PROJECT REPORT

ON

AgriPy

REPORT SUBMITTED

TO

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

FOR THE PBL OF **PYTHON FOR ENGINEERS**

IN

Department of Electronics and Telecommunication

BY

Pragati Shidarayagouda Patil - 101147
Atharva Rajendra Joshi - 101149
Revati Tushar Aute - 101153
Manas Girish Kulkarni – 101158
Atharva Vishwas Deshpande - 101161

Class: FY **Division: A** **Batch: A3**

Batch Teacher: Ms. Rupa Kawchale Sonagi

INDEX

ABSTRACT

AgriPy is a Python-based project that aims at transforming agriculture by incorporating Computer Science and technology. It covers modules like Water Management, Automation for agriculture, Predict Crop yields, Logistics and Supply Chains, Plant Disease Detection, Crop Simulation and Livestock Management. Each module plays a crucial role in providing farmers with valuable insights and tools. Programming concepts like Tkinter (GUI framework), OpenWeatherMap API, Data Analysis with NumPy and Pandas, Linear Regression Model, File Handling (JSON), Image Processing with OpenCV and PIL, Datetime Module, Styling and Theming have been used. This report includes the detailed theory of every code.

INTRODUCTION

Welcome to AgriPy, a Python project designed to revolutionize agriculture by integrating cutting-edge technologies into farming practices. Our project encompasses Crop Simulation, Plant Disease Detection, Livestock Management, Logistics and Supply Chain, Soil Monitoring, Automation for Water Management, and Crop Yield Prediction.
In the vast expanse of AgriPy, each module plays a pivotal role in empowering farmers with advanced tools and insights. It provides a virtual environment where crops thrive and simulate growth patterns, enabling farmers to make informed decisions for optimal yield.

Automation for Water Management takes the guesswork out of irrigation, ensuring that crops receive the right amount of water at the right time.

Our Crop Yield Prediction module is able to forecast future harvests, empowering farmers with valuable insights for planning and resource allocation.

The Logistics and Supply Chain module produces a bill and streamlines the agricultural supply chain, ensuring timely delivery of produce from farm to market.

Our Plant Disease Detection feature utilizes image processing techniques to identify and address potential threats to crop health.

Livestock Management ensures that every aspect of animal well-being is tracked and optimized, fostering healthier and more productive livestock.

AgriPy isn't just a project; it's a vision for a sustainable and technologically advanced future in agriculture. Join us on this journey as we leverage the power of Python to transform the way we cultivate, nurture, and harvest. Together, let's cultivate a smarter, more efficient and more prosperous agricultural landscape with AgriPy.

THEORY

Water Management

Farmers in India currently use manual methods to control crop irrigation, manually turning water flow on and off. This code offers a solution to automate the process, providing a more efficient and convenient approach for farmers.

1. Water Flow Simulation:
- The program simulates water flow for crop irrigation. It prompts the user to input the amount of water in liters. If the water level is below 70 liters, it initiates the water flow to the crops and displays the amount of water to be released. Otherwise, it indicates that the water level is sufficient.

2. Time-Related Functions:
- The application includes functions related to time. It allows the user to check the current time and set a specific time. If the current time matches the user-set time, it suggests turning off the water flow; otherwise, it recommends keeping the flow on. The time-related functions use the `datetime` module for time calculations.

3. Tank Filling Time Calculation:
- The program calculates the time required to fill a water tank based on user inputs such as the tank's capacity and the amount of water already filled. It uses a constant rate of flow (assumed to be 100 liters per hour) to estimate the time left to fill the tank. The result includes the time left and the expected time when the tank will be full.

4. Graphical User Interface (GUI):
- The code utilizes the `tkinter` library to create a graphical user interface. It includes input fields for water amount, time, tank capacity, and water filled. Buttons trigger specific functions, and the results are displayed on a label. The GUI has a dark theme with green font for a visually appealing design.

5. Styling and Theming:
- The GUI elements are styled using a consistent font (Helvetica, bold) and color scheme. The background is set to a dark color (#1C1C1C), and the font color is a distinctive green shade (#4FA687), providing a visually cohesive and aesthetically pleasing appearance to the application.

Automation for Agriculture

Farmers often face challenges in determining the optimal watering schedule for their crops. Our code addresses this issue by utilizing the OpenWeatherMap API to extract soil moisture information of a particular region. The code then provides clear indications, stating whether the soil is dry, moist, or wet. This information serves as valuable guidance for farmers, helping them make informed decisions on whether watering their crops is necessary.

1. API Integration:
- The code integrates with the OpenWeatherMap API to fetch current weather data for a specific city.
- An API key is used for authentication and access to weather information.

2. GUI Design with Tkinter:
- The script utilizes the Tkinter library to create a graphical user interface (GUI) for the application.
- The GUI includes an entry field for the user to input the city name and a button to trigger the weather and moisture level retrieval.

3. Weather Data Retrieval:
- The `get_weather` function is executed when the user clicks the "Get Weather and Moisture Level" button.
- It sends a request to the OpenWeatherMap API, retrieves the JSON response, and extracts relevant weather information such as main weather

conditions, temperature, and humidity.

4. Moisture Level Calculation:
- The code calculates the moisture level based on the humidity data obtained from the weather API response.
- Different messages are displayed in the result label to indicate whether the soil is dry, moist, or wet, providing guidance on watering plants accordingly.

5. User Interface Feedback:
- The GUI presents a user-friendly interface, allowing users to input a city, fetch weather information, and receive feedback on both weather conditions and soil moisture levels.
- The result label dynamically updates to display the fetched data and moisture level information. The overall design enhances user interaction and understanding.

Predict Crop Yields

Farmers often face challenges in predicting their crop yields. Our code addresses this issue by leveraging rainfall data obtained from the OpenWeatherMap API to predict crop yields. This prediction serves as a valuable tool for farmers, enabling them to plan their finances proactively based on anticipated agricultural outcomes.

1. Weather and Crop Yield Prediction: The code allows users to input a city and the corresponding rainfall (in mm). It utilizes the OpenWeatherMap API to fetch current weather details for the specified city, including temperature, humidity & Rainfall.

2. Linear Regression Model: The program employs a simple linear regression model to predict crop yield based on rainfall. It uses pre-defined example data for rainfall, temperature, and crop yield to calculate the slope (m) and intercept (c) for the linear regression equation.

3. Prediction: The rainfall is then used to predict the crop yield by applying the linear regression equation. The predicted crop yield is displayed as the result.

4. Graphical User Interface (GUI): The code utilizes the Tkinter library to create a graphical user interface (GUI). Users can enter their city through entry widgets, and the result is displayed on the GUI.

5. Visual Representation: The code provides a practical example of using weather data (rainfall) to make predictions about agricultural outcomes. It showcases the potential applications of data-driven decision-making in agriculture, aiding farmers in anticipating crop yields based on weather conditions.

Logistics & Supply Chain

Keeping track of crop inventory poses challenges for farmers. Our code addresses this issue by allowing farmers to efficiently place bulk orders for crops, and it instantly generates a detailed bill for their convenience.

