VIT®

BHOPAL

CHALLENGING TASK

"Milk Production Tracker"

Student: Narode Yuvraj Badrinath
Reg No: 25BCY10129
Course/Semester: Python Essentials

1. TITLE PAGE

Project Title: Milk Production Tracker
Course: Python Essentials
Student Name: Yuvraj Narode
Academic Year: 2025

2. INTRODUCTION

Milk production is a daily activity in dairy farms, especially small-scale rural farms. These farmers usually record milk quantities in notebooks or on loose papers. This method is prone to errors, pages get damaged, and it becomes difficult to calculate total daily milk output.

This project aims to build a simple Milk Production Tracker using Python and SQLite. The system allows farmers (or operators) to register farmers, register animals, store daily milk entries, and generate reports. It is designed to be lightweight, user-friendly and easy for beginners.

3. PROBLEM STATEMENT

Small dairy farmers lack an easy and reliable method to maintain milk production records. Manual entries on paper are:

>Hard to maintain

>Easily lost or damaged

>Not searchable

>Difficult to summarize

There is a need for a small, digital system to store farmer details, animals, and daily milk production in a structured format.

4. OBJECTIVES

1. Create a simple milk-tracking system for small dairy farms.
2. Digitalize farmer and animal data using a structured database.
3. Allow recording of morning and evening milk entries per animal.
4. Automatically calculate total milk per day.
5. Provide a simple reporting interface for daily totals and averages.
6.Build the project using beginner-friendly tools like Python and SQLite.

5. SCOPE OF THE PROJECT

This project includes:

>Farmer registration
>Animal registration under each farmer
>Recording milk production (morning + evening)
>Viewing all entries and reports
>Generating total milk produced on a given day
>A command-line interface (CLI)
>SQLite database for storage

Scope does NOT include:

>Mobile application
>Website / online version
>Authentication system
>Automated milk sensors

6. FUNCTIONAL REQUIREMENTS

6.1 Farmer Management

>Add farmer details such as name, phone and address.
>View list of all farmers.

6.2 Animal Management

>Add animals under a farmer.
>Each animal has: tag ID, name, breed, DOB, lactation start date.
>List animals for a specific farmer or all farmers.

6.3 Milk Entry Management

>Record daily milk entries (morning and evening).
>Auto-calculate total liters per entry.
>Optional notes.
>View list of all milk entries.

6.4 Reporting

>Show milk production for a specific date.
> Show total milk per animal.
>Calculate average milk per animal.

7. NON-FUNCTIONAL REQUIREMENTS

7.1 Usability
Simple text-based menu, easy for beginners.

7.2 Reliability
Uses SQLite database with foreign keys and unique tag IDs.

7.3 Maintainability
Code split into modules: db.py, models.py, cli.py.

7.4 Performance
Handles small datasets very fast.

7.5 Portability
Works on Windows, macOS and Linux with Python 3.

8. SYSTEM ARCHITECTURE

The project follows a three-layer structure:

8.1 Data Layer

SQLite database (milk.db) stores:

>farmers
>animals
>milk_entries

8.2 Business Logic Layer

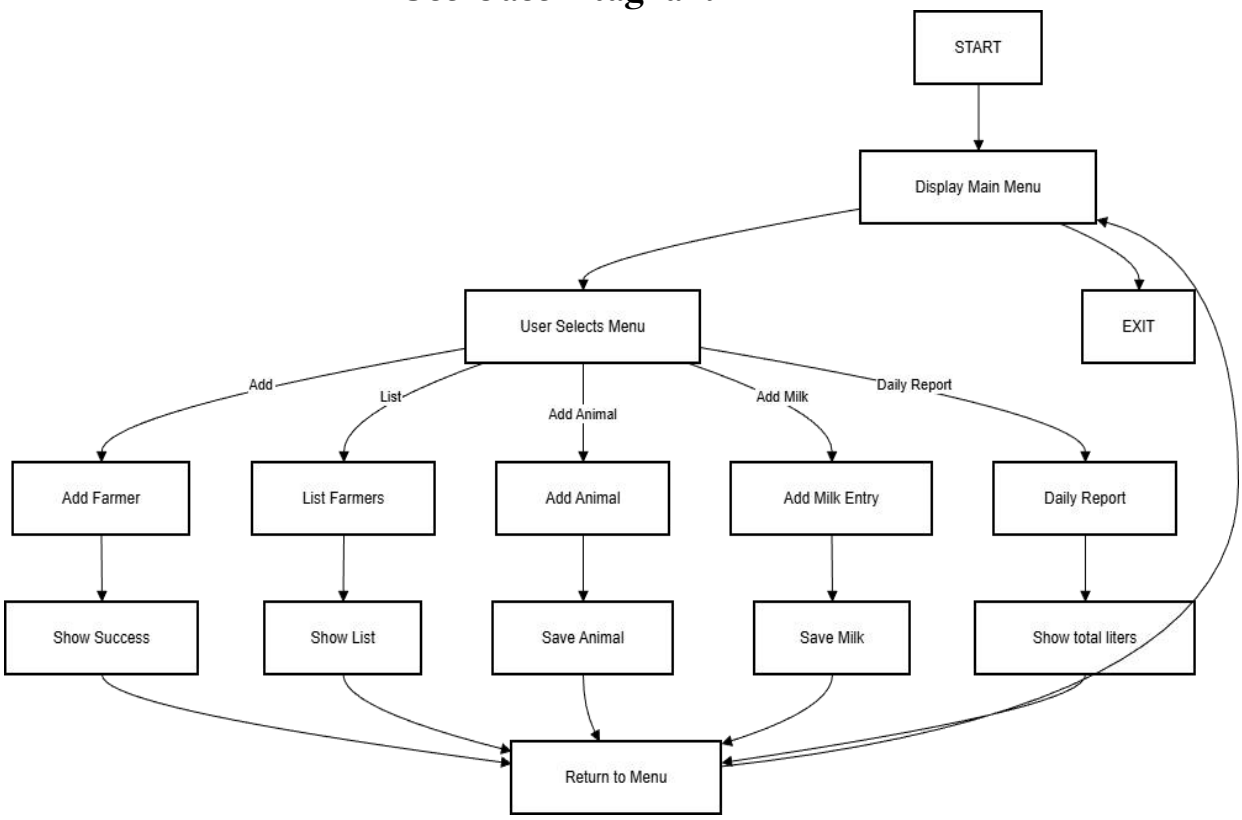
db.py handles DB connections.
models.py contains functions for CRUD operations and reports.

