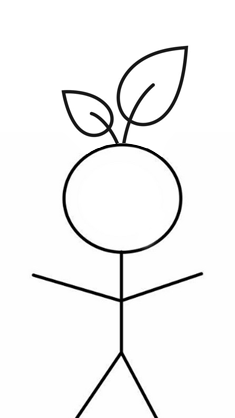
**Growing Pains – An Online Plant Store System**

**(OPSS)**

**Object Oriented Software Development - Project**

****

**By: Mark Lambert (C00192497)**

**Course: Software Development (CW\_KCSOF\_B)**

**Submission Date: 11th April 2025**

A black and grey logo

Description automatically generated**Table of Contents Page**

[**Summary** 3](#_Toc194566409)

[**Requirements** 4](#_Toc194566410)

[**Functional** 5](#_Toc194566411)

[**Non-Functional** 7](#_Toc194566412)

[**Database Tables** 9](#_Toc194566413)

[**Customer Table** 9](#_Toc194566414)

[**Product Table** 10](#_Toc194566415)

[**Orders** 11](#_Toc194566416)

[**ER Diagram** 12](#_Toc194566417)

[**Interesting Source Code Snippets** 14](#_Toc194566418)

[**Test Cases** 15](#_Toc194566419)

# **Summary**

A rapidly growing houseplant store wants to expand its business to build an Online Plant Store System (OPSS) to manage its expanding business and improve customer engagement. The system should aim to meet the following requirements:–

1. Facilitate the buying of a diverse range of plants and plant accessories
2. A comprehensive marketplace experience, making the interface accessible to the user
3. A responsive, dynamic application that responds to user inputs and updates the backend database as the user interacts with the application
4. The store catalogue must have a filter feature to enhance user experience, allowing users to sort items and accessories by price, type etc
5. The user must be able to edit their account information as well as view and cancel any orders made
6. Users must also be able to set personal reminders, notifying them of when to water their plants

The system should also feature a user-interface which keeps the design aesthetics of the houseplant store in mind.

The system should allow users to browse the store catalogue, which will include filtering options such as plant species, type, price and accessory. Each plant listing will include brief descriptions, care instructions and pricing information, which can be visible when a customer selects a plant.

Once the user selects a plant, they can add it to their cart to proceed with the checkout process. Users may update their cart or remove items. When the user initiates the checkout process, they must enter in payment details before finally placing the order.

Another integrated feature should allow users to set a reminder by selecting a date. The reminder will take input on the plant type and species (e.g., succulent, tropical, houseplant) and notify the user of when to next water their plant.

# **Requirements**

The following document outlines the requirements for the Online Plant Store System (OPSS). To ensure that all corners of the requirements finding process were covered, the **FURPS+** model to assess functional and on-functional requirements was considered.

**The functionality** of the system is defined by the behaviours and features of the system, ensuring that the user’s core needs and expectations of the system are met regarding what it does.

**The non-functional** requirements describe the systems usability, reliability, performance and supportability, ensuring that it operates as efficiently as possible and is scalable for future iterations.

## **Functional**

|  |  |
| --- | --- |
| **Requirement ID :** | FR001 |
| **Requirement Name :** | Select Item |
| **Definition:** | The system shall display detailed information when a user selects an item from the catalogue. |
| **Specification:** | - On click of a product in the BrowsePanel the system should:  - Form a SELECT query on the Product table  - Display a new ProductPanel with the following using JLabels: 200x200 image of the product, product name, price and description |

|  |  |
| --- | --- |
| **Requirement ID :** | FR002 |
| **Requirement Name :** | Cart Management |
| **Definition:** | Registered users may add products to their cart, from which they may alter the product quantity via a JSpinner |
| **Specification:** | - On click of the “Add to Cart” button the system must:  - Validate the user is first logged in  - Update the users cart object to display the product quantities and total price |

|  |  |
| --- | --- |
| **Requirement ID :** | FR003 |
| **Requirement Name :** | Checkout Process |
| **Definition:** | Users complete the Order by initiating a checkout process, validated by inputting payment details |
| **Specification:** | - On click of the “Checkout” button the system must:  - Build a form to input: Card Number, Card Holder, Address, CVV and Expiration Date (via JCombo boxes)  - On submit, the system will generate an INSERT query into the Orders table |

|  |  |
| --- | --- |
| **Requirement ID :** | FR004 |
| **Requirement Name :** | Browse Catalogue |
| **Definition:** | The system shall display a populated catalogue of items with a scrollable UI |
| **Specification:** | - A JPanel displaying a series of product item containers which hold information about each product in the Product table.  - Products are retrieved via a SELECT query in the Products table |

|  |  |
| --- | --- |
| **Requirement ID :** | FR005 |
| **Requirement Name :** | Order History |
| **Definition:** | Users must be able to view past orders with the aim of cancelling orders should they wish |
| **Specification:** | - A JTable displaying a history of all orders made by the logged in user.  - The table is populated via a SELECT query on the Customer table which INNER JOINS with the Orders table  - When an order is selected, the user may cancel the order by clicking the “Cancel Order” button.  - Onclick, a DELETE query in the Orders table is generated |

