

ICT 373 Assignment 2

Lim Wen Chao 35368872/CT0360379

1 April 2024

This is the documentation for ICT 373 Assignment 2

Table of Contentss

[Requirements/Specification 4](#_Toc162879929)

[Assumptions 5](#_Toc162879930)

[User Guide 5](#_Toc162879931)

[Structure/Design 11](#_Toc162879932)

[Specification 11](#_Toc162879933)

[Diagrams 13](#_Toc162879934)

[Pseudocode 14](#_Toc162879935)

[Limitations 17](#_Toc162879936)

[Testing 17](#_Toc162879937)

[Create Magazine Service 17](#_Toc162879938)

[Create Magazine 17](#_Toc162879939)

[Create Supplement(s) 18](#_Toc162879940)

[Create Customer(s) 21](#_Toc162879941)

[Load Magazine Service 32](#_Toc162879942)

[View Magazine Service 33](#_Toc162879943)

[Add Supplement 37](#_Toc162879944)

[Add Customer(s) 40](#_Toc162879945)

[Add Paying Customer 40](#_Toc162879946)

[Add Associate Customer 45](#_Toc162879947)

[Edit Magazine 49](#_Toc162879948)

[Edit Supplement 51](#_Toc162879949)

[Edit Customer(s) 53](#_Toc162879950)

[Edit Paying Customer 53](#_Toc162879951)

[Edit Associate Customer 57](#_Toc162879952)

[Delete Supplement(s) 61](#_Toc162879953)

[Delete Customer(s) 64](#_Toc162879954)

[Delete Associate Customer 64](#_Toc162879955)

[Delete Paying Customer 66](#_Toc162879956)

# Requirements/Specification

For this project, the objective is to design and implement a simple graphical user interface(GUI) program using JavaFX. The program will also need to have the ability to save and use data across different run times.

The program is about a magazine subscription service where the main part of the magazine is made available to all customers along with their chosen extra supplements.

A supplement has a name and a weekly cost.

The main magazine also has a weekly cost.

A customer is expected to have a name, an address, an email address, and a list of supplements.

A customer's address should have a street number, street name, suburb, and postcode.

A customer can either be a paying customer or an associate customer.

A paying customer has a payment method and a list of associate customers they are paying for.

The payment method could be either a credit card or direct debit from a bank account.

An associate customer is linked to a paying customer who will pay for the subscription cost of the associate customer.

The GUI program has three different modes, create, edit, and viewing.

In viewing mode, users will be able to view the list of supplements, list of customers, and an information panel.

When a supplement is clicked, the name, cost, and list of customers that are subscribed to the supplement will be displayed on the information panel.

When a customer is clicked, the name, address, email address, list of supplements, type of customer, list of associate customers if the customer is a paying customer or the paying customer if the customer is an associate customer, and the billing history if the customer is a paying customer.

In edit mode, users will be able to change existing information about the supplements and customers, such as changing of information, adding new customers, deleting customers, adding and deleting supplements. When adding new associate customers, the paying customer should be selectable from a drop-down list of existing paying customers.

In create mode, the user will create a magazine before adding supplements and customers.

## Assumptions

* The program is not for magazine delivery. Contents of the magazine are not stored or distributed through it. The program is only designed for an administrator of the magazine service to manage the service.
* A customer must be either a paying or associate customer.
* All customers must have a name, an address, and an email address. Supplements are optional.
* A customer name can be any string of the English alphabet with spaces.
* An email address can be any string of characters with an ‘@’ in between characters, followed by ‘.’ in between characters.
* An address must have a street number, street name, suburb, and postcode.
* A Street number can only an integer and it is mandatory.
* A Street name can be any character and it is mandatory.
* A Postcode can only be an integer and it is mandatory.
* Paying customers must have a payment method. List of associate customers linked to the paying customer are optional.
* Payment detail can only be an integer.
* An associate customer must have one and only one paying customer.
* The billing history of the paying customer are not stored and are instead dynamically generated using current costs.
* When a Paying Customer is deleted, all its associate customers are also deleted.
* When an associate customer is deleted, it is also removed from the paying customer’s list of associate customers.
* When a supplement is deleted, it is removed from all customer’s list of supplements.

# User Guide

This user guide will assume that the user is using Visual Studio Code as IDE to run this JavaFx application. The user is also expected to have at least Java SDK 11 and JavaFx 17. This user guide will also be using a pre-created Magazine Service instead of having the user to create themselves.

1. Open the Java project inside the src folder. The explorer should look like this:

A screenshot of a computer

Description automatically generated

Figure 1 Explorer View

1. Next, using ‘CTRL + Shift + P’ to open the command palette and select ‘Java: Configure Java Runtime’. Ensure your Java Version is correct.

A screenshot of a computer

Description automatically generated

Figure 2 Runtime Configuration

1. Open the Java Projects tab in the explorer tab and add all the JavaFx .jar files you have downloaded to the project’s reference library.

A screenshot of a computer

Description automatically generated

Figure 3 Java Projects Reference Library

1. Run the application starting on client.java and click start.

A screenshot of a computer

Description automatically generated

Figure 4 Start Window

1. Click on the ‘Load Magazine’ button to load a .ser file containing a created magazine service onto the application.

A screenshot of a computer

Description automatically generated

Figure 5 Create Mode

1. If the magazine service is loaded successfully, it will display the View mode

A screenshot of a computer

Description automatically generated

Figure 6 View Mode

1. You can click on each supplement or customer to see further information on the information panel

A screenshot of a computer

Description automatically generated

Figure 7 View Supplement Info

A screenshot of a computer

Description automatically generated

Figure 8 View Paying Customer Info

A screenshot of a computer

Description automatically generated

Figure 9 View Associate Customer Info

1. Switching to the edit mode, you should see all the existing customers and supplements, and the various buttons for editing, adding, and deleting them.

A screenshot of a computer

Description automatically generated

1. To edit or delete an customer or supplement, you will need to select a customer or supplement first
2. Fill in the form shown after clicking on an add or edit button and save them. You can check if the information is add/changed by looking at the view mode.

# Structure/Design

For this project, the Model-View-Controller (MVC) design pattern was used. MVC separates the application logic into three distinct but connected parts; the model, view and controller.

The model contains the data logic of the application and is responsible for the management of data, for example, the create, read, update and deletion of data.

The view is responsible for both displaying the data retrieved from the model and the collection of user inputs.

The controller acts as an intermediary between the model and the view, it will retrieve user inputs from view and pass it to the model as necessary, and it will also retrieve data from model and pass it to the view as necessary.

The MVC design pattern provides a clear separation between business logic and its Graphical User Interface (GUI), making the code easier to understand and modify. This design pattern also encourages modularity and reusability since each component could be developed and tested separately.

## Specification

There are the following packages and classes created for this application:

* Magazine - Holds all classes related to magazine.
  + Magazine - A class that represent a magazine service, also represents a magazine.
    - WeeklyCost - A double attribute representing the weekly cost of the base magazine.
    - Supplements - An ArrayList attribute of supplement object(s) containing all possible and existing supplement available for the magazine service.
    - Customers - An ArrayList attribute of customer object(s) containing all existing customers for the magazine service.
  + Supplement - A class that represents a supplement
    - Name - A string attribute representing the unique name of a supplement
    - WeeklyCost - A double attribute representing the weekly cost of a supplement
* Customer - Holds all classes related to customer.
  + Customer - An abstract class representing a customer.
    - CustomerType - A string attribute to indicate if a customer is paying or associate customer.
    - Name - A string attribute containing the name of a customer.
    - Email - A string attribute containing the email address of a customer.
    - Address - An Address object attribute representing the address of a customer.
    - Supplements - An ArrayList attribute of supplement object(s) representing the supplements a customer chose to subscribe to.
  + Address - A class representing an address of a customer.
    - StreetNumber - An integer attribute containing the street number of an address.
    - StreetName - A string attribute containing the street name of an address.
    - Suburb - A string attribute containing the suburb name of an address.
    - Postcode - An integer attribute containing the postcode of an address.
  + PayingCustomer - A class representing a paying customer.
    - PaymentMethod - A string attribute representing which type of payment method the customer uses (Card/Bank).
    - PaymentDetail - An integer attribute containing the card/bank number of the customer.
    - AssociateCustomers - An ArrayList of AssociateCustomer object(s)
  + AssociateCustomer - A class representing all the associate customer(s) the paying customer is paying the magazine for.
* Controller - Holds all controller classes for the application
* Fxml - Holds all view classes for the application
* Model - Holds all model classes for the application
* Helper - Holds all other classes that do not fit into the above categories

## Diagrams

A screenshot of a computer screen

Description automatically generated

Figure 10 Class Diagram

A screenshot of a computer

Description automatically generated

Figure 11 State Trasition Diagram

## Pseudocode

Pseudocode for showing Supplement or Customer information on View mode’s information panel.