8.3 Presentation Layer

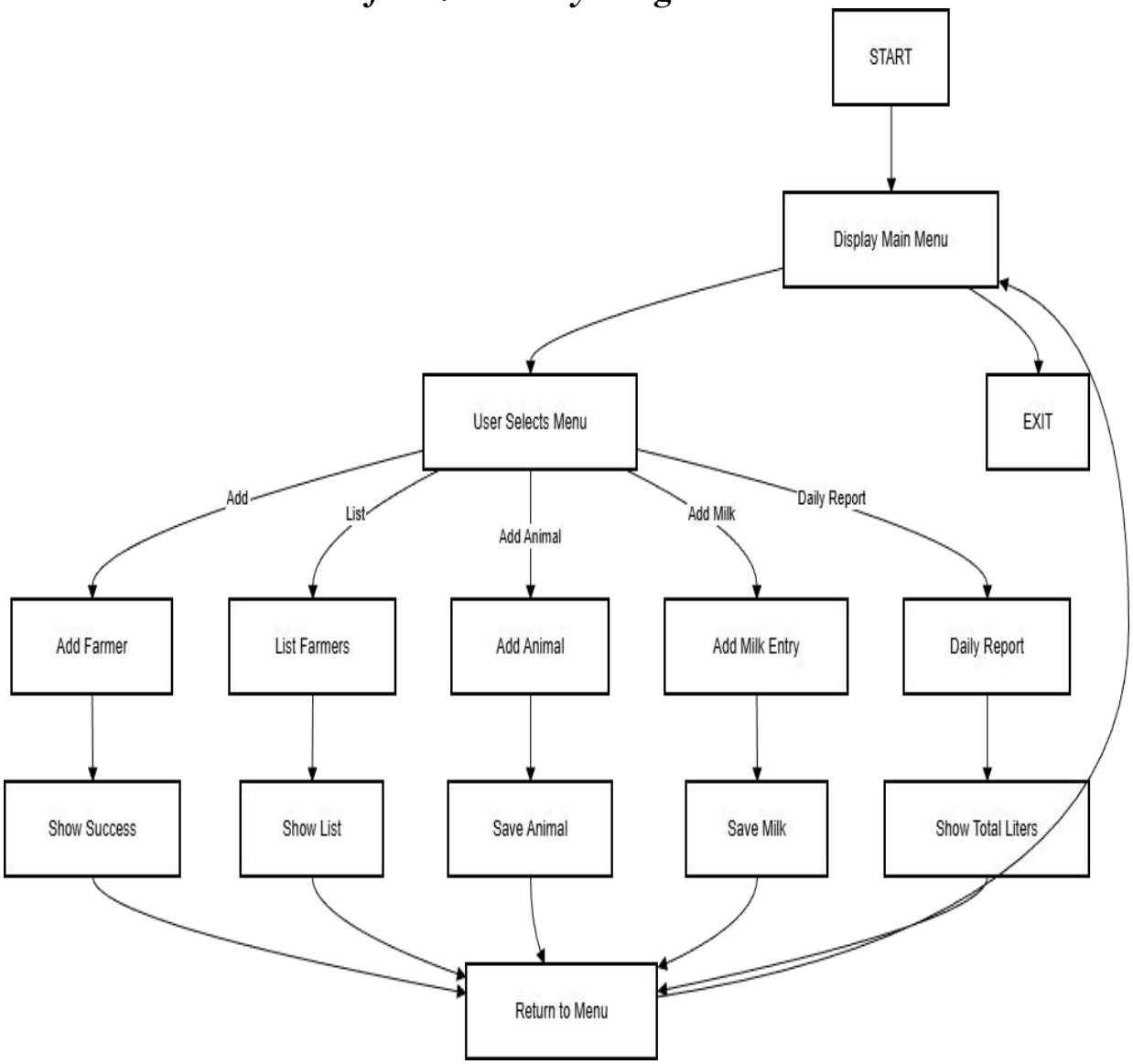
cli.py provides a text menu.
notebooks/ used for testing.

