UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

# COS214

# Final Project Specification

Release Date: 25-09-2025

Due Date: 02-11-2025

Total Marks: 100

# Contents

# 1 Project Description

It's spring in South Africa. The air is warm, the sun is bright, and everywhere you look, new life is pushing through the soil. Inspired by the season, you decide to start your very own plant nursery. It's a place filled with seedlings, colourful flowers, leafy shrubs, and tall trees reaching for the sky. Every corner of your nursery is alive with growth, from the greenhouse where young plants are nurtured, to the sales area where curious customers wander, exploring the greenery and asking for advice.

Running a nursery is more than just planting seeds and waiting for them to grow. You quickly realise that each plant has its own needs — some need frequent watering, others thrive on neglect; some require special soil, others love the sun. At the same time, customers want guidance, suggestions, and the ability to personalise their plant purchases, while your staff move between caring for plants and assisting visitors.

In this project, your task is to design and implement a Plant Nursery Simulator in C++. The goal is to model the complexities of a working nursery in a way that is flexible, maintainable, and scalable. You will use object-oriented principles and design patterns to help structure your system: patterns can guide you in handling recurring problems like different plant care strategies, customer customisations, stock management, and staff coordination. The important part is to show how patterns help you bring order to complexity, just as spring brings order and beauty to the chaos of nature.

Your nursery can grow and evolve however you imagine. You might focus on plant growth, customer interactions, staff workflows, or any combination of these. You could even introduce new ideas as you go, creating a system that adapts and expands, just like a real nursery in springtime.

Remember: the focus is on creative design and effective application of patterns, not on replicating every detail of a real nursery. Your system should capture the spirit and complexity of running a nursery while showing clear, structured thinking and good software design.

There are three main areas you need to showcase during this project: the greenhouse or garden area, the customers on the sales floor, and the staff. The communication between these aspects of the nursery need to demonstrated.

## 1.1 Greenhouse/ Garden Area

This is where all life is created! From small succulents to prickly cactus to budding roses to aromatic lavender to iconic Baobab trees, the garden is where the magic happens.

The importance of this area cannot be undermined and your system will need to account for the following:

Strategy or State?(i think its state, because the staff can switch from sun exposure->watering->fertiliser. Command? -> user invokes command to water or exposure to sun??

- Plant Care: Different plants require different care including amounts of water, sunlight exposure or fertilizer. Some plants need constant attention while others thrive without much fuss. Command pattern
- Plant Life Cycle: The life cycle of a plant differs per plant type. Some grow to maturity much faster than others meaning that plants may be put up for sale at different times. Plants also have specific seasons that they thrive in - this is when they are usually sold.
- Inventory: There must be an inventory tracking subsystem in place so that staff know what is

on hand to sell to the customers. All plants should be accounted for, whether they are ready to sell or still growing to maturity. Iterator and Memento?

- Stock: New stock should be easily added as the nursery grows. Factory/ Abstract factory method? Also Prototype

## 1.2 The Staff Mediator. Staff-> mediator, Sales floor/

The staff are the backbone of your nursery. They move between the greenhouse and the sales floor, making sure both plants and customers are well taken care of. Your system should allow staff to:

- Plant Care: Tend to plants according to their unique care routines (watering, fertilizing, sunlight management, pruning, etc.).
- Plant Life Cycles: Monitor plant health and life cycles, and take action when plants need extra attention.
- Inventory: Track inventory and coordinate with other staff to ensure the sales floor reflects what is available in the greenhouse.
- Customer Interaction: Interact with customers by offering advice, answering questions, and helping them find suitable plants.
- Potentially be extended with roles like delivery staff, landscapers, or greenhouse managers as the nursery grows.

Staff in your simulation can be simple agents or classes, but they should demonstrate coordination and interaction with both plants and customers.

## 1.3 The Customers and The Sales Floor

Customers are very important, you want to keep them as happy as possible so that they purchase more from your nursery. They wander, explore, ask questions, and make decisions that affect your nursery's operations. Your system should model:

- Customer browsing: customers can view available plants, ask for information, or request recommendations.
- Personalisation: customers may want to customise their plants (e.g., decorative pots, gift wrapping, or special arrangements).
- Interaction with staff: they receive guidance, purchase assistance, and updates on availability or stock changes.
- Sales and transactions: the system should handle plant purchases, updating inventory, and optionally recording transaction details.