Function initialize:

    Create a root node for magazine service

    Create a child node for magazine information

    Add weekly cost to magazine information

    Add magazine information to root node

    Create a node for supplements

    For each supplement in the magazine model:

        Add supplement name to supplements node

    Add supplements node to root node

    Set root node as the root of the supplement list

    Create a root node for customers

    For each customer in the magazine model:

        Add customer email to customers node

    Set customers node as the root of the customer list

    Add a change listener to the supplement list that does the following when a supplement is selected:

        If newValue is not null and is a descendant of supplementsInfo:

            Get the selected supplement from the magazine model using newValue

            Create a node for supplement information

            Add supplement name and weekly cost to supplement information

            Set supplement information as the root of the info panel

    Add a change listener to the customer list that does the following when a customer is selected:

        If newValue is not null and its parent is rootCustomer:

            Get the selected customer from the magazine model using newValue

            Create a task that does the following:

                Create a node for customer information

                Add customer type, name, and email to customer information

                Create a node for address information

                Add street number, street name, suburb, and postcode to address information

                Add address information to customer information

                Create a node for supplements

                For each supplement in selectedCustomer:

                    Add supplement name to supplements node

                Add supplements node to customer information

                If selectedCustomer is a paying customer:

                    Create a node for associate customers

                    For each associate customer in payingCustomer:

                        Add associate customer email to associate customers node

                    Add associate customers node, payment method, and payment detail to customer information

                    Add billing history to customer information

                Else if selectedCustomer is an associate customer:

                    Add paying customer email to customer information

                Return customer information

            When the task succeeds:

                Set the result of the task as the root of the info panel

            When the task fails:

                Print the error message

                Display an error alert to the user

            Start the task

Pseudocode for a function that handles updating of associate customers. It has to ensure that changes to paying customer will cascade.

Function editAssociateCustomer(associateCustomer, name, address, email, selectedSupplements, selectedPayingCustomer)

    oldPayingCustomer = getPayingCustomer(associateCustomer)

    if oldPayingCustomer is not equal to selectedPayingCustomer then

        removeAssociateCustomer(oldPayingCustomer, associateCustomer)

        addAssociateCustomer(selectedPayingCustomer, associateCustomer)

    setName(associateCustomer, name)

    setAddress(associateCustomer, address)

    setEmail(associateCustomer, email)

    setSupplements(associateCustomer, selectedSupplements)

    setPayingCustomer(associateCustomer, selectedPayingCustomer)

Pseudocode for a function that handles deletion of supplements. It has to ensure that deletion of supplements cascades to all affected customers.

Function deleteSupplement(supplement)

    for each customer in getCustomers(magazine)

        removeSupplement(customer, supplement)

    removeSupplement(getSupplements(magazine), supplement)

Pseudocode for a function that handles deletion of customers. It has to ensure that deletion of customers cascades to all affected customers.

Function deleteCustomer(customer)

    removeCustomer(getCustomers(magazine), customer)

    if getCustomerType(customer) equals "Paying" then

        payingCustomer = castToPayingCustomer(customer)

        for each associateCustomer in getAssociateCustomers(payingCustomer)

            removeCustomer(getCustomers(magazine), associateCustomer)

    else

        payingCustomer = getPayingCustomer(castToAssociateCustomer(customer))

        removeAssociateCustomer(payingCustomer, castToAssociateCustomer(customer))

# Limitations

There are no limitations found with regards to the required functionality.

# Testing

Since this application will start in create mode and not allow user to switch to view or edit mode unless there is a magazine service, the testing will be done in the following sequence; Create, View, Edit.

The test strategy is to first test the positive scenario where the application is given all the correct/valid inputs. This is to first ensure the application can work as expected given an expected scenario. Next, the application will be tested on the negative scenario, where incorrect/invalid inputs are given, and the application are expected to handle these scenarios gracefully.

In create mode, there are two main functions to test, creating and saving of a new magazine service, and the loading of existing magazine service.

The testing of creating a new magazine service involves the creation of new magazine, supplement(s), and customer(s) of both paying and associate type.

In view mode, the test is to ensure that the magazine, supplements, and customer details are visible and correct.

In edit mode, we test the functions in the following sequence: create, edit, delete. This is so that we can create new supplements and customers before editing the newly created supplements and customers to test the edit function, lastly, deleting the newly created supplements and customers to test the delete functions.

## Create Magazine Service

### Create Magazine

1. Blank or empty cost inputs are rejected.

A computer screen shot of a computer screen

Description automatically generated

Figure 12 Test Empty Magazine Cost

1. None double typed inputs are rejected.

A computer screen shot of a computer

Description automatically generated

Figure 13Test Invalid Magazine Cost Input

1. Switches to Create Supplement form if the input is a valid double type.

A screenshot of a computer

Description automatically generated

Figure 14 Test Valid Magazine Cost Input

A screenshot of a computer program

Description automatically generated

Figure 15 Test Valid Magazine Cost Input

### Create Supplement(s)

1. Blank Supplement Name or Supplement Cost are rejected.

A screenshot of a computer

Description automatically generated

Figure 16 Test Mandatory Fields

1. Non-double typed inputs are rejected for Supplement Cost.

A screenshot of a computer

Description automatically generated

Figure 17 Test Invalid Supplement Cost Input

1. Check that ‘Create Another Supplement’ button works by giving a valid supplement name and cost, the scene will remain the same.

A screenshot of a computer program

Description automatically generated

Figure 18 Test Create Another Supplement button

A screenshot of a computer

Description automatically generated

Figure 19 Test Result

1. Duplicate Supplement Names are rejected.

A screenshot of a computer

Description automatically generated

Figure 20 Test Duplicate Supplement Name

1. Check that ‘Finish creating Supplement’ buttons work by giving a valid supplement name and cost, the scene should switch to create customer form.

A screenshot of a computer program

Description automatically generated

Figure 21 Test Finish Creating Supplement Button

A screenshot of a computer

Description automatically generated

Figure 22 Test Result

### Create Customer(s)

#### Create Paying Customer(s)

1. Check that the previously created supplements are correctly shown.

A screenshot of a computer

Description automatically generated

Figure 23 Check Supplement List

1. Blank name, email, street number, street name, or postcode are rejected.

A screenshot of a computer

Description automatically generated

Figure 24 Test Mandatory Fields

1. Users are alerted that attempting to create associate customer by clicking on ‘Create Associate Customer’ button when there are no existing paying customer are not allowed.

A screenshot of a computer

Description automatically generated

Figure 25 Test Create Associate Customer Button without Existing Paying Customer

1. Names with numbers are rejected, only English letters and spaces are allowed.



Figure 26 Test Invalid Customer Name

1. Email with wrong formats are rejected.



Figure 27 Test Invalid Customer Email Format



Figure 28 Test Invalid Customer Email Format

1. Street Number that are not integers are rejected.

A white rectangular object with a white border

Description automatically generated

Figure 29 Test Invalid Street Number



Figure 30 Test Invalid Street Number

1. Postcode that are not integers are rejected.



Figure 31 Test Invalid Postcode



Figure 32 Test Invalid Postcode

1. Check that ‘Create Paying Customer’ Button works by giving valid inputs for all mandatory fields only.

A screenshot of a computer

Description automatically generated

Figure 33 Test Create Paying Customer Mandatory Only

A screenshot of a computer

Description automatically generated

Figure 34 Scene Switches to Create Paying Customer

1. That Payment Method and Payment Detail are mandatory.

A screenshot of a computer

Description automatically generated

Figure 35 Test Mandatory Fields

1. Check that Payment Details that are not Integer are rejected.



Figure 36 Test Invalid Payment Detail



Figure 37 Test Invalid Payment Detail

1. Check that ‘Create Another Customer’ button works by giving valid inputs for all fields, the scene should switch back to Create Customer form if successful.

A screenshot of a computer

Description automatically generated

Figure 38 Test Create Another Customer Button

A screenshot of a computer

Description automatically generated

Figure 39 Test Result

1. Check that ‘Create Paying Customer’ Button works by giving valid inputs for all fields, test that user can select both single and multiple supplements.

A screenshot of a computer

Description automatically generated

Figure 40 Test Create Paying Customer with All Fields Valid Input

A screenshot of a computer

Description automatically generated

Figure 41 Create Another Customer

A screenshot of a computer

Description automatically generated

Figure 42 Test Multiple Supplement Selection

A screenshot of a computer

Description automatically generated

Figure 43 Create Another Customer

#### Create Associate Customer(s)

1. Check that ‘Create Associate Customer’ button works when given valid inputs for all mandatory fields.

A screenshot of a computer

Description automatically generated

Figure 44 Test Create Associate Customer with all Mandatory Fields only

A screenshot of a computer

Description automatically generated

Figure 45 Scene Switch to Create Associate Customer when Successful

1. Check that mandatory fields are required.

A screenshot of a computer

Description automatically generated

Figure 46 Test Mandatory Field

1. Check that ‘Create Another Customer’ button works after selecting a paying customer.

A screenshot of a computer

Description automatically generated

Figure 47 Test Create Another Customer

A screenshot of a computer

Description automatically generated

Figure 48 Scene Switch to Create Customer when Successful

1. Repeat step 13 twice but with all fields and 1 supplement once and 2 supplements the other. Spread the associate customer out so that 1 paying customer would have 0 associate customer, 1 paying customer would have 1 associate customer and 1 paying customer would have 2 associate customers. Stop before creating the last associate customer to test ‘Finish Creating customer’ button.

A screenshot of a computer

Description automatically generated

Figure 49 Test All Fields Input with 1 Supplement

A screenshot of a computer

Description automatically generated

Figure 50 Create Another Customer

A screenshot of a computer

Description automatically generated

Figure 51 Test Create Associate Customer all Fields Input with Multiple Supplement

1. Test ‘Finish Creating Customer’ to end creation of Magazine Service.

A screenshot of a computer

Description automatically generated

Figure 52 Test Finish Creating Customer Button

A screenshot of a computer

Description automatically generated

Figure 53 Save Created Magazine Service to File

## Load Magazine Service

1. Restart the application so that no magazine service are loaded.
2. Click on ‘Load Magazine’ button and select the previously created.