9. DESIGN DIAGRAMS

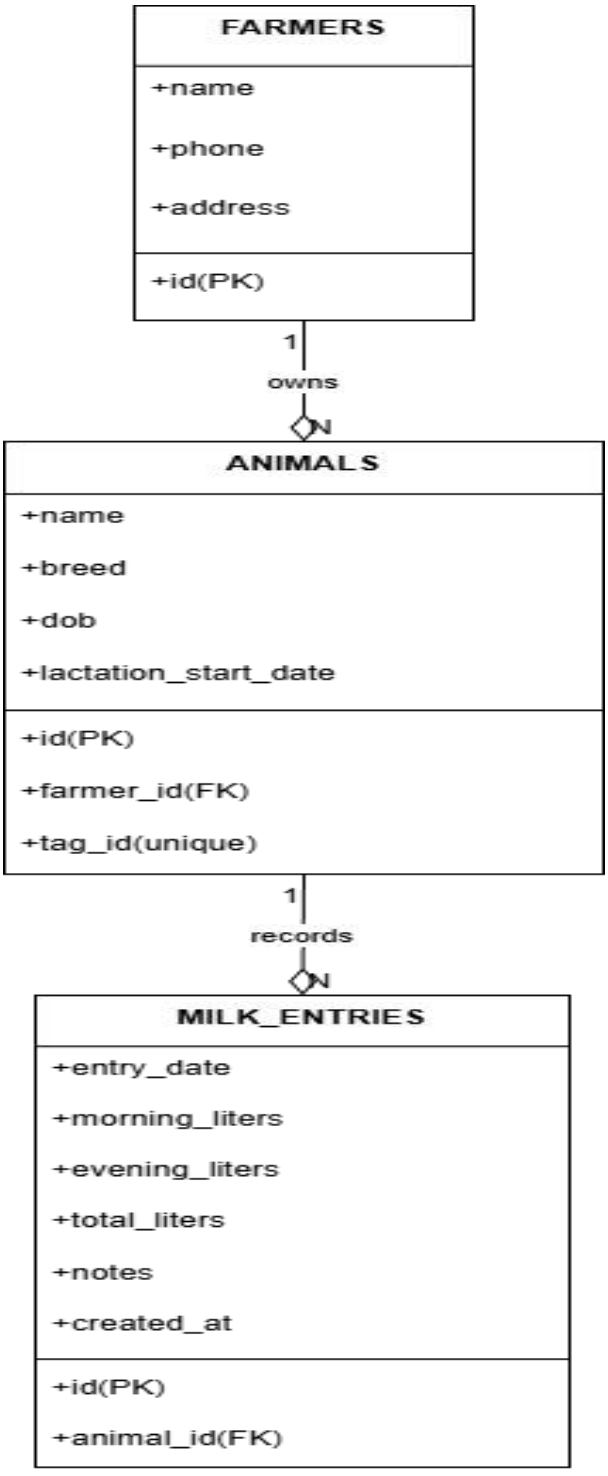
Use Case Diagram



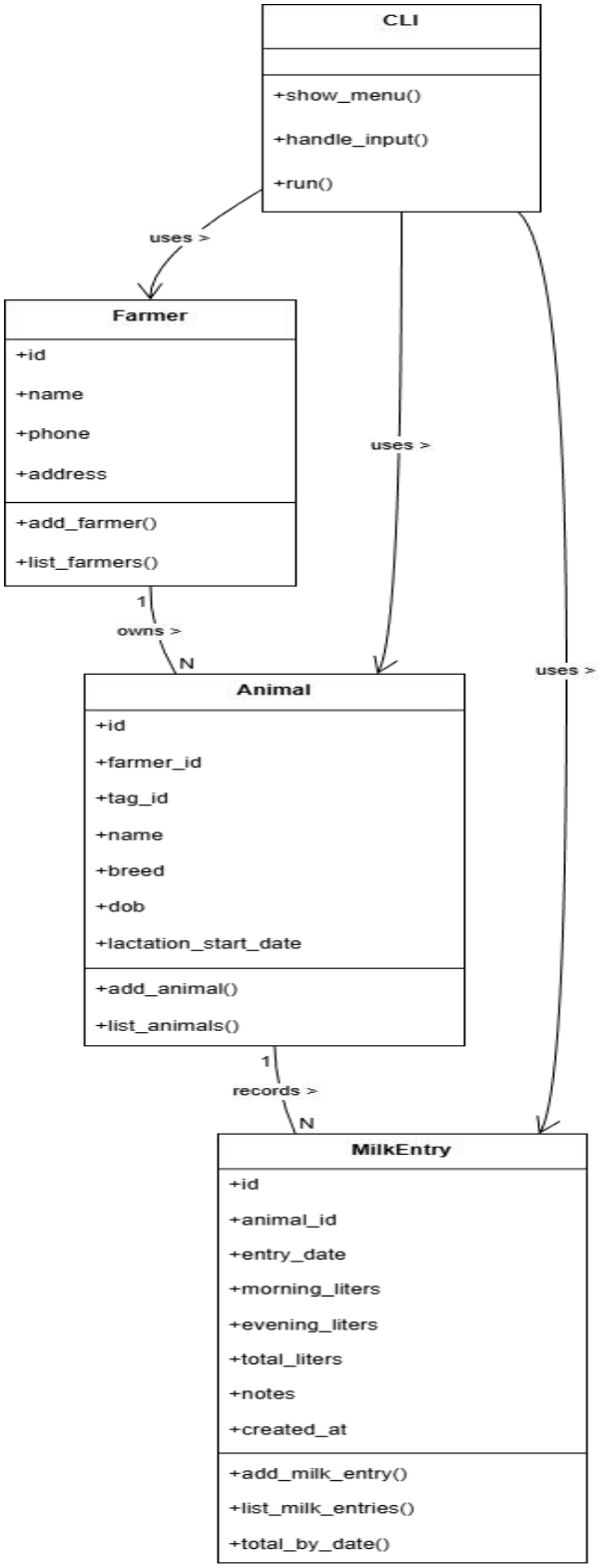
Workflow / Activity Diagram



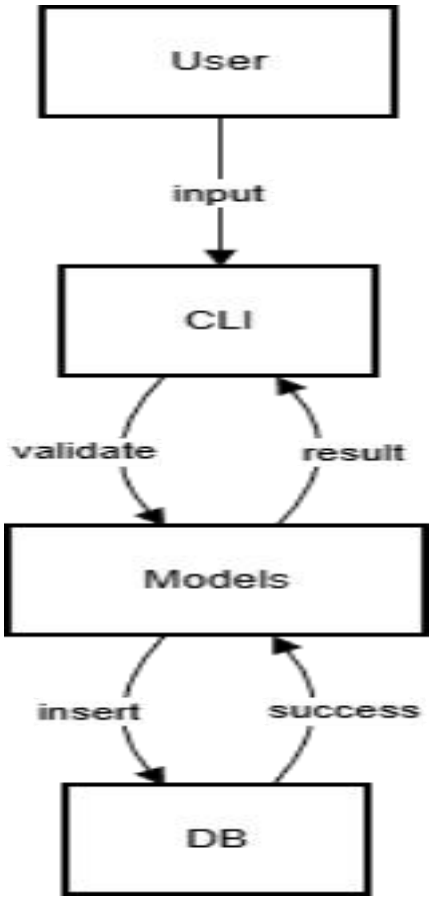
ER Diagram


FARMERS
+name
+phone
+address
+id(PK)
1
owns
N
ANIMALS
+name
+breed
+dob
+lactation_start_date
+id(PK)
+farmer_id(FK)
+tag_id(unique)
1
records
N
MILK_ENTRIES
+entry_date
+morning_liters
+evening_liters
+total_liters
+notes
+created_at
+id(PK)
+animal_id(FK)

Class Diagram



Sequence Diagram



10. DATABASE DESIGN

Detailed version:

```sql
User → CLI: Select "Add Milk Entry"
CLI → User: Ask for Animal ID, Date, Morning, Evening
