0;
                constraints.gridy = 1;
                constraints.insets = new Insets(10, 0, 0, 0);
                bottomPanel.add(highestOrdersValueLabel, constraints);
            }

            resultSet.close();
            preparedStatement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
        }

        add(bottomPanel, BorderLayout.SOUTH);

        add(imagePanel);
        farmerMenu.add(FarmerItem);
        farmerMenu.add(myInfoItem);
        farmerMenu.add(myFarmsItem);
        farmerMenu.add(myLivestockItem);
        farmerMenu.add(myProductsItem);
```

```
        farmerMenu.add(exitItem);
        farmerMenuBar.add(farmerMenu);
        setJMenuBar(farmerMenuBar);
        getContentPane().remove(radioPanel);
        revalidate();
        repaint();
    } else if (e.getSource() == customerRadioButton) {

         customer_id = JOptionPane.showInputDialog(radioPanel, "Enter Customer ID:", "Customer ID",
JOptionPane.PLAIN_MESSAGE);

         final JMenuBar customerMenuBar = new JMenuBar();
        final JMenu customerMenu = new JMenu("Customer");
        final JMenu customerItem = new JMenu("Customer");
        final JMenuItem myInfoItem = new JMenuItem("My Profile");
        final JMenuItem productsItem = new JMenuItem("View Products");
        final JMenuItem orderItem = new JMenuItem("Order");
        final JMenuItem myOrdersItem = new JMenuItem("My Orders");
        final JMenuItem exitItem = new JMenuItem("Exit");
        customerItem.add(new JMenuItem("New Customer")).addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                showInsertCustomerPanel();
            }
        });
        customerItem.add(new JMenuItem("Update Customer")).addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                showUpdateCustomerPanel();
            }
        });
        myInfoItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                showCustomerInfoPanel();
            }
        });
        productsItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                showProductsInfoPanel();
            }
        });
        orderItem.addActionListener(new ActionListener() {
```

```
            public void actionPerformed(ActionEvent e) {
                ShowInsertOrderPanel();
            }
        });
        myOrdersItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                ShowOrdersInfoPanel();
            }
        });
        exitItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });
        imagePanel = new JPanel();
        try {
            URL imageURL = new URL("https://wallpapercave.com/wp/wp8593074.jpg");
            ImageIcon = new ImageIcon(imageURL);
            Image = imageIcon.getImage().getScaledInstance(getWidth(), 500, Image.SCALE_SMOOTH);
            JLabel imageLabel = new JLabel(new ImageIcon(image));
            imagePanel.add(imageLabel, BorderLayout.SOUTH);
        } catch (IOException ex) {
            ex.printStackTrace();
        }

        JPanel bottomPanel = new JPanel(new GridBagLayout());
        GridBagConstraints constraints = new GridBagConstraints();
        constraints.gridx = 0;
        constraints.gridy = 0;
        constraints.anchor = GridBagConstraints.CENTER;
        constraints.insets = new Insets(10, 0, 0, 0);

        JLabel highestOrdersLabel = new JLabel("Highest Ordered By: ");
        highestOrdersLabel.setFont(new Font("Arial", Font.BOLD, 20)); // Increase the font size
        bottomPanel.add(highestOrdersLabel, constraints);

        // Execute the query to get the farmer with the highest orders
        try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
            String query = "SELECT c.CUSTOMER_ID, c.NAME, COUNT(o.ORDER_ID) AS ORDER_COUNT\r\n"
                    + "FROM customer c\r\n"
                    + "JOIN orders o ON c.CUSTOMER_ID = o.CUSTOMER_ID\r\n"
                    + "GROUP BY c.CUSTOMER_ID, c.NAME\r\n"
                    + "HAVING COUNT(o.ORDER_ID) = (\r\n"
```

```
                    + "  SELECT MAX(order_count)\r\n"
                    + "  FROM (\r\n"
                    + "    SELECT COUNT(ORDER_ID) AS order_count\r\n"
                    + "    FROM orders\r\n"
                    + "    GROUP BY CUSTOMER_ID\r\n"
                    + "  )\r\n"
                    + ")\r\n";
            PreparedStatement = connection.prepareStatement(query);
            ResultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                String customerName = resultSet.getString("NAME");
                JLabel highestOrdersValueLabel = new JLabel(customerName);
                highestOrdersValueLabel.setFont(new Font("Arial", Font.PLAIN, 18)); // Increase the font size

                constraints.gridx = 0;
                constraints.gridy = 1;
                constraints.insets = new Insets(10, 0, 0, 0);
                bottomPanel.add(highestOrdersValueLabel, constraints);
            }

            resultSet.close();
            preparedStatement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
        }

        add(bottomPanel, BorderLayout.SOUTH);

        add(imagePanel);

        customerMenu.add(customerItem);
        customerMenu.add(myInfoItem);
        customerMenu.add(productsItem);
        customerMenu.add(orderItem);
        customerMenu.add(myOrdersItem);
        customerMenu.add(exitItem);

        customerMenuBar.add(customerMenu);
        setJMenuBar(customerMenuBar);
        getContentPane().remove(radioPanel);
        revalidate();
```

```
        repaint();
      }
    }
    ////////////////////CUSTOMER////////////////////
    private void showInsertCustomerPanel()
    {
          removePreviousPanel();

          JLabel customeridLabel = new JLabel("Customer ID:");
      JTextField customeridTextField = new JTextField(20);
          JLabel nameLabel = new JLabel("Name:");
      JTextField nameTextField = new JTextField(20);
      JLabel phoneLabel = new JLabel("Phone:");
      JTextField phoneTextField = new JTextField(20);
      JLabel emailLabel = new JLabel("Email:");
      JTextField emailTextField = new JTextField(20);
      JLabel addressLabel = new JLabel("Address:");
      JTextField addressTextField = new JTextField(20);
      JLabel cityLabel = new JLabel("City:");
      JTextField cityTextField = new JTextField(20);
      JLabel stateLabel = new JLabel("State:");
      JTextField stateTextField = new JTextField(20);
      JLabel zipLabel = new JLabel("Zip Code:");
      JTextField zipTextField = new JTextField(20);
      JButton submitButton = new JButton("Submit");

      submitButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
         String customerId = customeridTextField.getText();
           String name = nameTextField.getText();
           String phone = phoneTextField.getText();
           String email = emailTextField.getText();
           String address = addressTextField.getText();
           String city = cityTextField.getText();
           String state = stateTextField.getText();
           String zip = zipTextField.getText();

           if (customerId.isEmpty() || name.isEmpty() || phone.isEmpty() || email.isEmpty() ||
address.isEmpty() || city.isEmpty() || state.isEmpty() || zip.isEmpty())   {
              JOptionPane.showMessageDialog(insertPanel, "Please fill in all fields", "Error",
JOptionPane.ERROR_MESSAGE);
              return;
           }
```

```
            try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
                String insertQuery = "INSERT INTO Customer (customer_id,name,phone, email,
address,city,state,zipcode) VALUES (?, ?, ?, ?,?,?,?,?)";
                PreparedStatement = connection.prepareStatement(insertQuery);
                preparedStatement.setString(1, customerId);
                preparedStatement.setString(2, name);
                preparedStatement.setString(3, phone);
                preparedStatement.setString(4, email);
                preparedStatement.setString(5, address);
                preparedStatement.setString(6, city);
                preparedStatement.setString(7, state);
                preparedStatement.setString(8, zip);

                int count = preparedStatement.executeUpdate();
                if (count > 0) {
                    JOptionPane.showMessageDialog(insertPanel, "Customer added successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(insertPanel, "Failed to add Customer", "Error",
JOptionPane.ERROR_MESSAGE);
                }

                customeridTextField.setText("");
                nameTextField.setText("");
                phoneTextField.setText("");
                emailTextField.setText("");
                addressTextField.setText("");
                cityTextField.setText("");
                stateTextField.setText("");
                zipTextField.setText("");

                preparedStatement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(insertPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    insertPanel = new JPanel();
    insertPanel.setLayout(new GridBagLayout());
```

```
GridBagConstraints constraints = new GridBagConstraints();
constraints.insets = new Insets(5, 5, 5, 5);

constraints.gridx = 0;
constraints.gridy = 0;
insertPanel.add(customeridLabel, constraints);

constraints.gridx = 1;
constraints.gridy = 0;
customeridTextField.setPreferredSize(new Dimension(150, 25));
insertPanel.add(customeridTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 1;
insertPanel.add(nameLabel, constraints);

constraints.gridx = 1;
constraints.gridy = 1;
nameTextField.setPreferredSize(new Dimension(150, 25));
insertPanel.add(nameTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 2;
insertPanel.add(phoneLabel, constraints);

constraints.gridx = 1;
constraints.gridy = 2;
phoneTextField.setPreferredSize(new Dimension(150, 25));
insertPanel.add(phoneTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 3;
insertPanel.add(emailLabel, constraints);

constraints.gridx = 1;
constraints.gridy = 3;
emailTextField.setPreferredSize(new Dimension(150, 25));
insertPanel.add(emailTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 4;
insertPanel.add(addressLabel, constraints);
```

```
    constraints.gridx = 1;
    constraints.gridy = 4;
    addressTextField.setPreferredSize(new Dimension(150, 25));
    insertPanel.add(addressTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 5;
    insertPanel.add(cityLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 5;
    cityTextField.setPreferredSize(new Dimension(150, 25));
    insertPanel.add(cityTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 6;
    insertPanel.add(stateLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 6;
    stateTextField.setPreferredSize(new Dimension(150, 25));
    insertPanel.add(stateTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 7;
    insertPanel.add(zipLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 7;
    zipTextField.setPreferredSize(new Dimension(150, 25));
    insertPanel.add(zipTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 8;
    constraints.gridwidth = 2;
    constraints.anchor = GridBagConstraints.CENTER;
    insertPanel.add(submitButton, constraints);

    getContentPane().add(insertPanel, BorderLayout.CENTER);
    revalidate();
    repaint();
}

private void showUpdateCustomerPanel() {
```

```
        updatePanel = new JPanel();
    removePreviousPanel();
    JLabel customeridLabel = new JLabel("Customer ID:");
    JTextField customeridTextField = new JTextField(20);
    customeridTextField.setText(customer_id);

    JLabel nameLabel = new JLabel("Name:");
    JTextField nameTextField = new JTextField(20);
    JLabel phoneLabel = new JLabel("Phone:");
    JTextField phoneTextField = new JTextField(20);
    JLabel emailLabel = new JLabel("Email:");
    JTextField emailTextField = new JTextField(20);
    JLabel addressLabel = new JLabel("Address:");
    JTextField addressTextField = new JTextField(20);
    JLabel cityLabel = new JLabel("City:");
    JTextField cityTextField = new JTextField(20);
    JLabel stateLabel = new JLabel("State:");
    JTextField stateTextField = new JTextField(20);
    JLabel zipLabel = new JLabel("Zip Code:");
    JTextField zipTextField = new JTextField(20);
    JButton updateButton = new JButton("Modify");

    // Retrieve previous values from the database using customerId
    try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "manisaiganesh",
"vasavi")) {
        String selectQuery = "SELECT name, phone, email, address, city, state, zipcode FROM Customer WHERE
customer_id = ?";
        PreparedStatement = connection.prepareStatement(selectQuery);
        preparedStatement.setString(1, customer_id);
        ResultSet = preparedStatement.executeQuery();
        if (resultSet.next()) {
            nameTextField.setText(resultSet.getString("name"));
            phoneTextField.setText(resultSet.getString("phone"));
            emailTextField.setText(resultSet.getString("email"));
            addressTextField.setText(resultSet.getString("address"));
            cityTextField.setText(resultSet.getString("city"));
            stateTextField.setText(resultSet.getString("state"));
            zipTextField.setText(resultSet.getString("zipcode"));
        }
        resultSet.close();
        preparedStatement.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(updatePanel, "Failed to connect to the database", "Error",
```

```
JOptionPane.ERROR_MESSAGE);
        }

        updateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String customerId = customeridTextField.getText();
                String name = nameTextField.getText();
                String phone = phoneTextField.getText();
                String email = emailTextField.getText();
                String address = addressTextField.getText();
                String city = cityTextField.getText();
                String state = stateTextField.getText();
                String zip = zipTextField.getText();

                try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
                    String updateQuery = "UPDATE Customer SET name = ?, phone = ?, email = ?, address = ?, city = ?,
state = ?, zipcode = ? WHERE customer_id = ?";
                    PreparedStatement = connection.prepareStatement(updateQuery);
                    preparedStatement.setString(1, name);
                    preparedStatement.setString(2, phone);
                    preparedStatement.setString(3, email);
                    preparedStatement.setString(4, address);
                    preparedStatement.setString(5, city);
                    preparedStatement.setString(6, state);
                    preparedStatement.setString(7, zip);
                    preparedStatement.setString(8, customerId);

                    int count = preparedStatement.executeUpdate();
                    if (count > 0) {
                        JOptionPane.showMessageDialog(updatePanel, "Customer details updated successfully",
"Success", JOptionPane.INFORMATION_MESSAGE);
                    } else {
                        JOptionPane.showMessageDialog(updatePanel, "Failed to update customer details", "Error",
JOptionPane.ERROR_MESSAGE);
                    }

                    customeridTextField.setText("");
                    nameTextField.setText("");
                    phoneTextField.setText("");
                    emailTextField.setText("");
                    addressTextField.setText("");
                    cityTextField.setText("");
                    stateTextField.setText("");
```

```
                zipTextField.setText("");

                preparedStatement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(updatePanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    updatePanel.setLayout(new GridBagLayout());

    GridBagConstraints constraints = new GridBagConstraints();
    constraints.insets = new Insets(5, 5, 5, 5);

    constraints.gridx = 0;
    constraints.gridy = 0;
    updatePanel.add(customeridLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 0;
    customeridTextField.setPreferredSize(new Dimension(150, 25));
    customeridTextField.setEditable(false); // Disable editing of customer ID
    updatePanel.add(customeridTextField, constraints);


    constraints.gridx = 0;
    constraints.gridy = 1;
    updatePanel.add(nameLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 1;
    nameTextField.setPreferredSize(new Dimension(150, 25));
    updatePanel.add(nameTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 2;
    updatePanel.add(phoneLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 2;
    phoneTextField.setPreferredSize(new Dimension(150, 25));
    updatePanel.add(phoneTextField, constraints);
```

```
constraints.gridx = 0;
constraints.gridy = 3;
updatePanel.add(emailLabel, constraints);

constraints.gridx = 1;
constraints.gridy = 3;
emailTextField.setPreferredSize(new Dimension(150, 25));
updatePanel.add(emailTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 4;
updatePanel.add(addressLabel, constraints);

constraints.gridx = 1;
constraints.gridy = 4;
addressTextField.setPreferredSize(new Dimension(150, 25));
updatePanel.add(addressTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 5;
updatePanel.add(cityLabel, constraints);

constraints.gridx = 1;
constraints.gridy = 5;
cityTextField.setPreferredSize(new Dimension(150, 25));
updatePanel.add(cityTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 6;
updatePanel.add(stateLabel, constraints);

constraints.gridx = 1;
constraints.gridy = 6;
stateTextField.setPreferredSize(new Dimension(150, 25));
updatePanel.add(stateTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 7;
updatePanel.add(zipLabel, constraints);

constraints.gridx = 1;
constraints.gridy = 7;
zipTextField.setPreferredSize(new Dimension(150, 25));
```

```
        updatePanel.add(zipTextField, constraints);

        constraints.gridx = 0;
        constraints.gridy = 8;
        constraints.gridwidth = 2;
        constraints.anchor = GridBagConstraints.CENTER;
        updatePanel.add(updateButton, constraints);

        getContentPane().add(updatePanel, BorderLayout.CENTER);
        revalidate();
        repaint();
    }


    private void showCustomerInfoPanel()
    {
            customerPanel = new JPanel();
        removePreviousPanel();


            try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
                String selectQuery = "SELECT * FROM Customer WHERE customer_id = ?";
                PreparedStatement = connection.prepareStatement(selectQuery);
                preparedStatement.setString(1, customer_id);

                ResultSet = preparedStatement.executeQuery();

                if (resultSet.next()) {
                    int customerIdResult = resultSet.getInt("customer_id");
                    String name = resultSet.getString("name");
                    String phone = resultSet.getString("phone");
                    String email = resultSet.getString("email");
                    String address = resultSet.getString("address");
                    String city = resultSet.getString("city");
                    String state = resultSet.getString("state");
                    String zipcode = resultSet.getString("zipcode");

                    JOptionPane.showMessageDialog(customerPanel, "Customer ID: " + customerIdResult +
"\nName: " + name + "\nPhone: " + phone + "\nEmail: " + email +"\nAddress: " + address +"\nCity: " + city
+"\nState: " + state + "\nZip Code: " + zipcode , "Customer Details", JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(customerPanel, "Customer not found", "Error",
JOptionPane.ERROR_MESSAGE);
```

```
            }

            resultSet.close();
            preparedStatement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(customerPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
        }




    getContentPane().setLayout(new BorderLayout());
    getContentPane().add(customerPanel, BorderLayout.CENTER);

    revalidate();
    repaint();
  }

  private void showProductsInfoPanel() {
    removePreviousPanel();

    JOptionPane.showMessageDialog(productsPanel, "Please note product_id before making an order...!",
"Alert", JOptionPane.INFORMATION_MESSAGE);

    try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "manisaiganesh",
"vasavi")) {
      String selectQuery = "SELECT * FROM PRODUCT";
      PreparedStatement = connection.prepareStatement(selectQuery);

      ResultSet = preparedStatement.executeQuery();

      DefaultTableModel tableModel = new DefaultTableModel() {
        public boolean isCellEditable(int row, int column) {
            return false;
        }
      };
      tableModel.addColumn("Product ID");
      tableModel.addColumn("Livestock ID");
      tableModel.addColumn("Type");
      tableModel.addColumn("Quantity");
      tableModel.addColumn("Unit Price");
      tableModel.addColumn("Date Produced");
```

```
        while (resultSet.next()) {
            int productIdResult = resultSet.getInt("product_id");
            int livestockId = resultSet.getInt("livestock_id");
            String type = resultSet.getString("type");
            String quantity = resultSet.getString("quantity");
            String unitPrice = resultSet.getString("unit_price");
            String dateProduced = resultSet.getString("date_produced");

            Object[] rowData = { productIdResult, livestockId, type, quantity, unitPrice, dateProduced };
            tableModel.addRow(rowData);
        }

        if (tableModel.getRowCount() == 0) {
            JOptionPane.showMessageDialog(productsPanel, "No Products found ", "Error",
JOptionPane.ERROR_MESSAGE);
        } else {
            JTable table = new JTable(tableModel);
            DefaultTableCellRenderer cellRenderer = new DefaultTableCellRenderer()
            {
                public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected,
boolean hasFocus, int row, int column)
                {
                    Component = super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row,
column);
                    component.setBackground(new Color(216, 237, 255));
                    return component;
                }
            };
            for (int i = 0; i < table.getColumnCount(); i++) {
                table.getColumnModel().getColumn(i).setCellRenderer(cellRenderer);
            }

            JScrollPane scrollPane = new JScrollPane(table);
            scrollPane.setPreferredSize(new Dimension(400, 200));

            productsPanel = new JPanel();
            productsPanel.setLayout(new BorderLayout());
            productsPanel.add(scrollPane, BorderLayout.CENTER);

            getContentPane().removeAll();
            getContentPane().add(productsPanel, BorderLayout.CENTER);

            revalidate();
```

```
            repaint();
          }
      } catch (SQLException ex) {
          ex.printStackTrace();
          JOptionPane.showMessageDialog(farmPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
      }

      revalidate();
      repaint();
   }


   private void ShowInsertOrderPanel()
   {
          removePreviousPanel();
          JLabel customeridLabel = new JLabel("Customer ID:");
      JTextField customeridTextField = new JTextField(20);
      customeridTextField.setText(customer_id);
          JLabel productidLabel = new JLabel("Product ID:");
      JTextField productidTextField = new JTextField(20);
      JButton orderButton = new JButton("Order");

      orderButton.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
          int orderId = generateRandomNumber();
            String customerId = customeridTextField.getText();
            String productId = productidTextField.getText();


            if (productId.isEmpty() || customerId.isEmpty()) {
                JOptionPane.showMessageDialog(insertPanel, "Please fill in all fields", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }


            try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
                String insertQuery = "INSERT INTO Orders (order_id,customer_id,product_id, order_date) VALUES
(?, ?, ?, SYSDATE)";
                PreparedStatement = connection.prepareStatement(insertQuery);
                preparedStatement.setInt(1, orderId);
                preparedStatement.setString(2, customerId);
```

```
                preparedStatement.setString(3, productId);

                int count = preparedStatement.executeUpdate();
                if (count > 0) {
                    JOptionPane.showMessageDialog(insertPanel, "Hurray...Order Successful..!", "Success",
JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(insertPanel, "Order Failed", "Error",
JOptionPane.ERROR_MESSAGE);
                }


                productidTextField.setText("");
                customeridTextField.setText("");

                preparedStatement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(insertPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    insertPanel = new JPanel();
    insertPanel.setLayout(new GridBagLayout());

    GridBagConstraints constraints = new GridBagConstraints();
    constraints.insets = new Insets(5, 5, 5, 5);

    constraints.gridx = 0;
    constraints.gridy = 0;
    insertPanel.add(productidLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 0;
    productidTextField.setPreferredSize(new Dimension(150, 25));
    insertPanel.add(productidTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 1;
    customeridTextField.setEditable(false);
    insertPanel.add(customeridLabel, constraints);
```

```
        constraints.gridx = 1;
        constraints.gridy = 1;
        customeridTextField.setPreferredSize(new Dimension(150, 25));
        insertPanel.add(customeridTextField, constraints);

        constraints.gridx = 0;
        constraints.gridy = 2;
        constraints.gridwidth = 2;
        constraints.anchor = GridBagConstraints.CENTER;
        insertPanel.add(orderButton, constraints);

        getContentPane().add(insertPanel, BorderLayout.CENTER);
        revalidate();
        repaint();
    }

    private void ShowOrdersInfoPanel()
    {
            orderPanel = new JPanel();
        removePreviousPanel();


            try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
                String selectQuery = "SELECT * FROM ORDERS WHERE CUSTOMER_ID = ?";
                PreparedStatement = connection.prepareStatement(selectQuery);
                preparedStatement.setString(1, customer_id);

                ResultSet = preparedStatement.executeQuery();

                DefaultTableModel tableModel = new DefaultTableModel() {
                    @Override
                    public boolean isCellEditable(int row, int column) {
                        return false;
                    }
                };
                tableModel.addColumn("Order Id");
                tableModel.addColumn("Customer Id");
                tableModel.addColumn("Product Id");
                tableModel.addColumn("Order Date");

                while (resultSet.next()) {
                      int orderIdResult = resultSet.getInt("order_id");
                    int customerIdResult = resultSet.getInt("customer_id");
```

```
                int productIdResult = resultSet.getInt("product_id");
                String orderDate = resultSet.getString("order_date");

                Object[] rowData = { orderIdResult, customerIdResult, productIdResult, orderDate };
                tableModel.addRow(rowData);
            }
            if (tableModel.getRowCount() == 0) {
                JOptionPane.showMessageDialog(orderPanel, "No Orders found for the given Customer ID..!",
"Error", JOptionPane.ERROR_MESSAGE);
            } else {
                JTable table = new JTable(tableModel);

                JScrollPane scrollPane = new JScrollPane(table);
                scrollPane.setPreferredSize(new Dimension(400, 200));

                orderPanel = new JPanel();
                orderPanel.setLayout(new BorderLayout());
                orderPanel.add(scrollPane, BorderLayout.CENTER);

                getContentPane().removeAll();
                getContentPane().add(orderPanel, BorderLayout.CENTER);

                revalidate();
                repaint();
            }
        }

         catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(orderPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
        }




    getContentPane().setLayout(new BorderLayout());
    getContentPane().add(orderPanel, BorderLayout.CENTER);

    revalidate();
    repaint();
  }
```

```
/////////////////////FARMER/////////////////////


private void showInsertFarmerPanel() {
    removePreviousPanel();

    JLabel idLabel = new JLabel("Farmer ID:");
    JTextField idTextField = new JTextField(20);
    JLabel nameLabel = new JLabel("Name:");
    JTextField nameTextField = new JTextField(20);
    JLabel emailLabel = new JLabel("Email:");
    JTextField emailTextField = new JTextField(20);
    JLabel phoneLabel = new JLabel("Phone:");
    JTextField phoneTextField = new JTextField(20);
    JButton addButton = new JButton("Add");

    idTextField.setPreferredSize(new Dimension(200, 30));
    nameTextField.setPreferredSize(new Dimension(200, 30));
    emailTextField.setPreferredSize(new Dimension(200, 30));
    phoneTextField.setPreferredSize(new Dimension(200, 30));
    addButton.setPreferredSize(new Dimension(100, 30));

    addButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String farmerId = idTextField.getText();
            String name = nameTextField.getText();
            String email = emailTextField.getText();
            String phone = phoneTextField.getText();


            if (farmerId.isEmpty() || name.isEmpty() || email.isEmpty() || phone.isEmpty()) {
                JOptionPane.showMessageDialog(insertPanel, "Please fill in all fields", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }


            try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
                String insertQuery = "INSERT INTO Farmer (farmer_id, name, email, phone) VALUES (?, ?, ?, ?)";
                PreparedStatement = connection.prepareStatement(insertQuery);
                preparedStatement.setString(1, farmerId);
                preparedStatement.setString(2, name);
                preparedStatement.setString(3, email);
```

```
                preparedStatement.setString(4, phone);

                int count = preparedStatement.executeUpdate();
                if (count > 0) {
                    JOptionPane.showMessageDialog(insertPanel, "Farmer added successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(insertPanel, "Failed to add farmer", "Error",
JOptionPane.ERROR_MESSAGE);
                }


                idTextField.setText("");
                nameTextField.setText("");
                emailTextField.setText("");
                phoneTextField.setText("");

                preparedStatement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(insertPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    insertPanel = new JPanel();
    insertPanel.setLayout(new GridBagLayout());

    GridBagConstraints constraints = new GridBagConstraints();
    constraints.insets = new Insets(5, 5, 5, 5);

    constraints.gridx = 0;
    constraints.gridy = 0;
    insertPanel.add(idLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 0;
    insertPanel.add(idTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 1;
    insertPanel.add(nameLabel, constraints);
```

```
    constraints.gridx = 1;
    constraints.gridy = 1;
    insertPanel.add(nameTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 2;
    insertPanel.add(emailLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 2;
    insertPanel.add(emailTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 3;
    insertPanel.add(phoneLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 3;
    insertPanel.add(phoneTextField, constraints);

    constraints.gridx = 1;
    constraints.gridy = 4;
    insertPanel.add(addButton, constraints);

    getContentPane().add(insertPanel, BorderLayout.CENTER);
    revalidate();
    repaint();
}

private void showUpdateFarmerPanel() {
    removePreviousPanel();

    JLabel nameLabel = new JLabel("Name:");
    JTextField nameTextField = new JTextField(20);
    JLabel emailLabel = new JLabel("Email:");
    JTextField emailTextField = new JTextField(20);
    JLabel phoneLabel = new JLabel("Phone:");
    JTextField phoneTextField = new JTextField(20);
    JButton addButton = new JButton("Modify");

    nameTextField.setPreferredSize(new Dimension(250, 30));
    emailTextField.setPreferredSize(new Dimension(250, 30));
    phoneTextField.setPreferredSize(new Dimension(250, 30));
    addButton.setPreferredSize(new Dimension(100, 30));
```

```
    try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "manisaiganesh",
"vasavi")) {
        String selectQuery = "SELECT name, email, phone FROM FARMER WHERE farmer_id = ?";
        PreparedStatement = connection.prepareStatement(selectQuery);
        preparedStatement.setString(1, farmer_id);

        ResultSet = preparedStatement.executeQuery();
        if (resultSet.next()) {
            // Retrieve existing values
            String existingName = resultSet.getString("name");
            String existingEmail = resultSet.getString("email");
            String existingPhone = resultSet.getString("phone");

            // Display existing values in text fields
            nameTextField.setText(existingName);
            emailTextField.setText(existingEmail);
            phoneTextField.setText(existingPhone);
        }

        addButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String name = nameTextField.getText();
                String email = emailTextField.getText();
                String phone = phoneTextField.getText();

                try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
                    String updateQuery = "UPDATE FARMER SET name = ?, email = ?, phone = ? WHERE farmer_id =
?";
                    PreparedStatement updateStatement = connection.prepareStatement(updateQuery);
                    updateStatement.setString(1, name);
                    updateStatement.setString(2, email);
                    updateStatement.setString(3, phone);
                    updateStatement.setString(4, farmer_id);

                    int count = updateStatement.executeUpdate();
                    if (count > 0) {
                        JOptionPane.showMessageDialog(updatePanel, "Farmer updated successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
                    } else {
                        JOptionPane.showMessageDialog(updatePanel, "Failed to update farmer", "Error",
JOptionPane.ERROR_MESSAGE);
                    }
```

```
                // Clear text fields
                nameTextField.setText("");
                emailTextField.setText("");
                phoneTextField.setText("");

                updateStatement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(updatePanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    updatePanel = new JPanel();
    updatePanel.setLayout(new GridBagLayout());

    GridBagConstraints constraints = new GridBagConstraints();
    constraints.insets = new Insets(5, 5, 5, 5);

    constraints.gridx = 0;
    constraints.gridy = 1;
    updatePanel.add(nameLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 1;
    updatePanel.add(nameTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 2;
    updatePanel.add(emailLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 2;
    updatePanel.add(emailTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 3;
    updatePanel.add(phoneLabel, constraints);

    constraints.gridx = 1;
    constraints.gridy = 3;
    updatePanel.add(phoneTextField, constraints);
```