A screenshot of a computer

Description automatically generated

Figure 54 Test Loading of Magazine Service

A screenshot of a computer

Description automatically generated

Figure 55 Name of the loaded file is displayed

## View Magazine Service

1. Check that all the created Magazine, Supplements and Customers have the correct details.

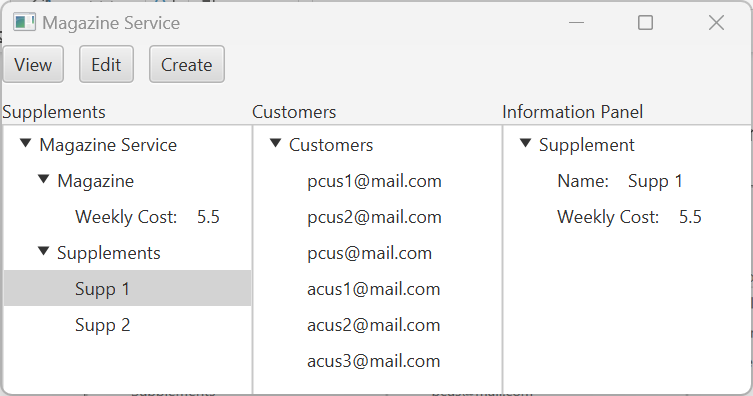


Figure 56 Check Supplement

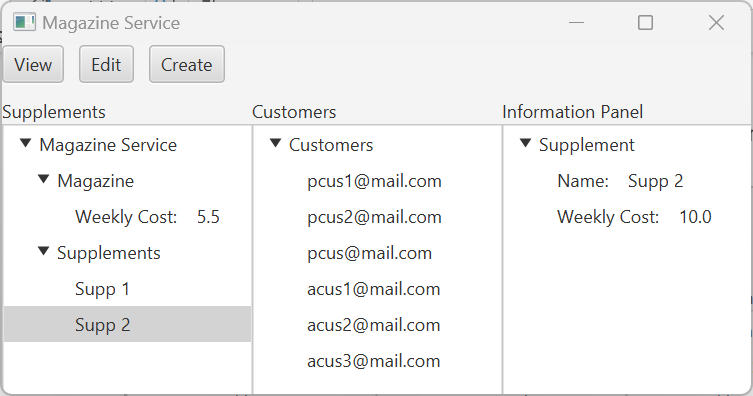


Figure 57 Check Supplement

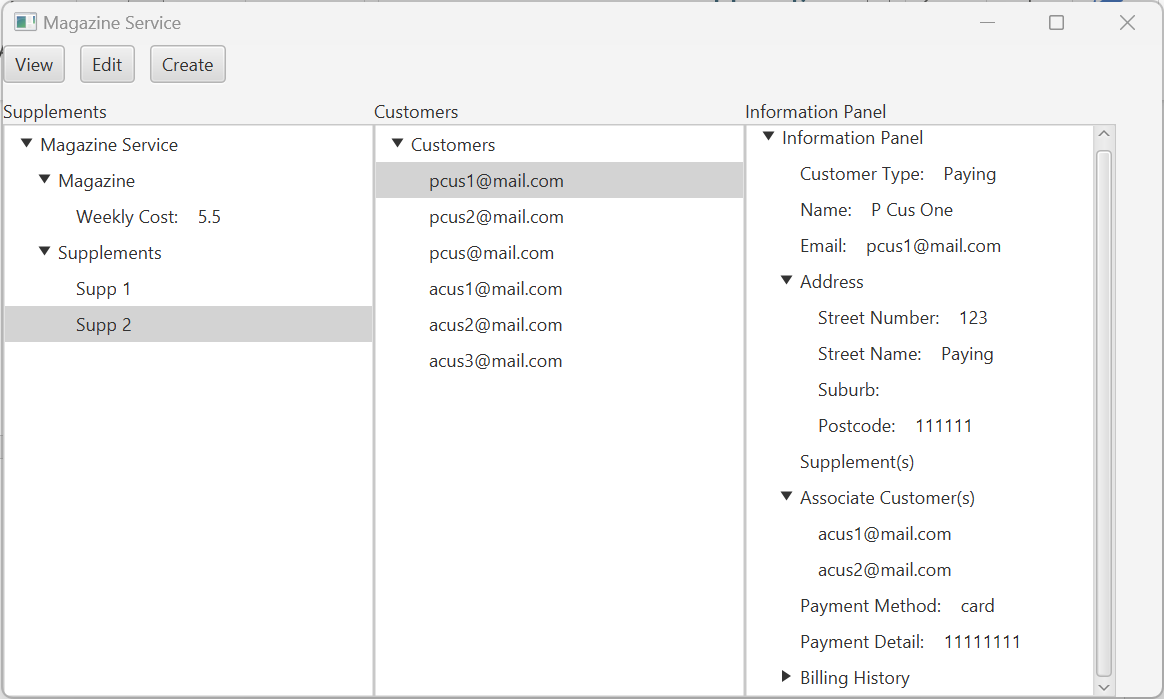


Figure 58 Check Paying Customer

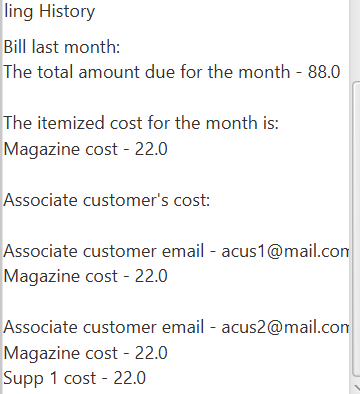


Figure 59 Check Billing History

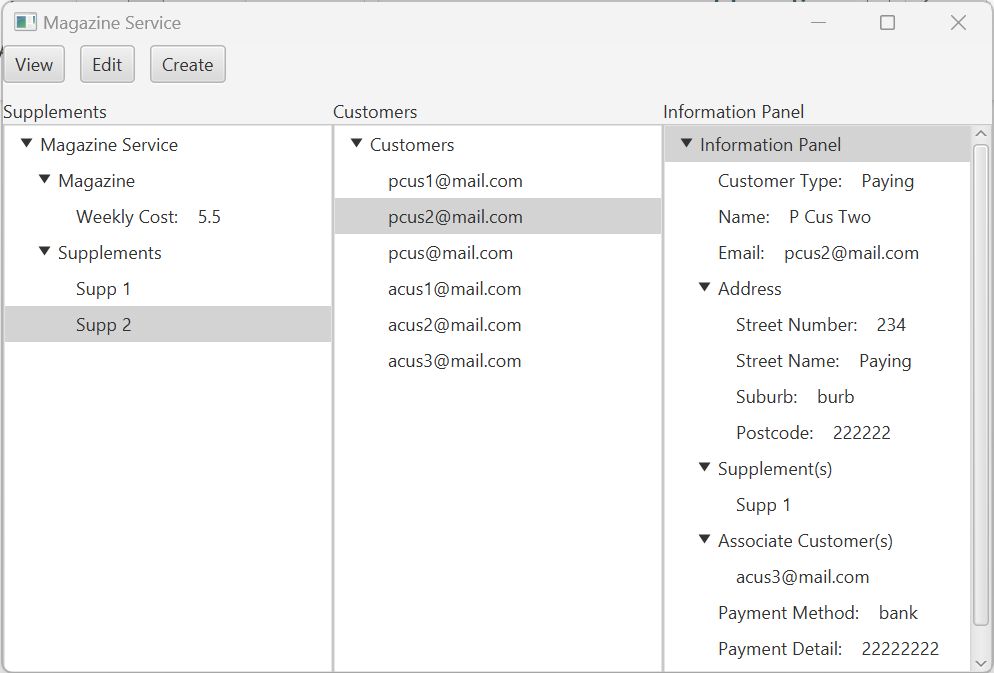


Figure 60 Check Paying Customer

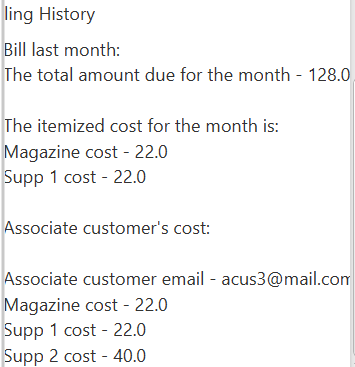


Figure 61 Check Billing History

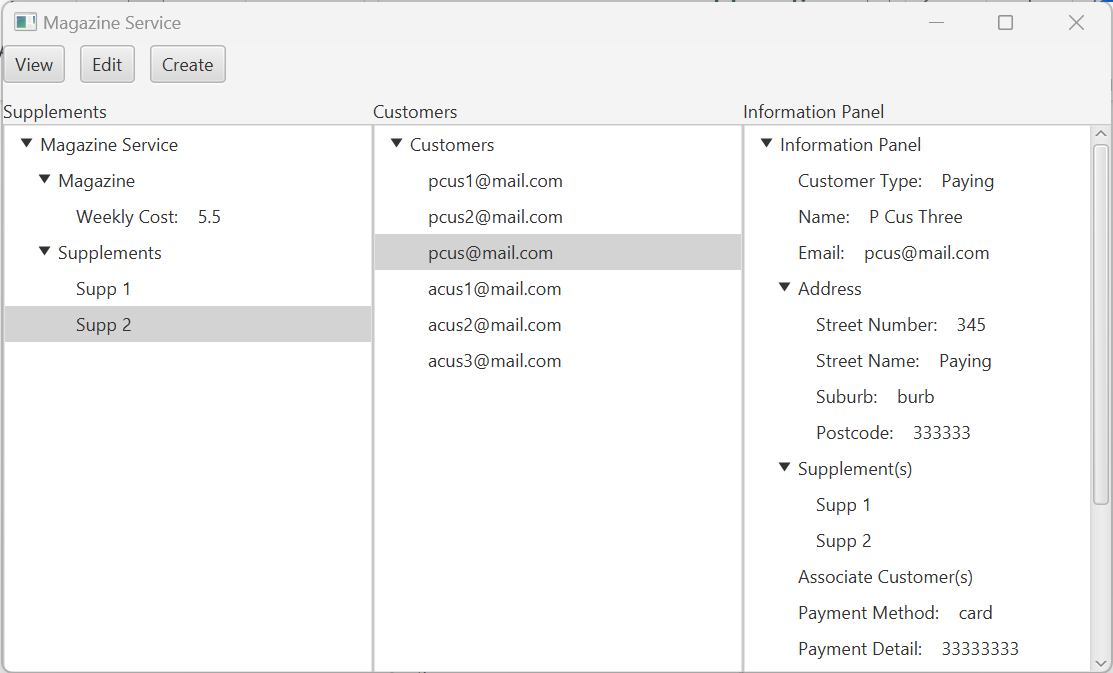


Figure 62 Check Paying Customer

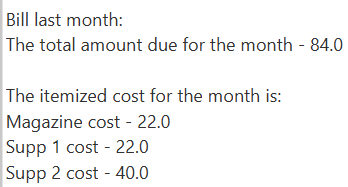


Figure 63 Check Billing History

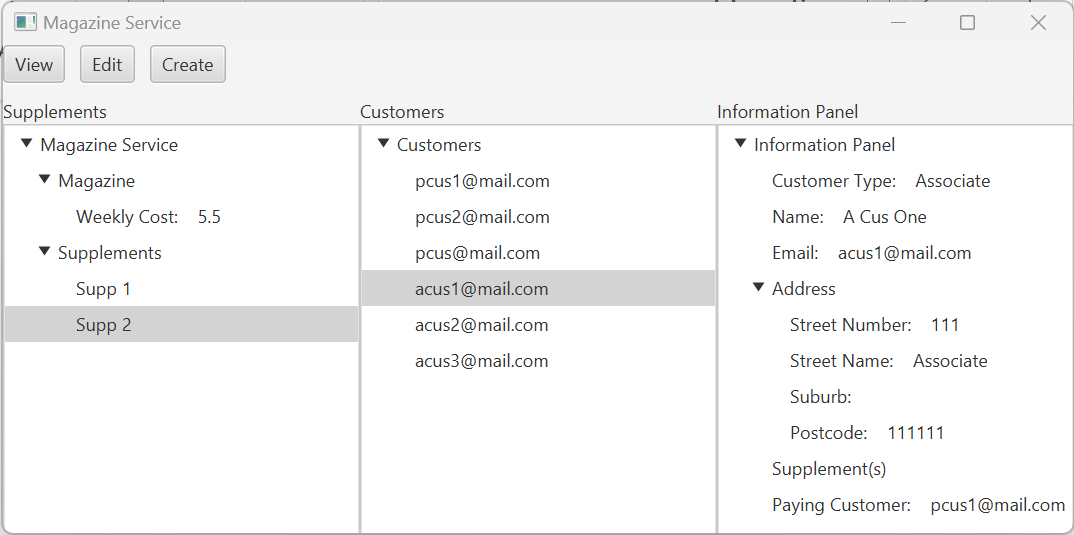


Figure 64 Check Associate Customer