## 1.4 Objectives

By completing this project, you will:

- Design a system using design patterns of your choosing to address design issues.
- Explore object-oriented design by modelling a dynamic, evolving nursery system.
- Build a flexible, maintainable, and scalable simulator in C++ that can grow as your nursery expands.
- Develop creative problem-solving skills.
- Identify functional requirements of a system and translate them into design pattern use cases.
- Develop UML diagrams.
- Compile documentation detailing the design, implementation, and usage of the simulation.
- Develop collaboration skills by working efficiently with a development team.

## 1.5 Expected Outcomes

At the end of this project, you should have:

- A working Plant Nursery Simulator in C++ that models greenhouse operations, staff, and customer interactions.
- Hands-on experience applying design patterns to solve software design problems.
- An appreciation and experience working with a team in a collaborative space.
- At least ten (10) design patterns applied meaningfully within your system.
- Detailed documentation outlining the system design and architecture, the implemented design patterns and how the system demonstrates extensibility, flexibility and maintainability.

## 1.6 Constraints

1. Team Composition: Teams must consist of 5 or 7 members. Come up with a creative name for your team.
2. Design Patterns: The system must use **at least 10** design patterns. More details in subsequent sections.
3. UML Documentation: Comprehensive UML diagrams must be provided, created in Visual Paradigm.
4. Version Control: Git must be used as the version control system, with all team members contributing regularly.

5. Documentation Generation: Doxygen must be utilised to produce documentation for the project.

6. Demo: All projects will be presented in person and all members of the group must be present.

# 2 Plagiarism

The Department of Computer Science considers plagiarism a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with permission) and copying material from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to `http://www.library.up.ac.za/plagiarism`. If you have any questions regarding this, please ask one of the lecturers to avoid misunderstanding.

# 3 Timeline

Your team will be expected to meet with your assigned tutor/AL weekly. This is to ensure that your team does not miss any deadlines and that consistent progress is made. The initial design component will serve as practical 5.

1. 28 September - Team registration of 5 or 7 members to be done. You will need to assign a team leader. The registration will take place on the CS portal.

2. 6 October - Submit practical 5 consisting of the initial project design and discuss design with your tutor/AL.

3. 20 October - Implement necessary changes and provide skeleton implementation.

4. 27 October - Have a working version and begin demo prep.

5. 2 November - Submit report and diagrams to ClickUp and code to FitchFork.

6. 3-5 November - Project Demos.

Note: Stick to these deadlines and do not fall behind. This will give you an idea of what to expect in COS 301.

# 4 Mark Distribution

| Task | Marks |
|---|---|
| Practical Assignment 5 | 0 |
| Design | 30 |
| Implementation | 30 |
| Report | 15 |
| Development Practices | 10 |
| Demo & Presentation | 15 |
| Bonus Marks (Extra marks) | 10 |
| **Total** | **100** |

Table 1: Mark Allocation

**Note:** Practical 5 does not contribute to your project mark, but contributes as a normal practical. You cannot skip Practical 5.

## 4.1 Practical 5 Mark Distribution

| Task | Marks |
|---|---|
| Functional Requirements | 10 |
| UML Class Diagram | 20 |
| UML State Diagram | 10 |
| UML Activity Diagram | 10 |
| UML Sequence Diagram | 10 |
| UML Object Diagram | 10 |
| UML Communication Diagram | 10 |
| **Total** | **80** |

Table 2: Mark Allocation Prac 5

# 5 Tasks

The following tasks should be followed to ensure your success in the project.

## 5.1 Task 1: Practical Assignment 5

Practical Assignment 5 requires you to give a rough overview of the system that you intend to build. Remember that you may change the design after demonstrating the initial design to your assigned AL/tutor.
Referring to the mark distribution in section 4.1, the following should be completed:

### 5.1.1 Functional and Non functional Requirements (10 marks)

Outline the functional requirements of your system. Functional requirements are specifications for what the system must do. This includes features, functions and behaviours the system must be able to do.
An example could be: FR 1: The system will use a different watering strategy based on the type of plant.
Ensure that you have **at least one functional requirement per design pattern**. In other words, each functionality that is being implemented using a design pattern should have a corresponding functional requirement.