User → CLI: Inputs values
CLI → Models: add_milk_entry(animal_id, date, morning, evening)
Models → DB: INSERT INTO milk_entries (...)
DB → Models: Success
Models → CLI: Return milk entry ID
CLI → User: Show "Milk Entry Added"
```

Tables:

Farmer

>id (PK)
>name
>phone
>address

Animal

>id (PK)
>farmer_id (FK)
>tag_id (Unique)
>name
>breed
>DOB
>lactation_start_date

Milk Entry

>id (PK)
>animal_id (FK)
>entry_date
>morning_liters
>evening_liters
>total_liters
>notes
>created_at

Relationships:

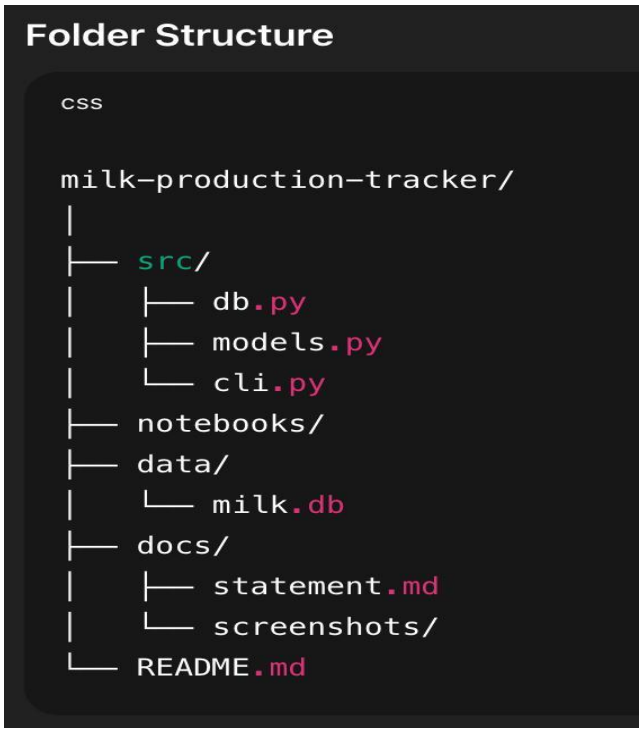
1 Farmer → many Animals
1 Animal → many Milk Entries

