
Project:
WWW-Based Pizza Order Application

Software Requirements Specification
(SRS)

August 1996

Version 1.0

Contents

1	Introduction	1
1.1	Purpose of SRS	1
1.2	Scope of Product	1
1.3	Definitions, Acronyms and Abbreviations	2
1.4	References	2
1.5	Document Overview	3
2	General Description	4
2.1	Product Perspective	4
2.2	System Functionality	5
2.3	User Characteristics	6
2.4	General Constraints	6
2.5	Assumptions and Dependencies	6
3	Specific Requirements	7
3.1	Functional Requirements	7
3.1.1	Class Model	7
3.1.2	Class Associations	11
3.1.3	Class Dictionary	12
3.2	External Interface Requirements	17
3.2.1	User Interfaces	17
3.2.2	Hardware Interfaces	22
3.2.3	Software Interfaces	23
3.3	Non-Functional Requirements	24
3.4	Database Requirements	25
4	Possible Product Evolution	26

Chapter 1

Introduction

1.1 Purpose of SRS

This is the Software Requirements Specification (SRS) for the *Pizza Order* application. The purpose of this document is to convey information about the application's requirements, both functional and non-functional, to the reader. This document provides (a) a description of the environment in which the application is expected to operate, (b) a definition of the application's capabilities, (c) a specification of the application's functional and non-functional requirements, including the (d) user-interface requirements, and (e) a list of tentative future extensions and enhancements to the application.

This document is intended to serve several groups of readers. First, it is anticipated that the SRS will primarily be used by the application designers. Designers will use the information recorded here as the basis for creating the application's design. Second, the client for the project is expected to review this document. The SRS will serve to establish a basis for agreement between the client and development team about the functionality to be provided by the application. Third, the application maintainers will review the document to clarify their understanding of what the application does. Fourth, test planners will use information from this document to derive test plans and test cases. Finally, the project manager will use this document during project planning and monitoring.

1.2 Scope of Product

The purpose of this software development project is to create a new application called: *Pizza Order*. The client for this project, Ethereum Pizza Co., wishes to enter the emerging web-based customer market. The *Pizza Order* application will be web-based, allowing customers to order a pizza for delivery via the world wide web (WWW). This application will provide the following capabilities:

- The application will be accessible via the world wide web.

- The application will allow a customer to place an order for a pizza to be delivered to the customer's location.
- Customers will be able to specify the size of the pizza, specific toppings they prefer, a delivery address, and a method of payment.
- Orders placed by customers will be recorded where the restaurant's kitchen staff can find and process the order.

The project's client has determined that this application will provide the following benefits:

- Provide additional ordering flexibility and convenience to the restaurant's customers.
- Increase the restaurant's share of the home delivery pizza market.
- Allow the restaurant to gain new customers among hungry WWW users.

1.3 Definitions, Acronyms and Abbreviations

This section identifies and explains terms, acronyms, and abbreviations that may appear in this document.

Browser	:	An application that provides the ability to locate, retrieve, and view information contained on the world-wide-web.
Home Page	:	Information on the Internet is presented on pages that are viewed through a browser. The first page in a web document is the home page.
HTML	:	Hyper-Text Mark up Language. Web pages are formatted using this language.
WWW	:	World Wide Web. A vast collection of interconnected documents, spanning the world. This WWW could be described as a global, interactive, dynamic cross-platform, distributed graphical hypertext information system that runs over the Internet.
OOA	:	Object Oriented Analysis.
Object Model	:	Captures the static structure of a system by showing the objects in the system, relationships between the objects, and the attributes and operations that characterize each class of objects.
Web Browser	:	See <i>Browser</i> .

1.4 References

Other documents or material of which the reader should be aware:

None.

1.5 Document Overview

A brief description of the content of each chapter is given below.

Chapter 1 : Introduction - Provides an overview of the project. Summarizes the major capabilities of the product.

Chapter 2 : General Description - The definition of requirements. Presents the environment in which the application is expected to operate, provides an overview of the system requirements, describes assumptions about possible users of the application, possible constraints on the project, and the underlying assumptions that on which the requirements analysis is based.

Chapter 3 : Specific Requirements - The specification of requirements. Contains a complete description of the application's requirements, both functional and non-functional.

Chapter 4 : Possible Product Evolution - Provides a list of features or capabilities that are expected to change or be added in the future.

