Computerized Gardening System

Divyanth Chalicham
Wineel Wilson Dasari
Varshit Purna Amrutham

Problem Statement

- Managing large, diverse gardens is labor-intensive and prone to errors, requiring consistent irrigation, temperature control, pest management, and activity monitoring. Traditional methods are inefficient and unsustainable for complex ecosystems.
- The solution is a Computerized Gardening System (CGS) that automates these tasks, integrates multiple subsystems, and provides detailed logs, ensuring efficient garden maintenance with minimal manual effort.

- **Purpose:** Automate and enhance the management of large, diverse gardens.
- **Challenges Addressed:**
  - Managing different plants and their requirements.
  - Handling environmental factors like irrigation, temperature, and pests.
  - Reducing manual effort while ensuring optimal garden health.
- **Core Features:**
  - Watering System.
  - Heating System.
  - Pest Control.
- **Why It Matters:**
  - Ensures a thriving garden with minimal human intervention.
  - Provides a detailed logging mechanism for transparency and analysis.

- **Traditional Gardening Methods:**
  - Manual watering, pest control, and temperature regulation remain the most common practices.
  - Depend heavily on human effort and expertise, leading to inefficiencies and errors.
- **Automated Gardening Tools:**
  - Limited commercial solutions like smart sprinklers and IoT devices for monitoring moisture or temperature exist.
  - Most focus on isolated functionalities, such as irrigation systems, rather than comprehensive garden management.
- **Academic Research:**
  - Studies have explored IoT and AI integration for gardening systems, focusing on optimizing resource use and plant growth.
  - Few implementations combine automation with user-friendly interfaces for large-scale gardens.

Motivation:

- Complexity of Garden Management: Large gardens with diverse plants require careful coordination of watering, temperature control, and pest management, which can be overwhelming for gardeners.
- Inefficiency of Current Systems: Existing solutions often address single issues (e.g., watering) but lack the ability to integrate multiple aspects of garden care.
- Need for Automation: Automating repetitive tasks saves time, reduces human errors, and ensures consistency in garden care.
- Environmental Sustainability: Efficient resource usage, such as optimized water distribution, reduces waste and promotes sustainability.

Methodology

Tools, Technologies, and Frameworks Used

- **Programming Language:**
  - Java: Core programming language for implementing the system.
- **User Interface:**
  - JavaFX: For developing a responsive, user-friendly graphical interface.
- **Development Tools:**
  - IntelliJ IDEA / Eclipse: IDEs for efficient coding and debugging.
- **Design Tools:**
  - Draw.io: For creating UML diagrams like class diagrams, sequence diagrams, and activity diagrams.

Workflow

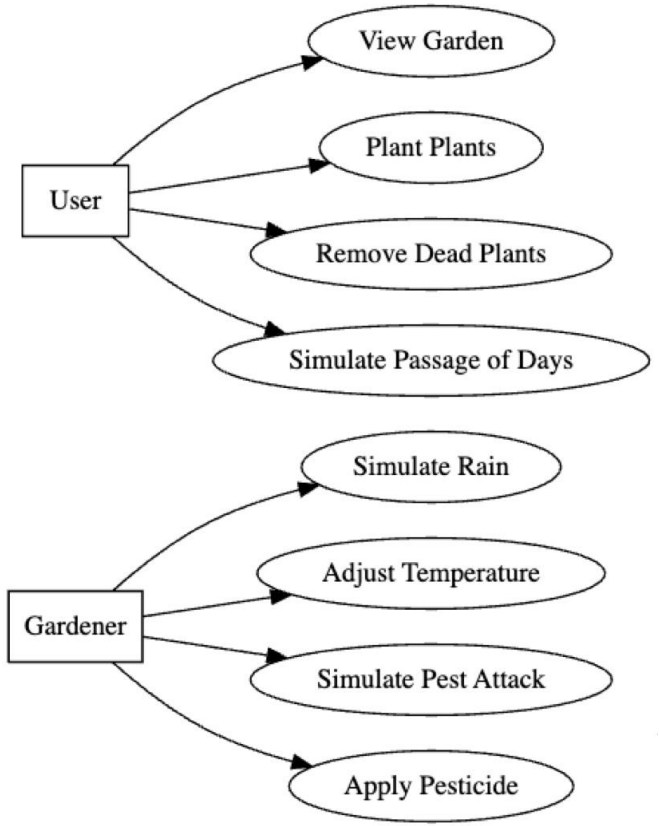
- **Requirements Analysis**
  - Gather requirements from gardening resources, gardeners, and online research.
  - Identify key features (e.g., watering, pest control, heating).
- **Design Phase**
  - Create UML diagrams (Class Diagram, Sequence Diagram, Use Case Diagram).
- **Implementation Phase**
  - Develop modules in Java.
  - Build a GUI using JavaFX.
- **Testing**
  - Unit testing for individual modules.
  - Integration testing to ensure all components work together.
  - Stress testing for prolonged operation.
- **Deployment**
  - Package the system for use.
  - Prepare the help manual and provide detailed documentation.