1. Order Processing Interface: The code creates a graphical user interface (GUI) using the Tkinter library to facilitate the ordering process for agricultural crops.

2. User Input and Validation: The interface prompts the user to enter their name, contact number, choice of crop (from a predefined list), the desired quantity in quintals, and the final destination. The code performs input validation to ensure that the entered data is within acceptable ranges.

3. Bill Calculation: Upon clicking the "Display the bill" button, the code calculates and opens a new window displaying order details. It includes the current date and time, customer information, selected crop details (name and price per quintal), the ordered quantity, remaining quantity, and the total cost of the order.

4. Choice and Crop Information: The predefined crop options include tomatoes, potatoes, lemons, cucumbers, and carrots. The user's choice is used to fetch the corresponding crop's name and price for further processing.

5. Order Limitation Notice: The code provides information in the interface, specifying that users can order a maximum of 100 quintals of any crop. This limitation is intended to guide users and prevent orders beyond the specified quantity.

Plant Disease Detection

Farmers often struggle to identify whether their crops are diseased or healthy. Our code addresses this issue by providing a user-friendly plant disease detection application. Farmers can conveniently capture real-time images of their crops using their mobile's camera through our intuitive graphical interface. The application analyzes the images, assessing the average intensity to determine if a disease is present. The result is displayed on the interface, allowing farmers to quickly identify the health status of their crops, facilitating timely interventions and promoting better crop management practices.

1. GUI Setup:
- The code uses the tkinter library to create a graphical user interface (GUI) for a plant disease detection application.
- It includes "Open Image" and "Capture Image" buttons for loading images from files or capturing them using the default camera.

2. Image Analysis:

- The `PlantDiseaseDetectionApp` class contains methods for opening and capturing images, and it uses OpenCV and NumPy for image processing.
- The `analyze_image` method converts the image to grayscale and calculates the average intensity. If the intensity exceeds a threshold (200 in this case), it indicates a disease.

3. Result Display:
- The GUI displays the analysis result with a label, stating whether a disease is detected or not.

4. Image Display:
- The captured or opened image is displayed on the GUI using the `ImageTk` module from the Pillow (PIL) library.
- The `update_image_label` and `update_image_label_from_frame` methods handle the conversion and updating of images on the GUI.

5. Camera Integration:
- The code integrates the default camera using OpenCV, allowing users to capture real-time images for disease detection.
- The `update` method continuously updates the camera feed on the GUI, creating a live preview for users.

Crop Simulation

Botanists, also known as plant biologists, are people who investigate various aspects of plant life, whether in a controlled laboratory setting or within their native surroundings. The Crop Simulation code proves valuable to them, enabling the simulation of crop growth and accurate prediction of its height.

The code works in the following stages:

1. Weather Data Retrieval:
- The code includes a function `get_weather_data` that uses the OpenWeatherMap API to fetch current weather data based on the user-provided city name. The function checks for errors in the city name input and handles cases where the city is not found.

2. GUI for Weather Input:
- The main function `manas_kulkarni_crop_simulation` creates a GUI using the `tkinter` library. It prompts the user to enter a city name and fetches the weather data for that city when the "Fetch Weather" button is clicked. The weather information, including temperature, humidity, and rainfall, is displayed on the GUI.

3. Crop Simulation Logic:
- The code provides a crop simulation function `CropSimulation` that takes into account weather data and user-inputted values such as initial crop height, growth rate, and simulation duration (in days). It calculates the daily growth of the crop based on temperature, rainfall, and soil moisture factors using a mathematical model.

4. Input Validation and Error Handling:
- The program incorporates input validation to ensure that the user enters valid numerical values for parameters like initial crop height, growth rate, and simulation duration. It uses a try-except block to catch `ValueError` exceptions and displays an error message if invalid input is detected.

5. Multi-Window Interface:
- The GUI is designed with a multi-window interface. After fetching weather data, the user can proceed to the next window to input parameters for crop simulation. The simulation results are displayed in a separate window, providing a clear and organized user interface. The code uses various `tkinter` widgets such as labels, entry fields, buttons, and styles to create a visually appealing and user-friendly interface.

Livestock Management

Indian farmers possess extensive livestock, yet lack digital tools for effective management. Our code addresses this gap by providing a solution that allows farmers to digitally track crucial information about their livestock, such as names, health status, age, and more.

1. LivestockAnimal Class:
- The code defines a class `LivestockAnimal` to represent livestock animals with attributes such as name, breed, age, gender, weight, and health status. Instances of this class will store information about individual animals on a farm.

2. Graphical User Interface (GUI):
- The code uses the `tkinter` library to create a graphical user interface for livestock management. It includes functions for adding a new animal, listing all animals, updating health status, and removing an animal. The GUI provides a visually appealing and interactive way for users to manage livestock data.

3. File Handling and Data Persistence:
- The program includes functions for saving and loading data to/from a JSON file (`livestock_data.json`). The data is stored in JSON format, allowing for persistent storage of livestock information between program executions.

4. User Input Handling and Validation:
- The code utilizes functions like `get_user_input` to obtain user input in a structured way, ensuring a consistent interface. It includes error handling for invalid input, displaying error messages in case of incorrect options or input types.

5. Interactive Livestock Management Loop:
- The main function `main()` implements a loop that continuously prompts the user for actions such as adding, listing, updating, or removing livestock animals. After each operation, the program checks if the user wants to continue or exit. Invalid options or inputs are handled gracefully with informative error messages. The program runs until the user chooses to exit.

FLOW OF THE PROGRAM

Here's a basic flow outline for our AgriPy project:

1. Common Window/GUI:

- Display options: Water Management, Automation, Crop Yield Prediction, Logistics, Disease Detection, Crop Simulation, Livestock Management.
- User inputs the option number.

2. Option Selection Handling:

- Based on the user input:
  - If 1 (Water Management): Navigate to Water Management module.
  - If 2 (Automation for Agriculture): Navigate to Automation module.
  - If 3 (Predict crop yields): Navigate to Crop Yield Prediction module.
  - If 4 (Logistics and Supply Chain): Navigate to Logistics and Supply Chain module.
  - If 5 (Plant Disease Detection): Navigate to Plant Disease Detection module.
  - If 6 (Crop Simulation): Navigate to Crop Simulation module.
  - If 7 (Livestock Management): Navigate to Livestock Management module.

3. Water Management Module:

- Implement features related to water usage optimization, irrigation scheduling, etc.

4. Automation Module:

- Develop functionalities for automating tasks in agriculture, like watering the farm.

5. Crop Yield Prediction Module:

- Integrate algorithms using real-time weather API data for predicting crop yields.

6. Logistics Module:

- Implement features for supply chain optimization, including transportation and storage.