```
            constraints.gridx = 1;
            constraints.gridy = 4;
            updatePanel.add(addButton, constraints);

            getContentPane().add(updatePanel, BorderLayout.CENTER);
            revalidate();
            repaint();
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(updatePanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }

    private void showFarmerInfoPanel() {

            try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
                String selectQuery = "SELECT * FROM Farmer WHERE farmer_id = ?";
                PreparedStatement = connection.prepareStatement(selectQuery);
                preparedStatement.setString(1, farmer_id);

                ResultSet = preparedStatement.executeQuery();

                if (resultSet.next()) {
                    int farmerIdResult = resultSet.getInt("farmer_id");
                    String name = resultSet.getString("name");
                    String email = resultSet.getString("email");
                    String phone = resultSet.getString("phone");

                    JOptionPane.showMessageDialog(farmerPanel, "Farmer ID: " + farmerIdResult + "\nName: " +
name + "\nEmail: " + email + "\nPhone: " + phone, "Farmer Details", JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(farmerPanel, "Farmer not found", "Error",
JOptionPane.ERROR_MESSAGE);
                }

                resultSet.close();
                preparedStatement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(farmerPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
```

```
            }

        farmerPanel = new JPanel();
        revalidate();
        repaint();
    }

    private void showProductInfoPanel() {
        productPanel = new JPanel();
        removePreviousPanel();
        try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "manisaiganesh",
"vasavi")) {
            String selectQuery = "SELECT p.* FROM Product p, Livestock l, Farm f WHERE p.LIVESTOCK_ID =
l.LIVESTOCK_ID AND l.FARM_ID = f.FARM_ID AND f.FARMER_ID = ?";
            PreparedStatement = connection.prepareStatement(selectQuery);
            preparedStatement.setString(1, farmer_id);

            ResultSet = preparedStatement.executeQuery();

            DefaultTableModel tableModel = new DefaultTableModel() {
                @Override
                public boolean isCellEditable(int row, int column) {
                    return true;
                }
            };
            tableModel.addColumn("Product Id");
            tableModel.addColumn("LiveStock Id");
            tableModel.addColumn("Type");
            tableModel.addColumn("Quantity");
            tableModel.addColumn("Unit Price");
            tableModel.addColumn("Date Produced");

            while (resultSet.next()) {
                int productIdResult = resultSet.getInt("product_id");
                int livestockIdResult = resultSet.getInt("livestock_id");
                String type = resultSet.getString("type");
                String quantity = resultSet.getString("quantity");
                String unit_price = resultSet.getString("unit_price");
                String date_produced = resultSet.getString("date_produced");

                Object[] rowData = {productIdResult, livestockIdResult, type, quantity, unit_price, date_produced};
                tableModel.addRow(rowData);
            }
            if (tableModel.getRowCount() == 0) {
```

```
            JOptionPane.showMessageDialog(productPanel, "No Products found for the given Farmer ID",
"Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JTable table = new JTable(tableModel);

            JScrollPane scrollPane = new JScrollPane(table);
            scrollPane.setPreferredSize(new Dimension(400, 200));

            productPanel = new JPanel();
            productPanel.setLayout(new BorderLayout());
            productPanel.add(scrollPane, BorderLayout.CENTER);

        // Create Save button
            JButton saveButton = new JButton("Save");
            saveButton.addActionListener(e -> {
                int rowCount = tableModel.getRowCount();
                if (rowCount == 0) {
                    JOptionPane.showMessageDialog(productPanel, "No rows to save", "Error",
JOptionPane.ERROR_MESSAGE);
                } else {
                    int confirm = JOptionPane.showConfirmDialog(productPanel, "Are you sure you want to save the
changes?", "Confirm Save", JOptionPane.YES_NO_OPTION);
                    if (confirm == JOptionPane.YES_OPTION) {
                        try (Connection updateConnection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "manisaiganesh", "vasavi")) {
                            String updateQuery = "UPDATE Product SET type = ?, quantity = ?, unit_price = ?,
date_produced = TO_DATE(?, 'YYYY-MM-DD') WHERE product_id = ?";
                            PreparedStatement updateStatement =
updateConnection.prepareStatement(updateQuery);

                            for (int row = 0; row < rowCount; row++) {
                                int productId = (int) tableModel.getValueAt(row, 0);
                                String type = (String) tableModel.getValueAt(row, 2);
                                String quantity = (String) tableModel.getValueAt(row, 3);
                                String unitPrice = (String) tableModel.getValueAt(row, 4);
                                String dateProduced = (String) tableModel.getValueAt(row, 5);

                                updateStatement.setString(1, type);
                                updateStatement.setString(2, quantity);
                                updateStatement.setString(3, unitPrice);
                                updateStatement.setString(4, dateProduced);
                                updateStatement.setInt(5, productId);

                                updateStatement.executeUpdate();
```

```
                    }

                    JOptionPane.showMessageDialog(productPanel, "Changes saved successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
                } catch (SQLException ex) {
                    ex.printStackTrace();
                    JOptionPane.showMessageDialog(productPanel, "Failed to save changes: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        }
    });


    // Create Delete button
    JButton deleteButton = new JButton("Delete");
    deleteButton.addActionListener(e -> {
        int selectedRow = table.getSelectedRow();
        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(productPanel, "No row selected", "Error",
JOptionPane.ERROR_MESSAGE);
        } else {
            int confirm = JOptionPane.showConfirmDialog(productPanel, "Are you sure you want to delete
the selected row?", "Confirm Delete", JOptionPane.YES_NO_OPTION);
            if (confirm == JOptionPane.YES_OPTION) {
                int productId = (int) tableModel.getValueAt(selectedRow, 0);
                try (Connection deleteConnection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "manisaiganesh", "vasavi")) {
                    String deleteQuery = "DELETE FROM Product WHERE product_id = ?";
                    PreparedStatement deleteStatement = deleteConnection.prepareStatement(deleteQuery);
                    deleteStatement.setInt(1, productId);
                    deleteStatement.executeUpdate();
                    tableModel.removeRow(selectedRow);
                    JOptionPane.showMessageDialog(productPanel, "Row deleted successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
                } catch (SQLException ex) {
                    ex.printStackTrace();
                    JOptionPane.showMessageDialog(productPanel, "Failed to connect to the database",
"Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        }
    });
```

```
            // Create button panel
            JPanel buttonPanel = new JPanel();
            buttonPanel.add(saveButton);
            buttonPanel.add(deleteButton);

            // Create main panel
            JPanel mainPanel = new JPanel();
            mainPanel.setLayout(new BorderLayout());
            mainPanel.add(scrollPane, BorderLayout.CENTER);
            mainPanel.add(buttonPanel, BorderLayout.SOUTH);

            getContentPane().removeAll();
            getContentPane().add(mainPanel, BorderLayout.CENTER);

            revalidate();
            repaint();
          }
        } catch (SQLException ex) {
          ex.printStackTrace();
          JOptionPane.showMessageDialog(productPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
        }

        getContentPane().add(productPanel, BorderLayout.CENTER);
        revalidate();
        repaint();
    }


    private void showInsertProductPanel()
    {
           removePreviousPanel();

           JLabel productidLabel = new JLabel("Product ID:");
        JTextField productidTextField = new JTextField(20);
           JLabel livestockidLabel = new JLabel("Livestock ID:");
        JTextField livestockidTextField = new JTextField(20);
        JLabel typeLabel = new JLabel("Type:");
        JTextField typeTextField = new JTextField(20);
        JLabel quantityLabel = new JLabel("Quantity:");
        JTextField quantityTextField = new JTextField(20);
        JLabel unitPriceLabel = new JLabel("Unit Price:");
        JTextField unitPriceTextField = new JTextField(20);
        JLabel dateProducedLabel = new JLabel("Date Produced(YYYY-MM-DD):");
```

```
JTextField dateProducedTextField = new JTextField(20);
JButton submitButton = new JButton("Submit");

productidTextField.setPreferredSize(new Dimension(250, 30));
livestockidTextField.setPreferredSize(new Dimension(250, 30));
typeTextField.setPreferredSize(new Dimension(250, 30));
quantityTextField.setPreferredSize(new Dimension(250, 30));
unitPriceTextField.setPreferredSize(new Dimension(250, 30));
dateProducedTextField.setPreferredSize(new Dimension(250, 30));
submitButton.setPreferredSize(new Dimension(100, 30));
;

submitButton.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    String productId = productidTextField.getText();
      String livestockId = livestockidTextField.getText();
      String type = typeTextField.getText();
      String quantity = quantityTextField.getText();
      String unitPrice = unitPriceTextField.getText();
      String dateProduced = dateProducedTextField.getText();


      if (productId.isEmpty() || livestockId.isEmpty() || type.isEmpty() || quantity.isEmpty() ||
unitPrice.isEmpty() || dateProduced.isEmpty()) {
         JOptionPane.showMessageDialog(insertPanel, "Please fill in all fields", "Error",
JOptionPane.ERROR_MESSAGE);
         return;
      }


      try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
         String insertQuery = "INSERT INTO Product (product_id,livestock_id,type, quantity,
unit_price,date_produced) VALUES (?, ?, ?, ?,?,TO_DATE(?, 'YYYY-MM-DD'))";
         PreparedStatement = connection.prepareStatement(insertQuery);
         preparedStatement.setString(1, productId);
         preparedStatement.setString(2, livestockId);
         preparedStatement.setString(3, type);
         preparedStatement.setString(4, quantity);
         preparedStatement.setString(5, unitPrice);
         preparedStatement.setString(6, dateProduced);

         int count = preparedStatement.executeUpdate();
         if (count > 0) {
```

```
                JOptionPane.showMessageDialog(insertPanel, "Product added successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(insertPanel, "Failed to add Product", "Error",
JOptionPane.ERROR_MESSAGE);
            }


            productidTextField.setText("");
            livestockidTextField.setText("");
            typeTextField.setText("");
            quantityTextField.setText("");
            unitPriceTextField.setText("");
            dateProducedTextField.setText("");

            preparedStatement.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(insertPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }
});

insertPanel = new JPanel();
insertPanel.setLayout(new GridBagLayout());
GridBagConstraints constraints = new GridBagConstraints();
constraints.fill = GridBagConstraints.HORIZONTAL;
constraints.insets = new Insets(5, 5, 5, 5);

constraints.gridx = 0;
constraints.gridy = 0;
insertPanel.add(productidLabel, constraints);

constraints.gridx = 1;
insertPanel.add(productidTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 1;
insertPanel.add(livestockidLabel, constraints);

constraints.gridx = 1;
insertPanel.add(livestockidTextField, constraints);
```

```
    constraints.gridx = 0;
    constraints.gridy = 2;
    insertPanel.add(typeLabel, constraints);

    constraints.gridx = 1;
    insertPanel.add(typeTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 3;
    insertPanel.add(quantityLabel, constraints);

    constraints.gridx = 1;
    insertPanel.add(quantityTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 4;
    insertPanel.add(unitPriceLabel, constraints);

    constraints.gridx = 1;
    insertPanel.add(unitPriceTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 5;
    insertPanel.add(dateProducedLabel, constraints);

    constraints.gridx = 1;
    insertPanel.add(dateProducedTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 6;
    constraints.gridwidth = 2;
    insertPanel.add(submitButton, constraints);

    getContentPane().add(insertPanel, BorderLayout.CENTER);
    revalidate();
    repaint();
}


private void showInsertLivestockPanel()
{
        removePreviousPanel();

        JLabel livestockidLabel = new JLabel("Livestock ID:");
```

```
JTextField livestockidTextField = new JTextField(20);
    JLabel farmidLabel = new JLabel("Farm ID:");
JTextField farmidTextField = new JTextField(20);
JLabel typeLabel = new JLabel("Type:");
JTextField typeTextField = new JTextField(20);
JLabel genderLabel = new JLabel("Gender:");
JTextField genderTextField = new JTextField(20);
JLabel dobLabel = new JLabel("DOB(YYYY-MM-DD):");
JTextField dobTextField = new JTextField(20);
JLabel breedLabel = new JLabel("Breed:");
JTextField breedTextField = new JTextField(20);
JLabel dateAddedLabel = new JLabel("Date Added(YYYY-MM-DD):");
JTextField dateAddedTextField = new JTextField(20);
JButton submitButton = new JButton("Submit");

submitButton.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
    String livestockId = livestockidTextField.getText();
      String farmId = farmidTextField.getText();
      String type = typeTextField.getText();
      String gender = genderTextField.getText();
      String dob = dobTextField.getText();
      String breed = breedTextField.getText();
      String dateAdded = dateAddedTextField.getText();


      if (livestockId.isEmpty() || farmId.isEmpty() || type.isEmpty() || gender.isEmpty() || dob.isEmpty() ||
breed.isEmpty() || dateAdded.isEmpty()) {
         JOptionPane.showMessageDialog(insertPanel, "Please fill in all fields", "Error",
JOptionPane.ERROR_MESSAGE);
         return;
      }


      try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
         String insertQuery = "INSERT INTO Livestock (livestock_id,farm_id,type, gender,
dob,breed,date_added) VALUES (?, ?, ?, ?,TO_DATE(?, 'YYYY-MM-DD'),?,TO_DATE(?, 'YYYY-MM-DD'))";
         PreparedStatement = connection.prepareStatement(insertQuery);
         preparedStatement.setString(1, livestockId);
         preparedStatement.setString(2, farmId);
         preparedStatement.setString(3, type);
         preparedStatement.setString(4, gender);
         preparedStatement.setString(5, dob);
```

```
                preparedStatement.setString(6, breed);
                preparedStatement.setString(7, dateAdded);

                int count = preparedStatement.executeUpdate();
                if (count > 0) {
                    JOptionPane.showMessageDialog(insertPanel, "Livestock added successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(insertPanel, "Failed to add Livestock", "Error",
JOptionPane.ERROR_MESSAGE);
                }


                livestockidTextField.setText("");
                farmidTextField.setText("");
                typeTextField.setText("");
                genderTextField.setText("");
                dobTextField.setText("");
                breedTextField.setText("");
                dateAddedTextField.setText("");

                preparedStatement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(insertPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    insertPanel = new JPanel();
    insertPanel.setLayout(new GridBagLayout());
    GridBagConstraints constraints = new GridBagConstraints();
    constraints.fill = GridBagConstraints.HORIZONTAL;
    constraints.insets = new Insets(5, 5, 5, 5);

    constraints.gridx = 0;
    constraints.gridy = 0;
    insertPanel.add(livestockidLabel, constraints);

    constraints.gridx = 1;
    livestockidTextField.setPreferredSize(new Dimension(150, 25));
    insertPanel.add(livestockidTextField, constraints);
```

```
constraints.gridx = 0;
constraints.gridy = 1;
insertPanel.add(farmidLabel, constraints);

constraints.gridx = 1;
farmidTextField.setPreferredSize(new Dimension(150, 25));
insertPanel.add(farmidTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 2;
insertPanel.add(typeLabel, constraints);

constraints.gridx = 1;
typeTextField.setPreferredSize(new Dimension(150, 25));
insertPanel.add(typeTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 3;
insertPanel.add(genderLabel, constraints);

constraints.gridx = 1;
genderTextField.setPreferredSize(new Dimension(150, 25));
insertPanel.add(genderTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 4;
insertPanel.add(dobLabel, constraints);

constraints.gridx = 1;
dobTextField.setPreferredSize(new Dimension(150, 25));
insertPanel.add(dobTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 5;
insertPanel.add(breedLabel, constraints);

constraints.gridx = 1;
breedTextField.setPreferredSize(new Dimension(150, 25));
insertPanel.add(breedTextField, constraints);

constraints.gridx = 0;
constraints.gridy = 6;
insertPanel.add(dateAddedLabel, constraints);
```

```
        constraints.gridx = 1;
        dateAddedTextField.setPreferredSize(new Dimension(150, 25));
        insertPanel.add(dateAddedTextField, constraints);

        constraints.gridx = 0;
        constraints.gridy = 7;
        constraints.gridwidth = 2;
        insertPanel.add(submitButton, constraints);

        getContentPane().add(insertPanel, BorderLayout.CENTER);
        revalidate();
        repaint();
    }

    private void showLivestockInfoPanel() {
        livestockPanel = new JPanel();
        removePreviousPanel();

        try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "manisaiganesh",
"vasavi")) {
            String selectQuery = "SELECT l.* FROM Livestock l, Farm f WHERE l.FARM_ID = f.FARM_ID AND
f.FARMER_ID = ?";
            PreparedStatement = connection.prepareStatement(selectQuery);
            preparedStatement.setString(1, farmer_id);

            ResultSet = preparedStatement.executeQuery();

            DefaultTableModel tableModel = new DefaultTableModel() {
                @Override
                public boolean isCellEditable(int row, int column) {
                    return column != 0; // Allow editing all columns except the first one (LiveStock Id)
                }
            };
            tableModel.addColumn("LiveStock Id");
            tableModel.addColumn("Farm Id");
            tableModel.addColumn("Type");
            tableModel.addColumn("Gender");
            tableModel.addColumn("Dob");
            tableModel.addColumn("Breed");
            tableModel.addColumn("Date Added");

            while (resultSet.next()) {
                int livestockIdResult = resultSet.getInt("livestock_id");
                int farmIdResult = resultSet.getInt("farm_id");
```

```
            String type = resultSet.getString("type");
            String gender = resultSet.getString("gender");
            String dob = resultSet.getString("dob");
            String breed = resultSet.getString("breed");
            String date_added = resultSet.getString("date_added");

            Object[] rowData = { livestockIdResult, farmIdResult, type, gender, dob, breed, date_added };
            tableModel.addRow(rowData);
        }
        if (tableModel.getRowCount() == 0) {
            JOptionPane.showMessageDialog(livestockPanel, "No Livestock found for the given Farmer ID",
"Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JTable table = new JTable(tableModel);

         // Create a Delete button
            JButton deleteButton = new JButton("Delete");
            deleteButton.addActionListener(e -> {
                int[] selectedRows = table.getSelectedRows();
                if (selectedRows.length == 0) {
                    JOptionPane.showMessageDialog(livestockPanel, "Please select rows to delete", "Error",
JOptionPane.ERROR_MESSAGE);
                } else {
                    int confirm = JOptionPane.showConfirmDialog(livestockPanel, "Are you sure you want to delete
the selected rows?", "Confirm Delete", JOptionPane.YES_NO_OPTION);
                    if (confirm == JOptionPane.YES_OPTION) {
                        try (Connection deleteConnection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "manisaiganesh", "vasavi")) {
                            String deleteQuery = "DELETE FROM Livestock WHERE livestock_id = ?";
                            PreparedStatement deleteStatement = deleteConnection.prepareStatement(deleteQuery);

                            int deleteCount = 0; // Track the number of rows deleted

                            for (int row : selectedRows) {
                                int livestockId = (int) table.getValueAt(row, 0);
                                deleteStatement.setInt(1, livestockId);
                                int rowsAffected = deleteStatement.executeUpdate();
                                deleteCount += rowsAffected;
                            }

                            if (deleteCount > 0) {
                                JOptionPane.showMessageDialog(livestockPanel, "Selected rows deleted successfully",
"Success", JOptionPane.INFORMATION_MESSAGE);
                                // Refresh the table
```

```
                    showLivestockInfoPanel();
                } else {
                    JOptionPane.showMessageDialog(livestockPanel, "No rows were deleted", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(livestockPanel, "Failed to delete selected rows", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    }
});



// Create a Save button
JButton saveButton = new JButton("Save");
saveButton.addActionListener(e -> {
    int rowCount = tableModel.getRowCount();
    if (rowCount == 0) {
        JOptionPane.showMessageDialog(livestockPanel, "No rows to save", "Error",
JOptionPane.ERROR_MESSAGE);
    } else {
        int confirm = JOptionPane.showConfirmDialog(livestockPanel, "Are you sure you want to save the
changes?", "Confirm Save", JOptionPane.YES_NO_OPTION);
        if (confirm == JOptionPane.YES_OPTION) {
            try (Connection updateConnection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "manisaiganesh", "vasavi")) {
                String updateQuery = "UPDATE Livestock SET type = ?, gender = ?, dob = TO_DATE(?, 'YYYY-
MM-DD'), breed = ?, date_added = TO_DATE(?, 'YYYY-MM-DD') WHERE livestock_id = ? and farm_id=?";
                PreparedStatement updateStatement =
updateConnection.prepareStatement(updateQuery);

                for (int row = 0; row < rowCount; row++) {
                    int livestockId = (int) tableModel.getValueAt(row, 0);
                    int farmId = (int) tableModel.getValueAt(row, 1);
                    String type = (String) tableModel.getValueAt(row, 2);
                    String gender = (String) tableModel.getValueAt(row, 3);
                    String dob = (String) tableModel.getValueAt(row, 4);
                    String breed = (String) tableModel.getValueAt(row, 5);
                    String date_added = (String) tableModel.getValueAt(row, 6);

                    updateStatement.setString(1, type);
```

```
                    updateStatement.setString(2, gender);
                    updateStatement.setString(3, dob);
                    updateStatement.setString(4, breed);
                    updateStatement.setString(5, date_added);
                    updateStatement.setInt(6, livestockId);
                    updateStatement.setInt(7, farmId);

                    updateStatement.executeUpdate();
                }

                JOptionPane.showMessageDialog(livestockPanel, "Changes saved successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(livestockPanel, "Failed to connect to the database",
"Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}});



        JScrollPane scrollPane = new JScrollPane(table);
        scrollPane.setPreferredSize(new Dimension(400, 200));

        livestockPanel.setLayout(new BorderLayout());
        livestockPanel.add(scrollPane, BorderLayout.CENTER);

        JPanel buttonPanel = new JPanel();
        buttonPanel.add(deleteButton);
        buttonPanel.add(saveButton);
        livestockPanel.add(buttonPanel, BorderLayout.SOUTH);

        getContentPane().removeAll();
        getContentPane().add(livestockPanel, BorderLayout.CENTER);

        revalidate();
        repaint();
    }
} catch (SQLException ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(livestockPanel, "Failed to connect to the database", "Error",
```

```
JOptionPane.ERROR_MESSAGE);
        }

        revalidate();
        repaint();
    }









    private void showInsertFarmsPanel()
    {
            removePreviousPanel();

            JLabel farmidLabel = new JLabel("Farm ID:");
        JTextField farmidTextField = new JTextField(20);
            JLabel farmeridLabel = new JLabel("Farmer ID:");
        JTextField farmeridTextField = new JTextField(20);
        JLabel farmNameLabel = new JLabel("Farm Name:");
        JTextField farmNameTextField = new JTextField(20);
        JLabel addressLabel = new JLabel("Address:");
        JTextField addressTextField = new JTextField(20);
        JLabel cityLabel = new JLabel("City:");
        JTextField cityTextField = new JTextField(20);
        JLabel stateLabel = new JLabel("State:");
        JTextField stateTextField = new JTextField(20);
        JLabel zipLabel = new JLabel("Zip Code:");
        JTextField zipTextField = new JTextField(20);
        JButton addButton = new JButton("Add");

        addButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
             String farmId = farmidTextField.getText();
                String farmerId = farmeridTextField.getText();
                String farmName = farmNameTextField.getText();
                String address = addressTextField.getText();
                String city = cityTextField.getText();
                String state = stateTextField.getText();
                String zip = zipTextField.getText();
```

```
            if (farmId.isEmpty() || farmerId.isEmpty() || farmName.isEmpty() || address.isEmpty() ||
city.isEmpty() || state.isEmpty() || zip.isEmpty()) {
                JOptionPane.showMessageDialog(insertPanel, "Please fill in all fields", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }


            try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"manisaiganesh", "vasavi")) {
                String insertQuery = "INSERT INTO Farm (farm_id,farmer_id, farm_name, address,
city,state,zipcode) VALUES (?, ?, ?, ?, ?, ?, ?)";
                PreparedStatement = connection.prepareStatement(insertQuery);
                preparedStatement.setString(1, farmId);
                preparedStatement.setString(2, farmerId);
                preparedStatement.setString(3, farmName);
                preparedStatement.setString(4, address);
                preparedStatement.setString(5, city);
                preparedStatement.setString(6, state);
                preparedStatement.setString(7, zip);

                int count = preparedStatement.executeUpdate();
                if (count > 0) {
                    JOptionPane.showMessageDialog(insertPanel, "Farm added successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(insertPanel, "Failed to add farm", "Error",
JOptionPane.ERROR_MESSAGE);
                }


                farmidTextField.setText("");
                farmeridTextField.setText("");
                farmNameTextField.setText("");
                addressTextField.setText("");
                cityTextField.setText("");
                stateTextField.setText("");
                zipTextField.setText("");

                preparedStatement.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(insertPanel, "Failed to connect to the database", "Error",
```

```
JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    insertPanel = new JPanel();
    insertPanel.setLayout(new GridBagLayout());
    GridBagConstraints constraints = new GridBagConstraints();
    constraints.fill = GridBagConstraints.HORIZONTAL;
    constraints.insets = new Insets(5, 5, 5, 5);

    constraints.gridx = 0;
    constraints.gridy = 0;
    insertPanel.add(farmidLabel, constraints);

    constraints.gridx = 1;
    farmidTextField.setPreferredSize(new Dimension(150, 25));
    insertPanel.add(farmidTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 1;
    insertPanel.add(farmeridLabel, constraints);

    constraints.gridx = 1;
    farmeridTextField.setPreferredSize(new Dimension(150, 25));
    insertPanel.add(farmeridTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 2;
    insertPanel.add(farmNameLabel, constraints);

    constraints.gridx = 1;
    farmNameTextField.setPreferredSize(new Dimension(150, 25));
    insertPanel.add(farmNameTextField, constraints);

    constraints.gridx = 0;
    constraints.gridy = 3;
    insertPanel.add(addressLabel, constraints);

    constraints.gridx = 1;
    addressTextField.setPreferredSize(new Dimension(150, 25));
    insertPanel.add(addressTextField, constraints);

    constraints.gridx = 0;
```

```
        constraints.gridy = 4;
        insertPanel.add(cityLabel, constraints);

        constraints.gridx = 1;
        cityTextField.setPreferredSize(new Dimension(150, 25));
        insertPanel.add(cityTextField, constraints);

        constraints.gridx = 0;
        constraints.gridy = 5;
        insertPanel.add(stateLabel, constraints);

        constraints.gridx = 1;
        stateTextField.setPreferredSize(new Dimension(150, 25));
        insertPanel.add(stateTextField, constraints);

        constraints.gridx = 0;
        constraints.gridy = 6;
        insertPanel.add(zipLabel, constraints);

        constraints.gridx = 1;
        zipTextField.setPreferredSize(new Dimension(150, 25));
        insertPanel.add(zipTextField, constraints);

        constraints.gridx = 0;
        constraints.gridy = 7;
        constraints.gridwidth = 2;
        insertPanel.add(addButton, constraints);

        getContentPane().add(insertPanel, BorderLayout.CENTER);
        revalidate();
        repaint();
    }



    private void showFarmsInfoPanel() {
        farmPanel = new JPanel();
        removePreviousPanel();

        try (Connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "manisaiganesh",
"vasavi")) {
            String selectQuery = "SELECT * FROM Farm WHERE farmer_id = ?";
            PreparedStatement = connection.prepareStatement(selectQuery);
            preparedStatement.setString(1, farmer_id);
```

```
        ResultSet = preparedStatement.executeQuery();

        DefaultTableModel tableModel = new DefaultTableModel() {
            @Override
            public boolean isCellEditable(int row, int column) {
                return true; // Allow editing0
            }
        };
        tableModel.addColumn("Farm ID");
        tableModel.addColumn("Farm Name");
        tableModel.addColumn("Address");
        tableModel.addColumn("City");
        tableModel.addColumn("State");
        tableModel.addColumn("Zip Code");

        while (resultSet.next()) {
            int farmIdResult = resultSet.getInt("farm_id");
            String farm_name = resultSet.getString("farm_name");
            String address = resultSet.getString("address");
            String city = resultSet.getString("city");
            String state = resultSet.getString("state");
            String zipcode = resultSet.getString("zipcode");

            Object[] rowData = { farmIdResult, farm_name, address, city, state, zipcode };
            tableModel.addRow(rowData);
        }
        if (tableModel.getRowCount() == 0) {
            JOptionPane.showMessageDialog(farmPanel, "No farms found for the given Farmer ID", "Error",
JOptionPane.ERROR_MESSAGE);
        } else {
            JTable table = new JTable(tableModel);

            // Create a Delete button
            JButton deleteButton = new JButton("Delete");
            deleteButton.addActionListener(e -> {
                int[] selectedRows = table.getSelectedRows();
                if (selectedRows.length == 0) {
                    JOptionPane.showMessageDialog(farmPanel, "Please select rows to delete", "Error",
JOptionPane.ERROR_MESSAGE);
                } else {
                    int confirm = JOptionPane.showConfirmDialog(farmPanel, "Are you sure you want to delete the
selected rows?", "Confirm Delete", JOptionPane.YES_NO_OPTION);
                    if (confirm == JOptionPane.YES_OPTION) {
```

```
                    try (Connection deleteConnection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "manisaiganesh", "vasavi")) {
                        String deleteQuery = "DELETE FROM Farm WHERE farm_id = ?";
                        PreparedStatement deleteStatement = deleteConnection.prepareStatement(deleteQuery);

                        for (int row : selectedRows) {
                            int farmId = (int) table.getValueAt(row, 0);
                            deleteStatement.setInt(1, farmId);
                            deleteStatement.executeUpdate();
                        }

