Sir Syed University of Engineering & Technology (SSUET)
Software Engineering Department

Course Name: Database Management System (SWE-209L)

Semester: 4th

Batch:2023F

PROJECT REPORT

Project Title: Farm-Master (FMS)

Project By:

Hassan Ahmed
Syed Hussain Ali
Muhammad Umar Intesab
Abdul Basit

TABLE OF CONTENTS

1. Introduction of the Project:

Farm Master is a comprehensive desktop-based application developed to centralize and streamline various farming operations through multiple user portals, including Admin, Manager, and User roles. The core aim of this project is to provide an all-in-one agricultural management system that covers animal management, crop production, field operations, and labour administration.

Animal Management Module

This module facilitates the effective management of livestock and related tasks:

- Feeding schedules
- Breeding programs
- Health care and vaccination records
- Addition and removal of livestock
- Placement and tracking of animals

Crop Management Module

This section focuses on optimizing crop production and profit analysis:

- Scheduling of planting and harvesting
- Yield calculation and prediction
- Benefit calculation and analysis

Field Management Module

This module manages the use and profitability of farming land:

- Adding new fields for farming
- Selling and purchasing of fields
- Planting crops in specific fields
- Managing activities such as irrigation, fertilization, and pest control
- Tracking field yields and calculating profits

Labour Management Module

The labour management section ensures smooth workforce operations by:

- Hiring new labourers
- Terminating services when required
- Managing and updating labour details
- Processing and sending salaries based on specific customer demands and requests
- Generating salary records and payroll management

Farm Master ultimately **aims** to:

- Enhance agricultural productivity
- Reduce unnecessary costs
- Improve the operational efficiency of farming activities through technology

This innovative platform supports modern digital farming practices and bridges the gap between traditional agricultural methods and contemporary management needs.

2. SQL Query used for Table Creation

TABLES CREATION:

1. livestock – Animal details

```
CREATE TABLE livestock (
   Animal_ID INT PRIMARY KEY,
   Animal_Name VARCHAR(50),
   Breed VARCHAR(50),
   Gender VARCHAR(10),
   DOB DATE,
   Health_Status VARCHAR(50)
);
```

2. labour – Labour details

```
CREATE TABLE labour (
   Labour_ID INT PRIMARY KEY,
   Labour_Name VARCHAR(50),
   Age INT,
   Gender VARCHAR(10),
   Contact VARCHAR(20),
   Position VARCHAR(50)
);
```

3. user – User login details

```
CREATE TABLE [user] (
   u_id INT PRIMARY KEY,
   u_name VARCHAR(50),
   password VARCHAR(50),
   email VARCHAR(100),
   u_type VARCHAR(20)
);
```

4. fields – Field details

```
CREATE TABLE fields (
   Field_ID INT PRIMARY KEY,
   Field_Name VARCHAR(50),
   Soil_Type VARCHAR(50),
   Area FLOAT,
   Status VARCHAR(20)
);
```

5. tasks – Field task assignments

```
CREATE TABLE tasks (
   Task_ID INT PRIMARY KEY,
   Task_Name VARCHAR(50),
   Field_ID INT,
   Labour_ID INT,
   Date DATE,
   Status VARCHAR(20)
);
```

6. salary_requests – Salary request submissions

```
CREATE TABLE salary_requests (
   request_id INT PRIMARY KEY AUTO_INCREMENT,
   employee_name VARCHAR(100),
   amount DECIMAL(10,2)
);
```

7. account_requests – New account requests

```
CREATE TABLE account_requests (
   request_id INT PRIMARY KEY AUTO_INCREMENT,
   user_name VARCHAR(50),
   password VARCHAR(50)
);
```

Other tables:

Table Name	Purpose (Comment)
vaccination	Records animal vaccinations (type, date, dose, linked to ID).
userprofile	Stores image path of uploaded user/labour profile photos.
cropstime	Stores estimated harvest time (in days) for each crop.
cropyielded	Records actual crop yield data in kg for yield tracking.
distinctions	Defines breed characteristics like gender, health, breeder.
seeds	Details seed name, type, price, and stock.
crops	Maintains crop name, type, and seasonal info.
breedings	Tracks breeding events with animal IDs, type, and outcome.