## **Non-Functional**

|  |  |
| --- | --- |
| **Requirement ID :** | NFR001 |
| **Requirement Name :** | Usability |
| **Definition:** | The system must be both learnable and accessible for new users |
| **Specification:** | - Learnability: Users must be able to comfortably adapt to the systems GUI, enabling them to purchase products quickly  - Accessible: The system must be designed bearing in mind users who may have vision impairments, such as font sizes, colours etc. |

|  |  |
| --- | --- |
| **Requirement ID :** | NFR002 |
| **Requirement Name :** | Reliability |
| **Definition:** | The system must reliably deal with invalid data input from the user |
| **Specification:** | - Data input must be handled appropriately according to what may constitute as “bad data” or malicious data.  - Preventative measures against SQL Injection by using prepared statements |

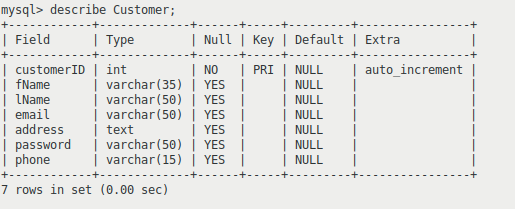
|  |  |
| --- | --- |
| **Requirement ID :** | NFR003 |
| **Requirement Name :** | Performance |
| **Definition:** | The system must respond quickly and appropriately to user input |
| **Specification:** | - Interaction between the system and database must be seamless, ensuring the customer is met with a responsive application |

|  |  |
| --- | --- |
| **Requirement ID :** | NFR004 |
| **Requirement Name :** | Supportability |
| **Definition:** | The system must be maintainable for future iterations and expansion |
| **Specification:** | - Code must be well documented and conform to standard Object Oriented principles |

|  |  |
| --- | --- |
| **Requirement ID :** | NFR005 |
| **Requirement Name :** | Security |
| **Definition:** | The system must be secure for the user to use |
| **Specification:** | Any sensitive or precious data must be handled with care |