Chapter 2

General Description

This chapter will help the reader to become familiar with the application to be built. Both the functional and non-functional requirements are summarized in this chapter. Chapter 3 serves to present the actual requirements.

2.1 Product Perspective

A conceptual model of the operating environment for the *Pizza Order* application is shown in Figure 2.1. This model shows the important relationships between the application and the entities with which it will interact. Each entity is described below.

Browser : A program such as Netscape that allows the customer to travel the web. A browser provides access to web-sites and home pages.

Customer : A customer is a user of the application who wishes to order a pizza. The customer will use a browser to access the client's home page and place an order.

Homepage : The client, Ethereal Pizza Company, will provide a home page that presents their restaurant to the web.

Pizza Order Application : The application to be developed for this project. A customer will enter this application via the home page. Once the application has been entered, a customer can read the restaurant's menu and place an order.

Order : The information for a customer's order including the customer's name, delivery address, and method of payment. This information is saved where the restaurant staff can access it.

Restaurant Staff : Typically the kitchen staff in the restaurant will receive orders placed by customers.

World-Wide Web : Provides the means of communication between the customer's browser and the restaurant.

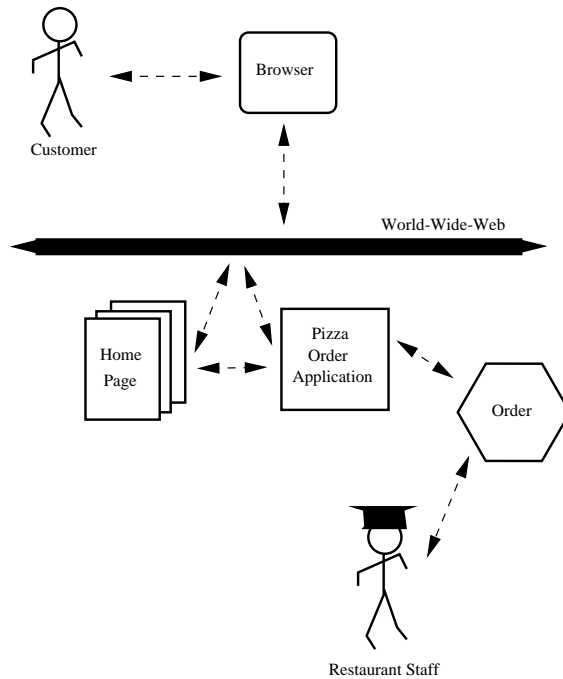


Figure 2.1: System Model

2.2 System Functionality

The purpose of the *Pizza Order* application is to enable customers to place an order via the world-wide web. A summary of the application's functionality is given below.

- The restaurant will provide a home page on the web. Customers will access the application via this home page.
- The application will provide support for the following information about an order to be entered:
 1. Size of pizza,
 2. Choice of toppings,
 3. Whether each topping should cover the entire pizza or just half,
 4. Customer's name,
 5. Delivery address,
 6. Method of payment. This includes cash, check, or credit card.
- The cost of the pizza order will be displayed and continuously updated as the customer enters or changes the order information.
- A customer will be required to enter a name and delivery address.

- A customer will be required to select a specific method of payment.
- The customer must confirm an order before it will be sent to the restaurant.
- When a customer places an order the order information will be saved such that restaurant staff can find and process the order.

2.3 User Characteristics

User of this application are customer's of the client who wish to place an order via the web. As such, the following assumptions have been made about any potential user.

- It is anticipated that users of this application are familiar with their web browser and its use.
- Users are assumed to have some familiarity with the nature of the web.
- Users are assumed to have some familiarity with how to place an order for a pizza.

2.4 General Constraints

This section lists considerations that may constrain the application's requirements or design. The following possibilities must be considered:

- Should orders be placed via a *secure* protocol? Orders will contain a customer's name, address, and payment method. Some people may prefer that such information not be sent openly across the web. This does not affect what the application does, but influence the selection of a communication mode.
- Even though it is not a client requirement, this application will be implemented using Java. Therefore, the home page and application will require that the customer's browser be Java-enabled.

2.5 Assumptions and Dependencies

The requirements analysis for this application was undertaken based on the following assumptions. Should any of these assumption later change or prove to be unreasonable, it will likely cause changes to occur in the application's requirements.