Modules

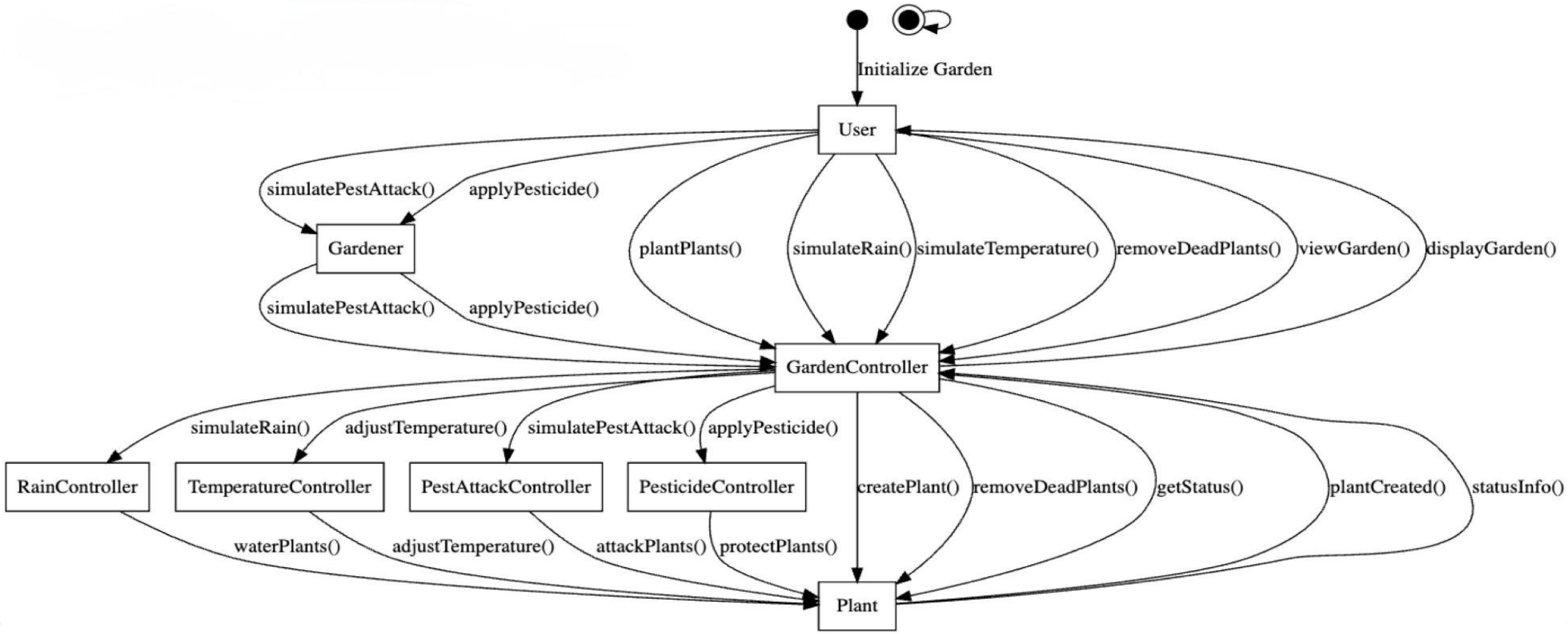
- **Garden Controller**
  - Manages overall garden operations and coordinates interactions between different controllers to ensure a seamless simulation.
- **Rain Controller**
  - Simulates rainfall and activates the sprinkler system if natural rainfall is insufficient to meet plant water requirements.
- **Temperature Controller**
  - Monitors and adjusts the garden's temperature, activating the heating system when temperatures drop below the safe threshold.
- **PestAttackController**
  - Simulates pest attacks and determines their impact based on plant susceptibility and pesticide application status.
- **Pesticide Controller**
  - Applies pesticides to enhance plant resistance against pests and reduces the likelihood of damage during pest attacks.
- **Sprinkler Controller**
  - Controls the sprinkler system, calculating and distributing water based on the specific requirements of each plant.
- **Heating Controller**
  - Manages the heating system to maintain a stable temperature, ensuring plant safety during cold conditions.