11. IMPLEMENTATION DETAILS

Languages & Tools Used

>Python 3
>SQLite
>Jupyter Notebook
>Terminal / CLI

>Git & GitHub



Key Files

>db.py: initializes database
>models.py: functions for farmers, animals, entries, reports
>cli.py: menu interface

12. TESTING APPROACH

Testing was performed in two phases:

12.1 Jupyter Notebook

>Added farmer → verified list
>Added animal → checked foreign key
>Added milk entry → checked total liters
>Tested reporting functions

12.2 CLI Testing

>End-to-end tests through the menu
>Tried valid and invalid inputs
>Confirmed database updates

13. SCREENSHOTS / RESULTS

CLI main menu

```
Last login: Sun Nov 23 12:27:33 on ttys000
narodeyuvraj113gmail.com@Yuvrajs-MacBook-Air ~ % cd ~/Desktop/milk-tracker
python3 src/cli.py

==============================
   MILK PRODUCTION TRACKER
==============================
1. Add Farmer
2. List Farmers
3. Add Animal
4. List Animals
5. Add Milk Entry
6. List Milk Entries
7. Daily Milk Report
8. Exit
Enter choice (1-8): 1
```

Adding farmer

```
==============================
   MILK PRODUCTION TRACKER
==============================
1. Add Farmer
2. List Farmers
3. Add Animal
4. List Animals
5. Add Milk Entry
6. List Milk Entries
7. Daily Milk Report
8. Exit
Enter choice (1-8): 2

--- Farmers ---
[1] Ramesh Patil | Phone: 9876543210 | Address: Village A
[2] Test Farmer | Phone: 9999999999 | Address: Village A
[3] Test Farmer | Phone: 9999999999 | Address: Village A
[4] Yuvraj Narode | Phone: 9216772670 | Address: Zodegaon
```

Adding animal

```
==============================
   MILK PRODUCTION TRACKER
==============================
1. Add Farmer
2. List Farmers
3. Add Animal
4. List Animals
5. Add Milk Entry
6. List Milk Entries
7. Daily Milk Report
8. Exit
Enter choice (1-8): 3

--- Add Animal ---
Farmer ID (owner): 1
Tag ID (unique): COW001
Animal name (optional): Gauri
DOB (YYYY-MM-DD, optional): 2020-01-01
Breed (optional): Jersey
Lactation start date (YYYY-MM-DD, optional): 2024-01-01
✅ Animal added with ID: 2
```

Daily report output

```
==============================
   MILK PRODUCTION TRACKER
==============================
1. Add Farmer
2. List Farmers
3. Add Animal
4. List Animals
5. Add Milk Entry
6. List Milk Entries
7. Daily Milk Report
8. Exit
Enter choice (1-8): 7

--- Daily Milk Report ---
Date (YYYY-MM-DD): 2024-11-23
Total milk on 2024-11-23: 9.5 liters
Average milk per animal (all dates): 18.50 liters
```

Milk Enteries

```
============================
   MILK PRODUCTION TRACKER
============================
1. Add Farmer
2. List Farmers
3. Add Animal
4. List Animals
5. Add Milk Entry
6. List Milk Entries
7. Daily Milk Report
8. Exit
Enter choice (1-8): 6

--- Milk Entries ---
[2] Animal 1 | Date: 2024-11-23 | Morning: 5.0 L | Evening: 4.5 L | Total: 9.5 L | Notes: Healthy Cow
[1] Animal 1 | Date: 2024-11-21 | Morning: 5.0 L | Evening: 4.0 L | Total: 9.0 L | Notes: Healthy cow
```