- For now, a customer will only be able to request a single pizza per order.
- The client does not currently provide drinks with delivery orders.
- The application will not be responsible for verifying that the delivery address entered by a customer is within the delivery area of the client's restaurant.

Chapter 3

Specific Requirements

This chapter presents a complete specification of the application requirements.

3.1 Functional Requirements

3.1.1 Class Model

The functional requirements for the *Pizza Order* application can be modeled by identifying eight classes. An overview diagram of the class model is shown in Figure 3.1.

A more detailed class model for this application is given in Figures 3.2, 3.3. These two figures show the attributes and methods for each class that have been identified during requirements analysis.

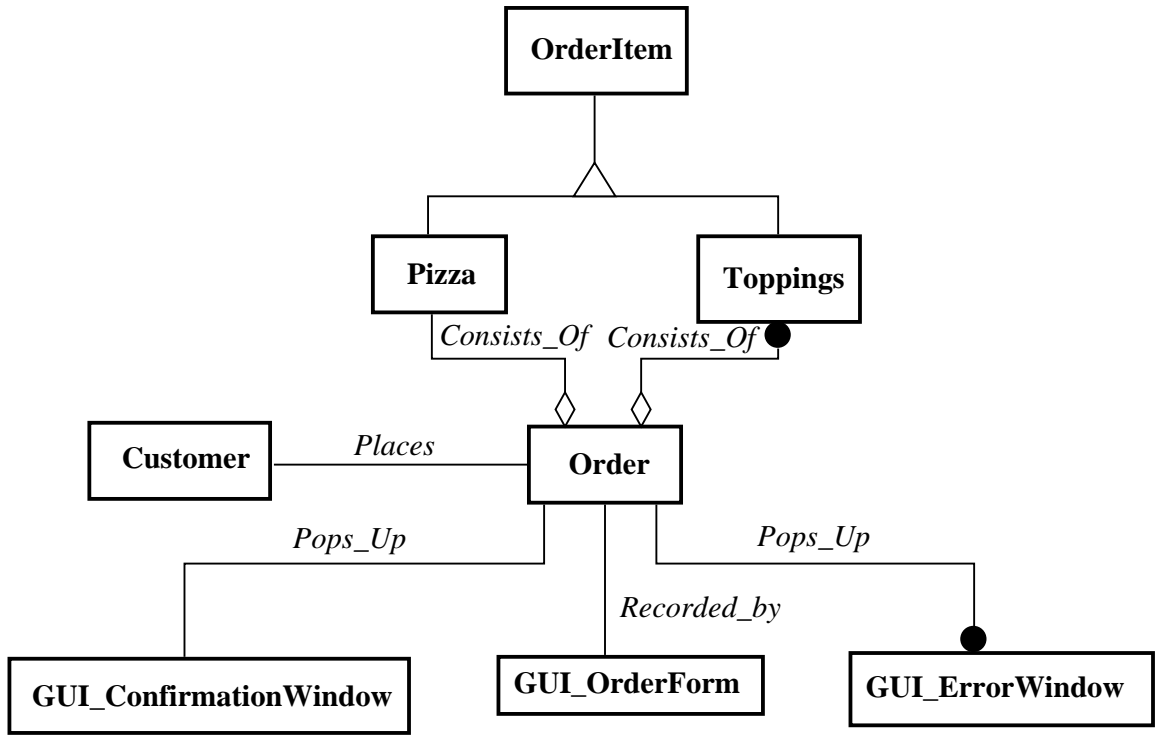