Outline the non functional requirements of the system. Non-functional requirements specify how the system performs rather than what it does. They deal with quality attributes such as scalability, reliability, usability, security, performance and so forth.
An example for scalability could be: The system can support 100 users without decreasing the uptime of the system.
Include **at least three non-functional requirements for the system as a whole**, and clearly indicate which quality attribute each addresses (e.g., scalability, reliability, usability, security).

### 5.1.2 UML Class Diagram (20 marks)

Your class diagram should show the overall structure of your nursery system. It must clearly capture the main components (plants, staff, customers, inventory, etc.) and their relationships (inheritance, composition, association). Use the class diagram to highlight where you've applied design patterns. Think of this as the blueprint of your nursery. For now it is not important to add all of the details to the class diagram. In other words, only the classes and their relationships are important for now. If there are specific attributes/ methods that you feel are necessary to include, feel free but do not add all of the attributes/ methods.
Remember, the tutors/ALs may offer suggestions and request changes.

### 5.1.3 UML State Diagram (10 marks)

Choose at least one important entity in your system (e.g., a plant or a customer) and model its different states and transitions. This diagram should show how an object's behaviour changes over time as it experiences different events.
As in your practical assignment, you should have at least four (4) states and there should be at least one state that can move "forwards"/"backward". There should also be a state where it is "final", as in there are no outgoing states.

### 5.1.4 UML Activity Diagram (10 marks)

Model the workflow of a key process in your nursery. Your activity diagram should show the sequence of steps and decision points in that process. Have at least four (4) steps in the diagram.

### 5.1.5 UML Sequence Diagram (10 marks)

Create a sequence diagram that shows the interactions over time between objects in a scenario. There should be at least thee (3) objects interacting.
This diagram highlights messages passed between objects and the order in which they occur.

### 5.1.6 UML Object Diagram (10 marks)

Provide a snapshot of your system at a given moment in time. An object diagram should show specific instances of classes and their relationships at that point. Include at least 5 objects in your diagram.

### 5.1.7 UML Communication Diagram (10 marks)

Model how different parts of your system collaborate to perform a function. A communication diagram is similar to a sequence diagram, but it emphasizes the network of relationships and how messages flow through them.

**NOTE:** All of your diagrams are based on the same system therefore there needs to be **consistency** between the different types of diagrams. This includes but is not limited to: class names, method and attribute names, and behaviours between classes. All of the diagrams are to be made using **Visual Paradigm**. Failure to use Visual Paradigm and not follow the constraints for each diagram will significantly decrease your marks for the relevant sections.
It is also important that all members work on these sections together. Do not "split up" the work here. Find time to discuss and outline the requirements and diagrams together to ensure that the system is cohesive.
This is an initial mock up meaning that these diagrams will most likely change for the final report. This practical lays the foundation for Task 2 of the project.

## 5.2 Task 2: Design

### 5.2.1 Identify Functional and Non Functional Requirements

Document your updated functional and non functional requirements from Task 1.1.
Make use of subsystems to organise the requirements neatly.

### 5.2.2 UML Diagrams

Update all of the different UML diagrams from Task 1 incorporating the changes that were suggested to you.
Additionally create a high level Activity Diagram showing the overall flow of the system.
**Advice:** Create this initially and update it as you find yourself making changes to the system design. This way you will not lose your original system flow.

### 5.2.3 Design Patterns

A **minimum of 10 distinct design patterns** are required. You will be heavily penalised for not including at least 10 patterns.
Consider the following constraints;

- You must have at least two (2) of each type of pattern (creational, behavioural and structural).
- At least five (5) of these design patterns must be used:
  - Observer
  - Iterator
  - Mediator
  - Command
  - Adapter
  - Chain of Responsibility
  - Builder
  - Interpreter
  - Bridge
  - Facade
  - Visitor
  - Proxy
  - Singleton
  - Flyweight

For each pattern identified, give an explanation of why you chose it and where you implemented it. Give the UML class diagram per pattern ensuring that all details are present.