3. SQL Query used for Views

VIEWS CREATION:

1. animalType – Groups animals by type

```
CREATE OR ALTER VIEW animalType AS
SELECT Animal_Type
FROM Livestock
GROUP BY Animal_Type;
GO
```

2.BreedingsView – Displays breeding information with animal type

```
CREATE VIEW BreedingsView AS
SELECT
   B.br_id AS br_id,
   B.male AS male,
   B.female AS female,
   B.pregnancyStatus AS PregnancyDays,
   L.Animal_Type
FROM
   Breeding B
JOIN
   Livestock L ON B.female = L.Animal_Name;
```

View Name	Purpose (Description)
ShowFeedings	**Shows morning and evening feeding records sorted by feeding date.**
AvailableFields	**Lists all fields where the status is marked as "Available".**
showTasks	**Displays field tasks assigned to labourers with dates and field names.**

FUNCTIONS:

1. getSire(@SireType) – Returns breedable male animals of a specific type

```
CREATE FUNCTION getSire(@SireType VARCHAR(20))
RETURNS TABLE
AS
RETURN(
   SELECT Animal_Name
   FROM Livestock
   WHERE gender = 'Male' AND Animal_Type = @SireType AND canBreed = 'Yes');
```

2. **getDam(@SireType) – Returns breedable female animals of a specific type**

```
CREATE FUNCTION getDam(@SireType VARCHAR(20))
RETURNS TABLE
AS
RETURN(
   SELECT Animal_Name
   FROM Livestock
   WHERE gender = 'Female' AND Animal_Type = @SireType AND canBreed = 'Yes'
);
```

Other Function Names

Function Name	Purpose (Description)
getFeed()	Counts and lists how many times each type of feed is used (morning + evening).

4. SQL Query used for Stored Procedures

STORED PROCEDURES:

1. animalHealth

```
-- Checks if the majority of animals of a given type are healthy
SELECT @total = COUNT(*)
FROM Livestock
WHERE Animal_Type = @typeAnimal;

SELECT @healthy = COUNT(*)
FROM Livestock
WHERE Animal_Type = @typeAnimal AND HealthStatus = 'Healthy';

-- Returns 'Healthy' if at least half of the animals are healthy, else 'Unhealthy'
IF @total > 0 AND @healthy * 2 >= @total
   SELECT 'Healthy' AS Status;
ELSE
   SELECT 'Unhealthy' AS Status;
```

2. VaccinationSchedule

```
-- Retrieves vaccination details and total dosage count for a specific animal type
SELECT
   vc.v_name,
   vc.v_date,
   COUNT(vc.dosage) AS total_dosage,
   L.Animal_Type
FROM
   Vaccination vc
INNER JOIN
   Livestock L ON vc.v_id = L.VaccineRecord
WHERE
   L.Animal_Type = @typeAnimal
GROUP BY
   vc.v_name, vc.v_date, L.Animal_Type;
```

3. AddBreeding

```
-- Inserts breeding record linking male and female animals along with pregnancy status
SELECT @male_id = Animal_id FROM Livestock WHERE Animal_name = @p_male AND Animal_Type =
@p_type;
SELECT @female_id = Animal_id FROM Livestock WHERE Animal_name = @p_female AND Animal_Type =
@p_type;

SET @pregnancyStatus = DATEDIFF(MONTH, @p_date, GETDATE());

INSERT INTO Breeding (
   male, maleId, female, femaleId, litterSize, pregnancyStatus
)
VALUES (
   @p_male, @male_id, @p_female, @female_id, @p_litterSize, @pregnancyStatus
);
```

4. PlantingSch

```
-- Adds a planting schedule for a crop in a field if the field is not already planted
SELECT TOP 1 @v_Crop_ID = Crop_ID FROM Crops WHERE Crop_Name = @p_Crop_name;
SELECT TOP 1 @v_Field_ID = Field_ID FROM Fields WHERE Field_Name = @p_Field;

IF NOT EXISTS (SELECT 1 FROM Fields WHERE Field_ID = @v_Field_ID AND Status = 'Planted')
BEGIN
   INSERT INTO Planting_Schedule (
      Crop_ID, Field_ID, Planting_Date_Time,
      Expected_harvest_Date_Time, Seed_Quantity, Status
   )
   VALUES (
      @v_Crop_ID,
      @v_Field_ID,
      GETDATE(),
      DATEADD(MONTH, @p_TimeRequired, GETDATE()),
      @p_seedQ,
      'Planted'
   );

   UPDATE Fields SET Status = 'Planted' WHERE Field_ID = @v_Field_ID;
END
```