Figure 3.1: Overview of *Pizza Order* Application Class Model

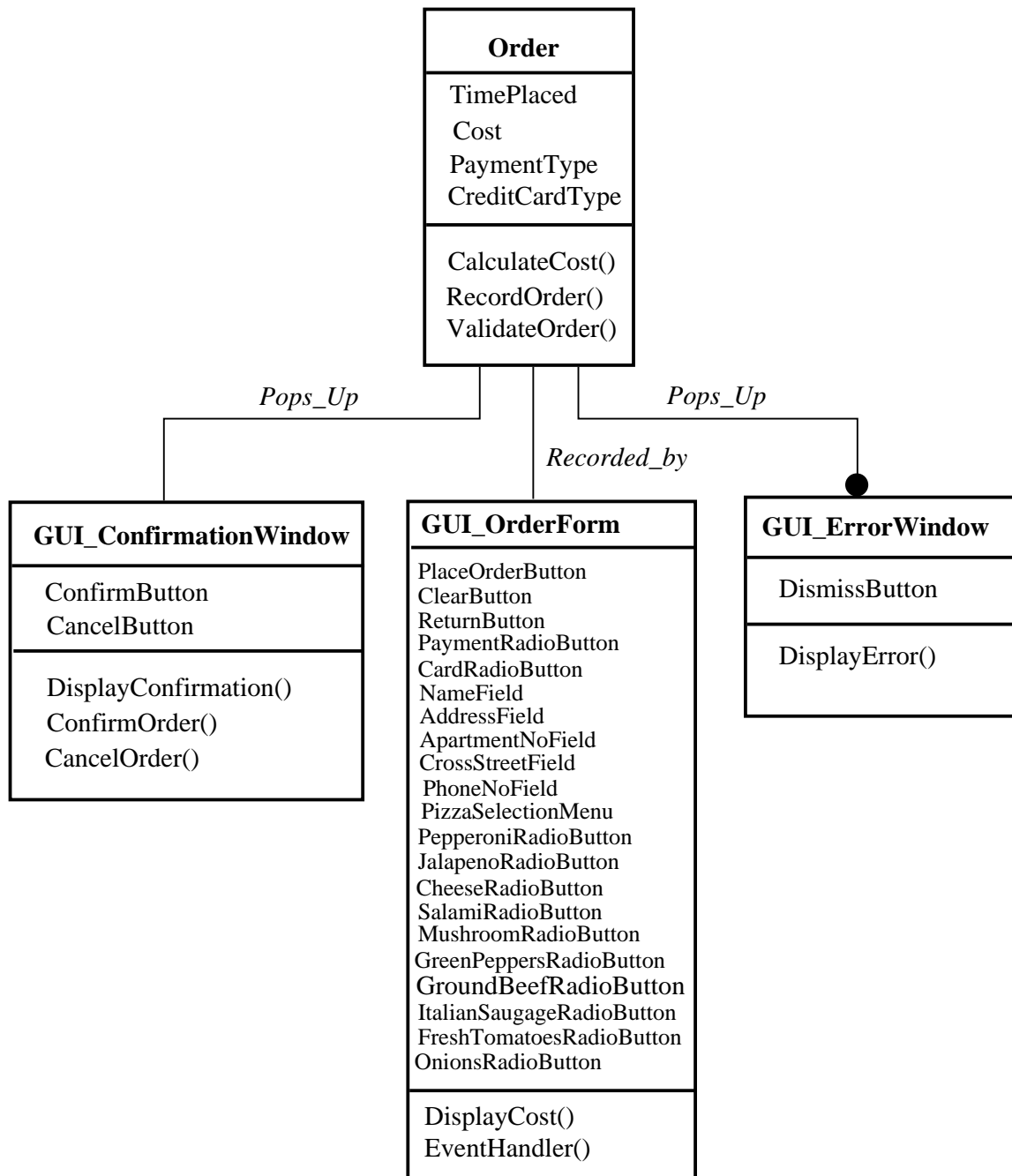


Figure 3.2: GUI Classes

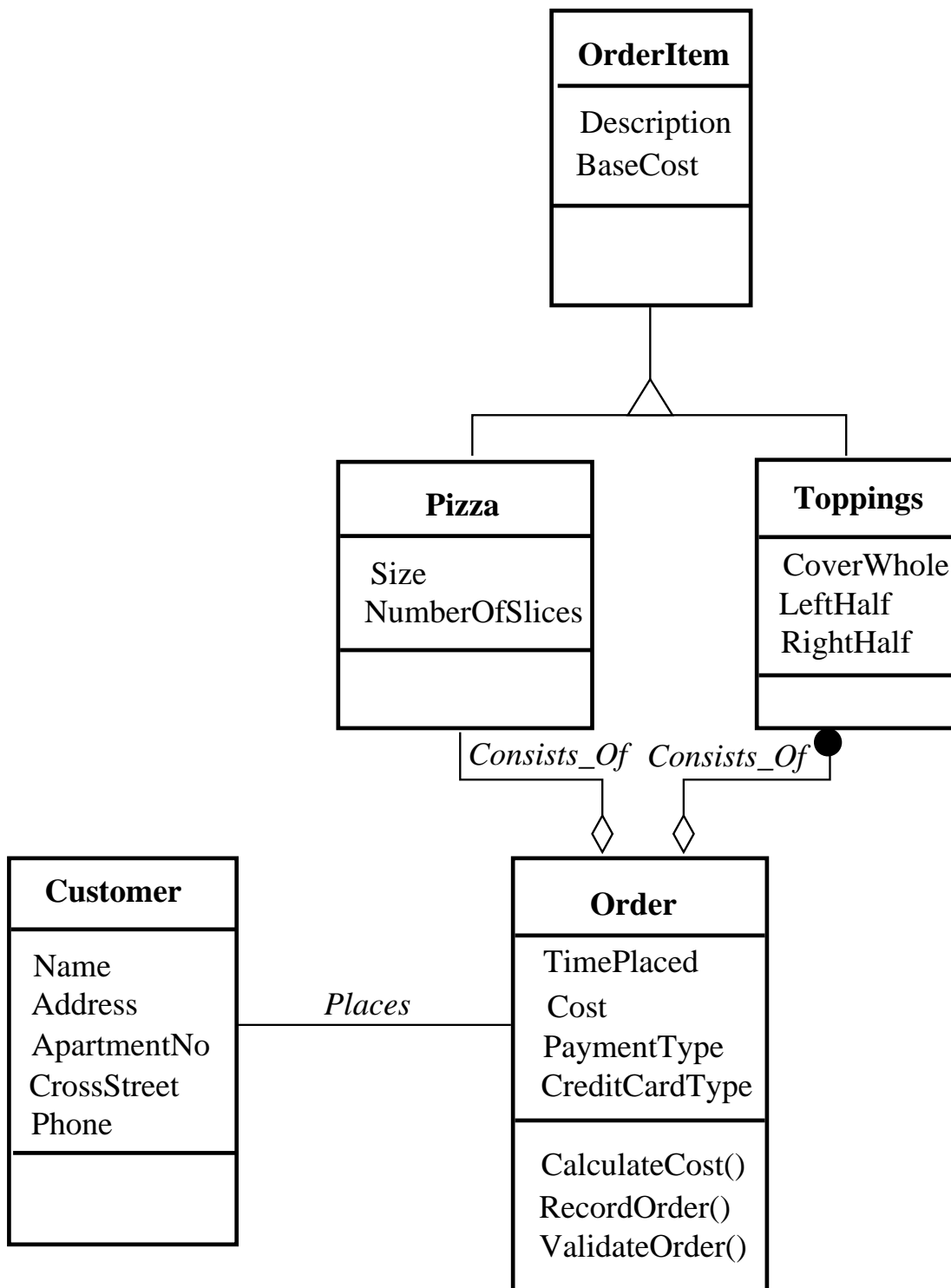


Figure 3.3: Classes Related to an Order

3.1.2 Class Associations

1. *Order* **Consists_Of** *Pizza*

A pizza order contains information about the kind of pizza to order. The order is always for one pizza.

2. *Order* **Consists_Of** *Toppings*

A pizza order contains information about the toppings to place on the pizza. There may be zero or more toppings.

3. *Customer* **Places** *Order*

An order is placed or submitted by a customer. An order must include information about who placed the order. An order is always placed by a single customer.

4. *Order* **Pops_Up** *GUI_ConfirmationWindow*

Before an order is recorded, the user must confirm the order information. There is a special window for confirming an order.

5. *Order* **Pops_Up** *GUI_ErrorWindow*

If incorrect information is detected with respect to an order, an error window is raised to display an error message.

6. *Order* **Recorded_by** *GUI_OrderForm*

A special menu window is used by a user to enter information about an order.

3.1.3 Class Dictionary

C1) Class: Customer

C1.1) Description : Represents information about the customer who is ordering the pizza.

C1.2) Attributes :

Name	Description
<i>Name</i>	: Customer name.
<i>Address</i>	: The delivery address.
<i>ApartmentNo</i>	: If the delivery address is an apartment the number goes here.
<i>CrossStreet</i>	: A major cross street near the delivery address.
<i>Phone</i>	: A phone number at the delivery address.

C1.3) Methods: None.



C2) Class: GUI_ConfirmationWindow

C2.1) Description : Encapsulates handling of the Confirmation window. This window provides the user with the choice of either confirming or cancelling an order.