                        JOptionPane.showMessageDialog(farmPanel, "Selected rows deleted successfully",
"Success", JOptionPane.INFORMATION_MESSAGE);
                        // Refresh the table
                        showFarmsInfoPanel();
                    } catch (SQLException ex) {
                        ex.printStackTrace();
                        JOptionPane.showMessageDialog(farmPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
        });

        // Create a Save button
        JButton saveButton = new JButton("Save");
        saveButton.addActionListener(e -> {
            int rowCount = tableModel.getRowCount();
            if (rowCount == 0) {
                JOptionPane.showMessageDialog(farmPanel, "No rows to save", "Error",
JOptionPane.ERROR_MESSAGE);
            } else {
                int confirm = JOptionPane.showConfirmDialog(farmPanel, "Are you sure you want to save the
changes?", "Confirm Save", JOptionPane.YES_NO_OPTION);
                if (confirm == JOptionPane.YES_OPTION) {
                    try (Connection updateConnection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "manisaiganesh", "vasavi")) {
                        String updateQuery = "UPDATE Farm SET farm_name = ?, address = ?, city = ?, state = ?,
zipcode = ? WHERE farm_id = ?";
                        PreparedStatement updateStatement =
updateConnection.prepareStatement(updateQuery);

                        for (int row = 0; row < rowCount; row++) {
                            int farmId = (int) table.getValueAt(row, 0);
```

```
                    String farmName = (String) table.getValueAt(row, 1);
                    String address = (String) table.getValueAt(row, 2);
                    String city = (String) table.getValueAt(row, 3);
                    String state = (String) table.getValueAt(row, 4);
                    String zipcode = (String) table.getValueAt(row, 5);

                    updateStatement.setString(1, farmName);
                    updateStatement.setString(2, address);
                    updateStatement.setString(3, city);
                    updateStatement.setString(4, state);
                    updateStatement.setString(5, zipcode);
                    updateStatement.setInt(6, farmId);

                    updateStatement.executeUpdate();
                }

                JOptionPane.showMessageDialog(farmPanel, "Changes saved successfully", "Success",
JOptionPane.INFORMATION_MESSAGE);
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(farmPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    }
});

// Create a panel to hold the buttons
JPanel buttonPanel = new JPanel();
buttonPanel.add(deleteButton);
buttonPanel.add(saveButton);

// Create a panel to hold the table and button panel
JPanel tablePanel = new JPanel(new BorderLayout());
tablePanel.add(new JScrollPane(table), BorderLayout.CENTER);
tablePanel.add(buttonPanel, BorderLayout.SOUTH);

farmPanel.setLayout(new BorderLayout());
farmPanel.add(tablePanel, BorderLayout.CENTER);

getContentPane().removeAll();
getContentPane().add(farmPanel, BorderLayout.CENTER);

revalidate();
```

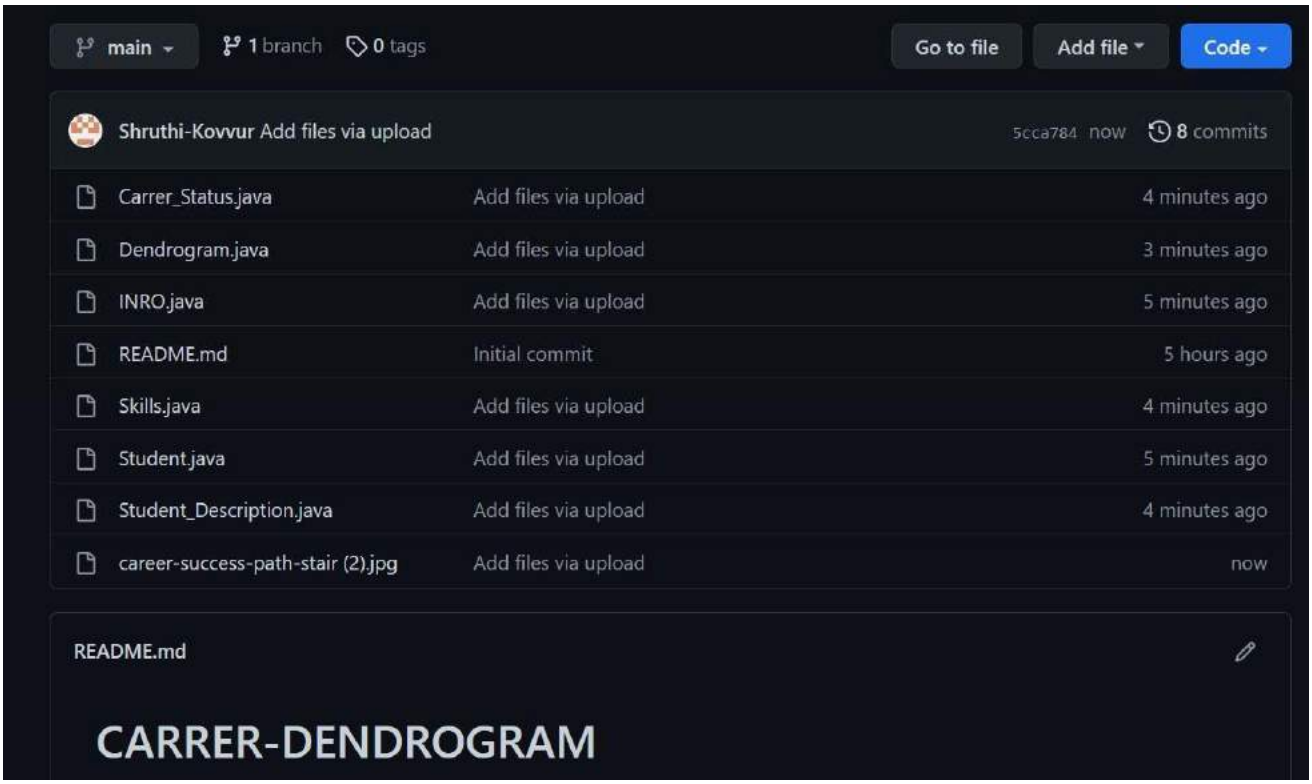
```
            repaint();
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(farmPanel, "Failed to connect to the database", "Error",
JOptionPane.ERROR_MESSAGE);
    }

    revalidate();
    repaint();
  }


  public static int generateRandomNumber() {
    Random = new Random();
    int randomNumber = random.nextInt(999) + 1;
    return randomNumber;
  }

  private void removePreviousPanel()
  {
    if (insertPanel != null) {
        getContentPane().remove(insertPanel);
    }
    if (updatePanel != null) {
        getContentPane().remove(updatePanel);
    }
    if (deletePanel != null) {
        getContentPane().remove(deletePanel);
    }
    if (farmerPanel != null) {
        getContentPane().remove(farmerPanel);
    }
    if (farmPanel != null) {
        getContentPane().remove(farmPanel);
    }
    if (livestockPanel != null) {
        getContentPane().remove(livestockPanel);
    }
    if (productPanel != null) {
        getContentPane().remove(productPanel);
    }
    if (customerPanel != null) {
        getContentPane().remove(customerPanel);
```

```
        }
        if (productsPanel != null) {
            getContentPane().remove(productsPanel);
        }
        if (orderPanel != null) {
            getContentPane().remove(orderPanel);
        }
        if (imagePanel != null) {
            getContentPane().remove(imagePanel);
        }
    }