Other Procedures Name and Description:

Procedure Name	Purpose / Description
animalHealth	Checks if most animals of a type are healthy.
checkcropYield	Evaluates if crop yield is above average.
checkField	Determines if a field is planted or available.
CheckVaccine	Verifies vaccination record exists for an animal type and vaccine.
CountCrops	Counts total crops of a certain type.
CropYield	Calculates total yield of a specific crop.
DeleteBreeding	Deletes breeding records between male and female animals.
DeleteCrop	Removes a crop entry by ID.
DeleteField	Deletes a field record.
DeleteHarvest	Deletes a harvest record for a crop.
DeleteLivestock	Removes a livestock animal by ID.
DeletePesticide	Deletes a pesticide record.
DeletePlantingSchedule	Deletes a planting schedule by ID.
DeleteTreatment	Removes an animal treatment record.
GetAnimalCount	Returns count of animals by type.
GetCropID	Fetches crop ID by crop name.
GetFieldID	Fetches field ID by field name.
GetLivestockID	Fetches livestock ID by animal name.
GetPesticideID	Fetches pesticide ID by name.
GetVaccineID	Fetches vaccine ID by name.
InsertCrop	Adds a new crop entry.
InsertField	Adds a new field entry.
InsertHarvest	Records a new harvest entry for a crop.
InsertLivestock	Adds a new livestock animal record.
InsertPesticide	Inserts a new pesticide record.

InsertTreatment	Adds an animal treatment record.
PlantingSch	Adds a new planting schedule if field is free.
recordVaccination	Records a vaccination event for livestock.
treatAnimals	Inserts animal treatment details including treatment type and date.
updateCrop	Updates existing crop details.
updateField	Updates field details.
updateHarvest	Updates harvest information for a crop.
updateLivestock	Updates animal details.
updatePesticide	Updates pesticide details.
updatePlantingSch	Updates planting schedule details.
updateTreatment	Updates treatment record for an animal.
updateVaccination	Updates vaccination records.
AddBreeding	Records breeding event between animals with pregnancy status.
BreedingStatus	Reports breeding status and litter size for female animal.
VaccineDue	Lists animals due for vaccination by type and vaccine.

5. SQL Query used for Stored Procedures

TRIGGERS:

-- Trigger to insert a new 'Incomplete' task status for a labourer when a new task is added

```
CREATE TRIGGER insert_task_status
ON task
AFTER INSERT
AS
BEGIN
   INSERT INTO task_status (employee_name, task, status, an_date)
   SELECT l.Labourer_Name, i.Task_Description, 'Incomplete', GETDATE()
   FROM inserted i
   JOIN labour l ON i.Labour_ID = l.Labour_ID;
END;
```

-- Trigger to update task_status to 'Completed' when a task status is updated to completed