7. Disease Detection Module:
- Use image processing techniques to detect and diagnose plant diseases.

8. Crop Simulation Module:
- Create simulations for crop growth based on various parameters and conditions.

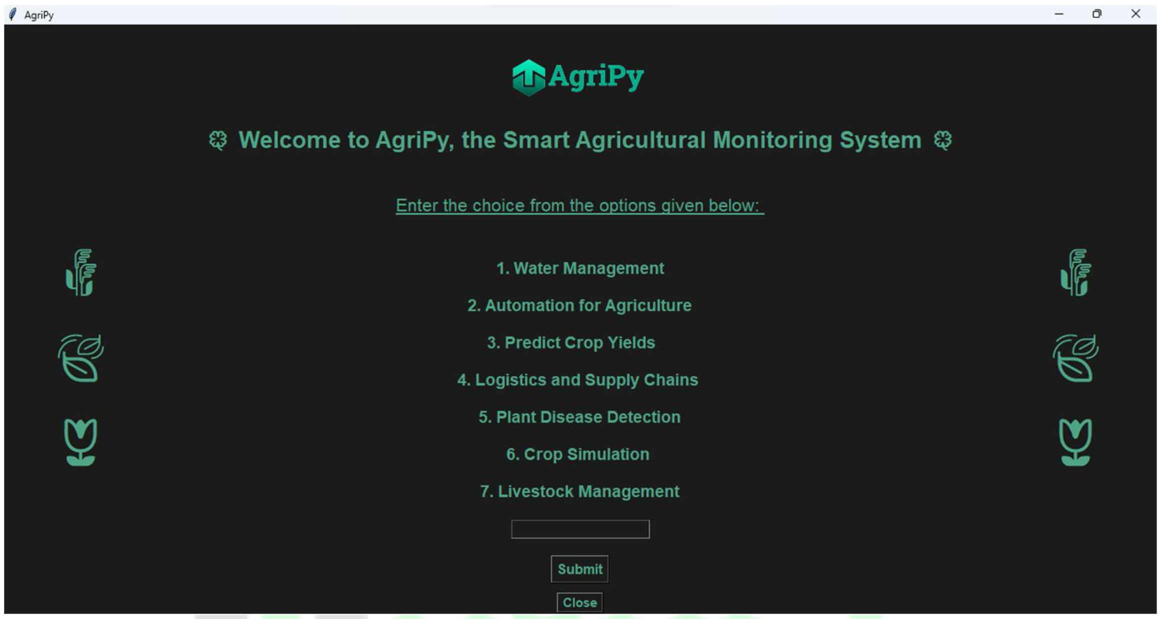
9. Livestock Management Module:
- Include features for monitoring and managing livestock health, breeding, and other relevant aspects.

This flow provides a structured approach for users to navigate through different aspects of AgriPy based on their specific needs in agriculture.

Source Code of the Program

Link- https://bit.ly/AgriPy

OUTPUT OF THE PROGRAM



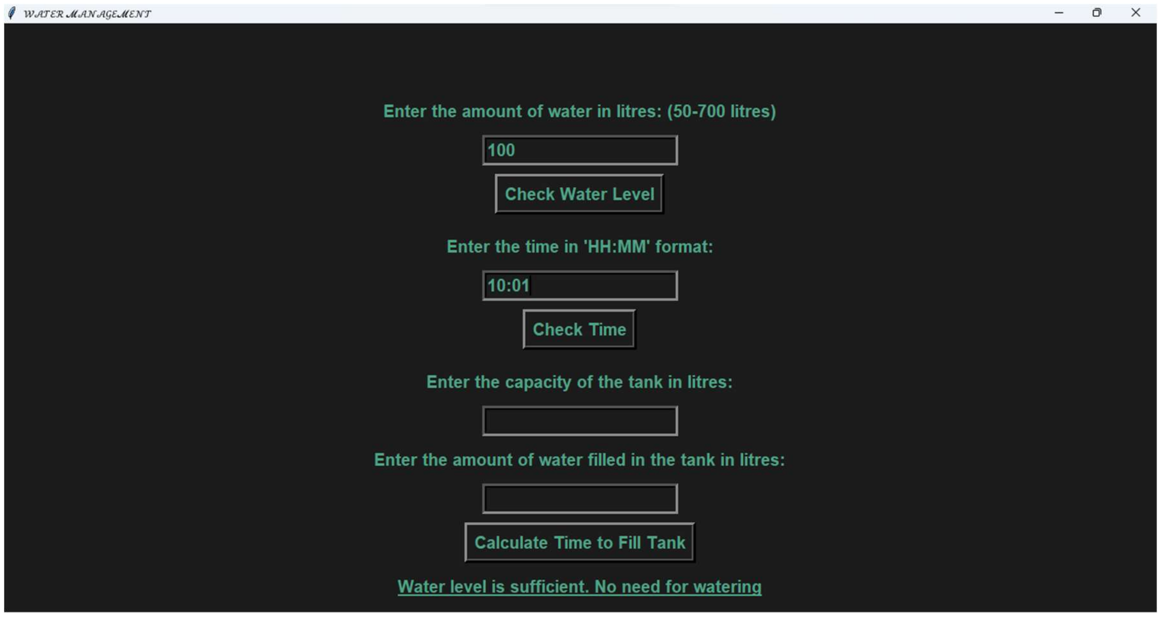
AgriPy

AgriPy

❁ Welcome to AgriPy, the Smart Agricultural Monitoring System ❁

Enter the choice from the options given below:

1. Water Management
2. Automation for Agriculture
3. Predict Crop Yields
4. Logistics and Supply Chains
5. Plant Disease Detection
6. Crop Simulation
7. Livestock Management



WATER MANAGEMENT

Enter the amount of water in litres: (50-700 litres)

100

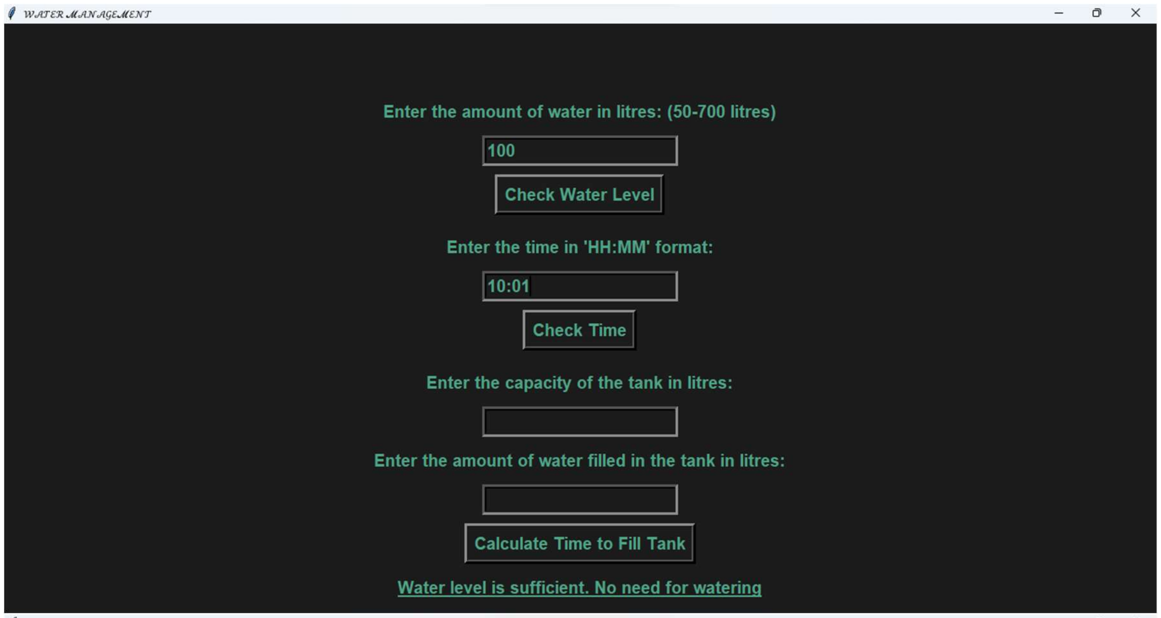
Enter the time in 'HH:MM' format:

10:01

Enter the capacity of the tank in litres:

Enter the amount of water filled in the tank in litres:

Water level is sufficient. No need for watering



WATER MANAGEMENT

Enter the amount of water in litres: (50-700 litres)

100

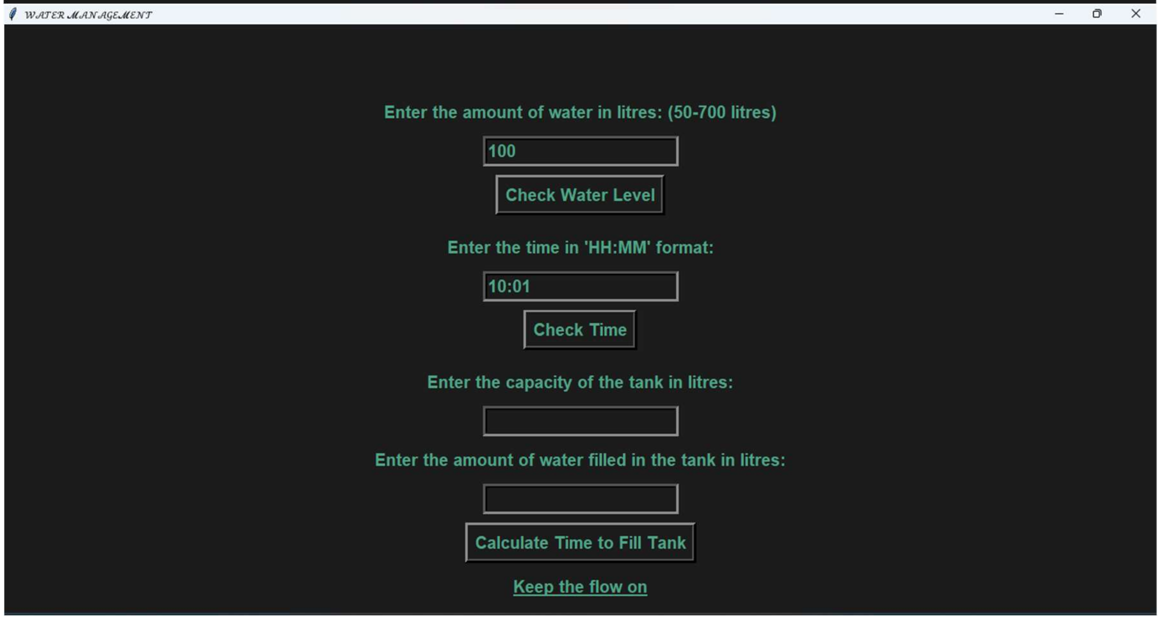
Enter the time in 'HH:MM' format:

10:01

Enter the capacity of the tank in litres:

Enter the amount of water filled in the tank in litres:

Water level is sufficient. No need for watering



WATER MANAGEMENT

Enter the amount of water in litres: (50-700 litres)

100

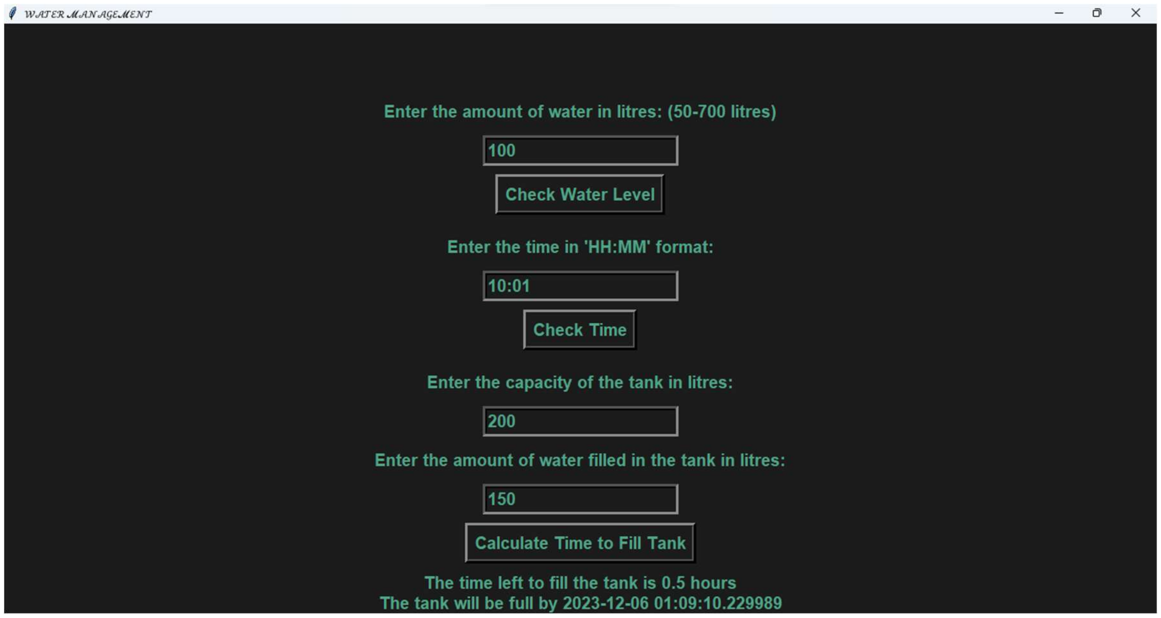
Enter the time in 'HH:MM' format:

10:01

Enter the capacity of the tank in litres:

Enter the amount of water filled in the tank in litres:

Keep the flow on



Enter the amount of water in litres: (50-700 litres)

100

Enter the time in 'HH:MM' format:

10:01

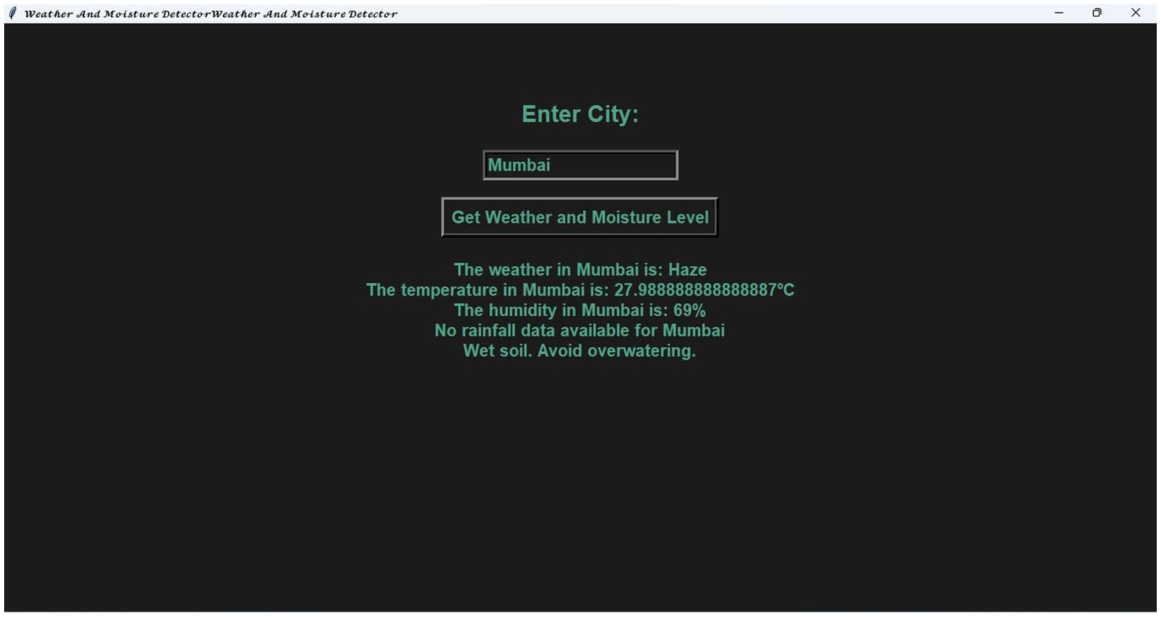
Enter the capacity of the tank in litres:

200

Enter the amount of water filled in the tank in litres:

150

```
The time left to fill the tank is 0.5 hours
The tank will be full by 2023-12-06 01:09:10.229989
```



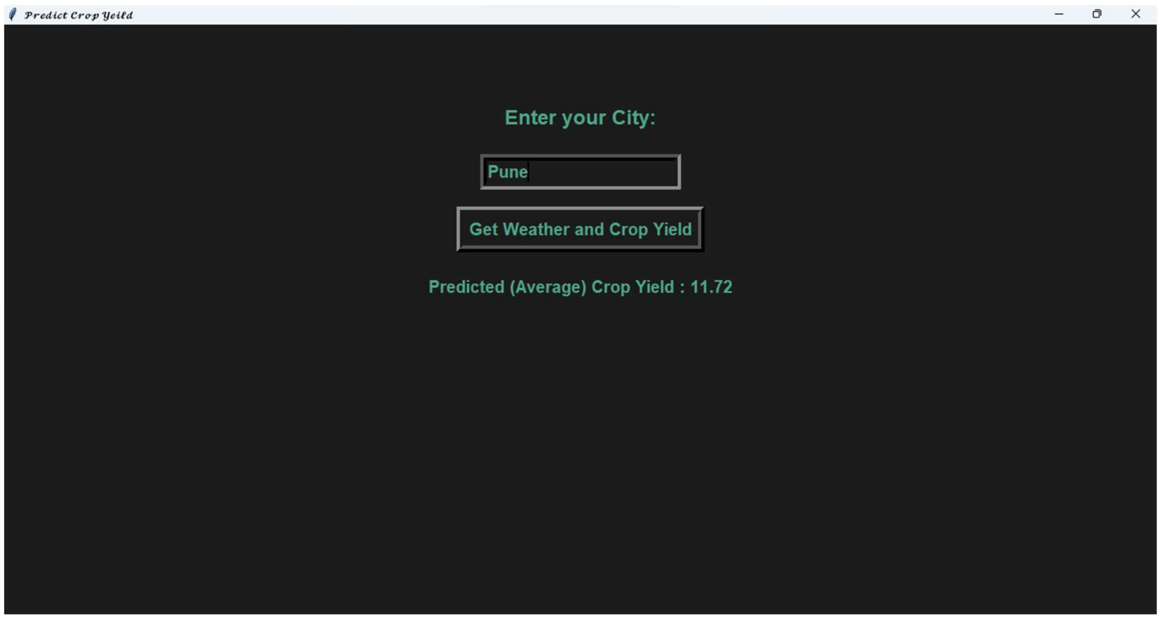
Weather And Moisture DetectorWeather And Moisture Detector

Enter City:

Mumbai

Get Weather and Moisture Level

```
The weather in Mumbai is: Haze
The temperature in Mumbai is: 27.988888888888887°C
The humidity in Mumbai is: 69%
No rainfall data available for Mumbai
Wet soil. Avoid overwatering.
```

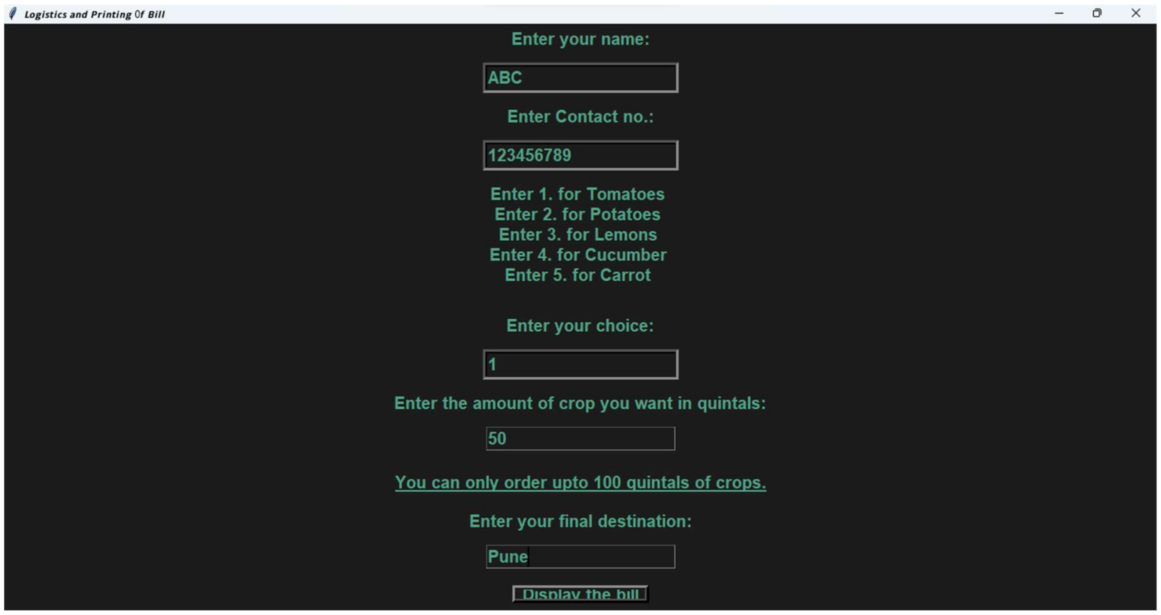

Predict Crop Yeild

Enter your City:

Pune

Get Weather and Crop Yield

Predicted (Average) Crop Yield : 11.72


Logistics and Printing Of Bill

Enter your name:

ABC

Enter Contact no.:

123456789

Enter 1. for Tomatoes
Enter 2. for Potatoes
Enter 3. for Lemons
Enter 4. for Cucumber
Enter 5. for Carrot

Enter your choice:

1

Enter the amount of crop you want in quintals:

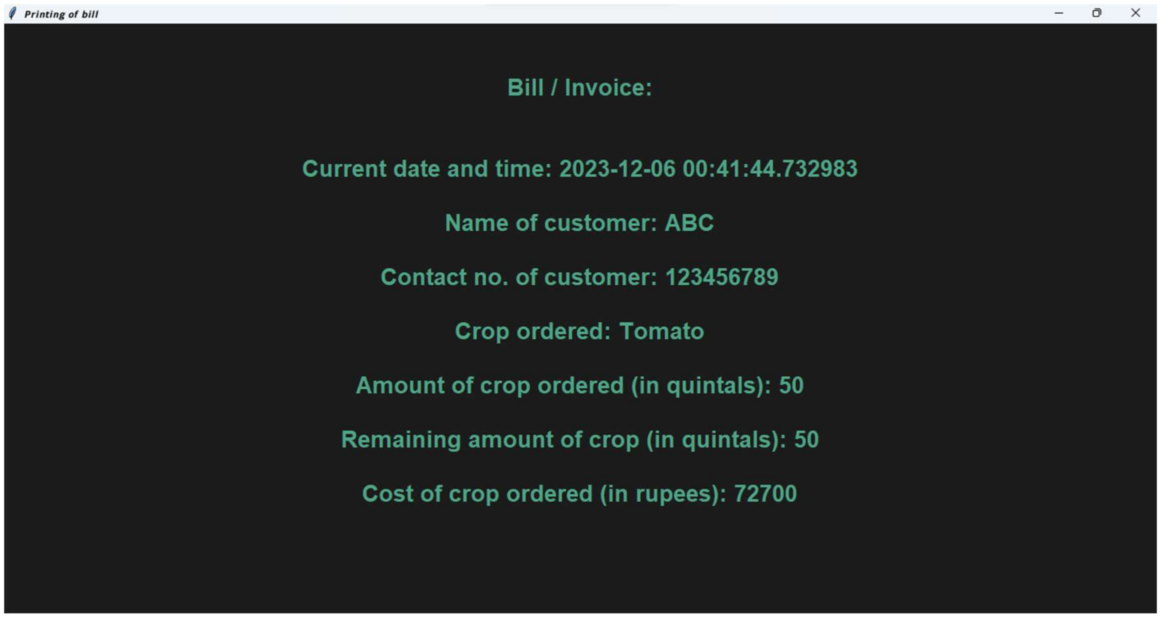
50

You can only order upto 100 quintals of crops.

Enter your final destination:

Pune

Display the bill



Bill / Invoice:

Current date and time: 2023-12-06 00:41:44.732983

Name of customer: ABC

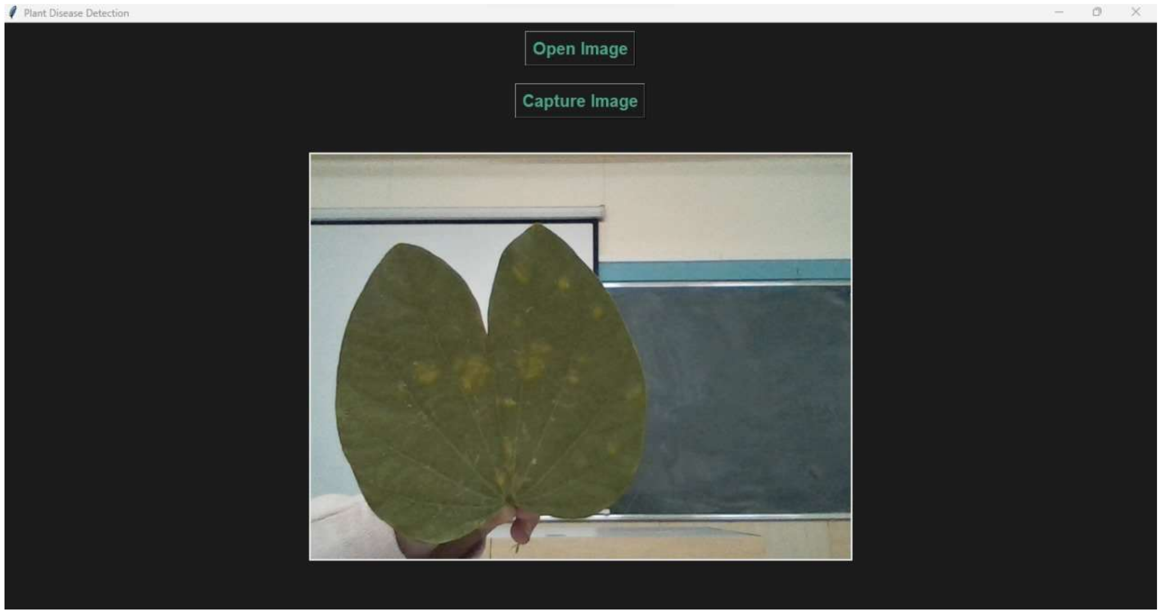
Contact no. of customer: 123456789

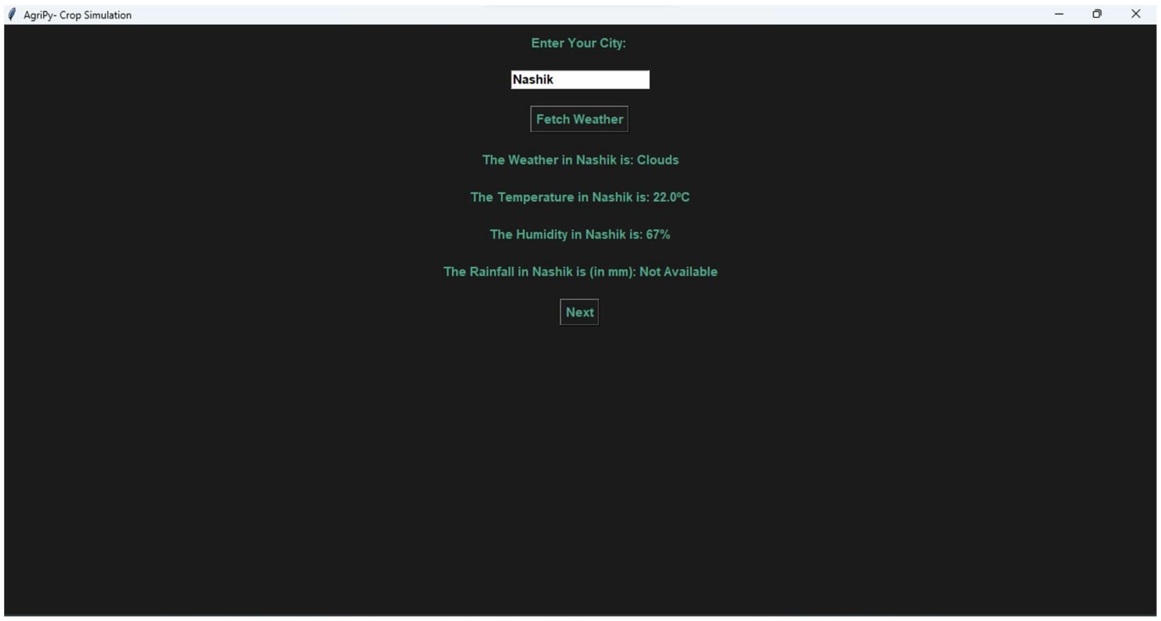
Crop ordered: Tomato

Amount of crop ordered (in quintals): 50

Remaining amount of crop (in quintals): 50

Cost of crop ordered (in rupees): 72700




AgriPy- Crop Simulation

Enter Your City:

Nashik

Fetch Weather

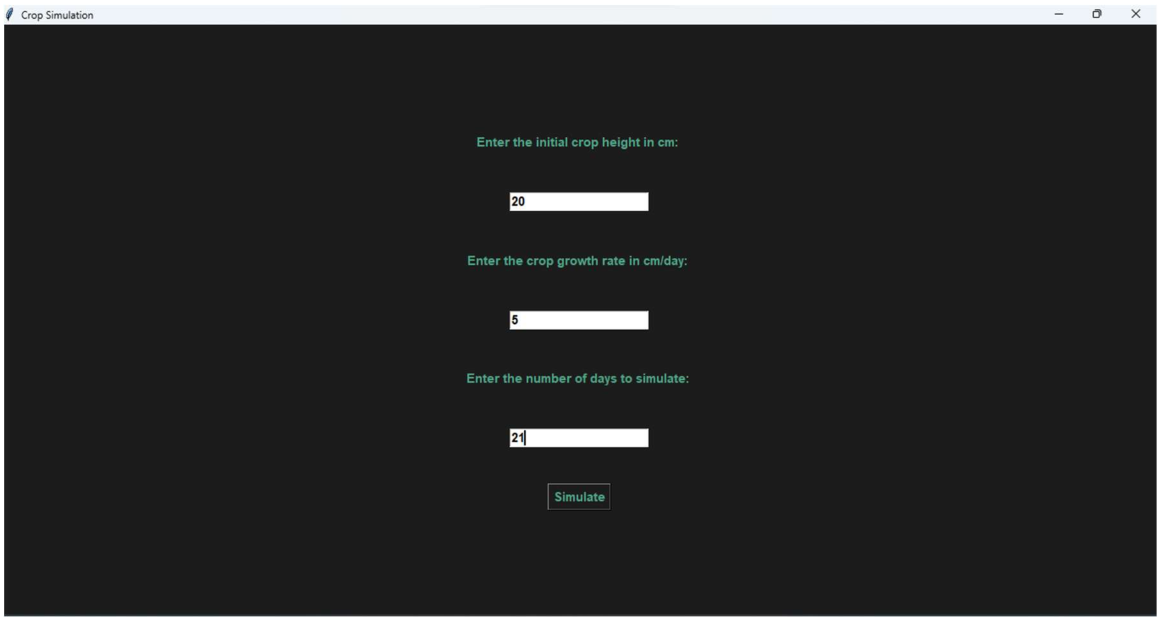
The Weather in Nashik is: Clouds

The Temperature in Nashik is: 22.0ºC

The Humidity in Nashik is: 67%

The Rainfall in Nashik is (in mm): Not Available

Next


Crop Simulation

Enter the initial crop height in cm:

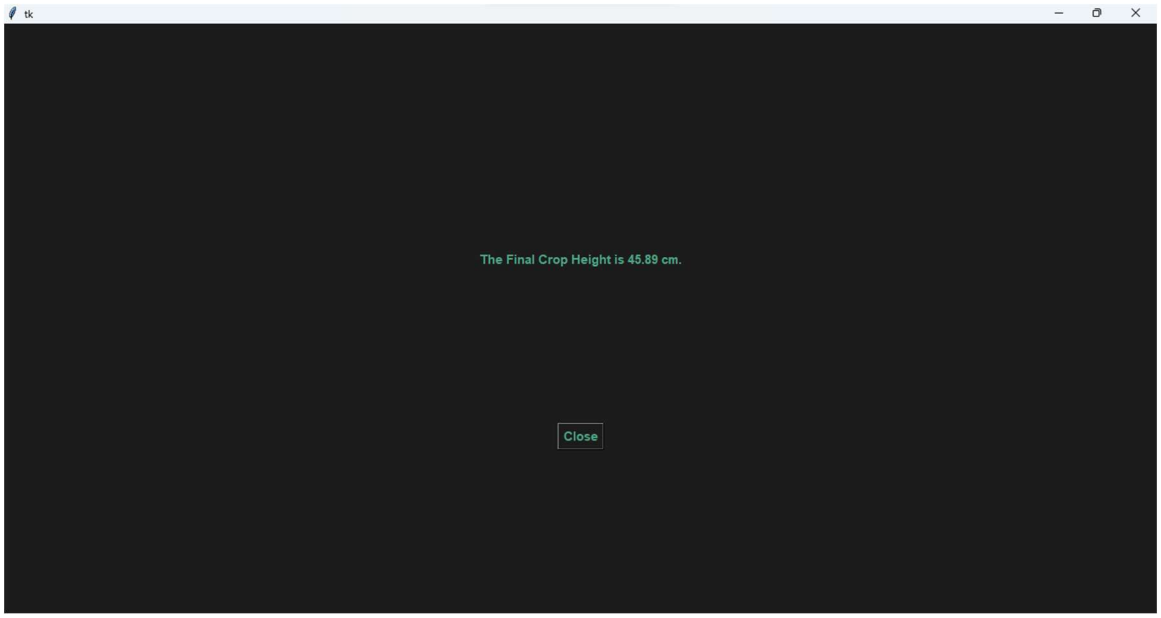
20

Enter the crop growth rate in cm/day:

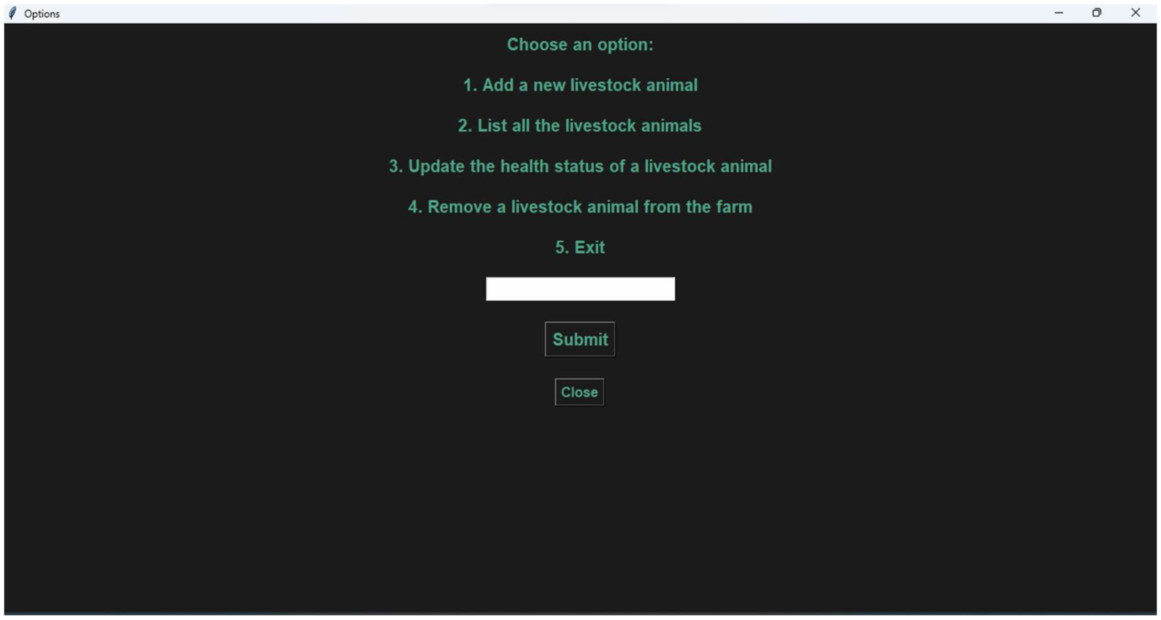
5

Enter the number of days to simulate:

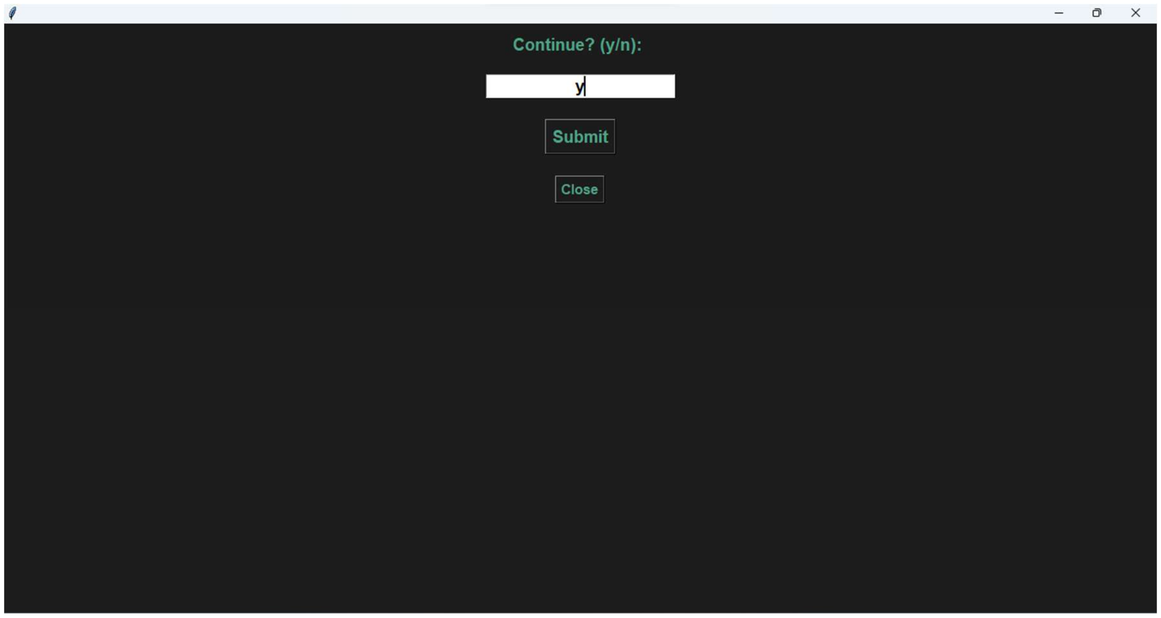
21



```
The Final Crop Height is 45.89 cm.
```



```
Choose an option:

1. Add a new livestock animal

2. List all the livestock animals

3. Update the health status of a livestock animal

4. Remove a livestock animal from the farm

5. Exit
```


Continue? (y/n):
y
Submit
Close

LIST OF THE TOPICS

- **Water Management & Automation for agriculture:**

Concepts within syllabus:

1) Loops

2)If-else statements

3) User-defined function

4) Function calling

Concepts out of syllabus:

1) Tkinter

2) Datetime library

3) weather API

- **Logistics & Supply Chain**

Concepts within syllabus:

1) Loops

2)If-else statements

3) User-defined function

4) Function calling

Concepts out of syllabus:

1) Tkinter

2) Datetime library

- **Plant Disease Detection:**

Concepts within the syllabus:

1) If-else statements

2) User-defined functions

3) Function calling

Concepts out of syllabus:

1)tkinter

2)cv2 library

3)Class

4)image processing

5)file handling

- **Crop Simulation**

Concepts within the syllabus:

1) If-else statements

2) User-defined functions

3) Function calling

4) math Library

Concepts out of syllabus:

1) tkinter

2) ttk

3) messagebox

4) *File Handling*

5) Weather API Handling

6) requests Library

7) Exception handling using try & except

- **Livestock Management**

*Concepts within the syllabu*s:

1) If-else statements

2) User-defined functions

3) Function calling

Concepts out of syllabus:

1) tkinter

2) messagebox

3) simpeldialog

4) Class

5) Constructors

6) File Handling

7) Exception handling using try & except

8)json

REFERENCES

1) Python playlist: https://www.youtube.com/watch?v=7wnove7K-ZQ&list=PLu0W_9lII9agwh1XjRt242xIpHhPT2llg

2) Tkinter tutorial playlist: https://www.youtube.com/watch?v=-Q4lm8eYulw&list=PLu0W_9lII9ajLcqRcj4PoEihkukF_OTzA

3) Geeks For Geeks: https://www.geeksforgeeks.org/python-gui-tkinter/

4) Tutorialspoint: https://www.tutorialspoint.com/python/python_gui_programming.htm

CONCLUSION

In conclusion, AgriPy is a helpful tool for farmers, making farming smarter and more efficient. As we wrap up this project, it's evident that the integration of Python programming, data analysis, and various technologies has provided us with a practical application of our academic knowledge. We have learned how to address real-world challenges in the agricultural sector. The collaborative nature of the project has honed our teamwork and communication skills, essential for successful project execution. Building a graphical user interface using Tkinter, integrating external APIs like OpenWeatherMap and going out of the way to learn new things have expanded our technical skill set significantly. As India is an Agrarian economy, it is important that we address real-world agricultural issues and make a positive contribution to society. It has given us an understanding of software development, problem-solving, and the relevance of technology in addressing contemporary challenges in agriculture. This project will surely contribute to our ongoing learning journey in the field of computer science and technology.