### 5.2.4 System Class Diagram

Now that you have identified all ten (10) design patterns and have their partial class diagrams, combine them to create one class diagram of the entire system. Include all methods and attributes, and relationships between classes and patterns.
Remember, one class may be part of multiple patterns.

## 5.3 Task 3: Implementation

### 5.3.1 Implement the Plant Nursery Simulation

Develop the simulation in C++, ensuring all functionalities are operational. The minimum is to use a text-based interface that users can interact with.
Take the time here to set up your system and dependencies such that team members can work simultaneously on separate aspects of the system. Your team must make use of git and GitHub and

every member is required to contribute equally. At completion, each member should have **at least ten (10) commits.**
Document your code using Doxygen.
Your team is also required to set up unit testing and every member must have written some unit tests. Full coverage is not a requirement.

#### 5.3.2 (Bonus) Implement a GUI

While not required, developing a graphical user interface (GUI) will provide bonus marks and enhance the user experience.
GUIs can be developed in a C++ GUI framework or as a website (with your C++ engine acting as the API). You may use any libraries.

### 5.4 Task 3: Report

As part of your project you are required to set a PDF document that includes the following:

#### 5.4.1 Research Brief

Write a short brief (1 page) on all the research you have done about managing a nursery and the components thereof. This can include plant types, their care, etc.
Include all references.
Explain how this influenced your design decisions and state any assumptions or relevant definitions here.

#### 5.4.2 Design Pattern Application

Add your reasoning and justifications from Task 5.2.3 here. This includes the partial class diagrams. Additionally match the functional requirements you have identified to the design pattern responsible.

#### 5.4.3 Other Items

You should include everything relevant to your system in the report. This includes all of the sections from section 5.2 Task 2: Design.

### 5.5 Task 5: Development Practices

#### 5.5.1 Version Control with Git

Use Git as the Version Control System and document your branching strategy in the report.
Ensure every team member makes at least 10 commits, documenting significant changes and updates.
Ensure that you have **at least 3 closed pull requests.**

### 5.5.2 Code Documentation

Document your code following C++ documentation standards.
Make sure to use meaningful comments and descriptions for classes, methods, and complex logic.
Use Doxygen to produce comprehensive documentation.

### 5.5.3 Automated Unit Testing

Implement unit and/or integration tests using an automated testing framework. While full coverage is not required, ensure that key functionalities are tested. Every team member must contribute to the creation of unit tests.

### 5.5.4 (Bonus) Github Actions Linter and Tester

While not required, develop and deploy CI/CD pipelines to lint, test (using your automated unit tests) and build your application using Github Actions). This will count for bonus marks.

## 5.6 Demo & Presentation

**Team Participation:** All team members must be present during the demo. Failure to attend will result in penalties.
**Professionalism:** The demo should be well-prepared, demonstrating all key features and functionalities of the simulation.
**Clarity:** Present your design decisions, implementation challenges, and how design patterns were utilised effectively.
**Engagement:** Engage the audience with clear explanations, visual aids, and a live demonstration of the simulation in action.

# 6 Submission Instructions

Each team is required to create a Git repository to manage the project. Use GitHub to host your repository. Ensure that the documentation includes a link to your GitHub repository. The repository must contain the following:

- **System Files:** All source files (`.h` and `.cpp`) and your Makefile.
- **Data Files:** Any necessary data files required to run the program.
- **Readme:** A `readme.md` in the root directory containing a short description of your project, explaining how to compile and run the program, and the placement of any data files.
- **Report:** A PDF version of your latest report in a folder named `Report`. The report must also be written in Google Docs, with a link to the Google Docs version included in the PDF.
- **Data Folder:** Place your data files, if any, in a folder named `Data`.

Only the team leader is required to submit the project to ClickUp.
The team leader should download the git repository as a `.zip` or `.tar.gz` file. This archive should then be uploaded to ClickUp before the submission deadline. A copy of your code must also be submitted to Fitchfork. Failure to upload the project to ClickUp and/or Fitchfork will result in the entire team receiving **zero (0)** for the project.

**Demo Scheduling:**
The demos will take place during the week of 3 November. Demo times and specific dates will be made available to you closer to the demo time.