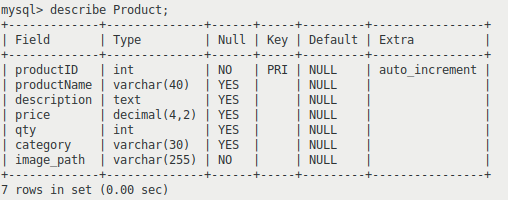
# **Database Tables**

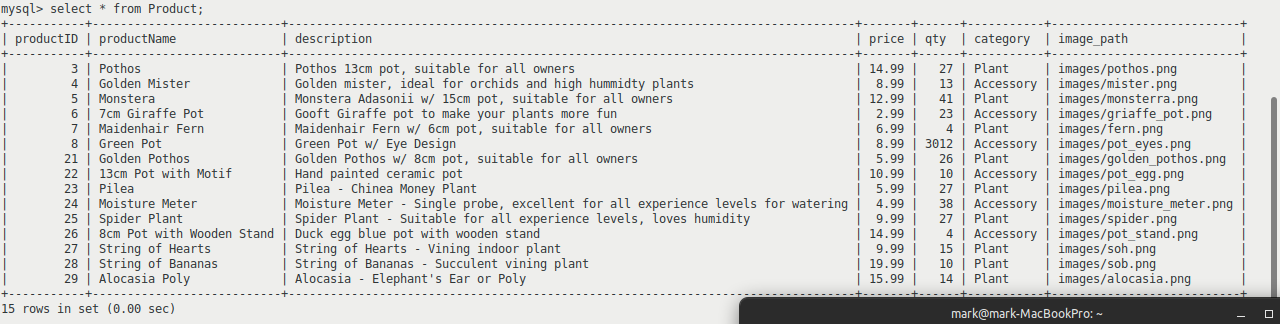
## **Customer Table**

**Structure** **of Customer Table**

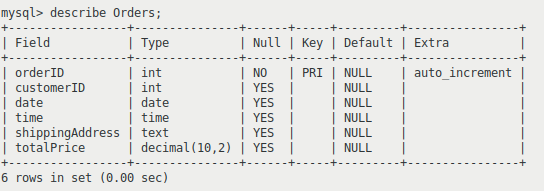
**Sample Data for Customer Table**

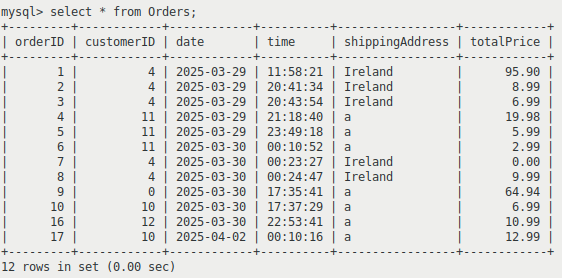
## **Product Table**

**Structure** **of Product Table**

**Sample Data for Product Table**

## **Orders**

**Structure of Orders Table**

**Sample Data for Orders Table**

# **ER Diagram**

**A screenshot of a diagram

AI-generated content may be incorrect.**

[[1]](#footnote-1)

# **Interesting Source Code Snippets**

# **Test Cases**

|  |  |
| --- | --- |
| **Name** | **TC-001:** Select Item |
| **Requirement** | Verify that the system successfully updates to display details of a selected item from a database of items. |
| **Preconditions** | The system is displaying the full catalogue |
| **Steps** | 1. Click on the **image** of the third item. 2. Return to previous page 3. Click on the **name** of the first item. 4. Return to previous page |
| **Expected Results** | 1. The item details window appears, with a larger image and more detailed information 2. Verify that items can be selected by clicking icon **or** thumbnail 3. Return to browsing catalogue |

|  |  |
| --- | --- |
| **Name** | **TC-002:** Add to Cart |
| **Requirement** | Verify that the system successfully allows a user to enter item(s) to cart. |
| **Preconditions** | The user is viewing the catalogue |
| **Steps** | 1. Click on the **second** product 2. Click the “Add to Cart” button 3. Return to catalogue 4. Click on the **fourth** product 5. Change **quantity** to 2, add to cart |
| **Expected Results** | 1. The item details window appears, with a larger image and more detailed information 2. The system alerts the user to the fact that the item has been added successfully 3. Return to browsing catalogue |

|  |  |
| --- | --- |
| **Name** | **TC-003:** Checkout |
| **Requirement** | Verify that the store system successfully allows the user to checkout their items |
| **Preconditions** | The customer has items in their cart |
| **Steps** | 1. Click on the **View Cart** button 2. Click on the **Proceed to Checkout** button 3. Enter your **login details** 4. Input **personal information** 5. Click on the **Confirm Order** button |
| **Expected Results** | 1. The system displays the users Cart 2. The system begins the Checkout process 3. System prompts user for login details 4. System prompts user for shipping & billing information 5. System updates to confirm to the user that their order has been successfully placed |

|  |  |
| --- | --- |
| **Name** | **TC-004:** Filter Catalogue |
| **Requirement** | Verify that the system allows the user to apply filter(s) to the Catalogue of Items |
| **Preconditions** | The system is displaying the full catalogue |
| **Steps** | 1. Click on the **Filter** button 2. Select **one** filter 3. Click the **Apply Filter** button 4. Click on the **Filter** button 5. Select **another** filter 6. Click on the **Apply Filter** button |
| **Expected Results** | 1. The system updates to show a list of filters to choose from 2. When applied, the system displays just items matching the filter tag 3. Applying another filter will result in a more specific list of items |

|  |  |
| --- | --- |
| **Name** | **TC-005:** Schedule Reminder |
| **Requirement** | Verify that the system successfully sets and alerts the user when a Reminder is Scheduled |
| **Preconditions** | The user is logged in |
| **Steps** | 1. Click on the **Schedule Reminder** button 2. Input **todays date** 3. Click on the **Set Reminder** button to confirm |
| **Expected Results** | 1. The system will display the Schedule Reminder page 2. The system will update to display the reminder the user has just input |

|  |  |
| --- | --- |
| **Name** | **TC-006:** Browse Catalogue |
| **Requirement** | Verify that the system allows the user to browse the catalogue of items |
| **Preconditions** | The user is on the page displaying the catalogue |
| **Steps** | 1. Click on the **Home** page 2. Scroll to browse the catalogue |
| **Expected Results** | 1. The system will update in real time to display the catalogue – containing items - for the user |

1. The Many to Many relationship between the Orders and Product tables in the current iteration of the OPSS has not been simplified to include an Order/Product table.

   Future expansion during the Summer of ’25 will ensure this implementation is appropriately handled. [↑](#footnote-ref-1)