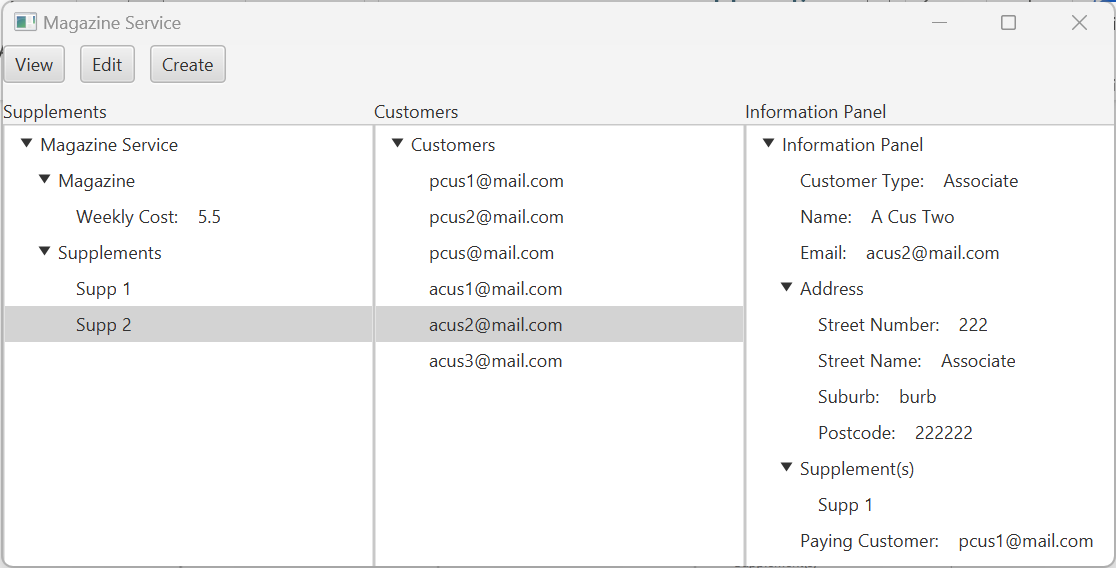


Figure 65 Check Associate Customer

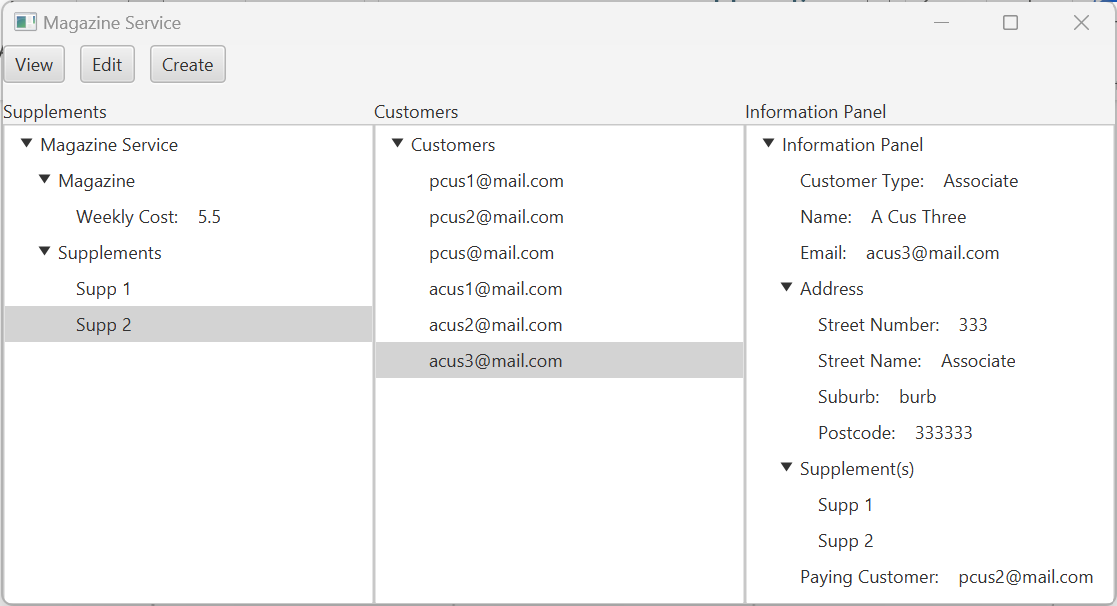


Figure 66 Check Associate Customer

## Add Supplement

1. Switch to Edit Mode and click on ‘Add Supplement’ button.

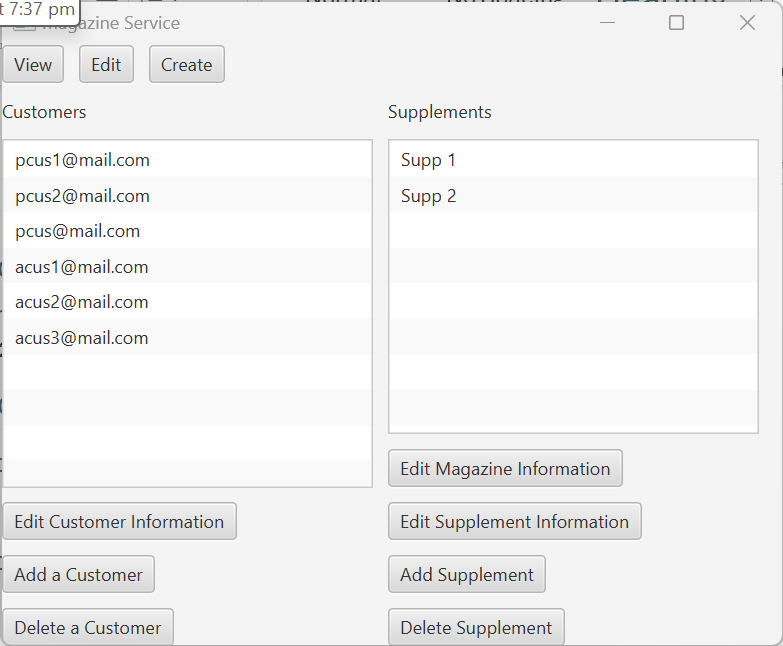


Figure 67 Edit Mode

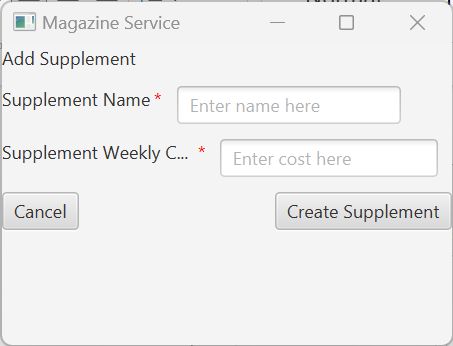


Figure 68 Scene Switch to Add Supplement

1. Check that empty inputs in mandatory fields are rejected.

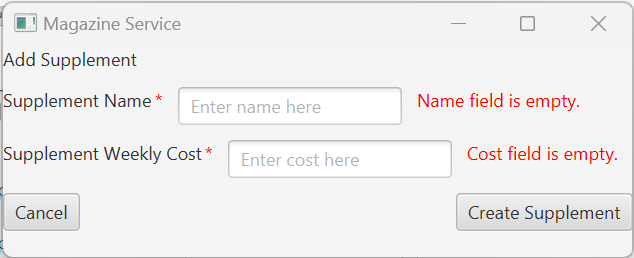


Figure 69 Check Mandatory Fields

1. Check that Duplicate supplement names are rejected.



Figure 70 Check Duplicate

1. Check that Weekly Cost that is not double type is rejected.



Figure 71 Check Invalid Supplement Cost

1. Check that ‘Create Supplement’ button works when all fields are given valid inputs

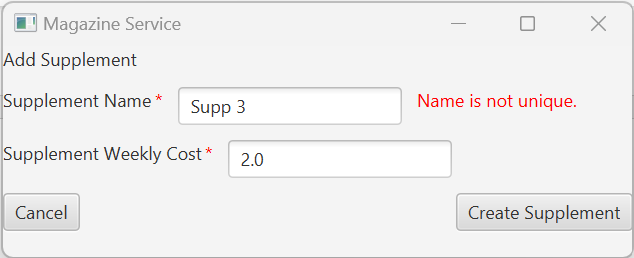


Figure 72 Test Valid Supplement Creation

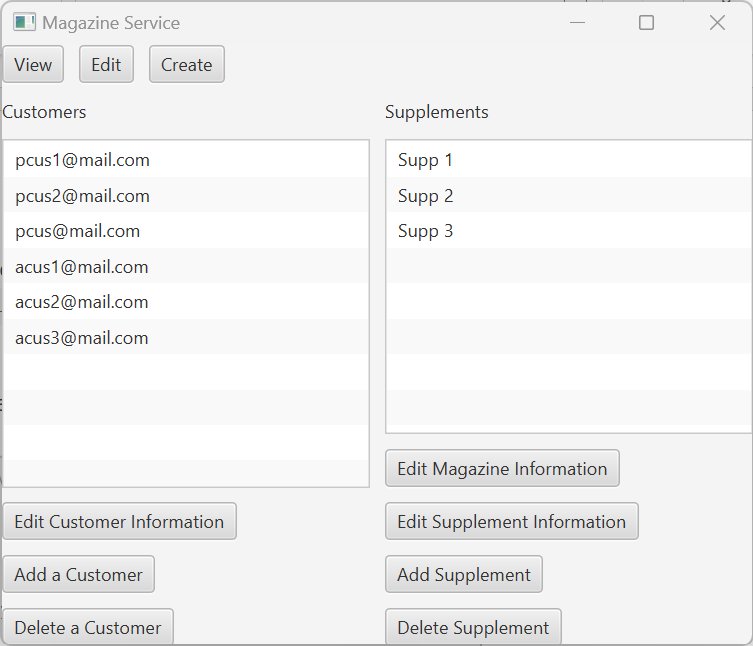


Figure 73 Check New Supplement Exist

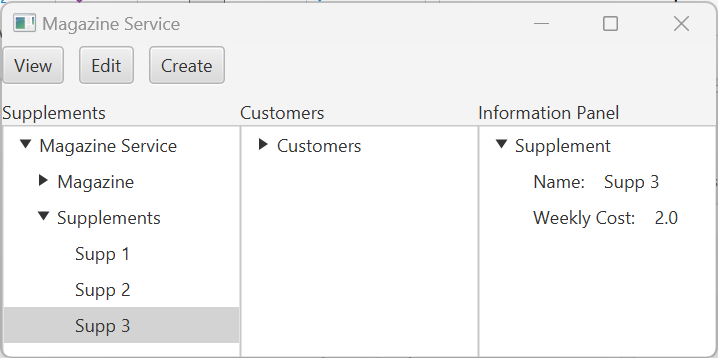


Figure 74 Check New Supplement Details in View Mode