    public static void main(String[] args)
    {
        SwingUtilities.invokeLater(new Runnable()
        {
            public void run()
            {
                new FarmerCustomer();
            }
        });
    }
}
```

GitHub Links and Folder Structure

Link: <gitlink to be given clearly>

Folder Structure:

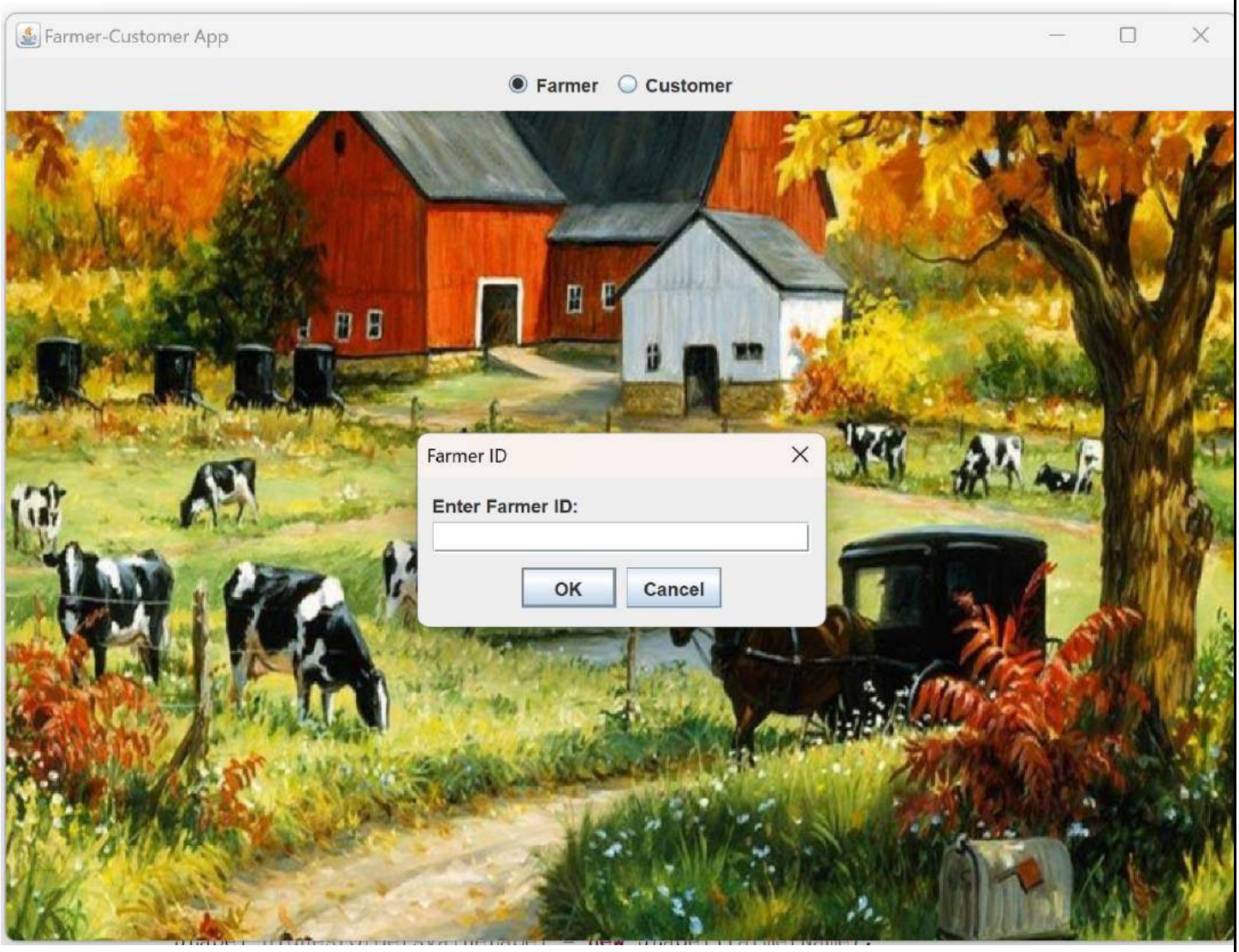


TESTING

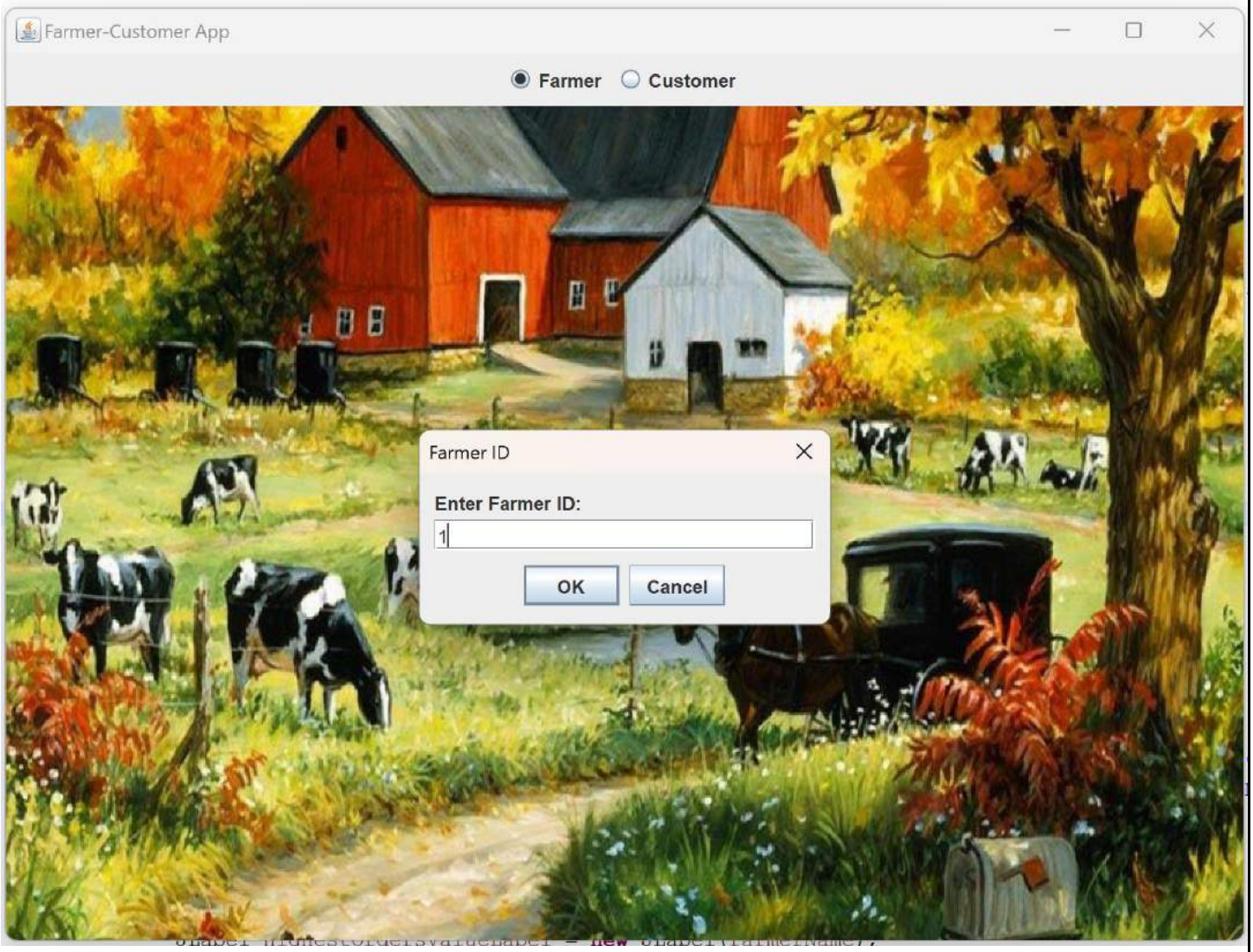
INTRODUCTION PAGE:

FARMER PAGE:

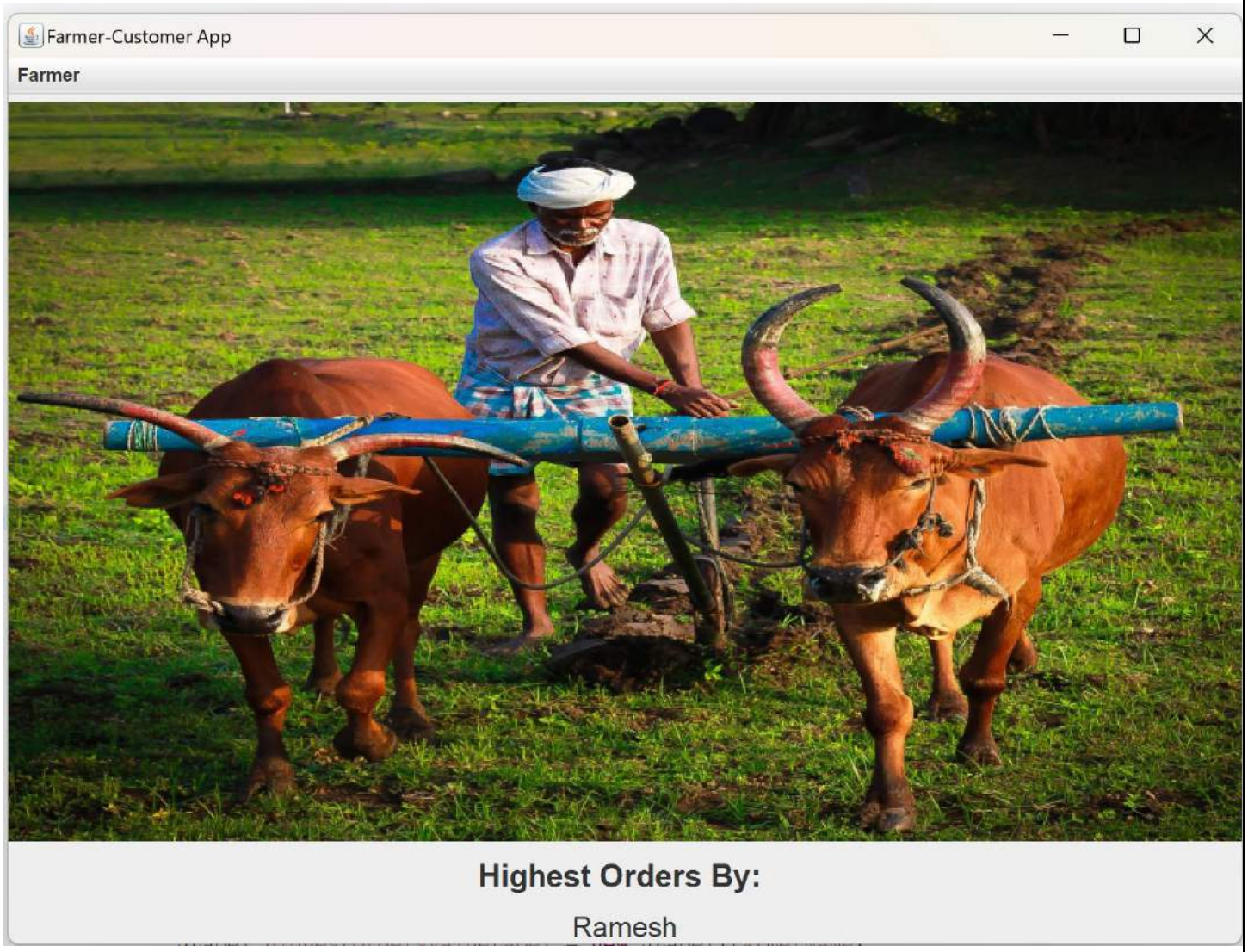
I.SELECT FARMER:



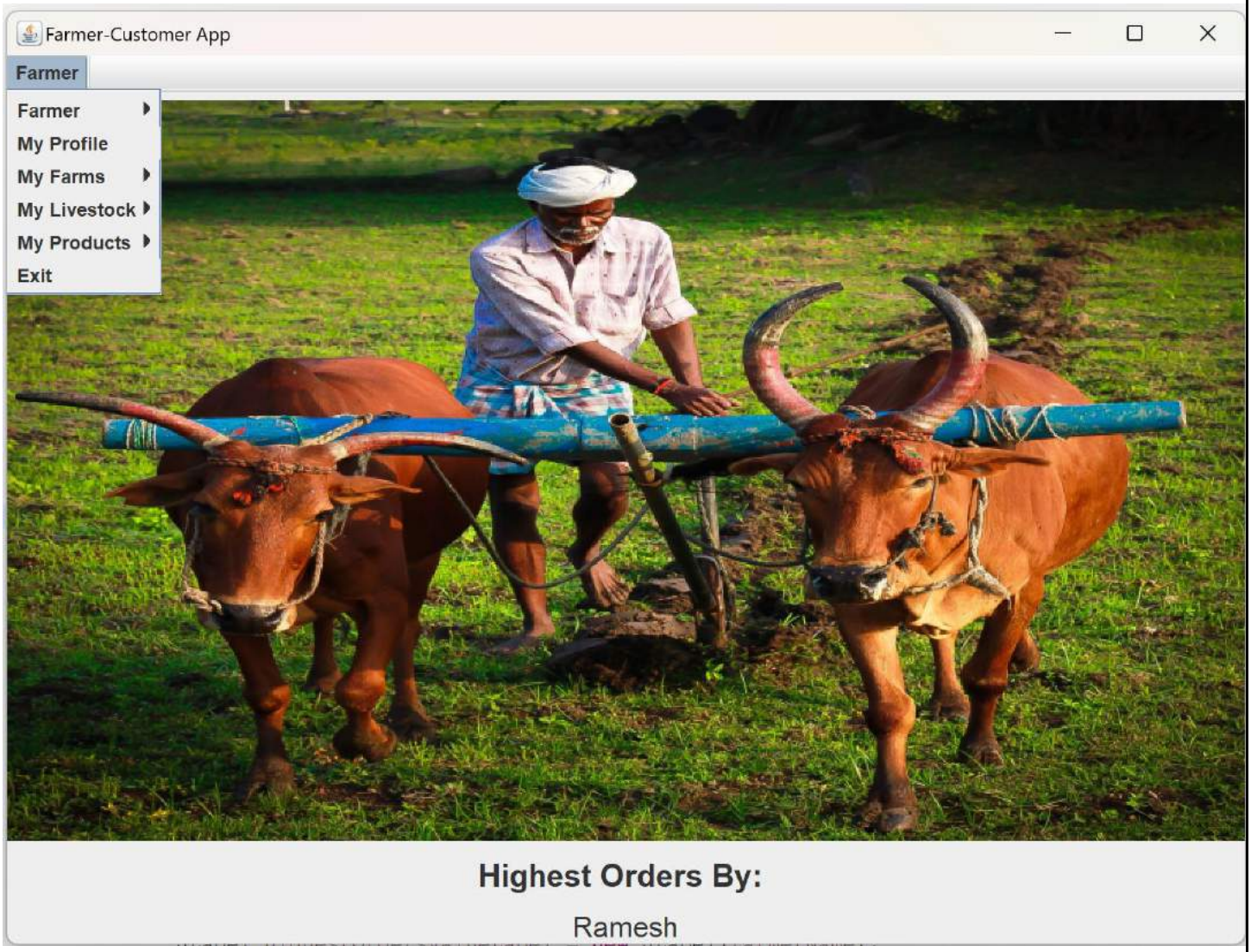
II. ENTER FARMER ID:



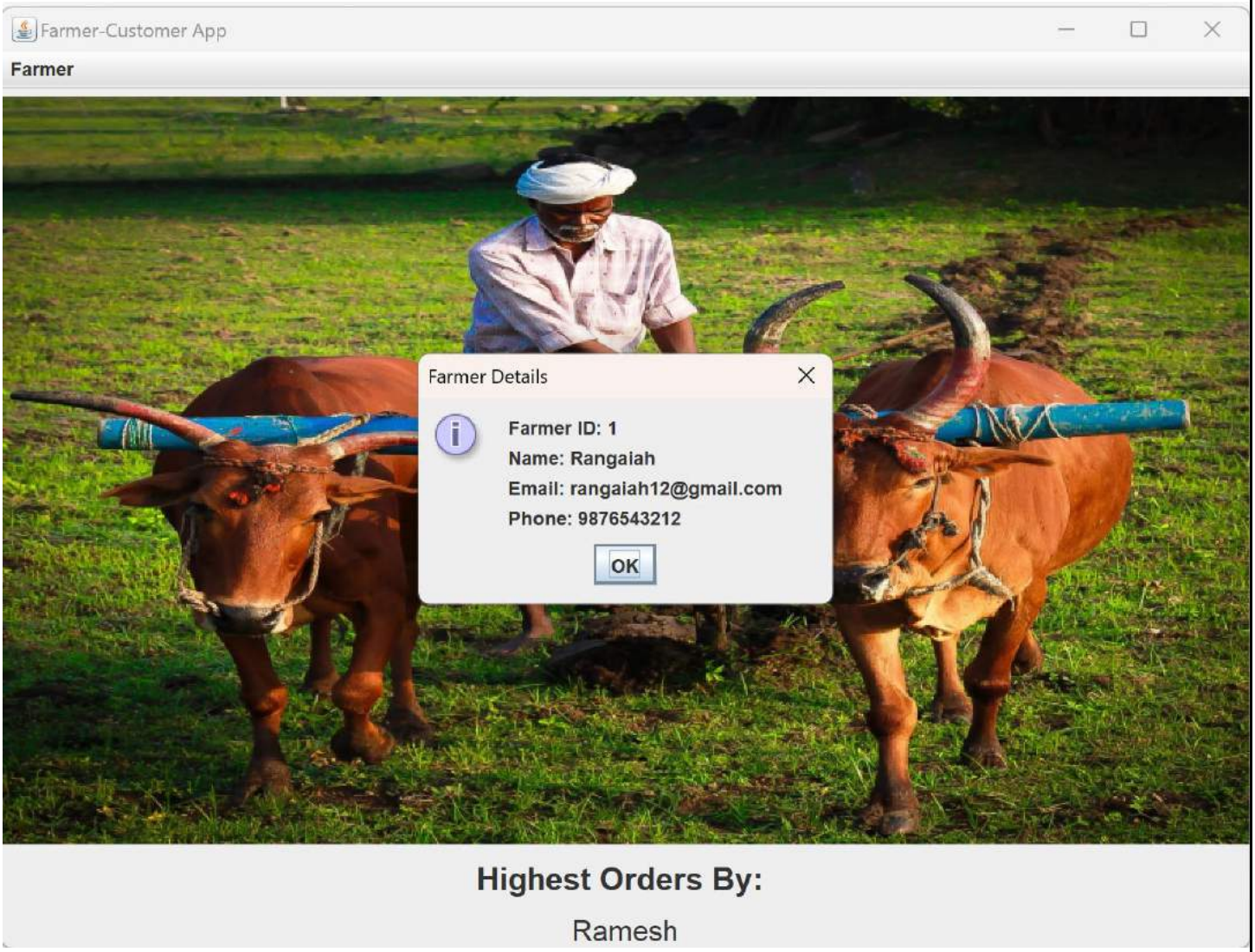
III. FARMER PAGE:



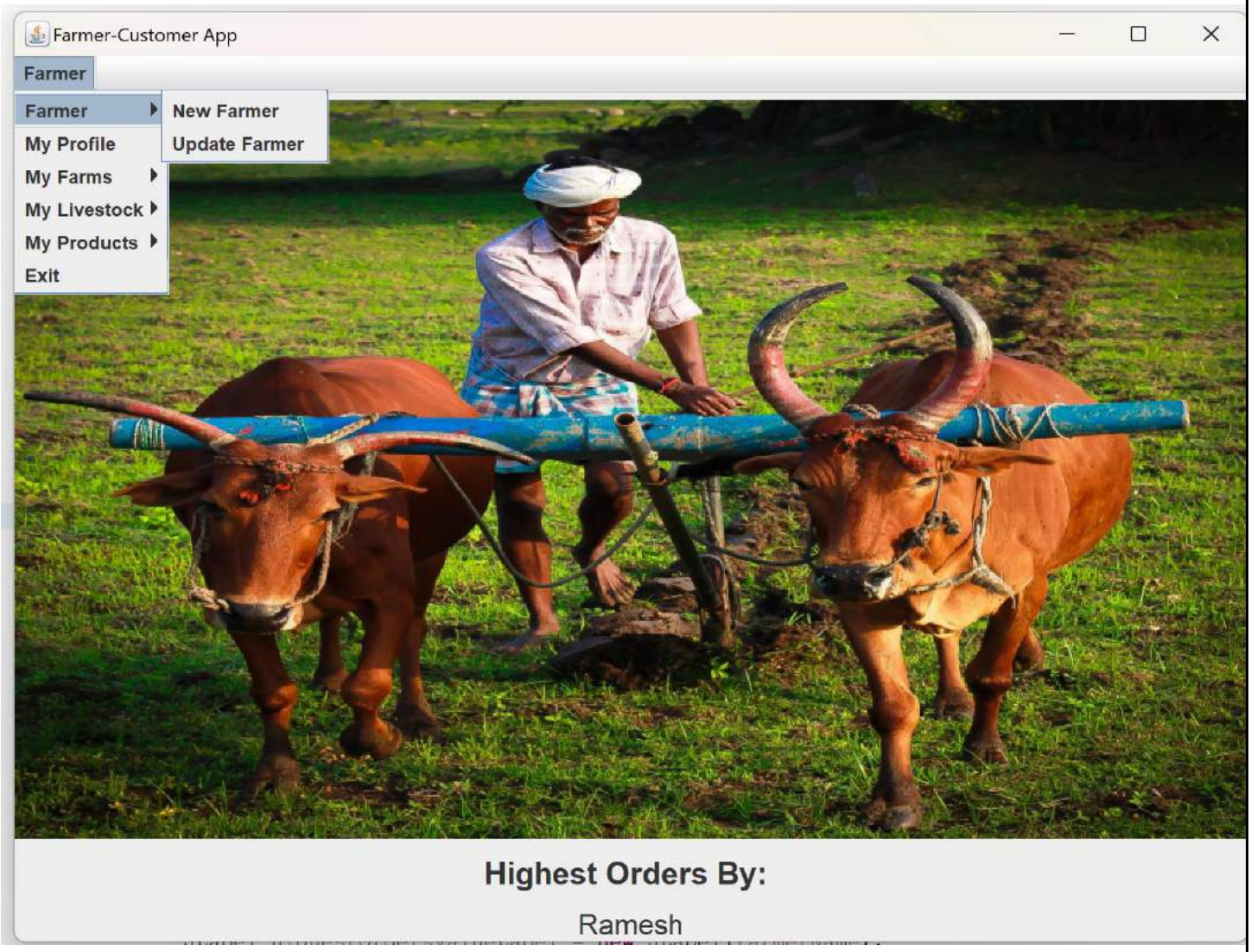
IV. CLICK ON MENU NAMED FARMER:



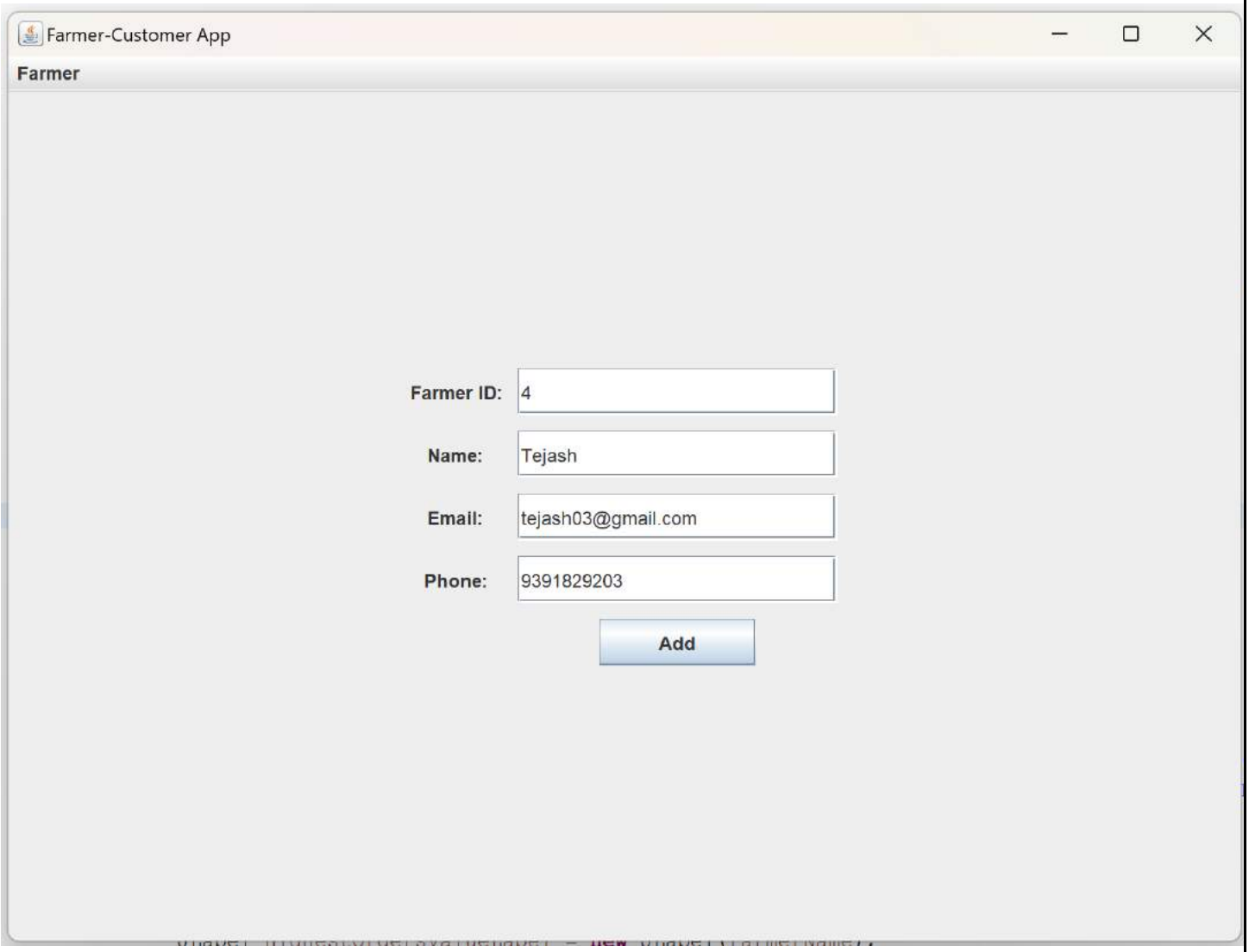
V. CLICK ON MY PROFILE:



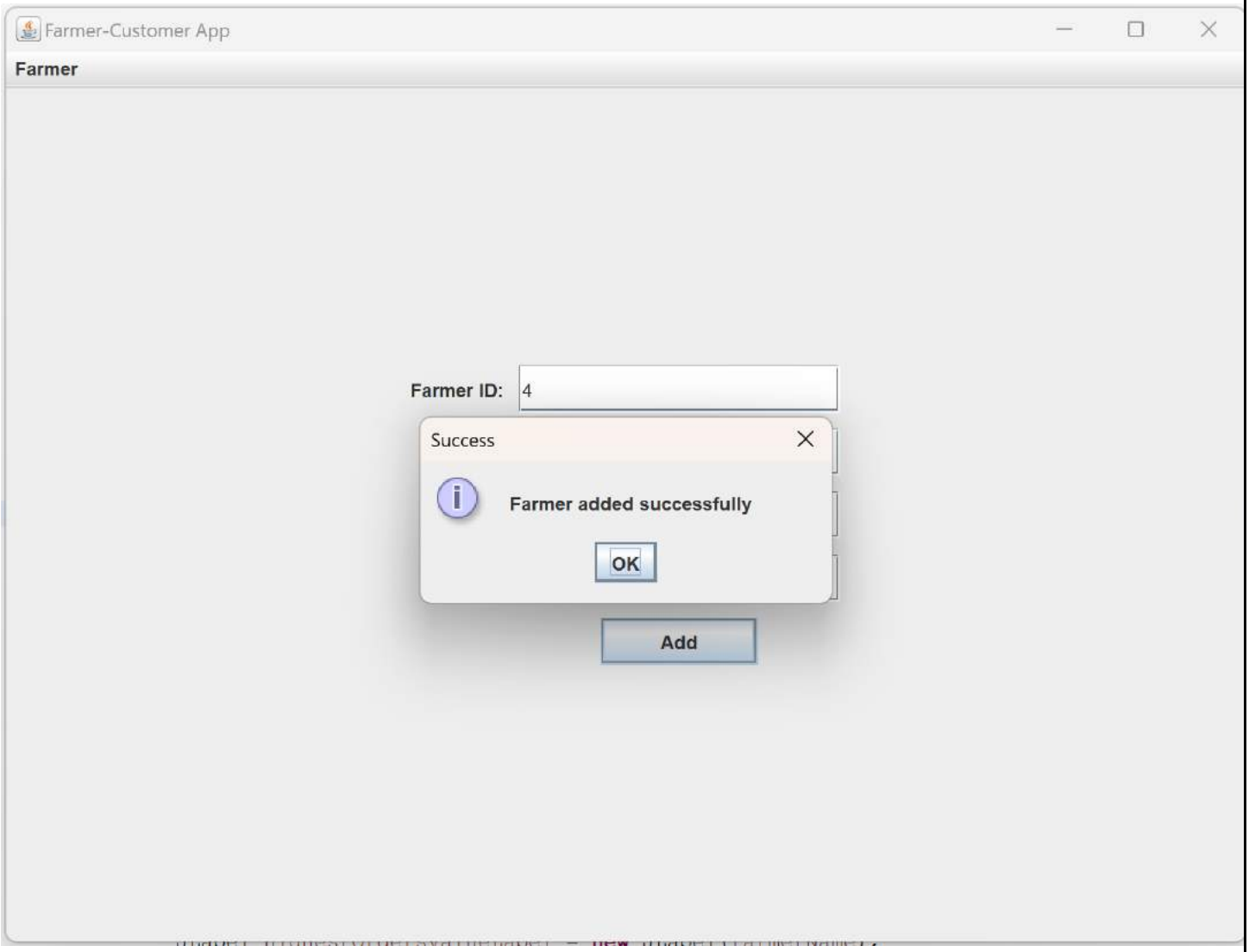
VI. CLICK OK THEN CLICK ON NEW FARMER FROM MENU BAR IF YOU ARE NOT REGISTERED AS A FARMER:



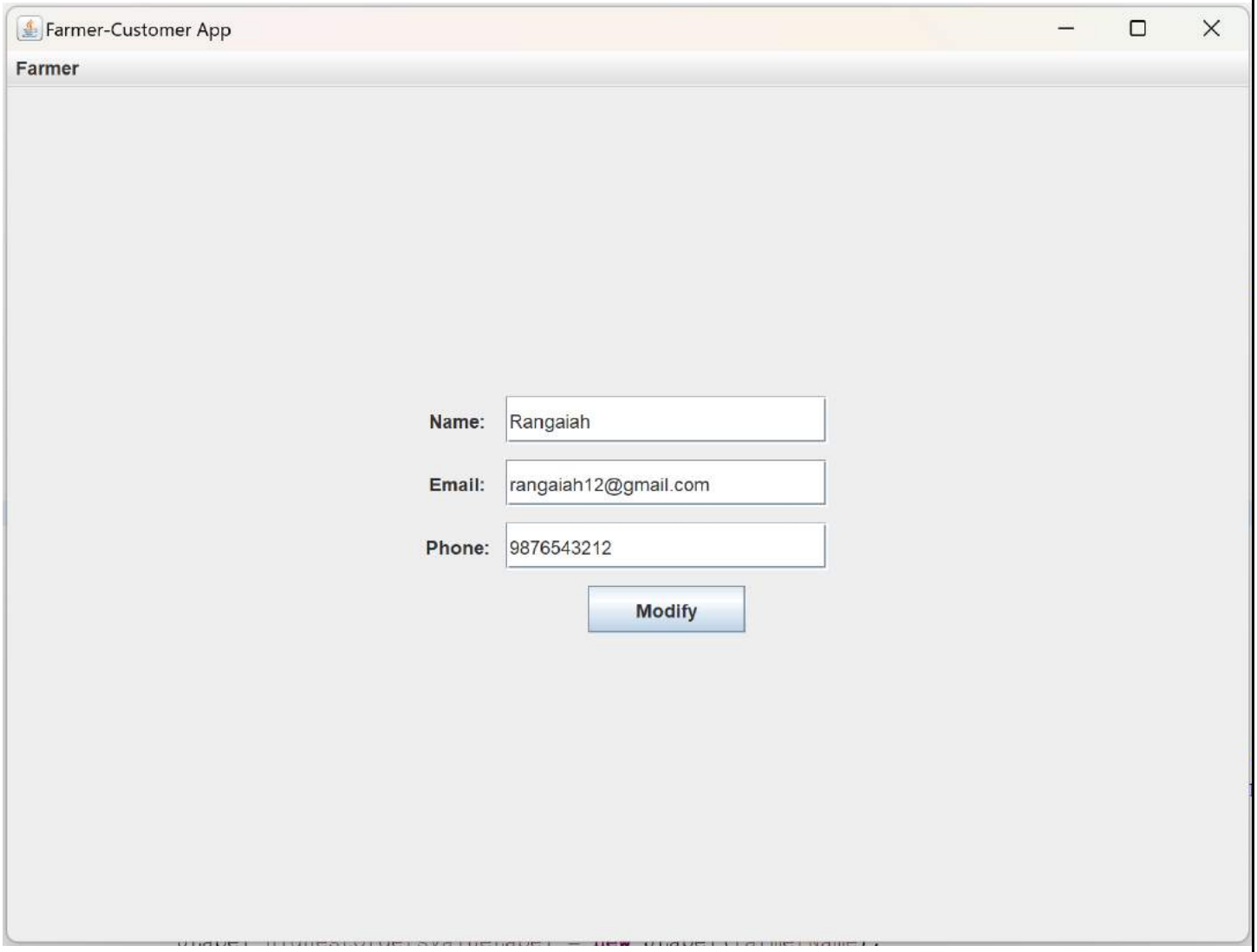
VII. GIVE NECESSARY DETAILS :



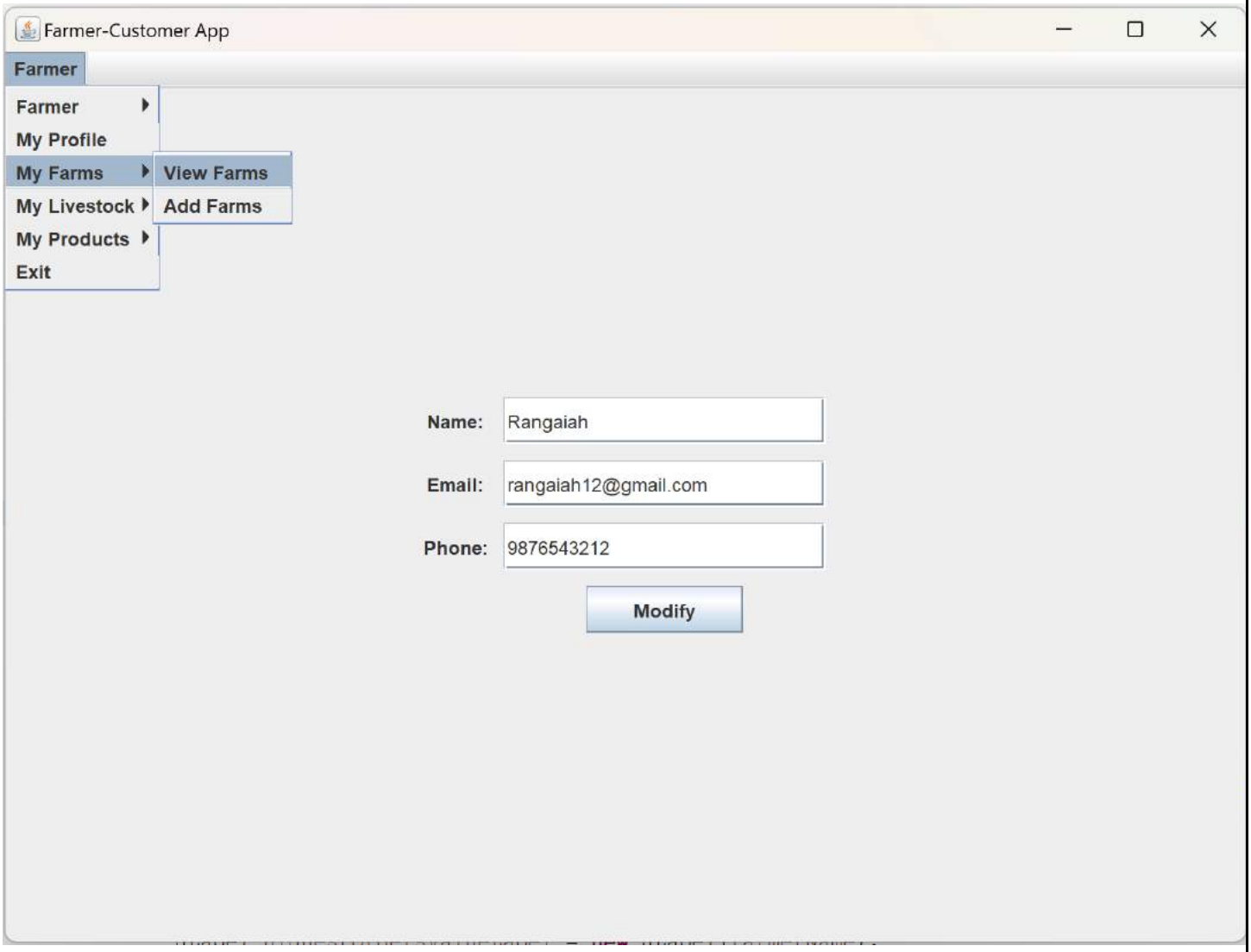
VIII. THEN CLICK ADD:



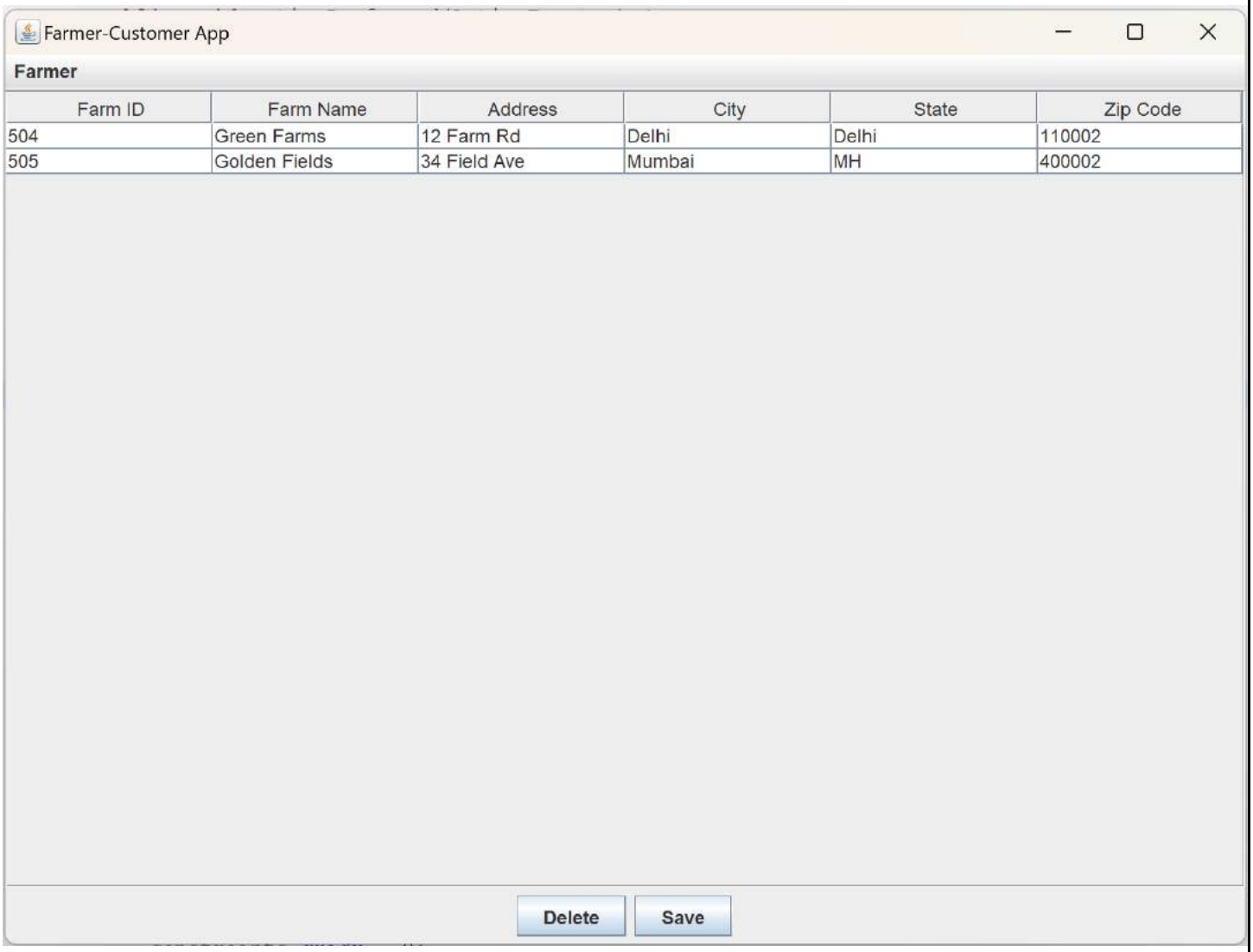
IX. TO UPDATE CLICK UPDATE FARMER BY NAVIGATING TO THE MENU BAR AND AFTER PROVIDING NECESSARY INFORMATION CLICK ON MODIFY:



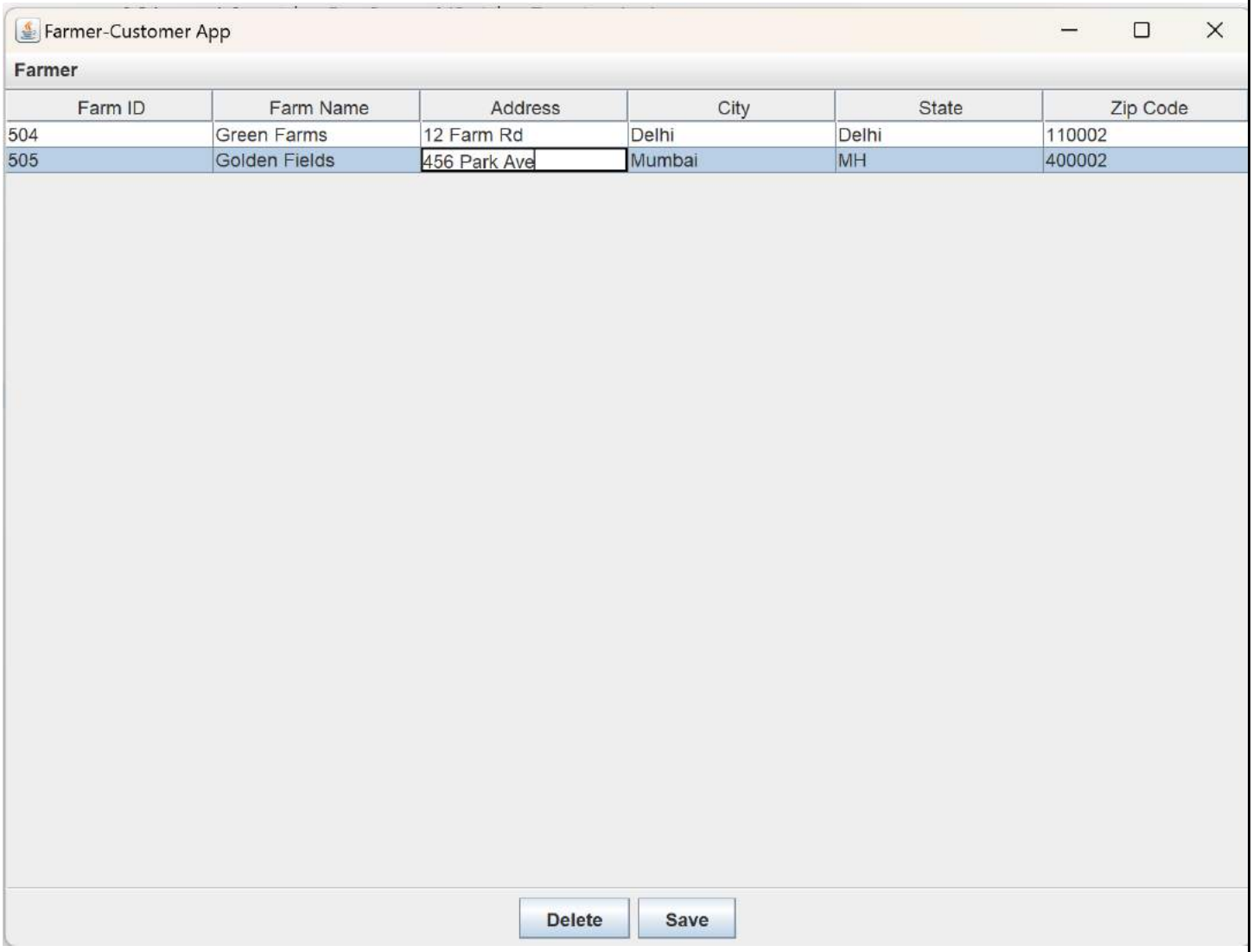
X. CLICK ON VIEW FARMS:

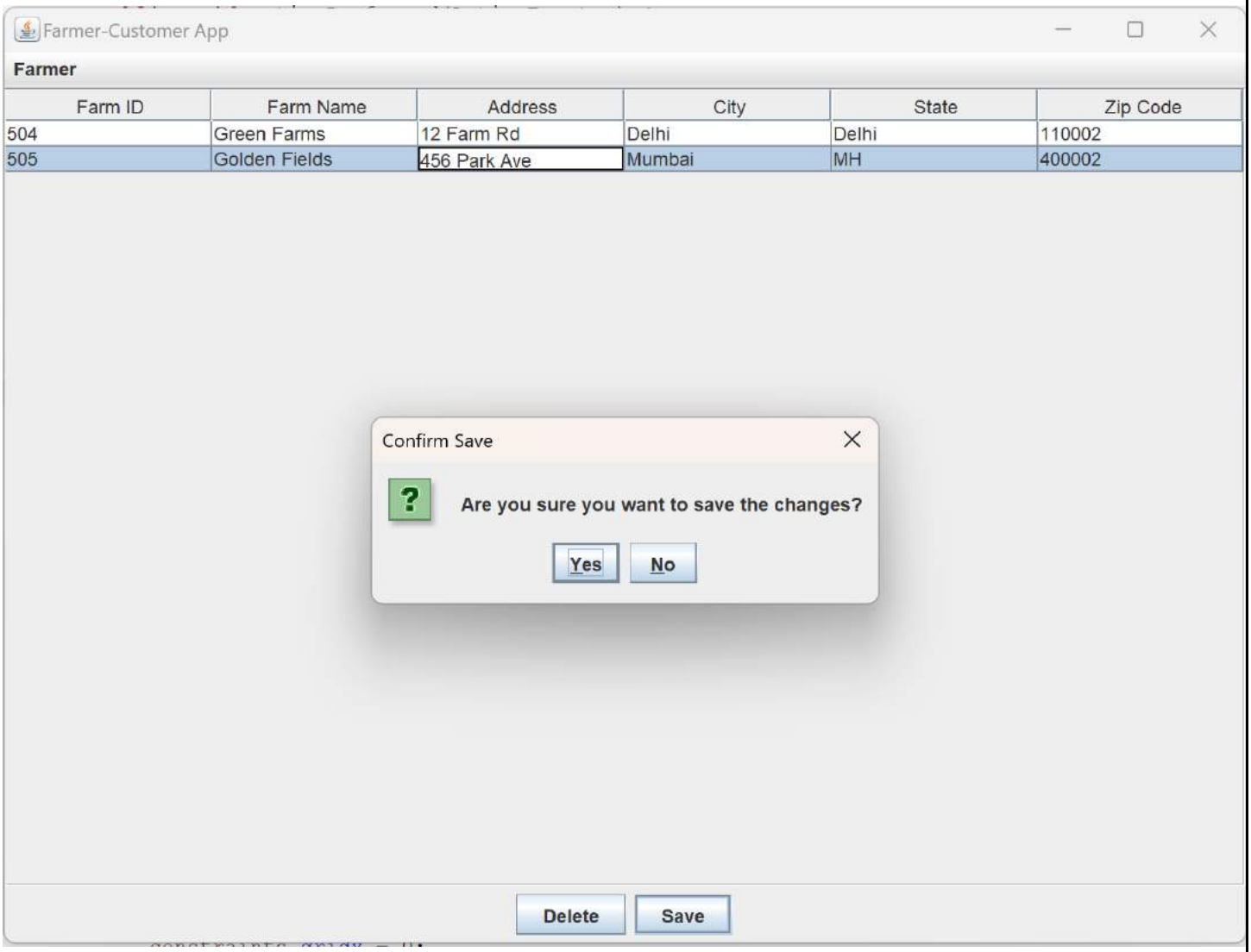


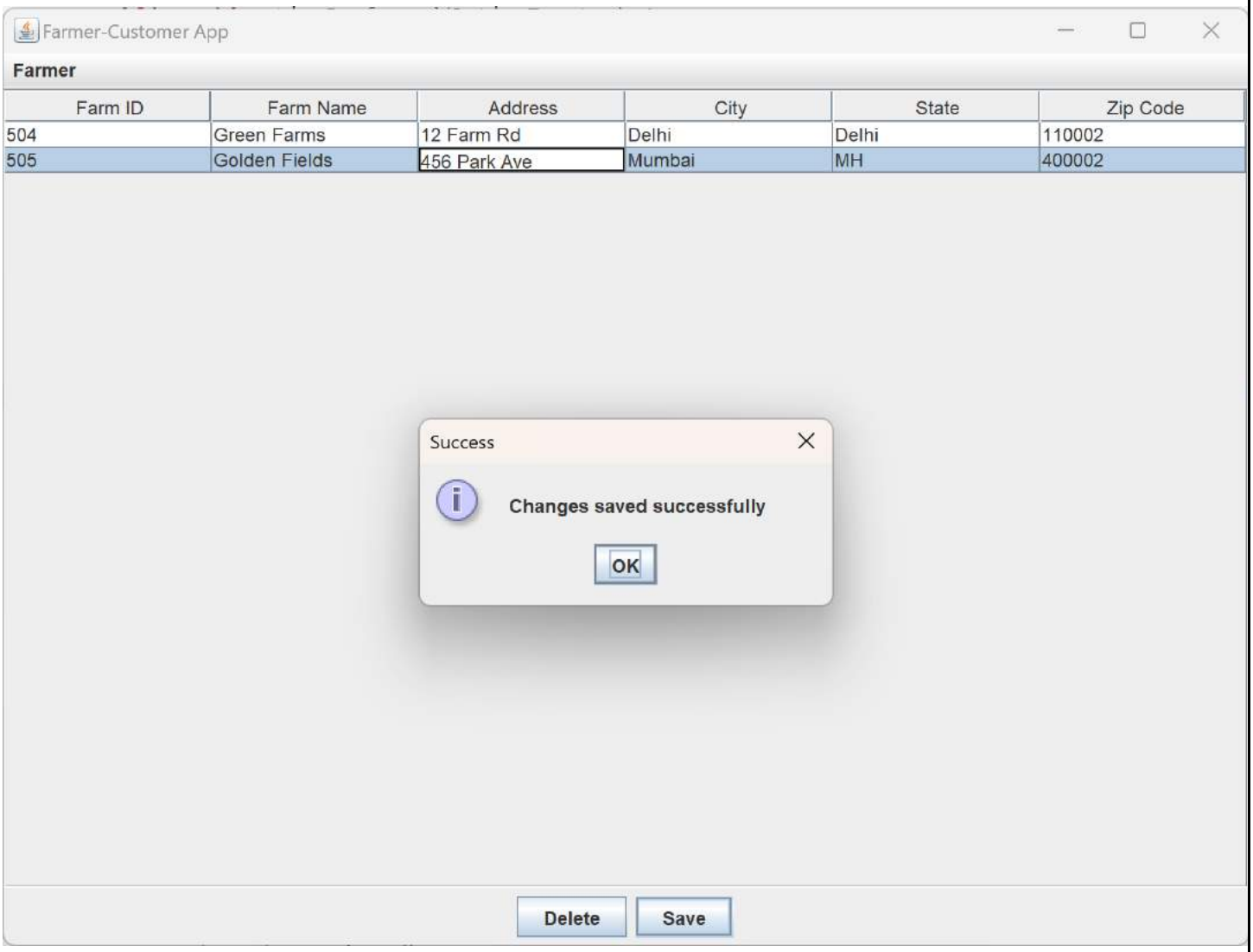
XI. THE FARMS FOR THE CORRESPONDING FARMER ARE:



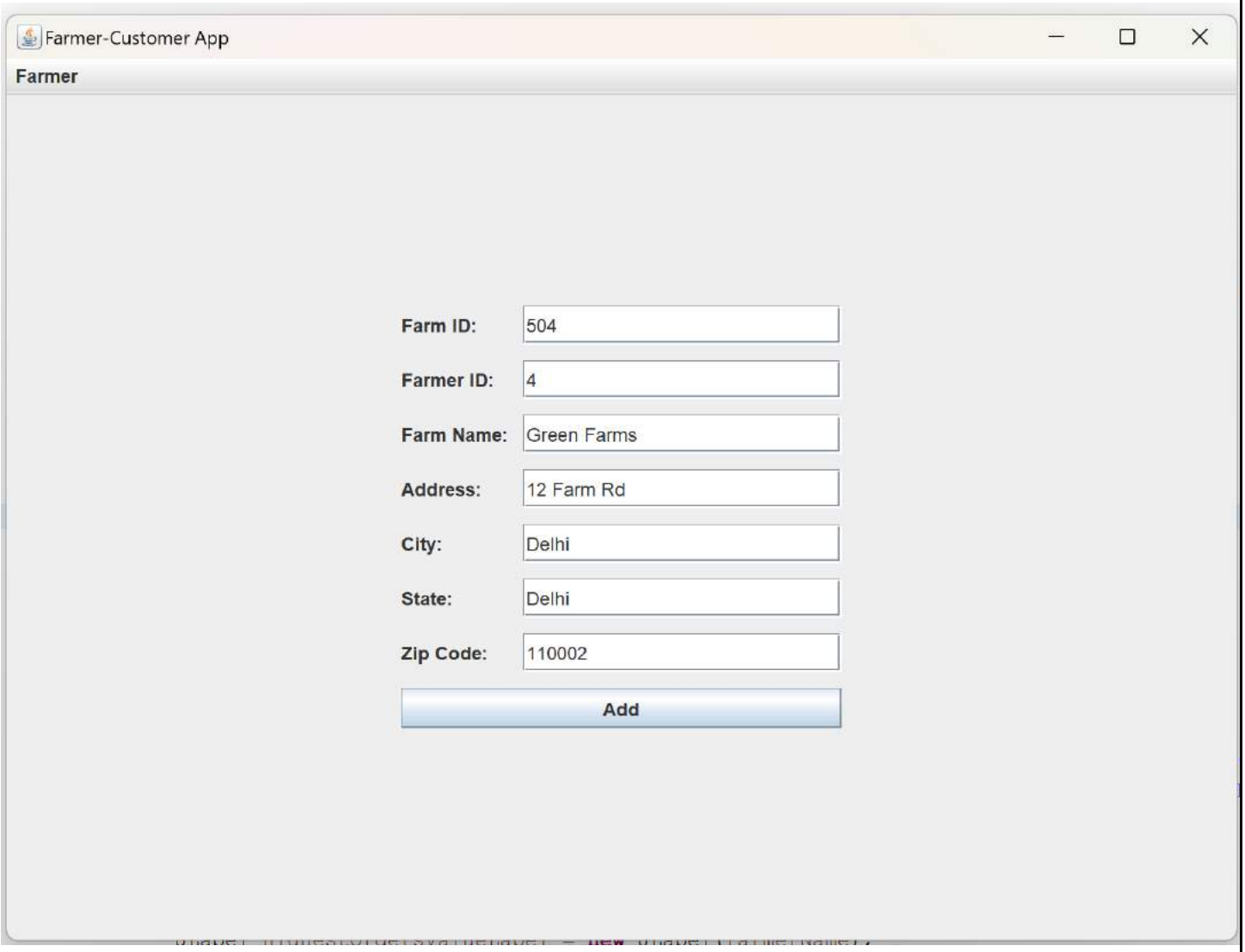
XII. HERE EDIT THE NECESSARY INFO AND CLICK SAVE FOR UPDATION AND SELECT REQUIRED ROWS AND PRESS ON DELETE FOR DELETION.


Farmer-Customer App
Farmer
Farm ID
Farm Name
Address
City
State
Zip Code
504
Green Farms
12 Farm Rd
Delhi
Delhi
110002
505
Golden Fields
456 Park Ave
Mumbai
MH
400002
Delete
Save

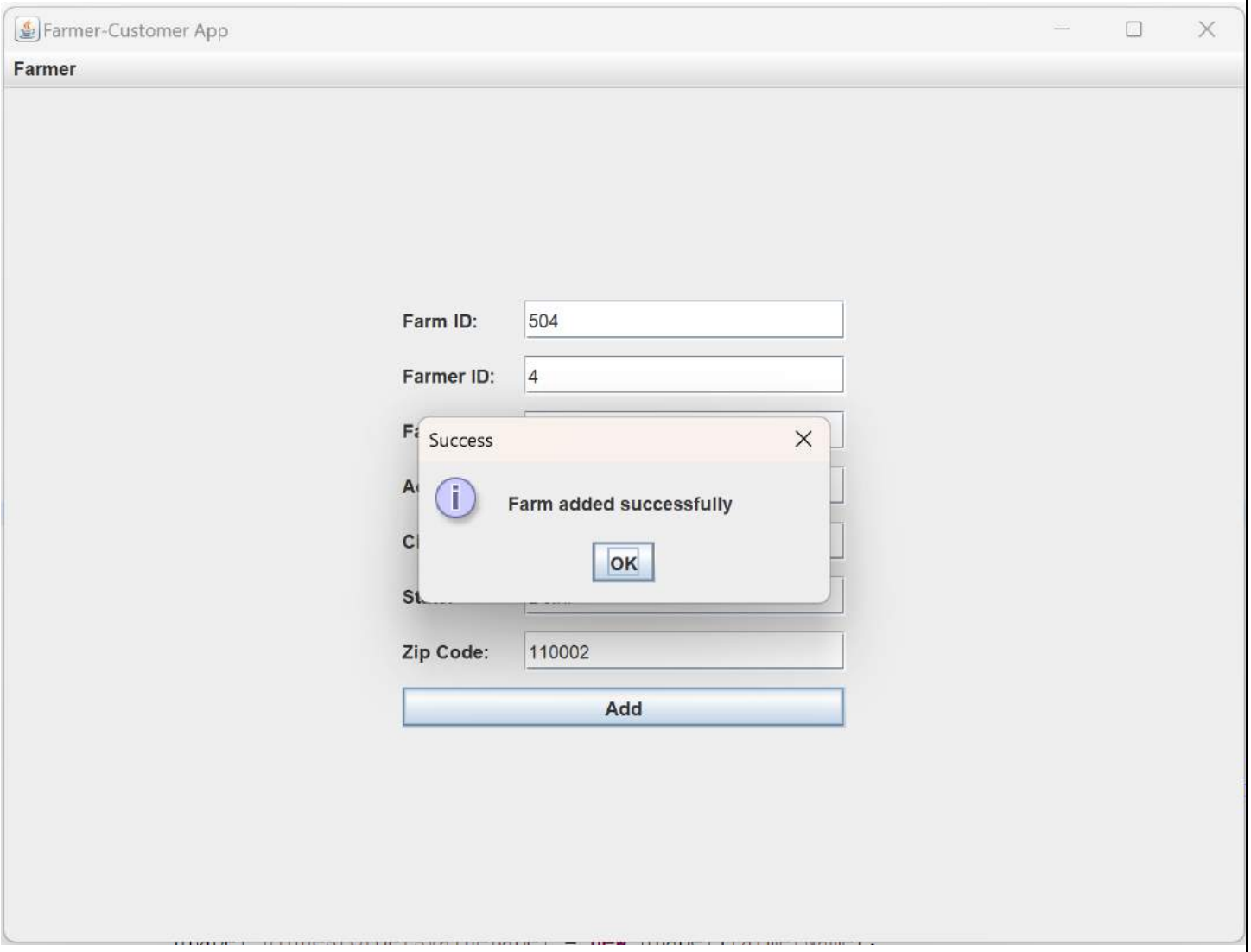

Farmer-Customer App
Farmer
Farm ID
Farm Name
Address
City
State
Zip Code
504
Green Farms
12 Farm Rd
Delhi
Delhi
110002
505
Golden Fields
456 Park Ave
Mumbai
MH
400002
Confirm Save
Are you sure you want to save the changes?
Yes
No
Delete
Save



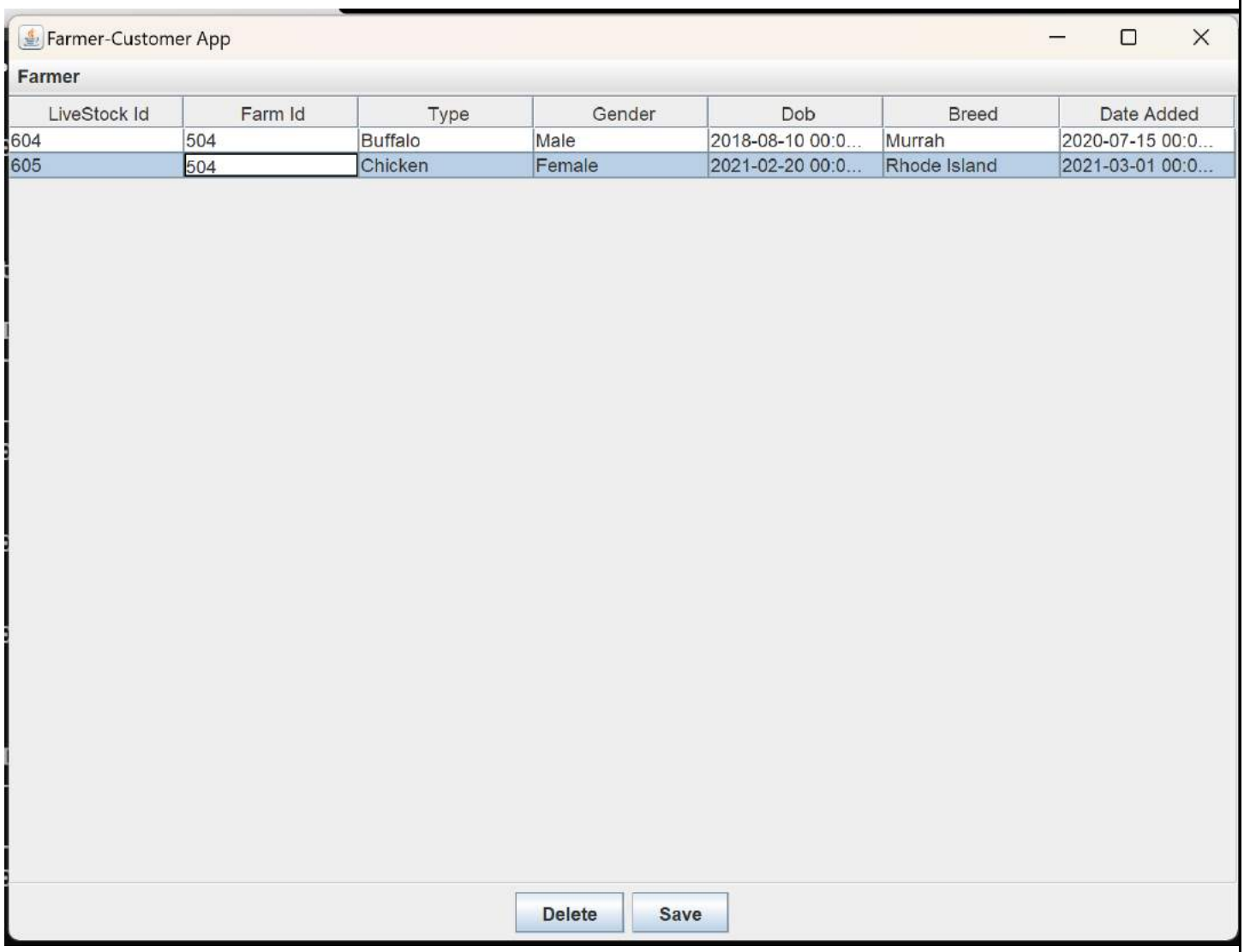
XIII. TO ADD FARM:



XIV. PROVIDE NECESSARY INFORMATION AND CLICK ON ADD:

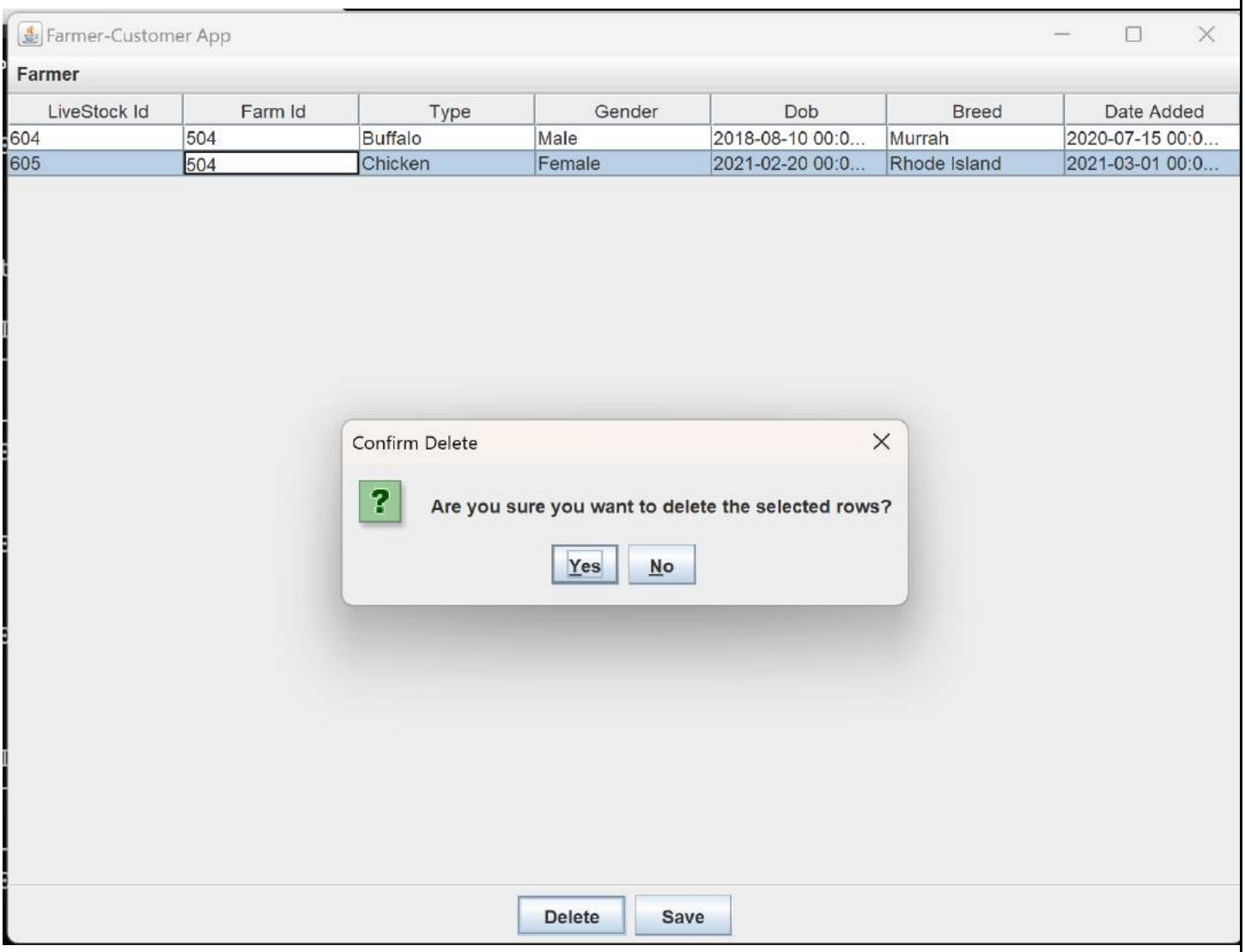


XV. VIEW LIVESTOCK:

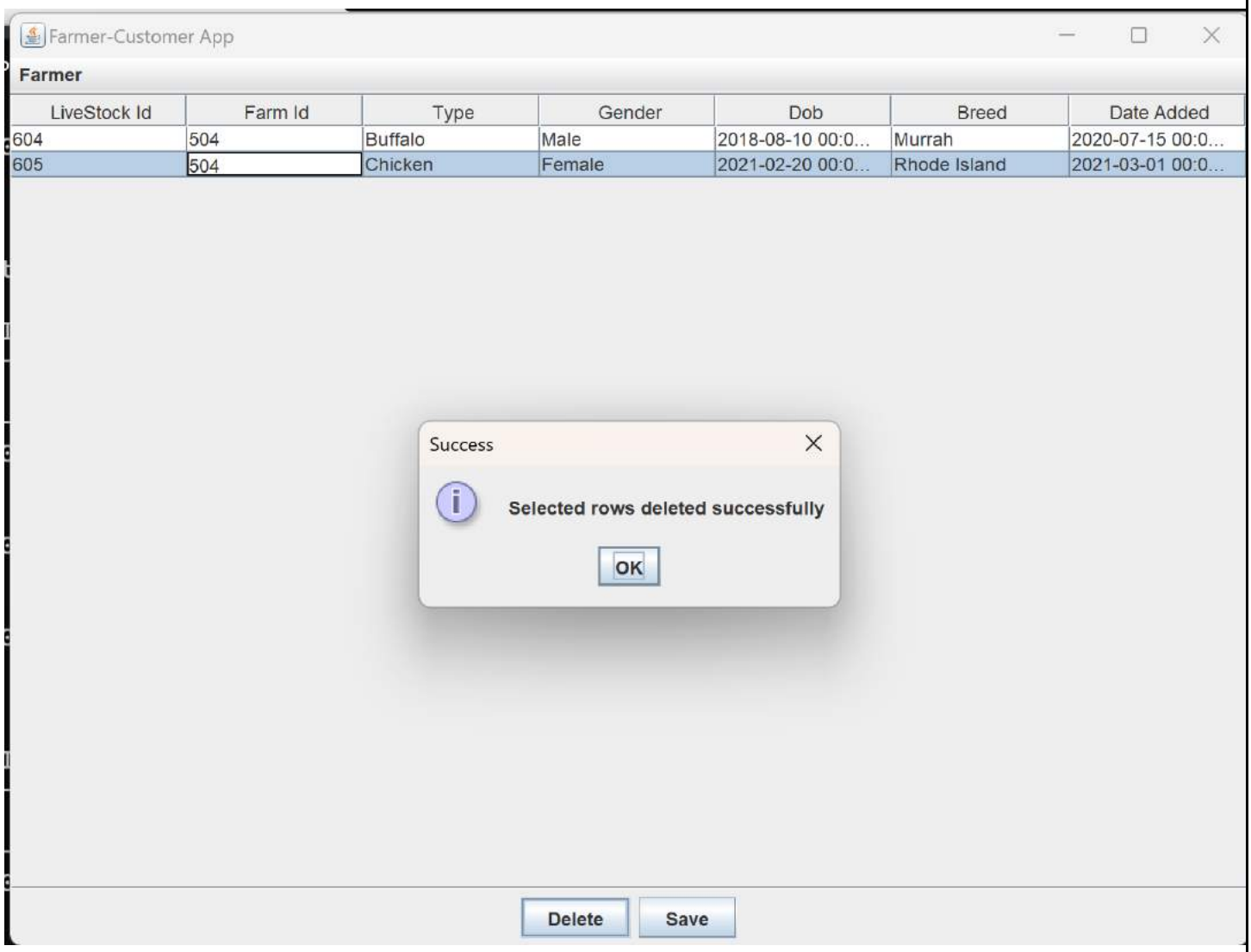


EDIT FOR MODIFICATION AND DELETION.

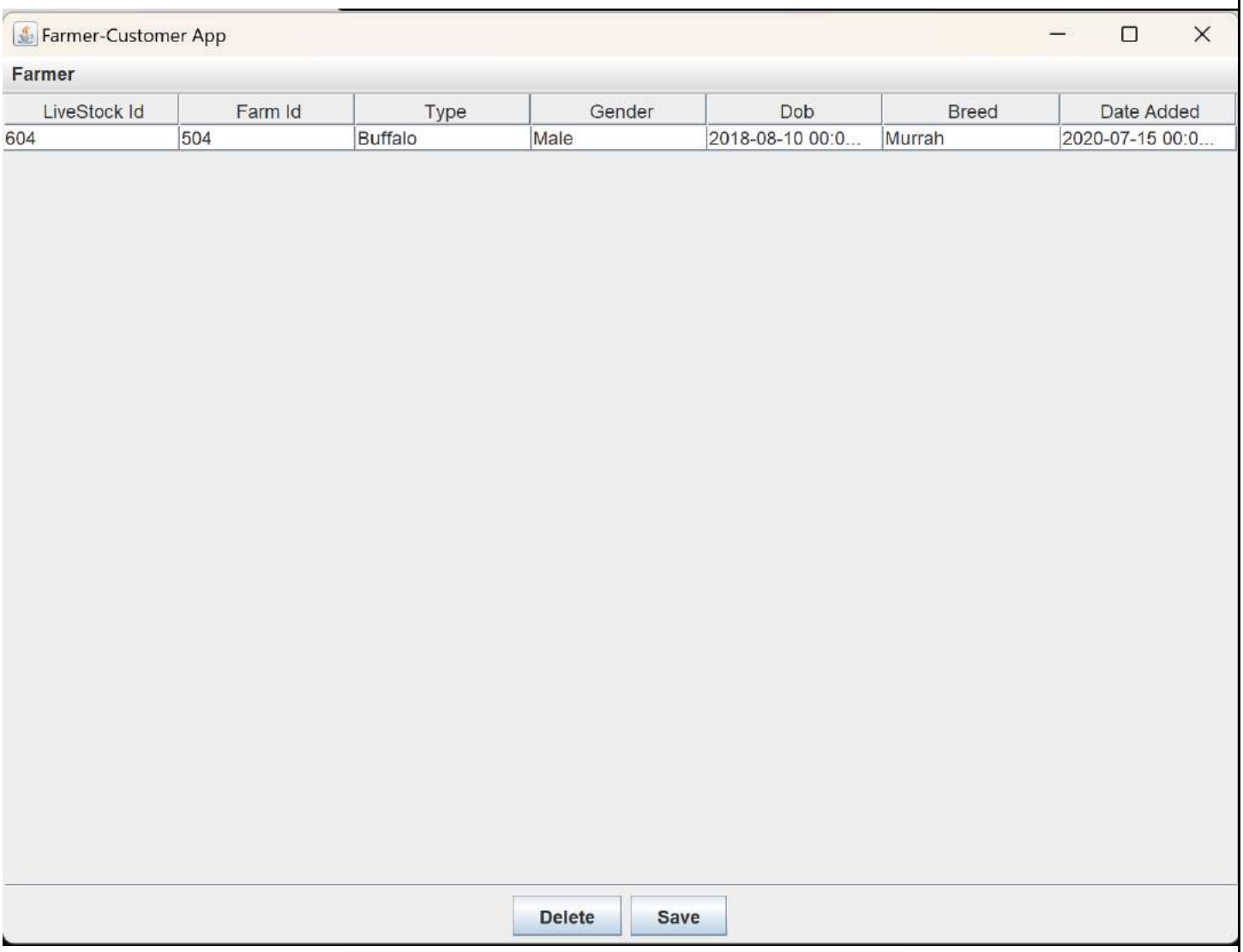
XVI. FOR DELETION SELECT ROWS AND CLICK ON DELETE:



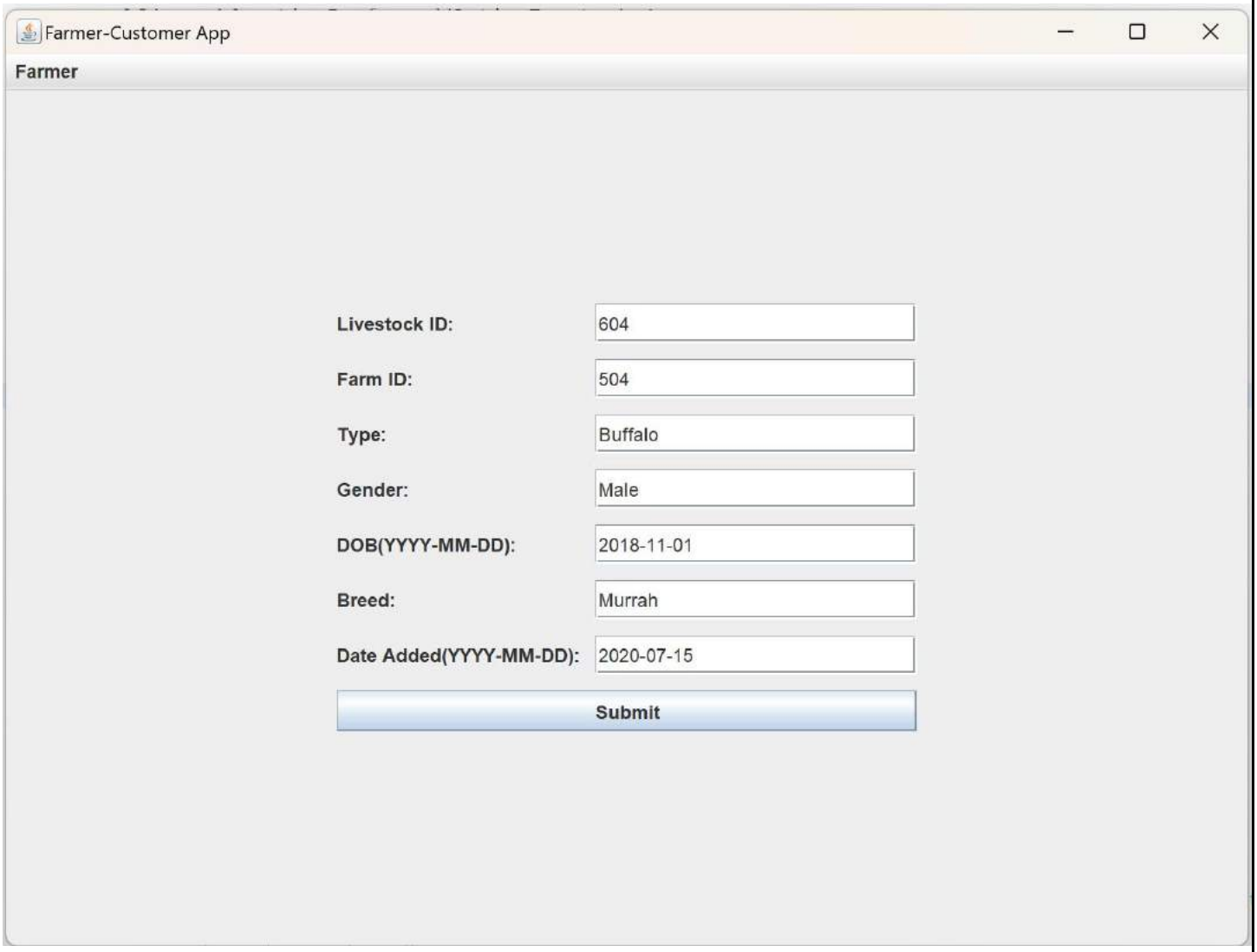
XVII. CLICK YES:

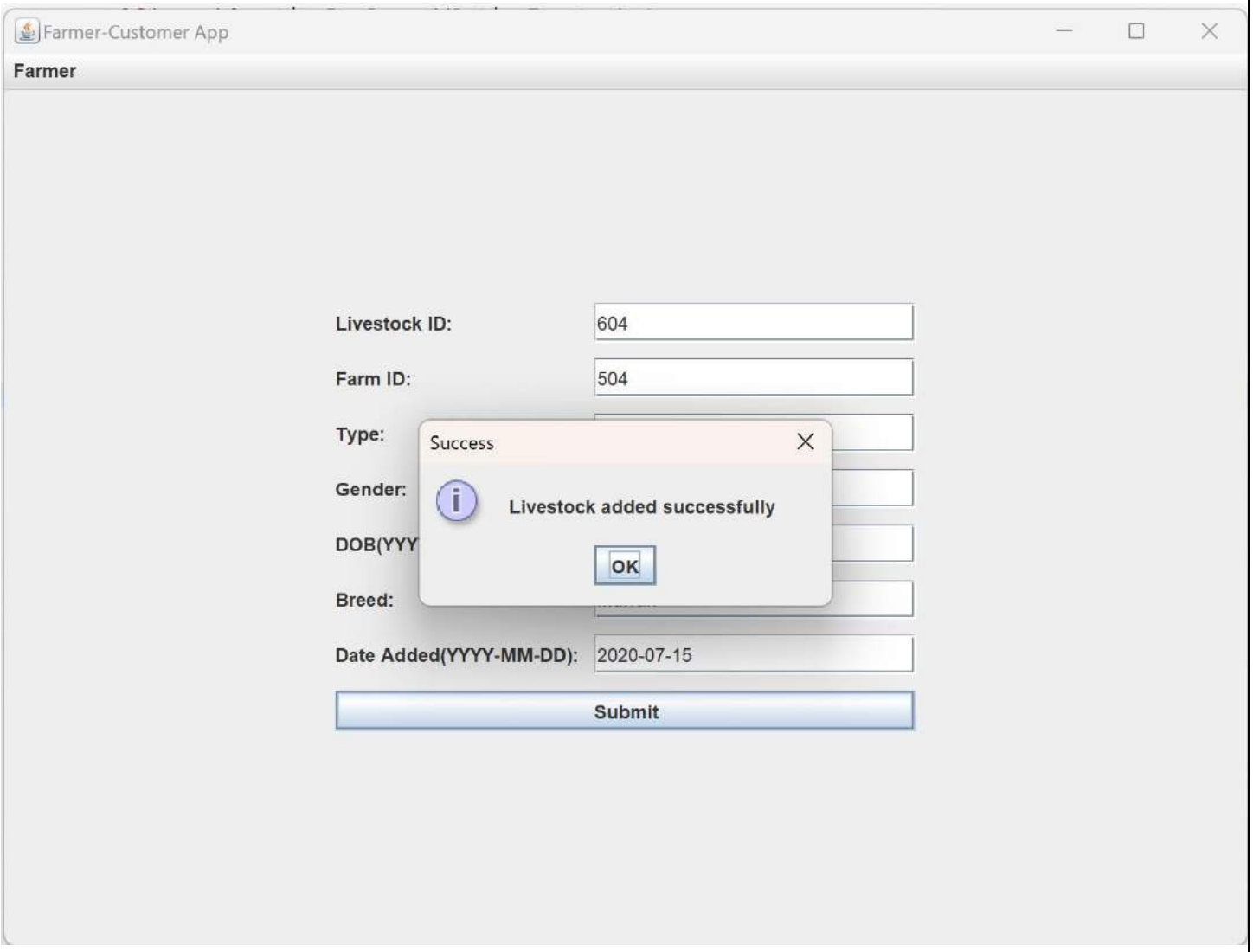


XVIII. THE PAGE AUTOMATICALLY RELOADS AFTER DELETION:

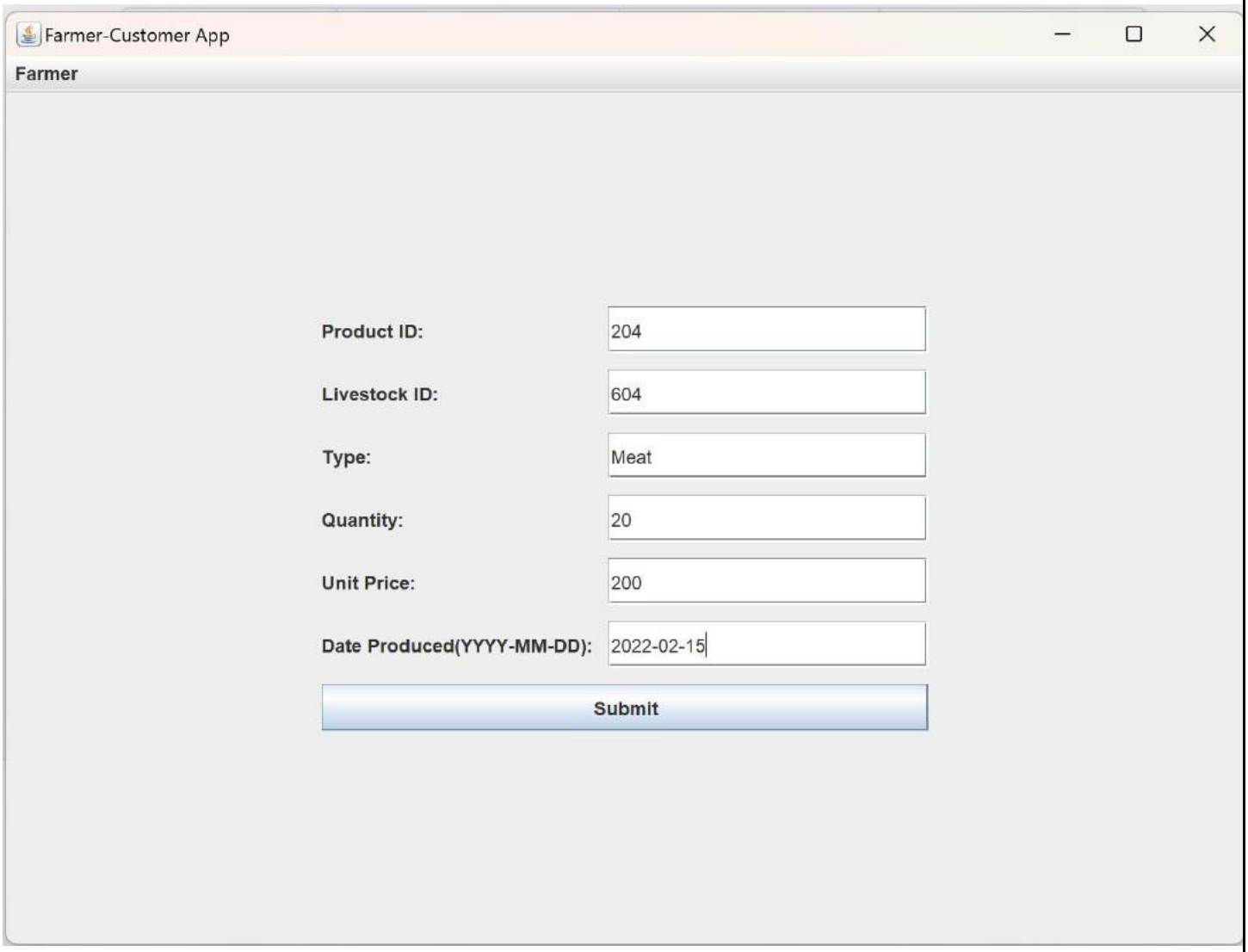


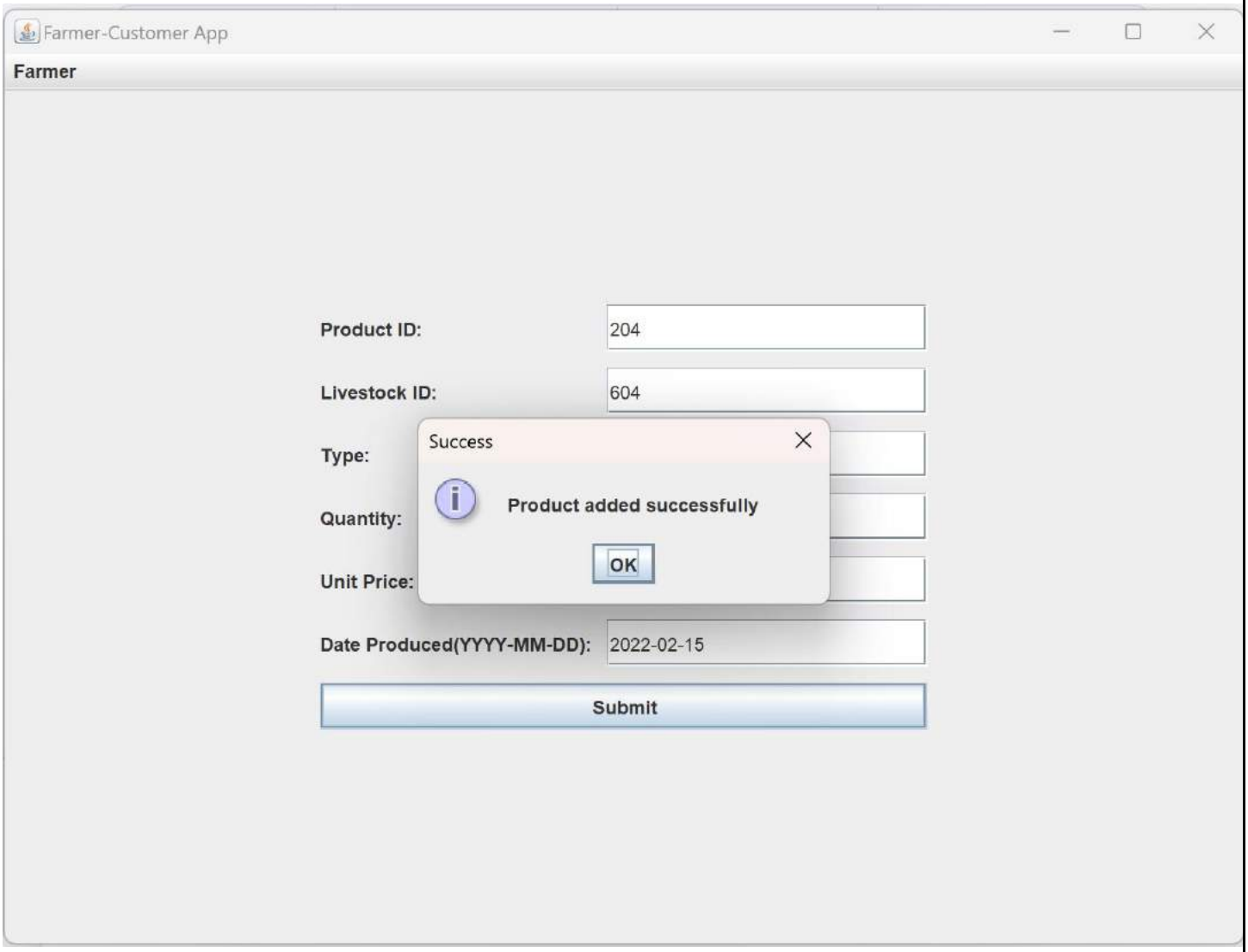
XIX. FOR INSERTING LIVESTOCK:


Farmer-Customer App
Farmer
Livestock ID:
604
Farm ID:
504
Type:
Buffalo
Gender:
Male
DOB(YYYY-MM-DD):
2018-11-01
Breed:
Murrah
Date Added(YYYY-MM-DD):
2020-07-15
Submit

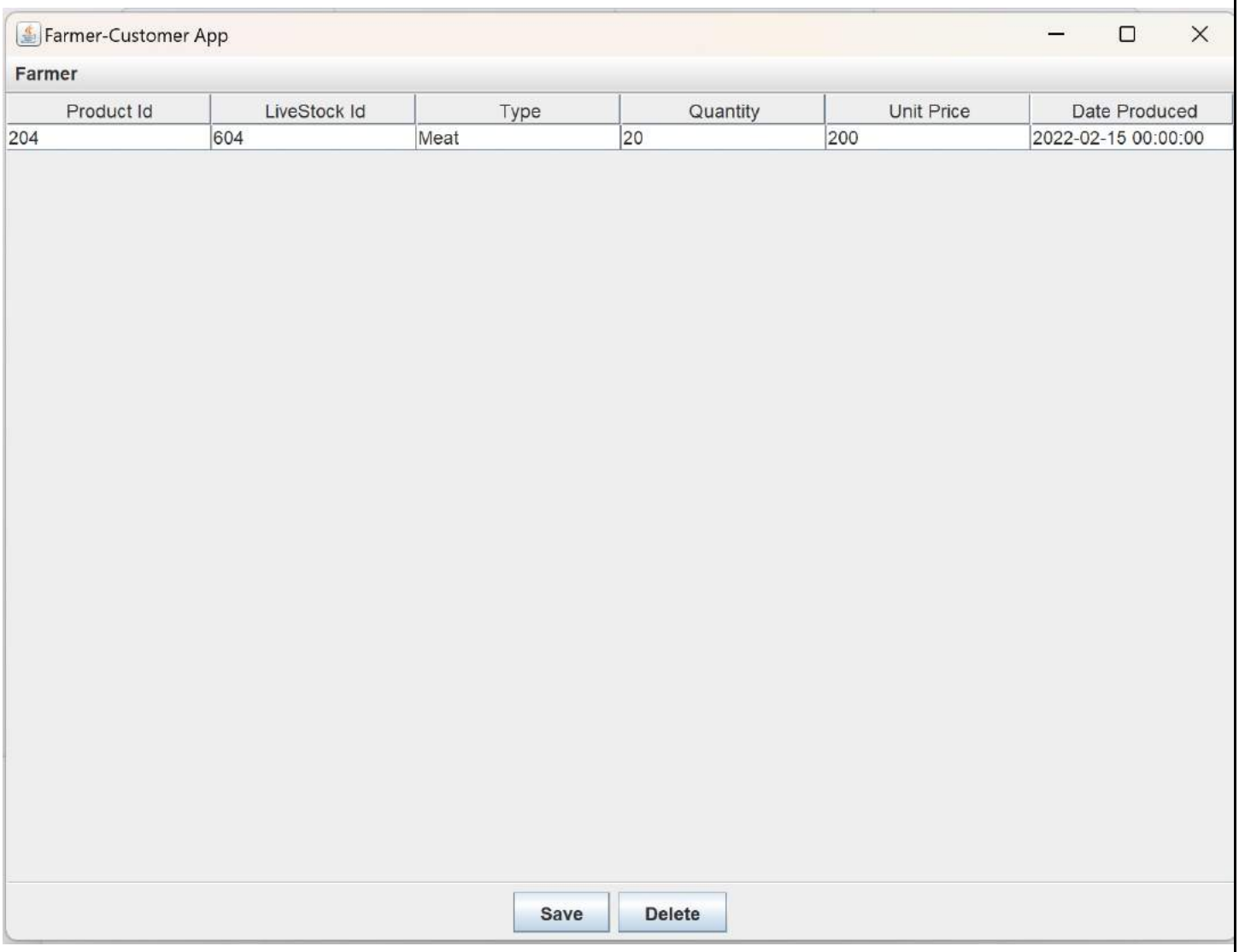


XX. FOR PRODUCTS ADDING:

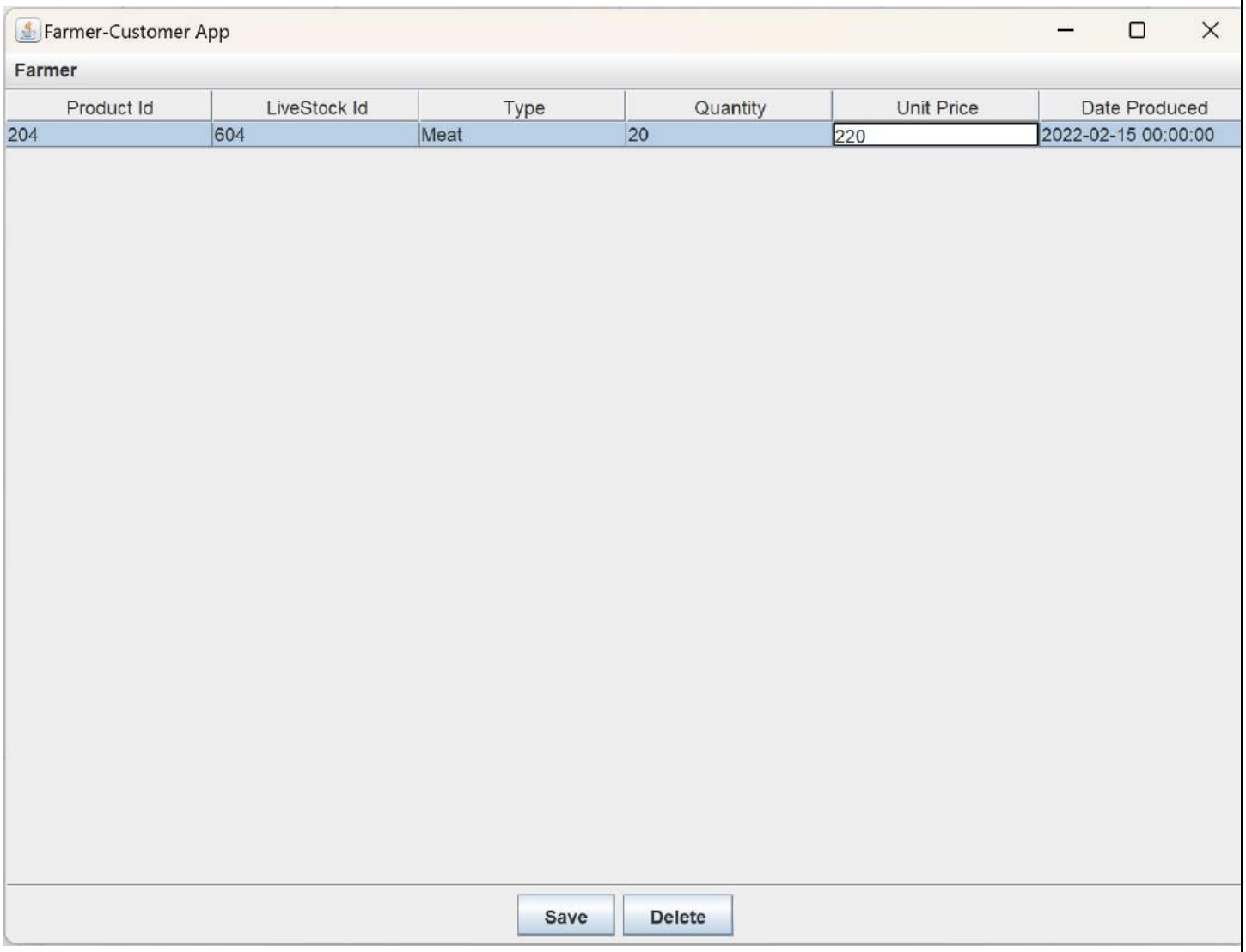

Farmer-Customer App
Farmer
Product ID:
204
Livestock ID:
604
Type:
Meat
Quantity:
20
Unit Price:
200
Date Produced(YYYY-MM-DD):
2022-02-15
Submit

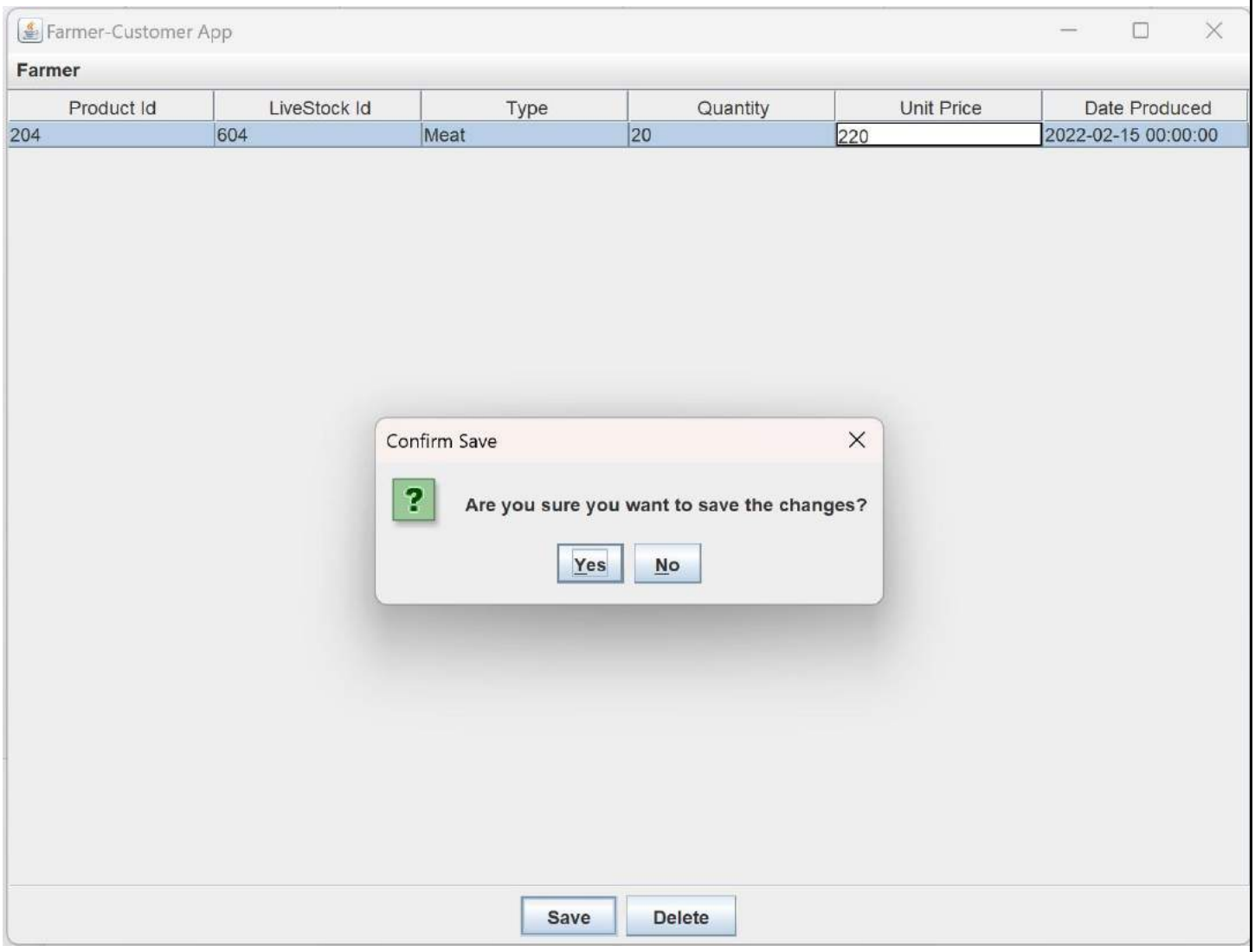


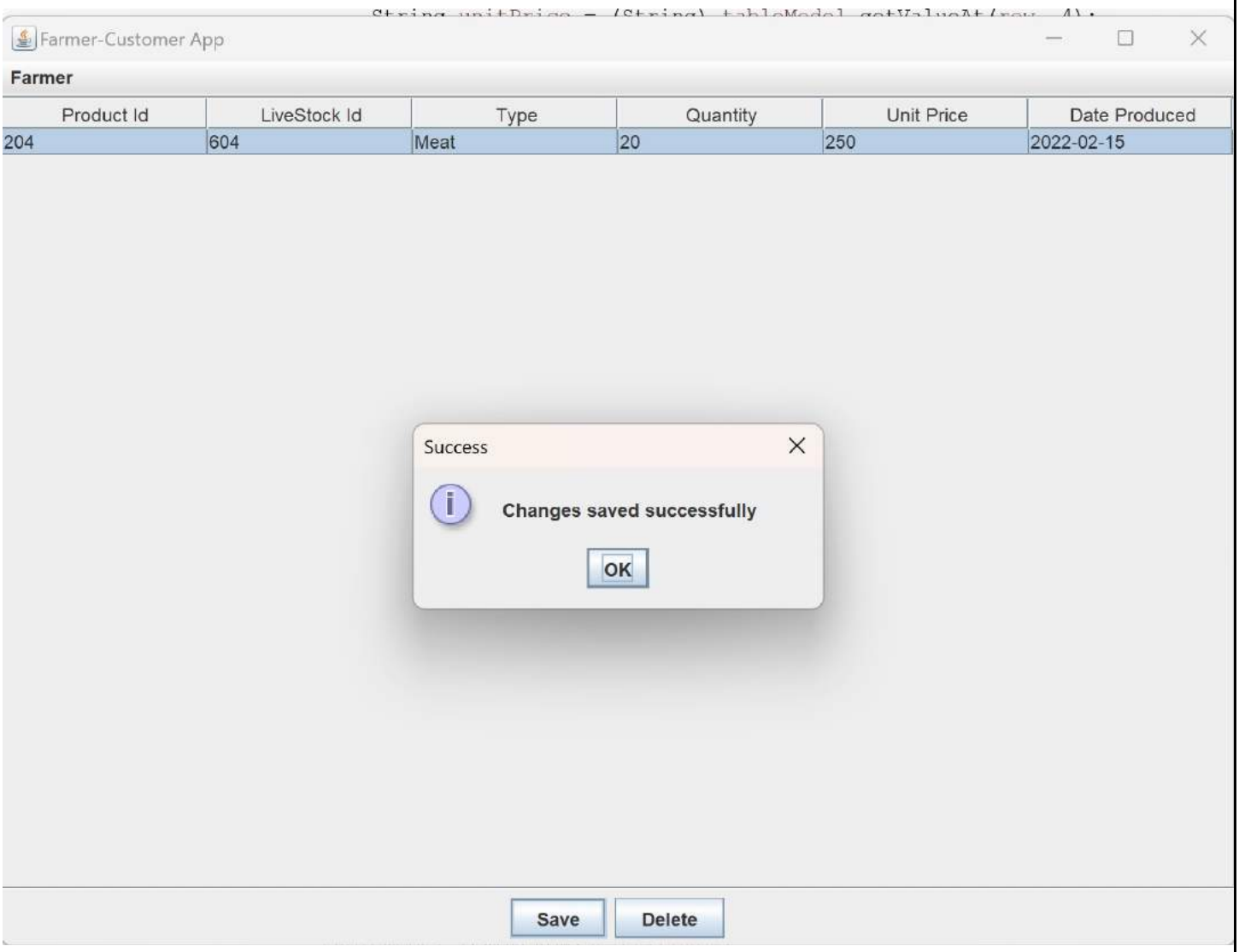
XXI. TO VIEW PRODUCTS:



XXII. TO EDIT OR UPDATE:

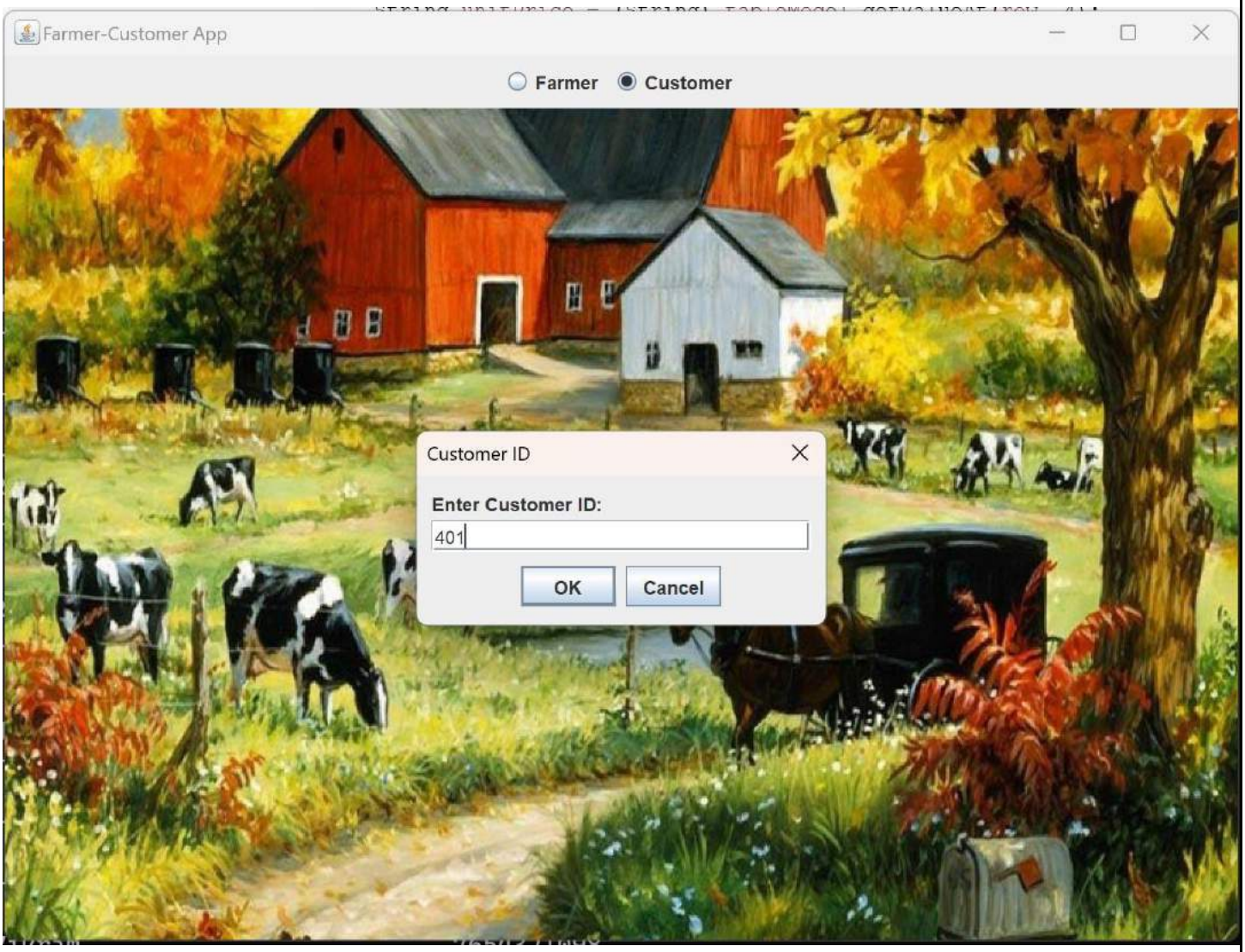

Farmer-Customer App
Farmer
Product Id
LiveStock Id
Type
Quantity
Unit Price
Date Produced
204
604
Meat
20
220
2022-02-15 00:00:00
Save
Delete


Farmer-Customer App
Farmer
Product Id
LiveStock Id
Type
Quantity
Unit Price
Date Produced
204
604
Meat
20
220
2022-02-15 00:00:00
Confirm Save
Are you sure you want to save the changes?
Yes
No
Save
Delete

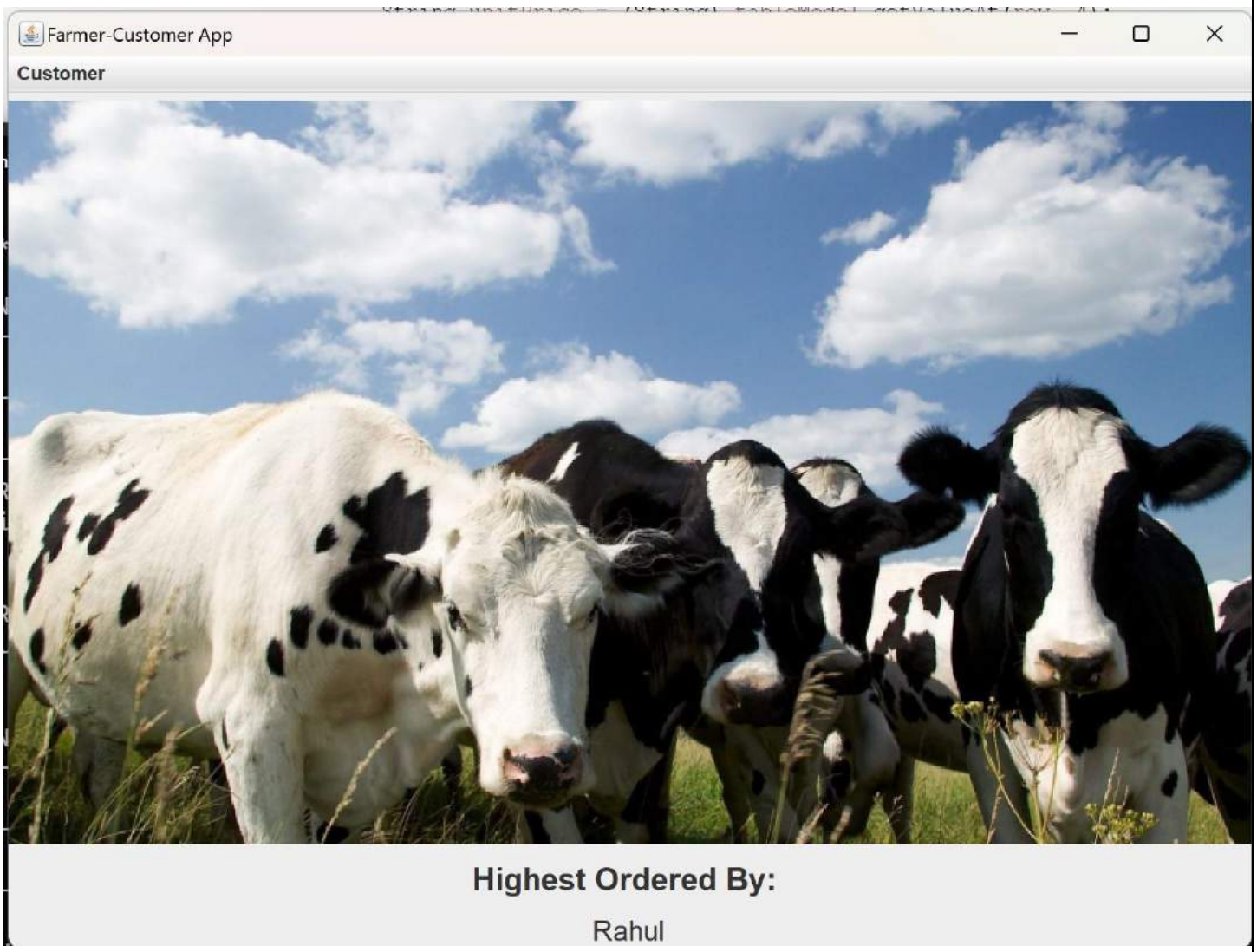

Farmer-Customer App
Farmer
Product Id
LiveStock Id
Type
Quantity
Unit Price
Date Produced
204
604
Meat
20
250
2022-02-15
Success
Changes saved successfully
OK
Save
Delete

CUSTOMER PAGE:

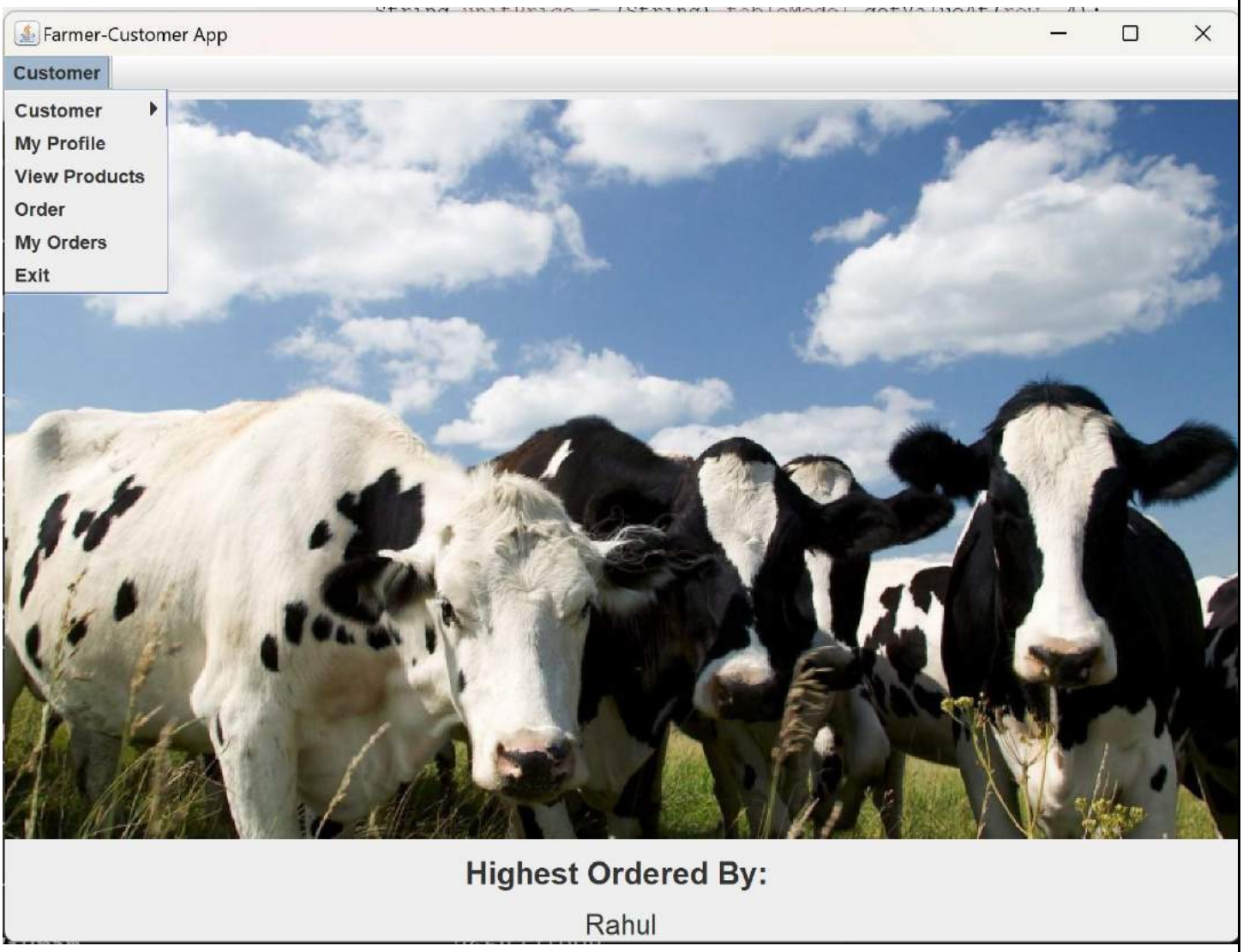
I.GIVE CUSTOMER ID:



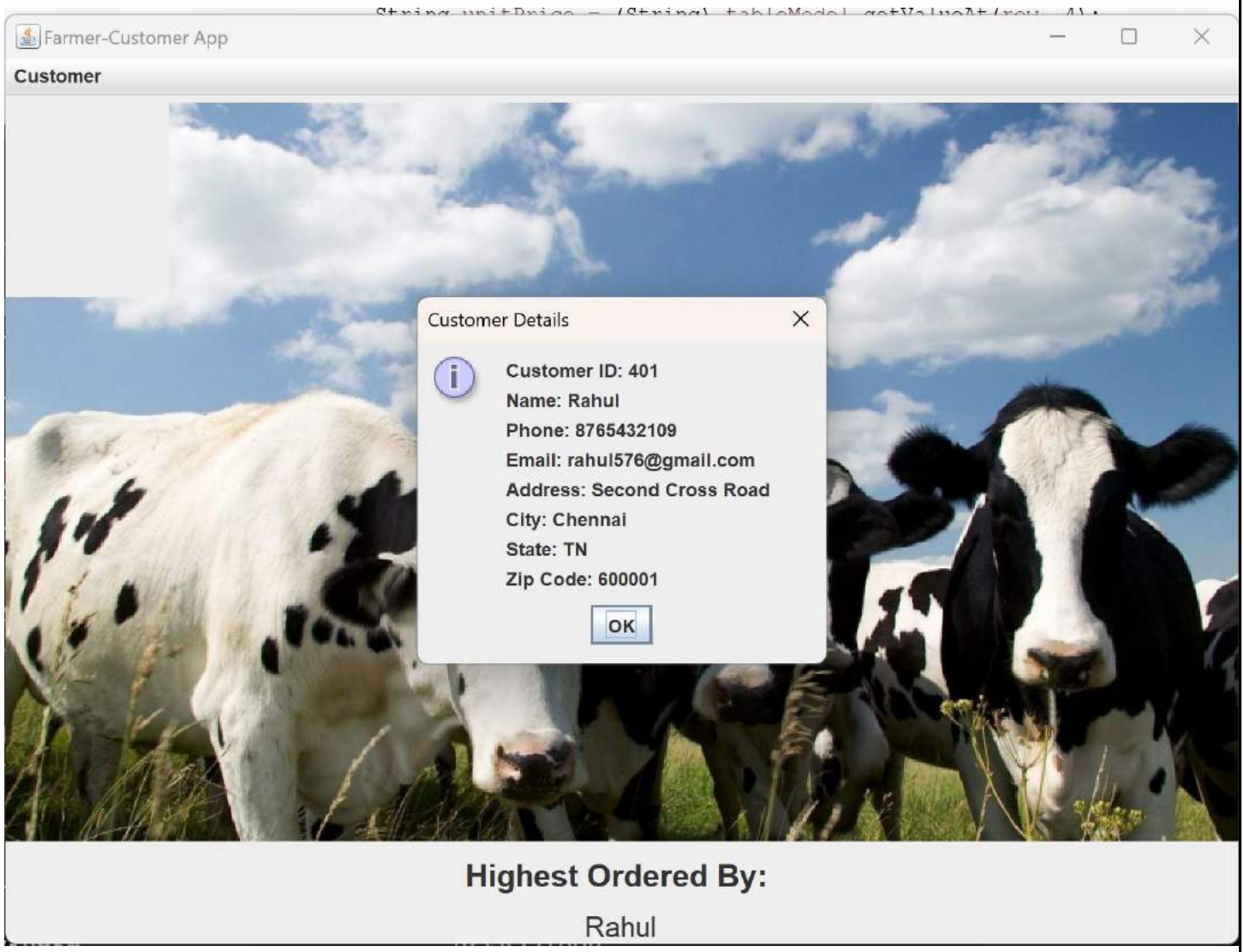
II.CUSTOMER PAGE:


Farmer-Customer App
Customer
Highest Ordered By:
Rahul

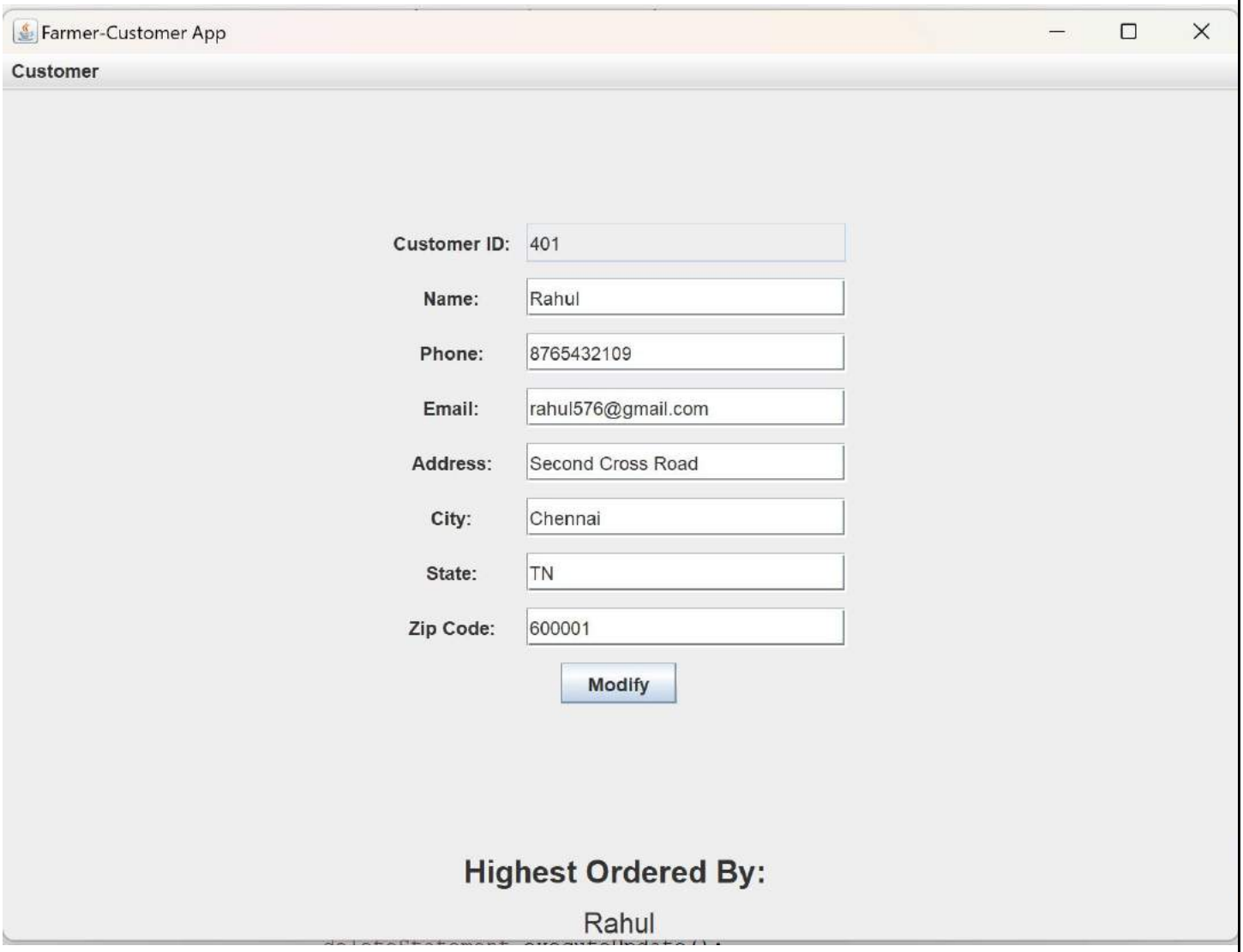
III. CUSTOMER MENU BAR:



IV. CUSTOMER PROFILE:

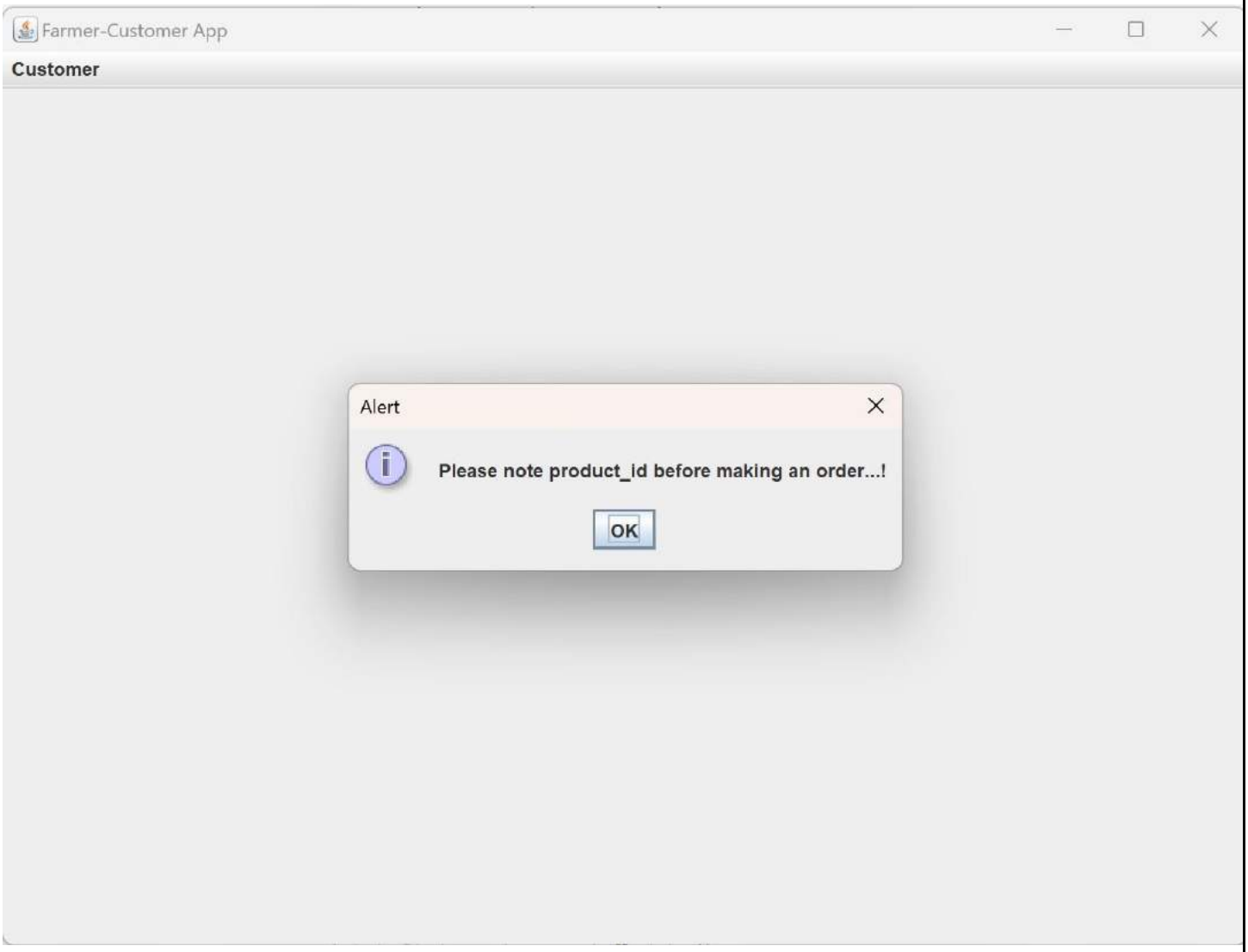


V.UPDATE CUSTOMER DETAILS:

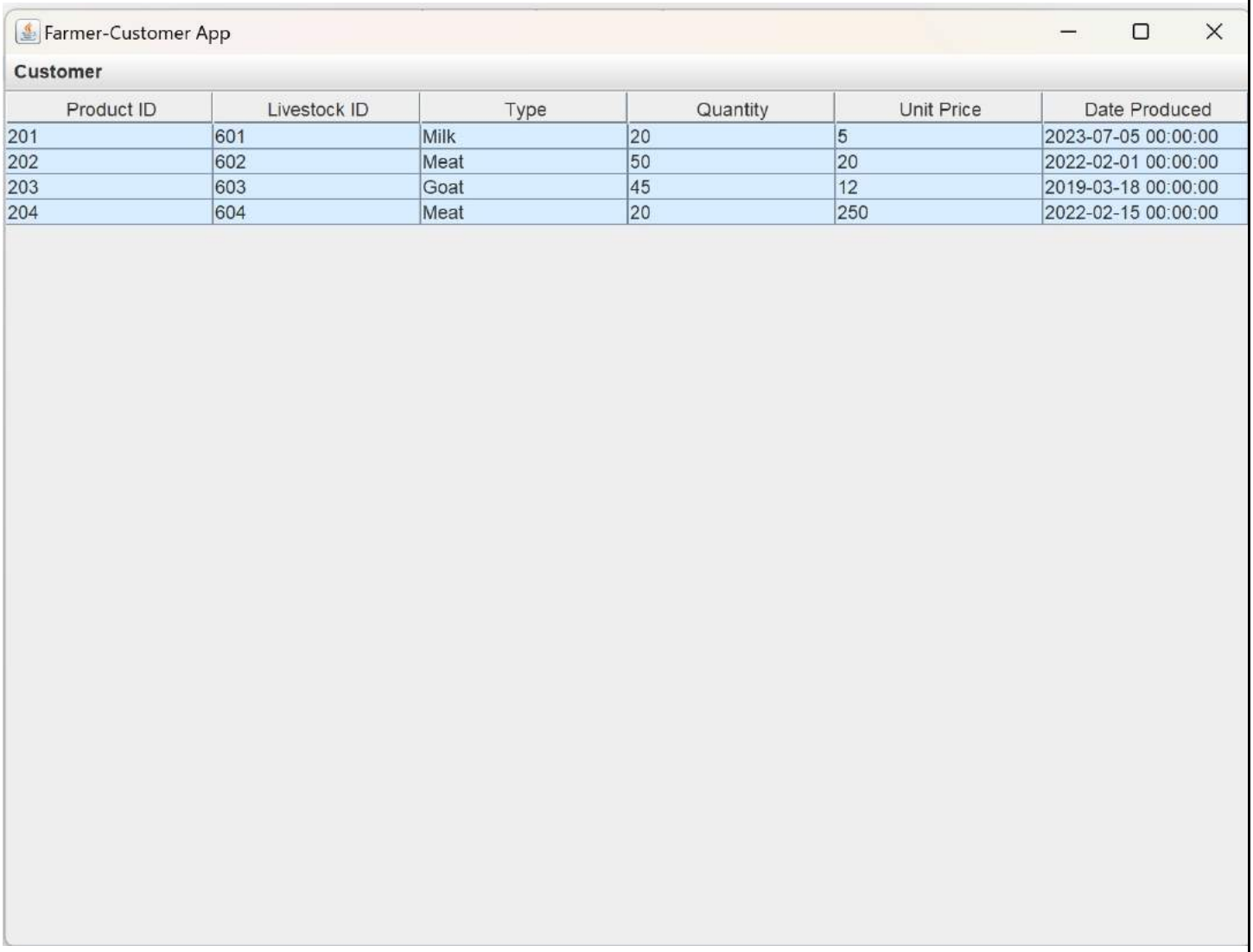


VI.VIEW PRODUCTS:

<u>IMPORTANT NOTE:</u>

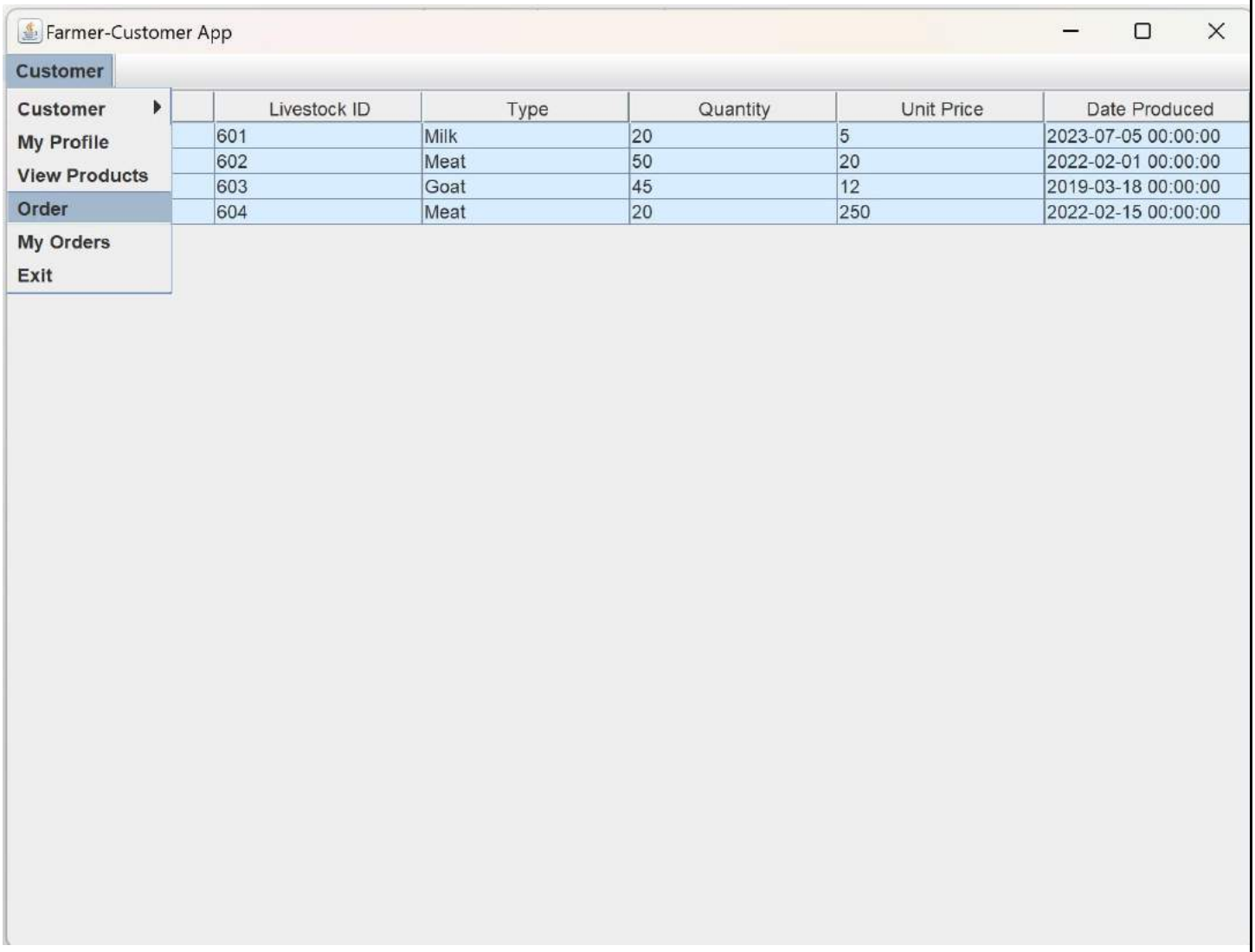


VII.PRODUCTS:

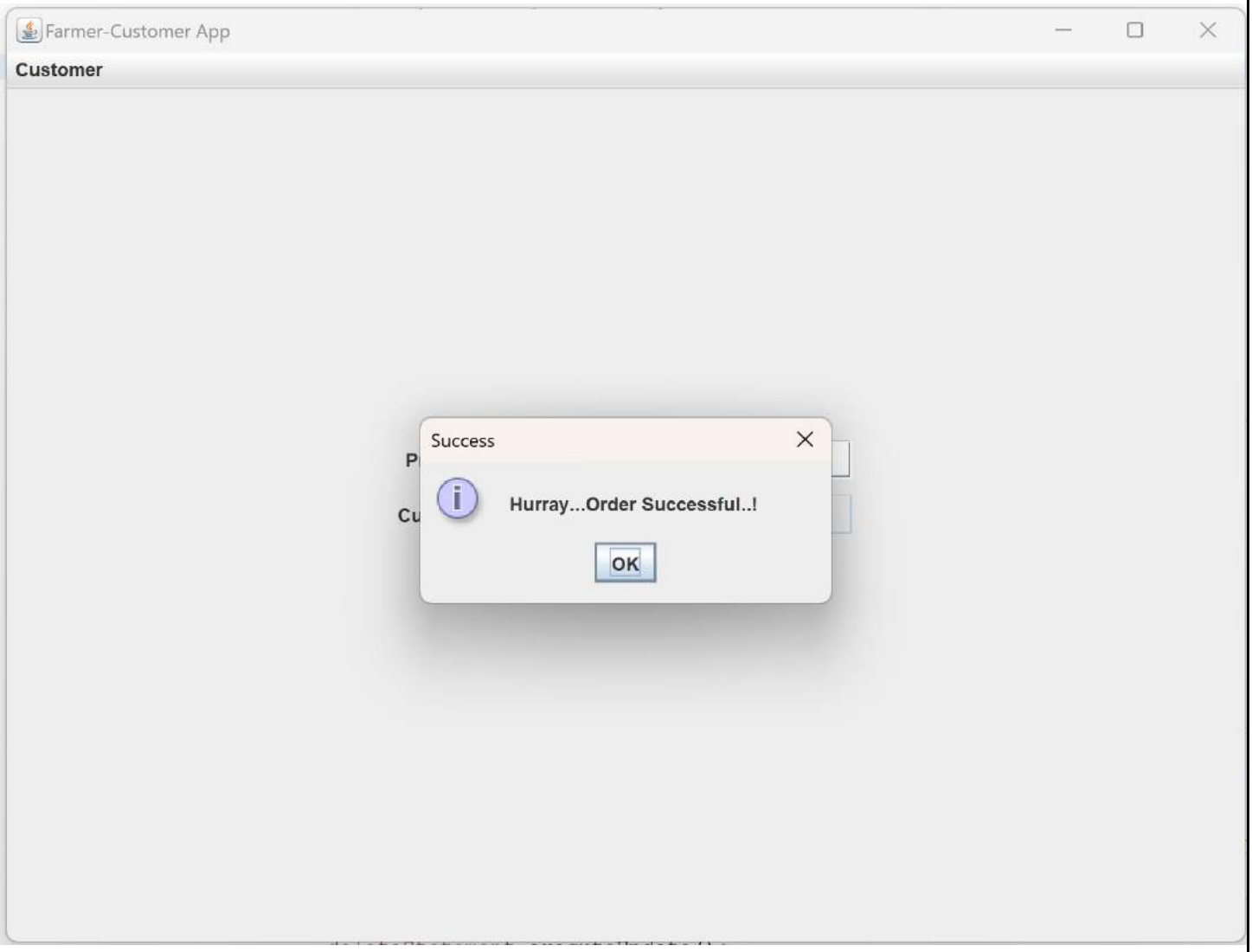


Product ID	Livestock ID	Type	Quantity	Unit Price	Date Produced
201	601	Milk	20	5	2023-07-05 00:00:00
202	602	Meat	50	20	2022-02-01 00:00:00
203	603	Goat	45	12	2019-03-18 00:00:00
204	604	Meat	20	250	2022-02-15 00:00:00

VIII.CLICK ON ORDER:



IX.GIVE THE PRODUCT ID YOU WANT TO ORDER:



X.TO VIEW YOUR ORDERS:

Farmer-Customer App

Customer

Order Id	Customer Id	Product Id	Order Date
451	401	202	2023-06-21 09:58:56
709	401	203	2023-06-21 22:55:52
242	401	202	2023-06-22 13:02:53
273	401	204	2023-07-06 00:59:43
875	401	202	2022-03-04 00:00:00
249	401	203	2023-06-22 13:49:15
465	401	203	2023-07-07 00:00:00
466	401	203	2023-07-07 00:00:00
467	401	203	2023-07-07 00:00:00
468	401	203	2023-07-07 00:00:00
469	401	203	2023-07-07 00:00:00

SQL*PLUS CONNECTIVITY:

```
Run SQL Command Line

SQL*Plus: Release 11.2.0.2.0 Production on Thu Jul 6 01:04:31 2023

Copyright (c) 1982, 2010, Oracle.  All rights reserved.

SQL> conn manisaiganesh/vasavi;
Connected.
SQL>
```

```
SQL> select * from Farmer;

FARMER_ID NAME                 EMAIL                          PHONE
--------- -------------------- ------------------------------ ---------------
        4 Tejash               tejash03@gmail.com             9391829203
        1 Rangaiah             rangaiah12@gmail.com           9876543212
        2 SreeRam              sundharam16@gmail.com          7329746295
        3 Ramesh               ramesh89@gmail.com             9763412794
```

```
Run SQL Command Line

SQL> conn manisaiganesh/vasavi;
Connected.
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
CUSTOMER                       TABLE
FARM                           TABLE
FARMER                         TABLE
LIVESTOCK                      TABLE
ORDERS                         TABLE
PRODUCT                        TABLE

6 rows selected.

SQL>
```

```
SQL> select * from Farmer;

FARMER_ID NAME                 EMAIL                          PHONE
--------- -------------------- ------------------------------ ----------------
        4 Tejash               tejash03@gmail.com             9391829203
        1 Rangaiah             rangaiah12@gmail.com           9876543212
        2 SreeRam              sundharam16@gmail.com          7329746295
        3 Ramesh               ramesh89@gmail.com             9763412794
```

```
SQL> select * from Farm;

FARM_ID FARMER_ID FARM_NAME           ADDRESS                      CITY           STATE          ZIPCODE
------- --------- ------------------- ---------------------------- -------------- -------------- ----------
    504         4 Green Farms         12 Farm Rd                   Delhi          Delhi          110002
    501         1 Happy Valleys       Madhulapalli                 Jagtial        Telangana      505452
    502         2 Sundharam Acres     Chintakunta                  Karimnagar     Telangana      518348
    505         4 Golden Fields       456 Park Ave                 Mumbai         MH             400002
    503         3 Ramesh Fields       Cherlapalli                  Jagtial        Telangana      505454
```

```
SQL> select * from Livestock;

LIVESTOCK_ID FARM_ID TYPE                 GENDER  DOB       BREED                DATE_ADDE
------------ ------- -------------------- ------- --------- -------------------- ---------
         604     504 Buffalo              Male    10-AUG-18 Murrah               15-JUL-20
         601     501 Cattle               Male    20-APR-19 Angus                05-MAY-19
         602     502 Sheep                Female  10-FEB-21 Dorper               01-MAR-21
         603     503 Goat                 Male    17-SEP-18 Boer                 01-OCT-18
```

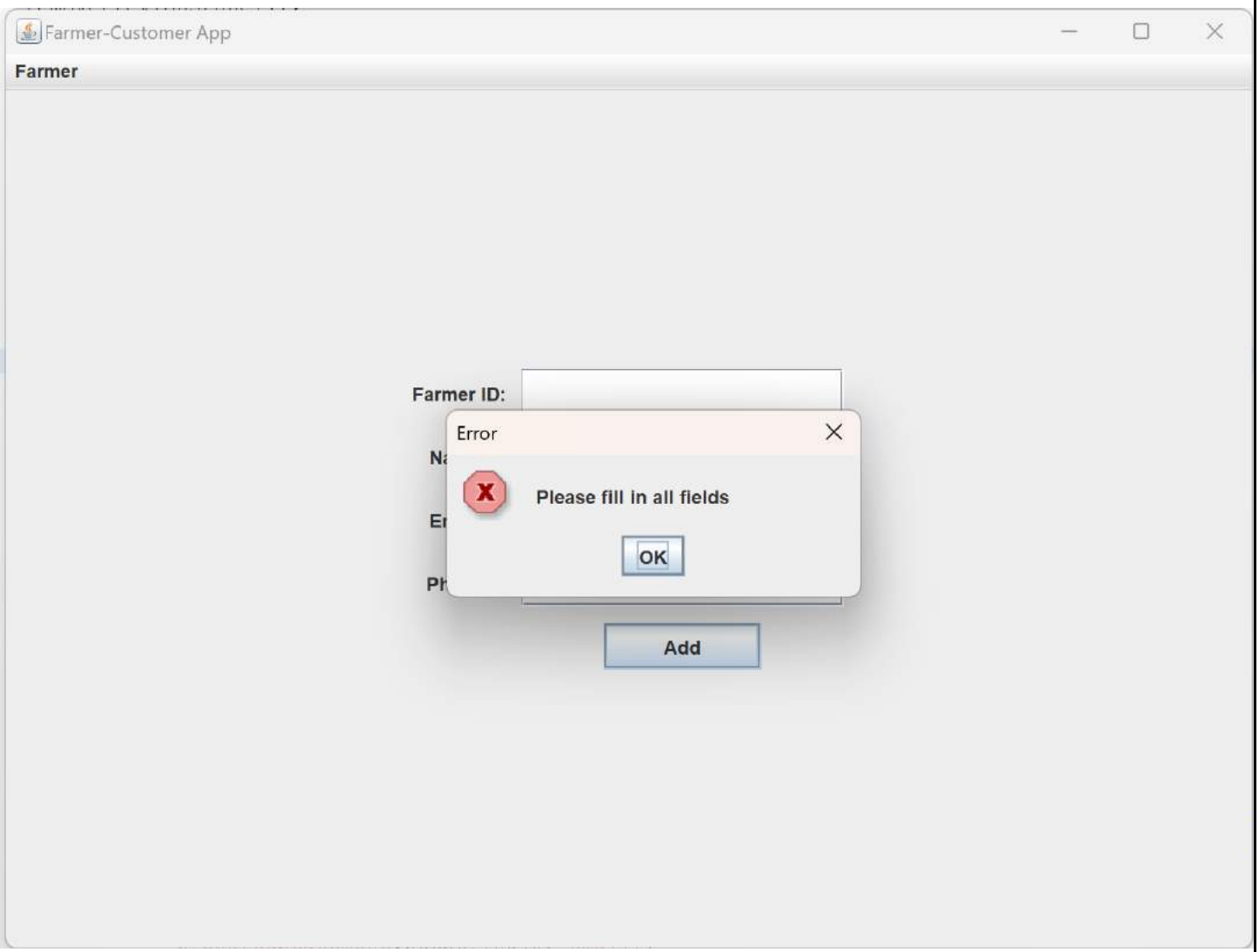
```
SQL> select * from Product;

PRODUCT_ID LIVESTOCK_ID TYPE         QUANTITY UNIT_PRICE DATE_PROD
---------- ------------ ---------- ---------- ---------- ---------
       201          601 Milk               20          5 05-JUL-23
       202          602 Meat               50         20 01-FEB-22
       203          603 Goat               45         12 18-MAR-19
       204          604 Meat               20        250 15-FEB-22
```

```
SQL> select * from Customer;

CUSTOMER_ID NAME                 PHONE        EMAIL                      ADDRESS                    CITY         STATE        ZIPCODE
----------- -------------------- ------------ -------------------------- -------------------------- ------------ ------------ --------
        401 Rahul                8765432109   rahul576@gmail.com         Second Cross Road          Chennai      TN           600001
        402 Rani                 9876543210   rani524@gmail.com          Gachibowli                 Hyderabad     TS          500032
        403 Vikram               7654321098   vikram298@gmail.com        Durgam Cheruvu             Hyderabad     TS          500081
```

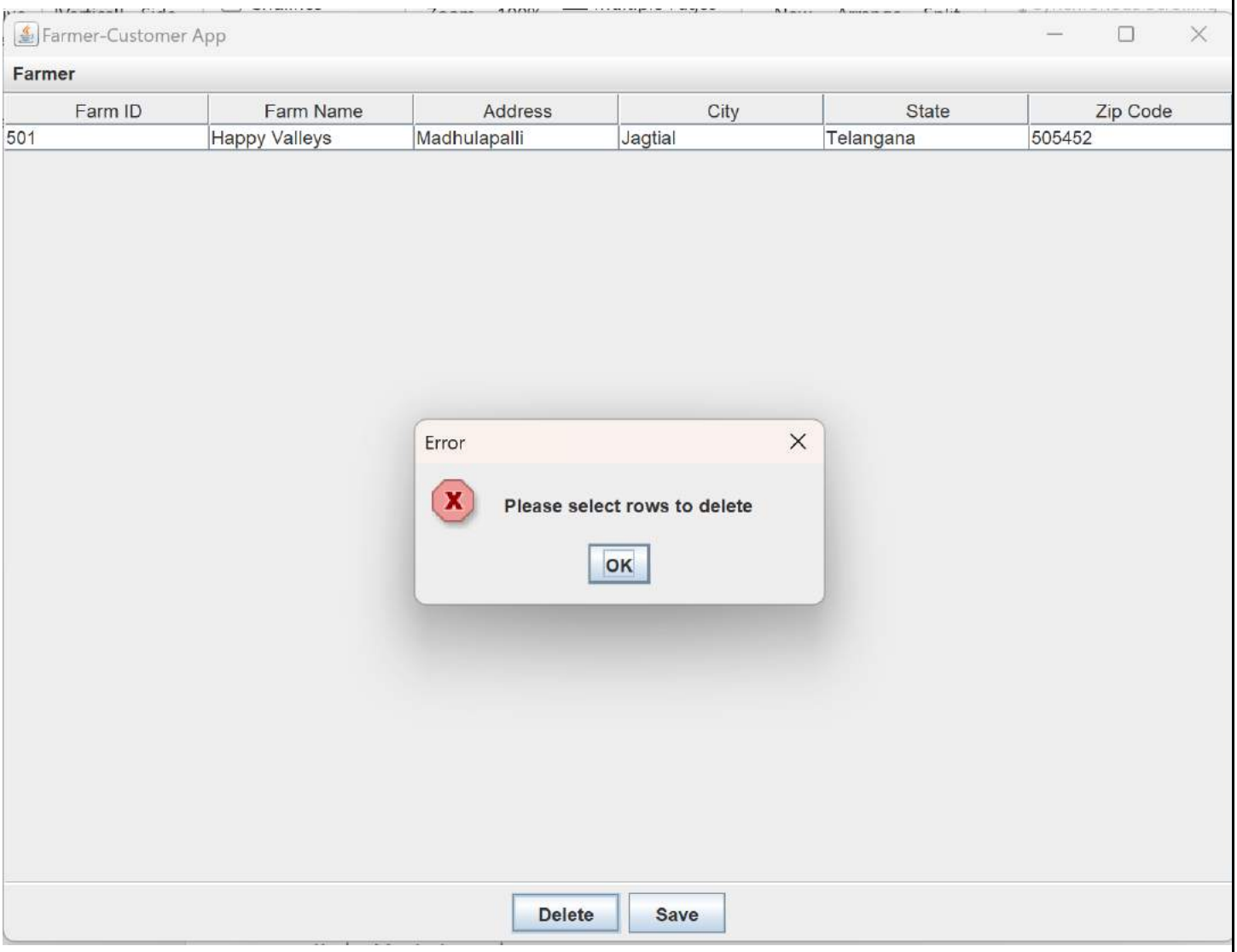
```
SQL> select * from Orders;

   ORDER_ID CUSTOMER_ID PRODUCT_ID ORDER_DAT
---------- ----------- ---------- ---------
        451         401        202 21-JUN-23
        982         402        202 21-JUN-23
        709         401        203 21-JUN-23
        242         401        202 22-JUN-23
        273         401        204 06-JUL-23
        875         401        202 04-MAR-22
        561         402        203 04-APR-19
        249         401        203 22-JUN-23
        807         403        202 22-JUN-23
        465         401        203 07-JUL-23
        466         401        203 07-JUL-23
        467         401        203 07-JUL-23
        468         401        203 07-JUL-23
        469         401        203 07-JUL-23

14 rows selected.
```

CHECKING SOME CONSTRAINTS:

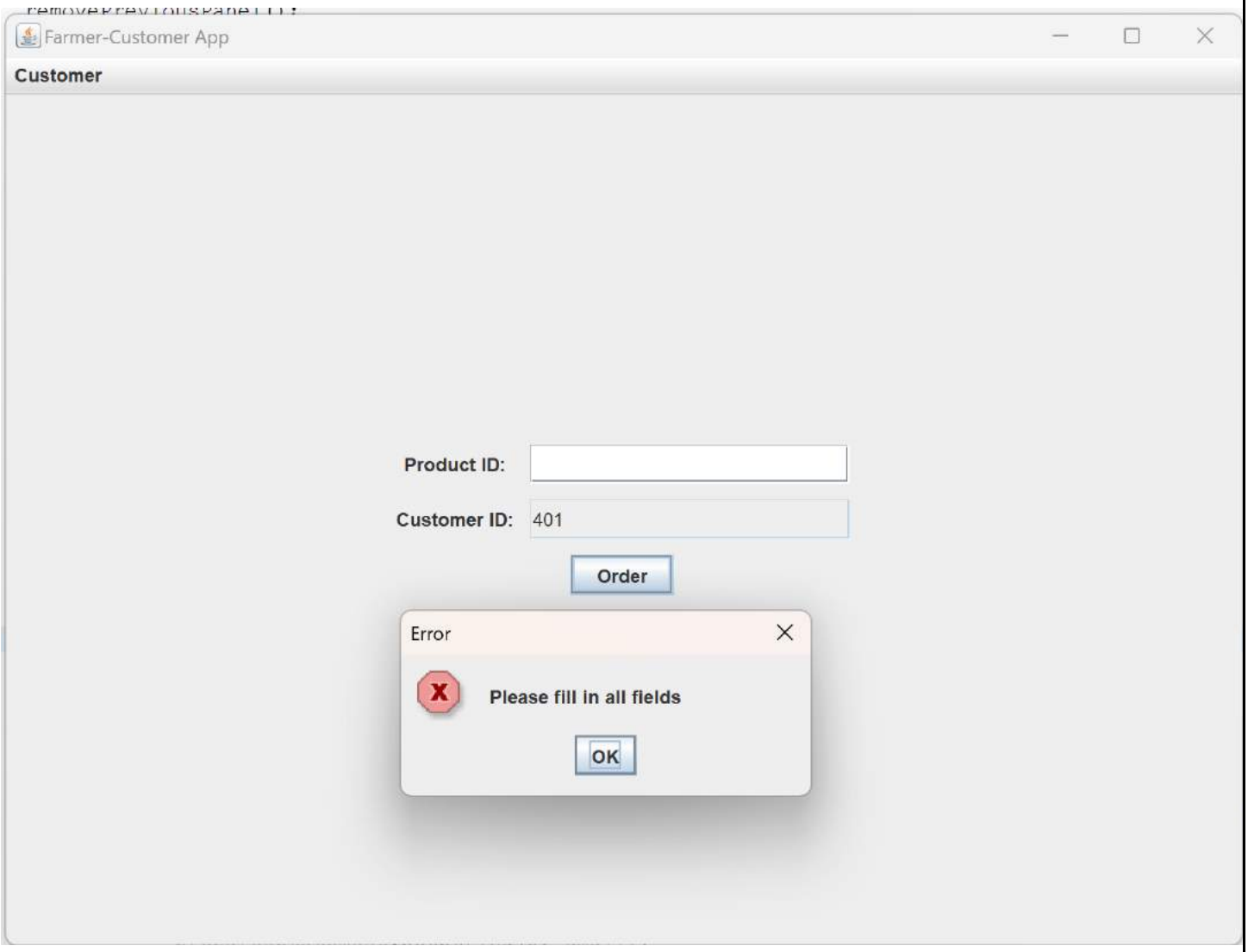
I.FOR ADDING WITHOUT FILLING:



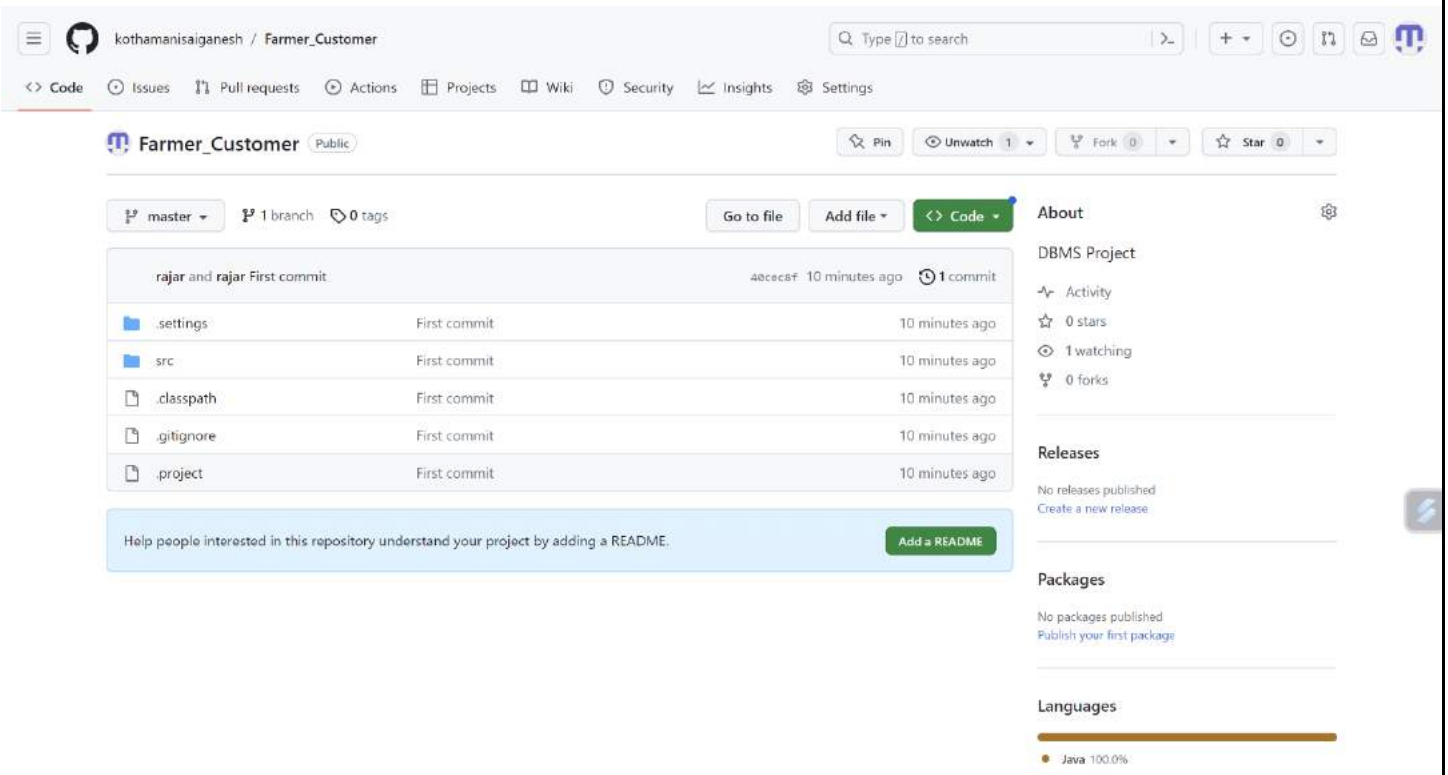
II.FOR DELETING WITHOUT SELECTING:

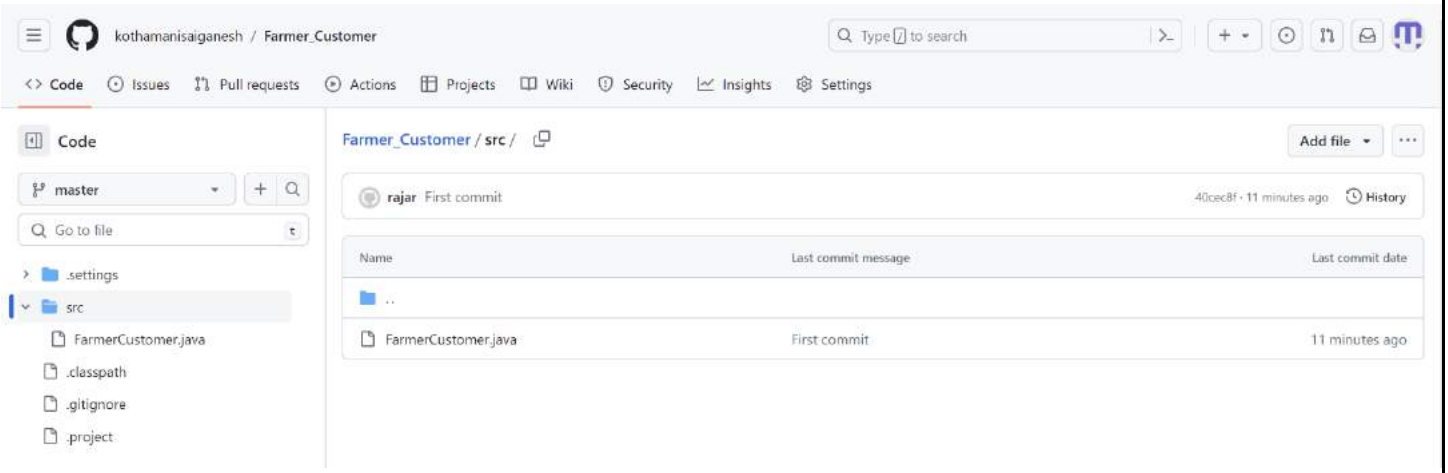


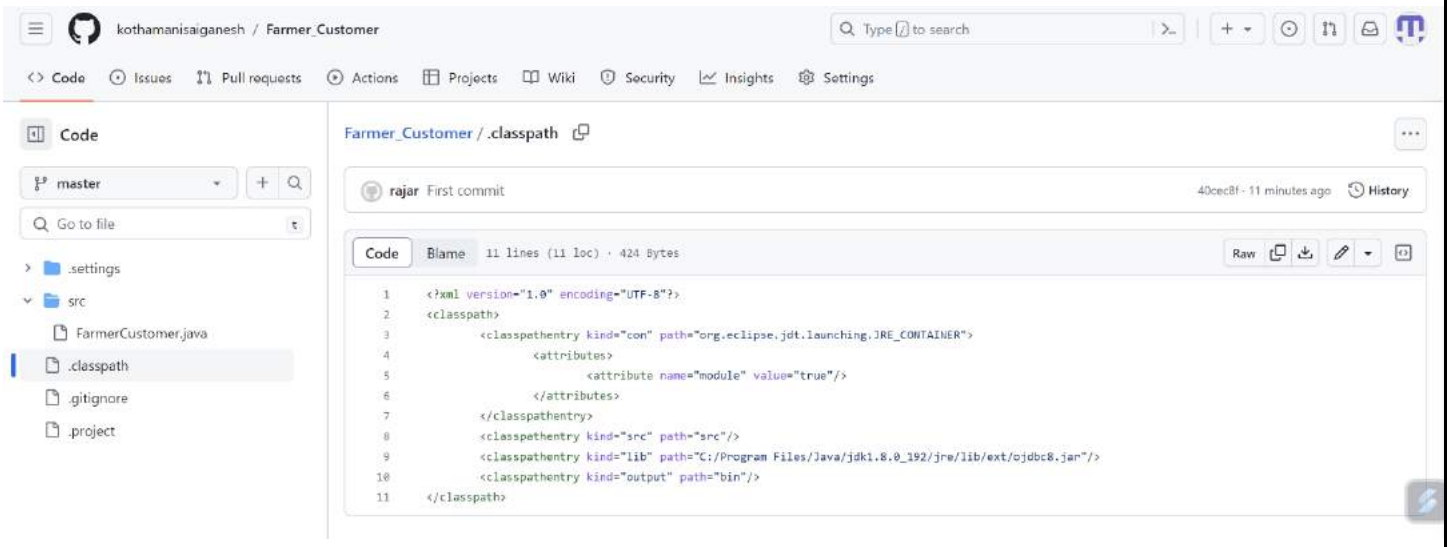
III. MAKING AN ORDER WITHOUT PRODUCT ID:

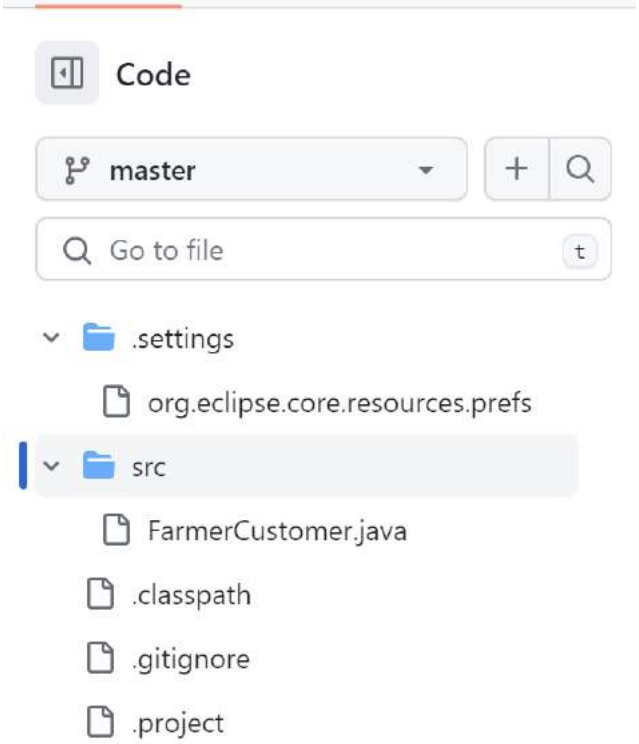

Farmer-Customer App
Customer
Product ID:
Customer ID:
401
Order
Error
Please fill in all fields
OK

GITHUB:


kothamanisaiganesh / Farmer_Customer
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Farmer_Customer
Public
Pin
Unwatch 1
Fork 0
Star 0
master
1 branch
0 tags
Go to file
Add file
Code
rajar and rajar First commit
40cec8f 10 minutes ago
1 commit
.settings
First commit
10 minutes ago
src
First commit
10 minutes ago
.classpath
First commit
10 minutes ago
.gitignore
First commit
10 minutes ago
.project
First commit
10 minutes ago
Help people interested in this repository understand your project by adding a README.
Add a README
About
DBMS Project
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
Java 100.0%


kothamanisaiganesh / Farmer_Customer
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Code
master
Go to file
.settings
src
FarmerCustomer.java
.classpath
.gitignore
.project
Farmer_Customer / src /
Add file
rajar First commit
40cec8f · 11 minutes ago
History
Name
Last commit message
Last commit date
..
FarmerCustomer.java
First commit
11 minutes ago





Github link:

https://github.com/kothamanisaiganesh/Farmer_Customer.git

RESULTS

I have successfully completed the mini-project ***"ONLINE REGISTRATION AND PROCESSING OF DAIRY FARMERS PRODUCING MILK AND OTHER LIVESTOCK PRODUCTS"*** .

DISCUSSION AND FUTURE WORK

The project's success relies on an intuitive and user-friendly interface for dairy farmers. It is important to gather feedback from users during the development process and continuously refine the user experience. Conducting usability testing and incorporating user suggestions can further enhance the system's usability and adoption rate.To create a comprehensive ecosystem for dairy farmers, consider integrating the system with external applications or platforms. This could include integration with financial systems for seamless payment processing, integration with weather APIs for real-time weather updates affecting farming practices, or integration with third-party logistics providers for efficient product distribution.

Implement real-time data synchronization between the front-end application and the back-end database to ensure up-to-date information availability.Integrate machine learning algorithms to predict milk production levels, detect anomalies, and optimize feed and nutrition management practices.Develop a recommendation engine that suggests best practices, breeding programs, and healthcare protocols based on historical data and industry benchmarks.Implement a feedback mechanism for farmers to share their experiences, suggestions, and concerns, enabling continuous improvement and addressing user needs effectively.

REFERENCES

- https://docs.oracle.com/javase/7/docs/api/
- https://www.javatpoint.com/java-swing
- https://stackoverflow.com/