- **Plant (Abstract Class)**
  - Represents a generic plant with shared properties, such as water requirements, temperature tolerance, and pest resistance.
- **Rose**
  - A specialized plant with unique water requirements and specific susceptibility to pests.
- **Tomato**
  - A specialized plant with its own set of water requirements and pest tolerances.
- **Orange**
  - A specialized plant designed with distinct water needs and pest resistance levels.
- **Garden Simulator**
  - Acts as the main entry point for the Command-Line Interface (CLI) version, managing the simulation lifecycle and daily garden operations.
- **Garden Simulator API**
  - Provides an API for interacting with the simulation, allowing users to initialize the garden and manipulate simulation parameters programmatically.
- **GardenThread**
  - A custom thread class designed to handle asynchronous tasks, such as time-sensitive garden activities.
- **MyTimer**
  - Tracks simulation time, maintaining day and hour progression for accurate scheduling of garden events.
- **GUIMain**
  - Serves as the entry point for the JavaFX-based graphical user interface, initializing and running the GUI application.
- **ViewController**
  - Handles user interactions within the GUI and bridges the gap between the user interface and the garden simulation logic.
- **hello-view.fxml**
  - Defines the layout and structure of the GUI using FXML, ensuring a clean and intuitive user interface.

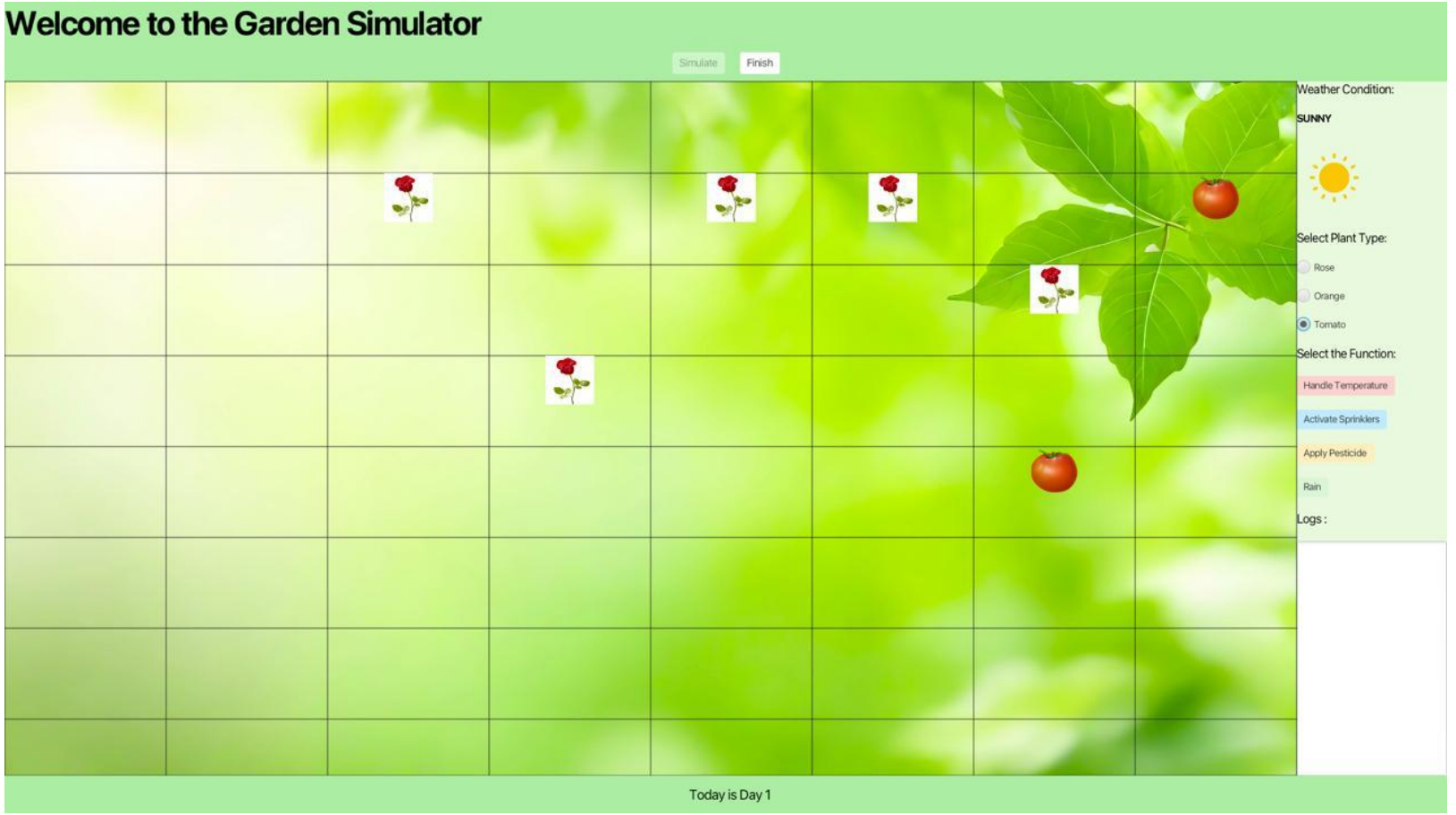
Use case


View Garden
Plant Plants
User
Remove Dead Plants
Simulate Passage of Days
Simulate Rain
Adjust Temperature
Gardener
Simulate Pest Attack
Apply Pesticide

Sequence diagram



UI


Welcome to the Garden Simulator
Simulate
Finish
Weather Condition:
SUNNY
Select Plant Type:
Rose
Orange
Tomato
Select the Function:
Handle Temperature
Activate Sprinklers
Apply Pesticide
Rain
Logs :
Today is Day 1

DEMO

Future work

- **Displaying All Log Details in the UI:**
  - Implemented functionality to load and display all log details in the log section of the user interface, providing real-time monitoring of system events and updates.
- **Visualizing Pest Attacks on Plants:**
  - Integrated a dynamic system that tracks and displays various pest attacks on plants, offering users insights into pest activity and its impact on garden health like displaying dead plants and removing them from the grid.