## Add Customer(s)

### Add Paying Customer

1. Click on ‘Add Customer’ button in Edit mode.

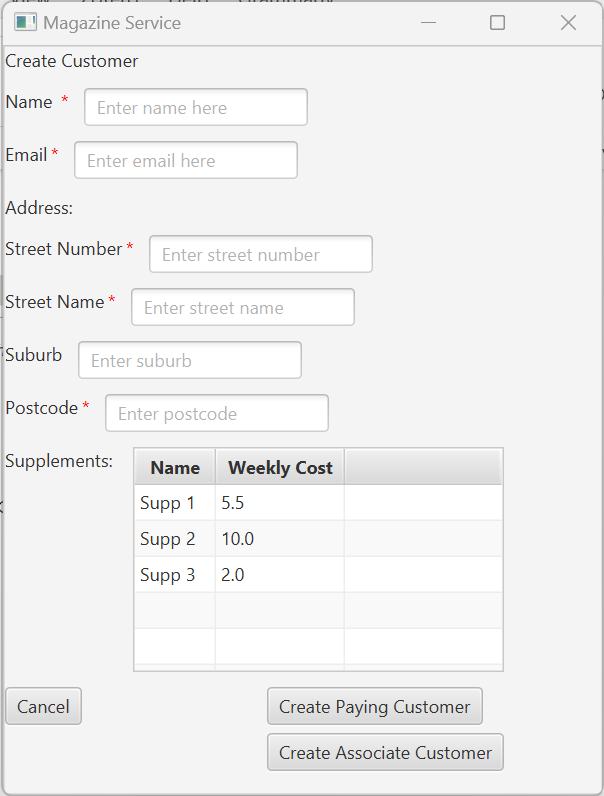


Figure 75 Check Add Customer Button & Supplement List

1. Check empty inputs for mandatory fields are rejected.

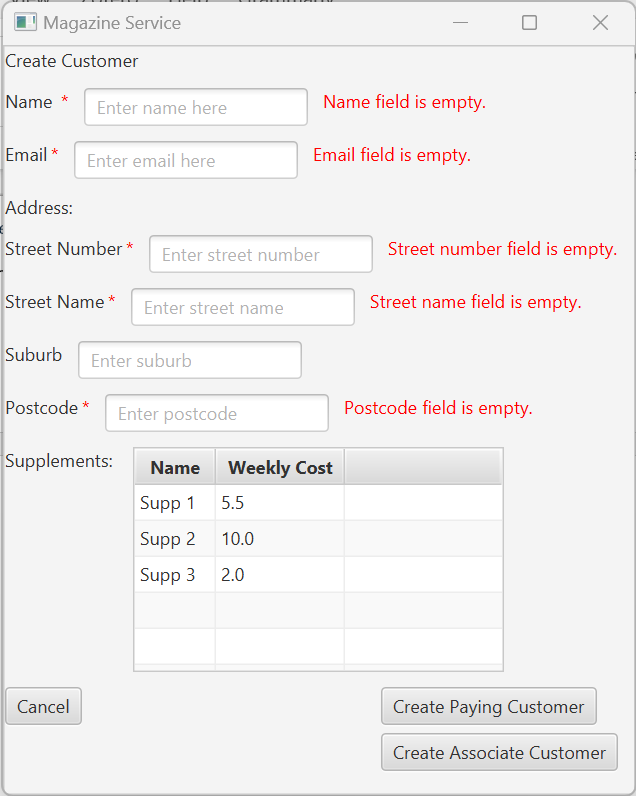


Figure 76 Check Mandatory

1. Check that name does not accepts numbers as inputs.



Figure 77 Check Invalid Name

1. Check that invalid email formats are rejected.



Figure 78 Check Invalid Email Format



Figure 79 Check Invalid Email Format

1. Check that duplicate emails are rejected.



Figure 80 Check Duplicate Email

1. Check that street number only accepts integers as inputs.



Figure 81 Check Invalid Street Number



Figure 82 Check Invalid Street Number

1. Check postcode only accepts integers as inputs.



Figure 83 Check Invalid Postcode



Figure 84 Check Invalid Postcode

1. Check that ‘Create Paying Customer works when given valid inputs for all fields with multiple selection of supplements.