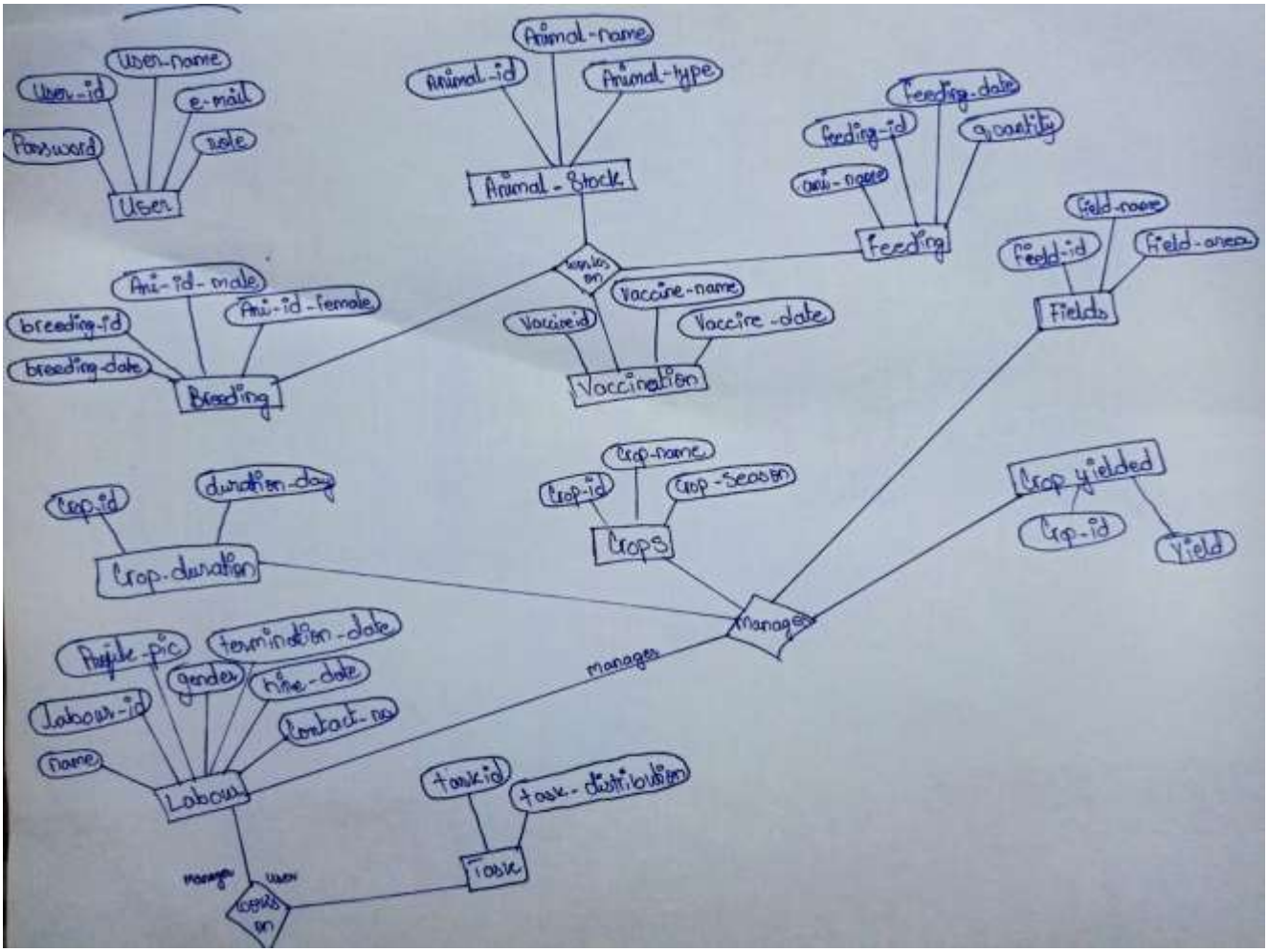
```
CREATE TRIGGER updateTasks
ON task
AFTER UPDATE
AS
BEGIN
   -- Update the existing task_status record instead of inserting new
   UPDATE ts
   SET
      ts.status = 'Completed',
      ts.an_date = CAST(GETDATE() AS DATE)
   FROM task_status ts
   JOIN inserted i ON ts.task = i.Task_Description
   JOIN labour l ON i.Labour_ID = l.Labour_ID
   JOIN deleted d ON i.Task_ID = d.Task_ID
   WHERE
      i.status = 'Completed'
      AND d.status <> 'Completed'
      AND ts.employee_name = l.Labourer_Name
      AND ts.status = 'Incomplete';
END;
```

7. Project Scheduling

FarmMaster Project Scheduling - Gantt Chart (Text Format)

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
1. Project Initiation	■									
2. Requirements Gathering		■	■							
3. System Design				■	■					
4. Backend Development					■	■	■			
4. Frontend Development					■	■	■			
5. Testing & QA								■	■	
6. Deployment & Review										■

8. ER Diagram:



9. SQL operations used in project

- INSERT
- UPDATE
- DELETE
- VIEWS
- FUNCTIONS
- STORED PROCEDURES
- TRIGGER

10. Constraints Used in Project:

Constraint	**Where Used**	**Why Used (One-liner)**
PRIMARY KEY	Labour, Customer, Order, Task	To uniquely identify each record in a table.
FOREIGN KEY	Order, Task, task_status	To establish relationships between tables.
NOT NULL	All key columns and names	To ensure important fields like names and IDs are always provided.
DEFAULT	task_status.status	To set a default task status as 'Incomplete' for new tasks.
CHECK	Labour.gender	To allow only valid values ('Male', 'Female') in gender field.
IDENTITY	Labour_ID, Task_ID, etc.	To auto-generate unique IDs for primary key fields.
UNIQUE	Labour.CNIC	To avoid duplicate CNIC entries for labourers.