Exit

```
==============================
   MILK PRODUCTION TRACKER
==============================
1. Add Farmer
2. List Farmers
3. Add Animal
4. List Animals
5. Add Milk Entry
6. List Milk Entries
7. Daily Milk Report
8. Exit
Enter choice (1-8): 8
Exiting. Bye! 👋
narodeyuvraj113gmail.com@Yuvrajs-MacBook-Air milk-tracker %
```

Jupyter notebook tests

```
if src_path not in sys.path:
    sys.path.insert(0, src_path)

print("\nTop 3 entries in sys.path:")
for p in sys.path[:3]:
    print("  ", p)

# 3) Try importing models and the functions
import models
print("\nmodels imported from:", models.__file__)

from models import (
    add_farmer,
    list_farmers,
    add_animal,
    list_animals,
    add_milk_entry,
    list_milk_entries,
    total_milk_by_date,
    total_milk_by_animal,
    average_milk_per_animal,
)

print("\n✅ Imports successful.")
```

```
Current working dir: /Users/narodeyuvraj113gmail.com/Desktop/milk-tracker/notebooks
Project path: /Users/narodeyuvraj113gmail.com/Desktop/milk-tracker
src_path: /Users/narodeyuvraj113gmail.com/Desktop/milk-tracker/src
models.py exists? True

Top 3 entries in sys.path:
   /Users/narodeyuvraj113gmail.com/Desktop/milk-tracker/src
   /Library/Frameworks/Python.framework/Versions/3.14/lib/python314.zip
   /Library/Frameworks/Python.framework/Versions/3.14/lib/python3.14

models imported from: /Users/narodeyuvraj113gmail.com/Desktop/milk-tracker/src/models.py

✅ Imports successful.
```

---

[2]:
```
fid = add_farmer("Test Farmer", "9999999999", "Village A")
aid = add_animal(fid, "TST001", "Gauri", "2020-01-01", "Jersey")

print("Farmer ID:", fid)
print("Animal ID:", aid)
```

```
Farmer ID: 3
Animal ID: 1
```

[3]:
```
entry_id = add_milk_entry(
    animal_id=aid,
    entry_date="2024-11-21",
    morning_liters=5,
    evening_liters=4,
    notes="Healthy cow"
)
print("Milk entry ID:", entry_id)
```

```
Milk entry ID: 1
```

[4]:
```
for row in list_milk_entries():
    print(dict(row))

print("Milk on 2024-11-21:", total_milk_by_date("2024-11-21"))
print("Total by Animal", aid, ":", total_milk_by_animal(aid))
print("Average per animal:", average_milk_per_animal())
```

```
{'id': 1, 'animal_id': 1, 'entry_date': '2024-11-21', 'morning_liters': 5.0, 'evening_liters': 4.0, 'total_liters': 9.0, 'notes': 'Healthy c
ow', 'created_at': '2025-11-23 06:26:41'}
Milk on 2024-11-21: 9.0
Total by Animal 1 : 9.0
Average per animal: 9.0
```

---

14. CHALLENGES FACED

>Import path errors in Jupyter
>Understanding SQLite constraints
>Learning command line basics on macOS
>Managing multiple Python files
>Learning GitHub version control
>Time management near submission

15. LEARNINGS

>Structuring Python projects
>Using SQLite databases
>Writing modular code
>Testing using Jupyter notebooks
>Building a CLI
>Using Git and GitHub
>Creating diagrams and documentation

16. FUTURE ENHANCEMENTS

>Add a graphical UI (desktop/web)
>Add login system for multiple users
>Add monthly/yearly summary graphs
>Export data to Excel/CSV
>Store data in cloud database
>Support local languages (Marathi)

17. CONCLUSION

The Milk Production Tracker successfully solves the problem of maintaining daily milk records in a structured way. It is simple, efficient and useful for small dairy farms. The project demonstrates database design, Python modules, CLI development and proper documentation. It can be extended further into a complete dairy management system.