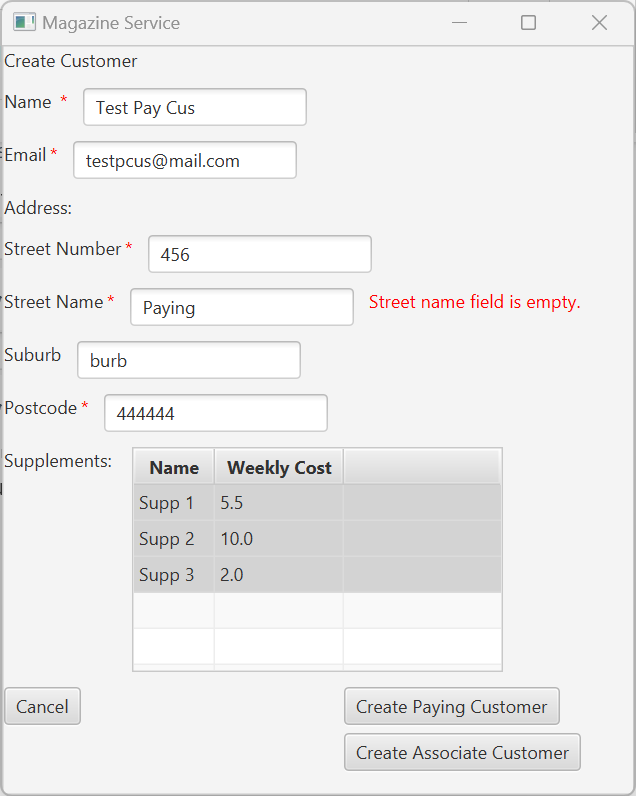


Figure 85 Check Create Paying Customer

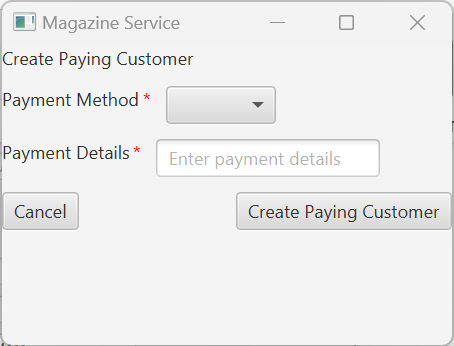


Figure 86 Scene Switch to Create Paying Customer

1. Check empty inputs are rejected for mandatory fields.

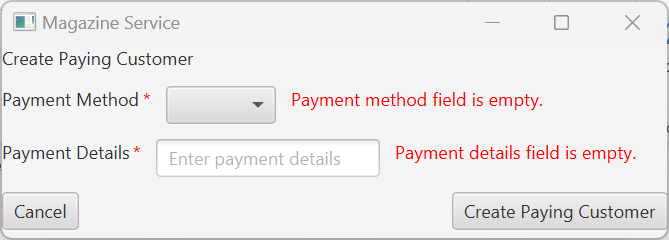


Figure 87 Check Mandatory

1. Check that payment details only accept integers as input.



Figure 88 Check Invalid Payment Details



Figure 89 Check Invalid Payment Details

1. Check ‘ Create Paying Customer’ button works when given valid inputs for all fields

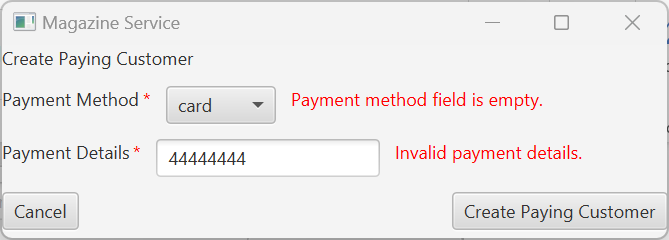


Figure 90 Check Create Paying Customer

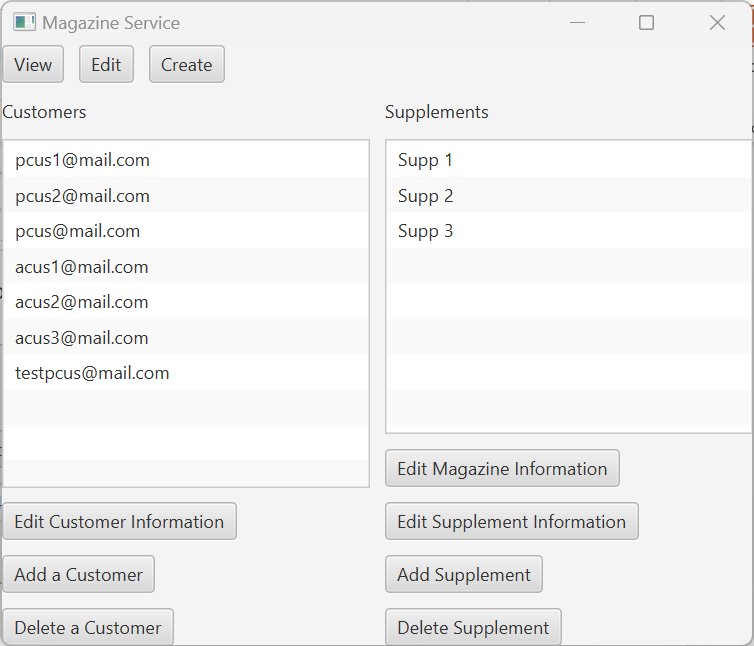


Figure 91 Check New Customer Exist in Edit Mode

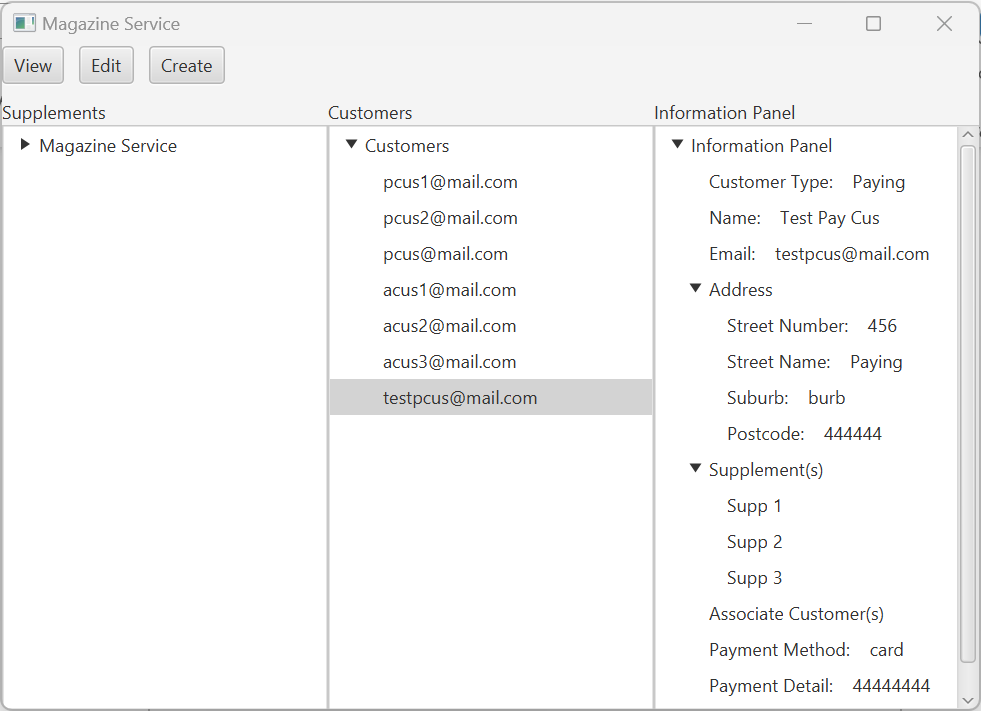


Figure 92 Check Customer Information in View Mode

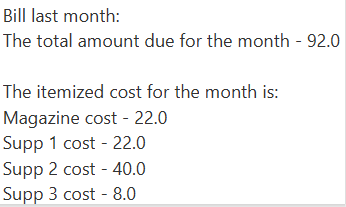


Figure 93 Check Billing History

### Add Associate Customer

1. Click on ‘Add Customer’ button in Edit Mode.
2. Check that empty inputs for mandatory fields are rejected.

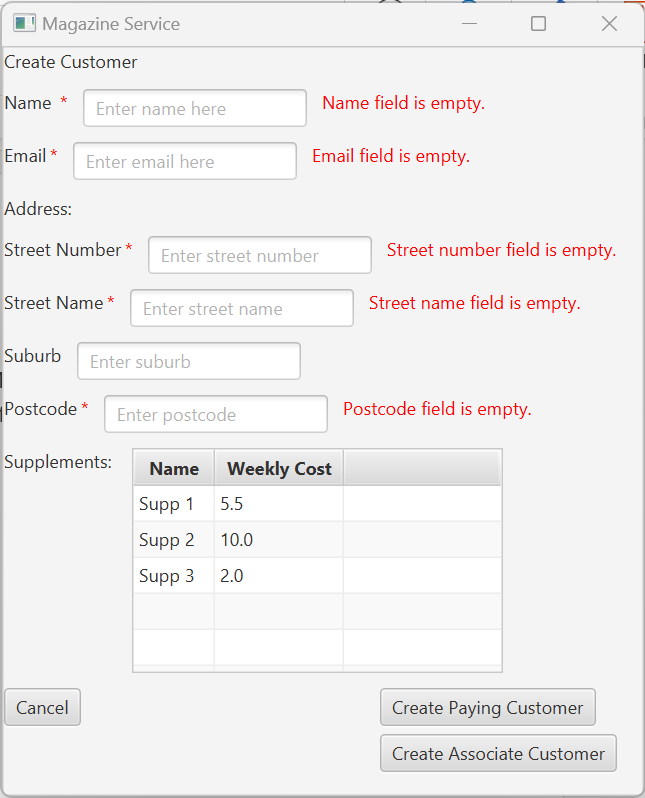


Figure 94 Check Mandatory

1. Check that numbers are rejected for name.



Figure 95 Check Invalid Name

1. Check that invalid email formats are rejected.



Figure 96 Check Invalid Email Format