- **Handling Weather Conditions and Their Impact on Plants:**
  - Incorporated weather condition simulations (e.g., rainfall, temperature changes) and their effects on plants, with real-time updates shown on the frontend to help users understand environmental impacts on plant growth.

Conclusion

- **Summary of the Project**
  - The Computerized Gardening System (CGS) is a comprehensive and automated solution designed to address the complexities of managing large and diverse gardens. By integrating advanced features such as automated watering, pest control, heating, and a robust logging system, the project showcases how modern technology can transform traditional gardening practices. Through a user-friendly JavaFX interface, the system simplifies garden management while ensuring sustainability and efficiency.
- **Key Takeaways**
  - Automation for Efficiency: The system minimizes manual effort and human error by automating key gardening tasks.
  - Modular Design: Independent modules ensure scalability and easier maintenance.
  - User-Centric Approach: An intuitive GUI and detailed logs enhance usability for both technical and non-technical users.
  - Resilience and Sustainability: The garden can operate autonomously over extended periods while optimizing resource usage.
- **Final Thoughts**
  - The CGS highlights the potential of combining software engineering with real-world applications to solve everyday problems. This project not only demonstrates the power of object-oriented design and programming but also emphasizes the importance of user-focused design and sustainable practices. With further enhancements, the CGS could serve as a model for future smart gardening solutions, bridging the gap between technology and environmental stewardship.

Acknowledgments

- We would like to express our sincere gratitude to the following individuals and organizations for their invaluable guidance and support throughout the development of the Computerized Gardening System:
  - Professor: For providing expert advice, constructive feedback, and continuous encouragement that shaped the project's direction and execution.
  - Santa Clara University: For providing the resources and platform to explore and implement this innovative idea.
- This project would not have been possible without their contributions, and we are deeply thankful for their help and inspiration.

Thank you