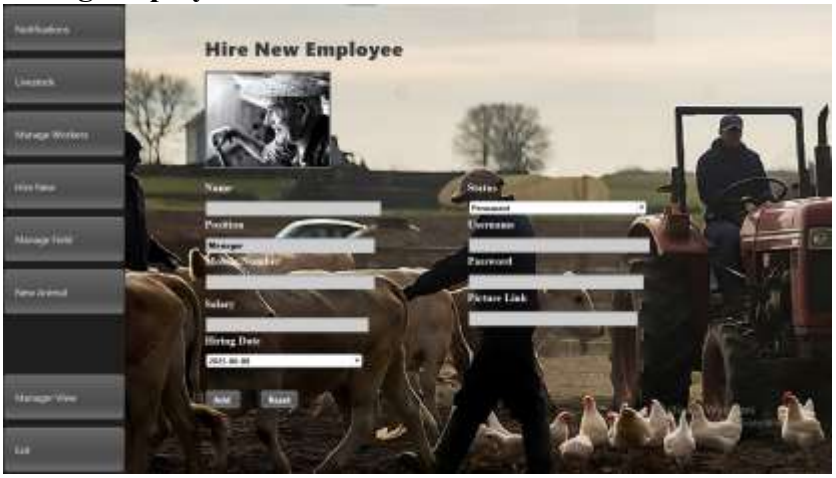
11. User Guide:

Entry Page:

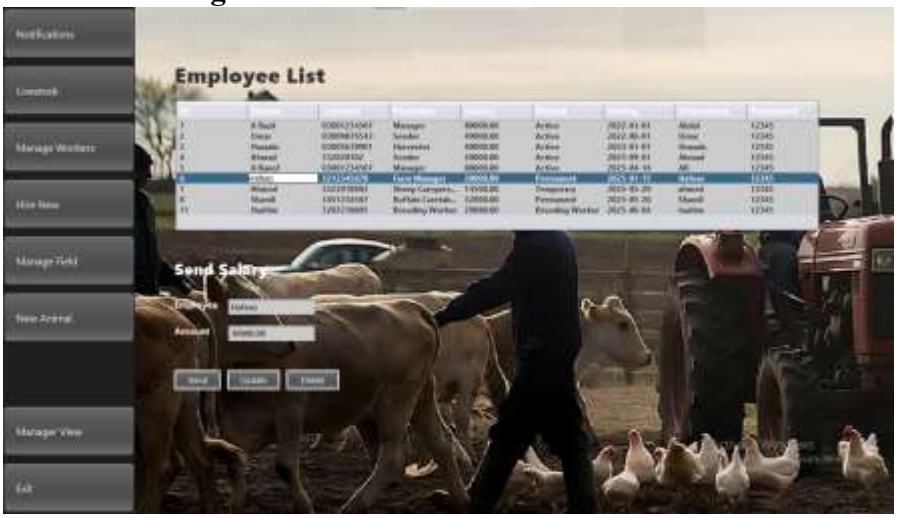
Login Page:

Fields Management:

Hiring Employee:



Workers Management:



New Animal Addition:

Animal Management:

Crop Management:

Task Management:

Worker Portal:

Profile and Task:

Notifications:

12. Conclusion

FarmMaster is a desktop application designed to help farmers manage daily agricultural tasks efficiently. It includes features like labor task scheduling, resource tracking, and task status updates. The system ensures smooth operation using a user-friendly interface and reliable data management through SQL Server. It minimizes manual errors, improves transparency, and supports small to medium-sized farm operations. Our team worked on backend development, UI design, database setup, and testing to meet both functional and non-functional requirements.

13. Future Expansion

To meet the growing demands of the agricultural sector and reach a wider audience, the future expansion of FarmMaster includes:

🌐 Web-Based Platform

We plan to convert FarmMaster from a desktop application into a web-based system, allowing:

- **Remote access** from any device with internet connectivity.
- **Global availability,** enabling farmers and organizations around the world to benefit from the tool.
- **Cloud-based** data storage for secure and centralized management.

🚀 Upcoming Features and Enhancements

- **Mobile application** integration for on-the-go task management.
- **Multilingual support** for farmers from different regions.
- **AI-based** crop and weather recommendations using real-time data.
- **Analytics** dashboard for visual insights into farm operations.
- **Role-based access control (RBAC)** to support different user roles like Admin, Supervisor, and Worker.