Figure 97 Check Invalid Email format

1. Check that only integers are accepted for street number.



Figure 98 Check Invalid Street Number



Figure 99 Check Invalid Street Number

1. Check that only integers are accepted for postcode.



Figure 100 Check Invalid Postcode



Figure 101 Check Invalid Postcode

1. Check that ‘Create Associate Customer’ button works when the form is given valid inputs for all fields.

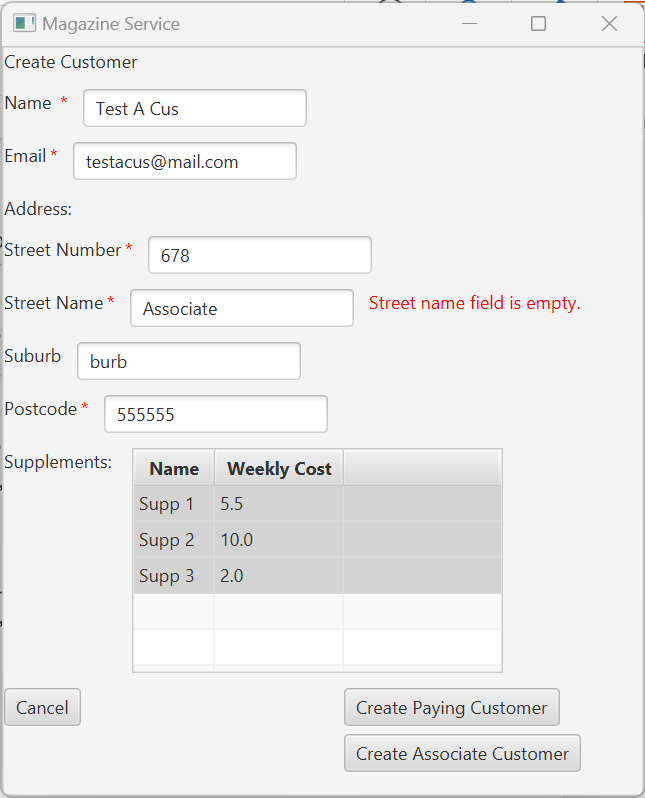


Figure 102 Check Create Associate Customer with all valid inputs

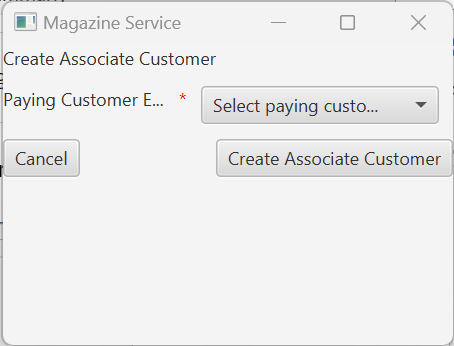


Figure 103 Scene Switch to Create Associate Customer

1. Check that empty inputs are rejected for mandatory fields

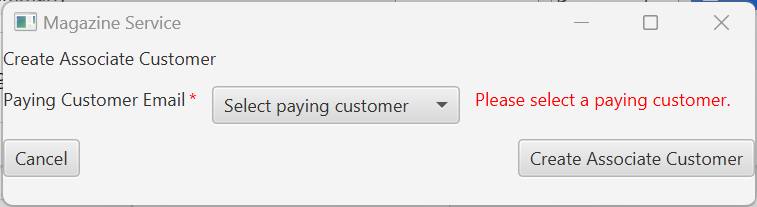


Figure 104 Check Mandatory

1. Check that ‘Create Associate Customer’ button works when given valid inputs for all fields.

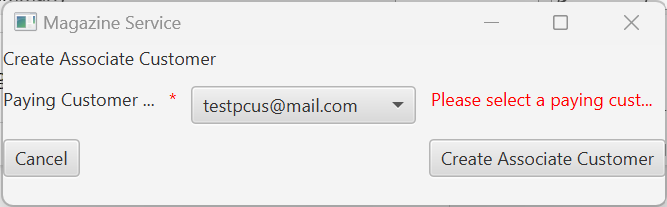


Figure 105 Check Create Associate Customer Button with Valid Inputs

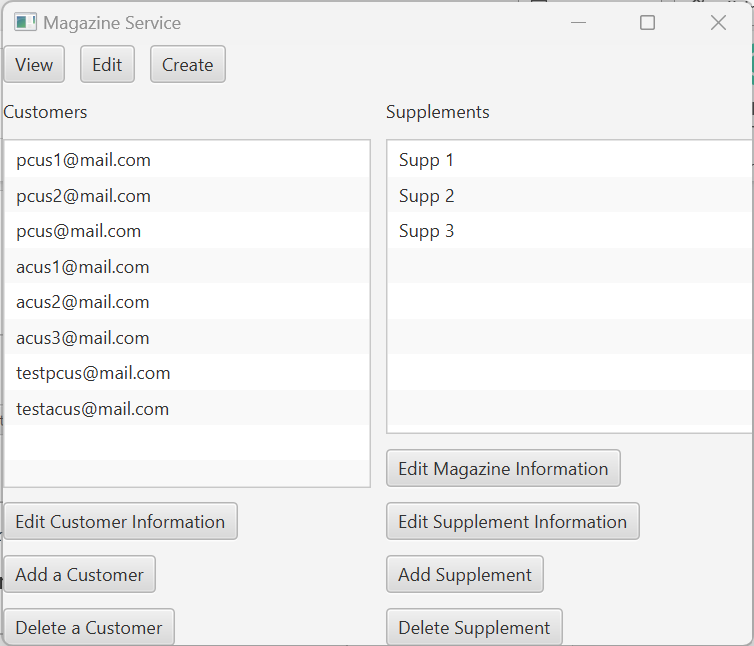


Figure 106 Check New Customer Exist in Edit Mode

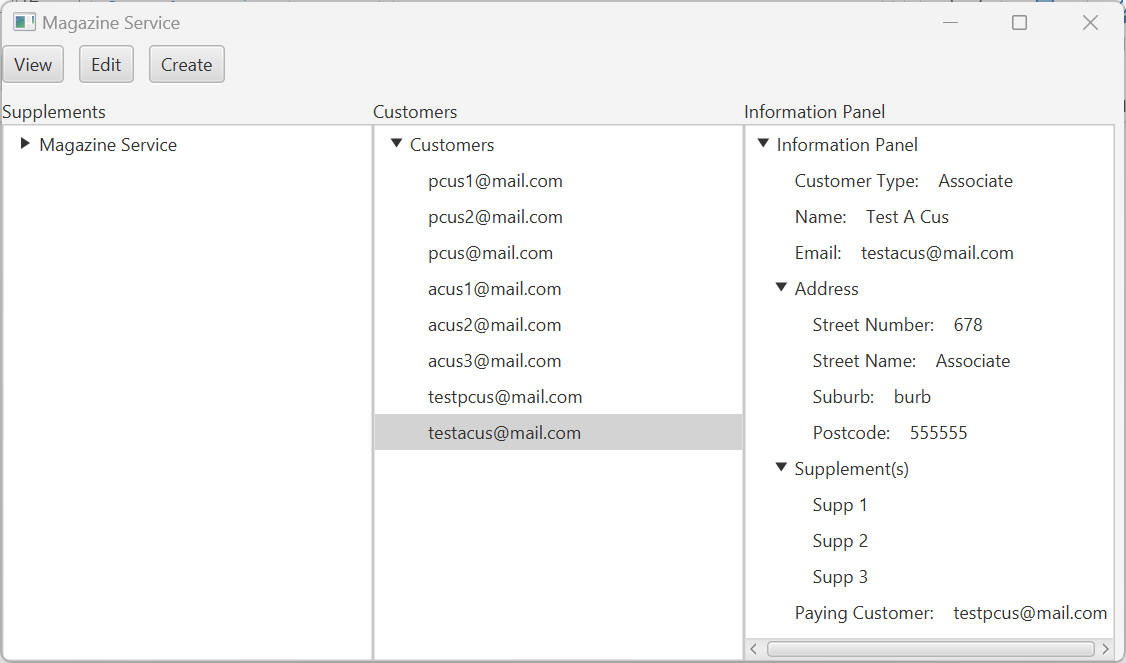


Figure 107 Check New Associate Customer's Information in View Mode

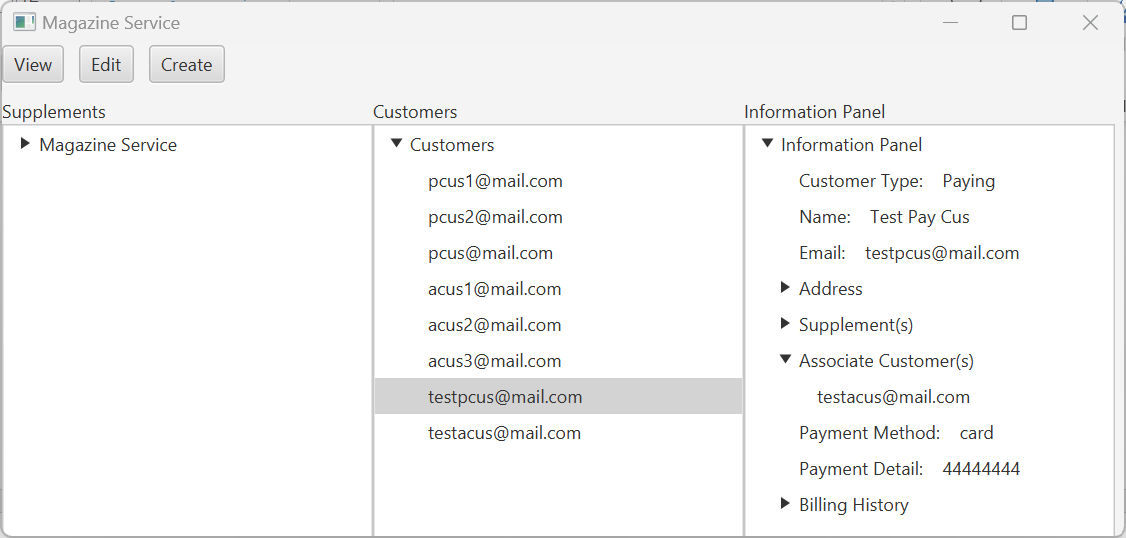


Figure 108 Check New Associate Customer's Paying Customer's List of Associate Customer

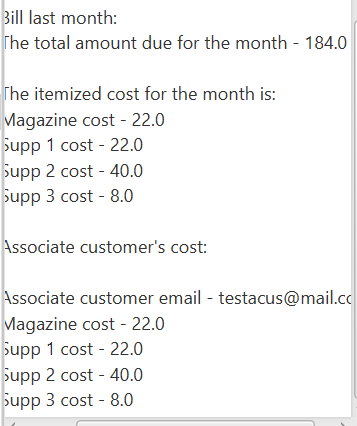


Figure 109 Check that the linked Paying Customer's Billing History contains the New Associate Customer's Cost

## Edit Magazine

1. Click on ‘Edit Magazine Information’ button, the cost should be pre-filled with the existing magazine cost.

A screenshot of a service

Description automatically generated

Figure 110 Check pre-filled cost is correct

1. Check that empty inputs for cost is rejected.

A computer screen shot of a computer screen

Description automatically generated

Figure 111 Test Mandatory

1. Check that only double type inputs are allowed.

A screenshot of a computer

Description automatically generated

Figure 112 Test Invalid Cost

1. Check that the magazine cost is updated when given a valid cost input and clicking on ‘save’ button.

A screenshot of a computer

Description automatically generated

Figure 113 Test Valid Cost Update

A screenshot of a magazine service

Description automatically generated

Figure 114 Check Cost Updated in View Mode

A screenshot of a phone

Description automatically generated

Figure 115 Check Bill History for Customers are also Updated

## Edit Supplement

1. Check that user is warned that a supplement must be selected to use ‘Edit Supplement Information’ button.

A white background with black text

Description automatically generated

Figure 116 Check User is Alerted to Select a Supplement

1. Select ‘Supp 3’ and click on ‘Edit Supplement Information’ button again.

A screenshot of a service

Description automatically generated

Figure 117 Check pre-filled fields are Correct

1. Check that empty inputs are not rejected for all mandatory fields.

A screenshot of a computer

Description automatically generated

Figure 118 Test Mandatory

1. Check that duplicate supplement name are rejected.



Figure 119 Test Duplicate

1. Check that only double type inputs are allowed for cost.



Figure 120 Check Invalid Cost

1. Check the supplement is correctly updated given valid input and clicking ‘Save’ button.

A screenshot of a computer

Description automatically generated

Figure 121 Check Valid Input is Updating Supplement

A screenshot of a computer

Description automatically generated

Figure 122 Check Edit Mode Supplement List Updated

A screen shot of a computer

Description automatically generated

Figure 123 Check Supplement Information Updated in View Mode

A screenshot of a computer

Description automatically generated

Figure 124 Check Paying Customer Bill Information is Updated

## Edit Customer(s)

1. Check that the user is warned when ‘Edit Customer Information’ button is clicked without any customer selected.

A white screen with black text

Description automatically generated

### Edit Paying Customer

1. Select ‘testpcus@mail.com’ and click ‘Edit Customer Information’ Button

A screenshot of a computer

Description automatically generated

Figure 125 Check that Form is Pre-filled with existing information

1. Test that empty inputs for mandatory fields are not allowed.

A screenshot of a computer screen

Description automatically generated

Figure 126 Check Mandatory

1. Test that name do not allow numbers.



Figure 127 Check Invalid Name

1. Test that invalid email formats are not allowed.



Figure 128 Check Invalid Email

A close up of a message

Description automatically generated

Figure 129 Check Invalid Email

1. Test that duplicates emails are not allowed.



Figure 130 Check Duplicate

1. Check that street number only allows integers.



Figure 131 Check Invalid Street Number



Figure 132 Check Invalid Street Number

1. Check that postcode only allows integers.



Figure 133 Check Invalid Postcode



Figure 134 Check Invalid Postcode

1. Check that payment detail only allows integers.



Figure 135 Check Invalid Payment Detail



Figure 136 Check Invalid Payment Detail

1. Check that, when given all valid inputs the selected paying customer’s details are updated accordingly.

A screenshot of a computer

Description automatically generated

Figure 137 Check Updating Paying Customer

A screenshot of a computer

Description automatically generated

Figure 138 Check Customer List in Edit Mode

A screenshot of a computer

Description automatically generated

Figure 139 Check Customer Information In View Mode

A screenshot of a computer

Description automatically generated

Figure 140 Check Bill History

### Edit Associate Customer

1. Select ‘testacus@mail.com’ and click ‘Edit Customer Information’ button.

A screenshot of a computer

Description automatically generated

Figure 141 Check Form is Pre-filled with Existing Information

1. Check that empty inputs are rejected for mandatory fields.

A screenshot of a computer

Description automatically generated

Figure 142 Check Mandatory

1. Check that name does not allow numbers.



Figure 143 Check Invalid Name

1. Check that duplicate emails are not allowed.



Figure 144 Check Duplicate

1. Check that invalid email formats are rejected.



Figure 145 Check Invalid Email Format



Figure 146 Check Invalid Email Format

1. Check that only integers are allowed for Street Number.



Figure 147 Check Invalid Street Number



Figure 148 Check Invalid Street Number

1. Check that only integers are allowed for Postcode.



Figure 149 Check Invalid Postcode



Figure 150 Check Invalid Postcode

1. Check that, when given valid inputs for all fields, the associate customer’s information is correctly updated.

A screenshot of a computer

Description automatically generated

Figure 151 Check Updating Associate Customer with Valid Inputs

A screenshot of a computer

Description automatically generated

Figure 152 Check Customer List in Edit Mode

A screenshot of a computer

Description automatically generated

Figure 153 Check Updated Information in View Mode

A screenshot of a computer

Description automatically generated

Figure 154 Check Linked Paying Customer Information is updated

A screenshot of a computer

Description automatically generated

Figure 155 Check Linked Paying Customer's Billing History

## Delete Supplement(s)

Before deleting supplements, save a backup copy the save file.

1. Check that user is alerted then trying to delete a supplement without selecting a supplement.

A white background with black text

Description automatically generated

Figure 156 No selected Supplement Alert

1. Check that supplement is successfully deleted after selecting ‘Test Supp’ and clicking ‘Delete Supplement’ button.

A screenshot of a computer

Description automatically generated

Figure 157 Click on OK to Delete Selected Supplement

A screenshot of a computer

Description automatically generated

Figure 158 Check that Supplement is Removed from Supplement List in Edit Mode

A screenshot of a computer

Description automatically generated

Figure 159 Check that Supplement is Removed from Supplement List in View Mode

A screenshot of a computer

Description automatically generated

Figure 160 Check that Supplement is Removed from Paying Customer

A screenshot of a computer

Description automatically generated

Figure 161 Check that Supplement is Removed from Associate Customer

A screenshot of a computer

Description automatically generated

Figure 162 Check that the Removed Supplement from Associate Customer is Reflected in the Linked Paying Customer

## Delete Customer(s)

Before deleting customers, save a backup copy of the save file.

1. Check that user is alerted when trying to delete a customer without selecting a customer.

A white screen with black text

Description automatically generated

Figure 163 Alert User to Select a Customer First

### Delete Associate Customer

1. Select ‘atestcus@mail.com’ and click on ‘Delete a customer’ Button.

A screenshot of a chat

Description automatically generated

Figure 164 Confirmation to Delete Selected Customer

A screenshot of a computer

Description automatically generated

Figure 165 Check Deleted Customer is removed from Customer List in Edit Mode

A screenshot of a computer

Description automatically generated

Figure 166 Check that Deleted Customer is Removed from Customer List in View Mode

A screenshot of a computer

Description automatically generated

Figure 167 Check that Deleted Associate Customer is Removed from Linked Paying Customer

A white background with black text

Description automatically generated

Figure 168 Check that the Paying Customer's Billing History Removed that Deleted Associate Customer's Cost

### Delete Paying Customer

1. Reload the magazine service from your backup save file.
2. Select ‘pcus@mail.com’ and click on ‘Delete a customer’ Button.

A screenshot of a chat box

Description automatically generated

Figure 169 Confirmation Box to Delete Selected Customer

A screenshot of a computer

Description automatically generated

Figure 170 Check that the Selected Paying Customer and its Associate Customer are Removed from Customer List in Edit Mode

A screenshot of a computer

Description automatically generated

Figure 171 Check that the selected Paying Customer and its Associate Customer are Removed from Customer List in View Mode