C2.2) Attributes :

Name	Description
<i>ConfirmButton</i>	: If this button is clicked the order will be recorded.
<i>CancelButton</i>	: If this button is clicked the order will be ignored.

C2.3) Methods:

C2.3.1.1 Method : *DisplayConfirmation()*

C2.3.1.2 Description : Provides the ability to draw the confirmation window on the screen.

C2.3.2.1 Method : *ConfirmOrder()*

C2.3.2.2 Description : Accepts the user's request to record an order.

C2.3.3.1 Method : *CancelOrder()*

C2.3.3.2 Description : Handles the task of ignoring an order.



C3) Class: GUI_ErrorWindow

C3.1) Description : Encapsulates handling of the error window. This window is used to display an error message to the user.

C3.2) Attributes :

Name	Description
<i>DismissButton</i>	: When this button is clicked the window is removed from the screen.

C3.3) Methods:

C3.3.1.1 Method : *DisplayError()*

C3.3.1.2 Description : Provides the ability to draw the error window on the screen.



C4) Class: GUIOrderForm

C4.1) Description : Encapsulates handling of the order form window. This window is used to enter the order.

C4.2) Attributes :

Name	Description
<i>PlaceOrderButton</i>	: This button is clicked when the user is ready to place the order. The confirmation window will be raised to handle this situation.
<i>ClearButton</i>	: Clears the order form and returns all fields and radio buttons to their initial setting.
<i>ReturnButton</i>	: Exits the order form application.
<i>CardRadioButton</i>	: Used to select exactly one of the available types of credit cards.
<i>PaymentRadioButton</i>	: Used to select exactly one from of payment.
<i>NameField</i>	: The user enters the customer name in this field.
<i>AddressField</i>	: The user enters the delivery address in this field.
<i>ApartmentNoField</i>	: The user enters an apartment number in this field.
<i>CrossStreetField</i>	: The user enters the name of a major cross street in this field.
<i>PhoneNoField</i>	: The user enters a phone number in this field.
<i>PizzaSelectionMenu</i>	: Provides the user with a selection of pizza sizes.
<i>PepperoniRadioButton</i>	: Used to select a topping.
<i>JalapenoRadioButton</i>	: Used to select a topping.
<i>CheeseRadioButton</i>	: Used to select a topping.
<i>SalamiRadioButton</i>	: Used to select a topping.
<i>MushroomRadioButton</i>	: Used to select a topping.
<i>GreenPeppersRadioButton</i>	: Used to select a topping.
<i>GroundBeefRadioButton</i>	: Used to select a topping.
<i>ItalianSaugageRadioButton</i>	: Used to select a topping.
<i>FreshTomatoesRadioButton</i>	: Used to select a topping.
<i>OnionsRadioButton</i>	: Used to select a topping.

C4.3) Methods:

C4.3.1.1 Method : *DisplayCost()*

C4.3.1.2 Description : Provides the ability to draw the Order Form window on the screen.

C4.3.1.1 Method : *EventHandler()*

C4.3.1.2 Description : Responds to events that occur in the order form window.



C5) Class: Order

C5.1) Description : Holds information about a pizza order collected through the GUI_OrderForm window.

C5.2) Attributes :

Name	Description
<i>TimePlaced</i>	: The time the order was recorded.
<i>Cost</i>	: The total cost of the order.
<i>PaymentType</i>	: The method of payment selected by the user. Choices are: cash, check, or credit.
<i>CreditCardType</i>	: The type of credit type to which to charge an order. The possible cards are: Visa, MasterCard, or Discovery.

C5.3) Methods:

C5.3.1.1 Method : *CalculateCost()*

C5.3.1.2 Description : Calculate the current total cost of the order and display it in the *Cost of Order* field in the GUI_OrderForm window.



C5.3.2.1 Method : *RecordOrder()*

C5.3.2.2 Description : Handles the task of actually collecting all of the order information from the GUI_OrderForm window and writing it to a file.



C5.3.3.1 Method : *ValidateOrder()*

C5.3.3.2 Description : Before an order can be recorded, it must be validated to be sure that all the necessary data has been supplied. This method handles that task.



C6) Class: OrderItem

C6.1) Description : Denotes general information about an item that can be ordered from the restaurant.

C6.2) Attributes :

Name	Description
<i>Description</i>	: The name of an item.
<i>BaseCost</i>	: How much the item costs.

C6.3) Methods: None.

██████████

C7) Class: Pizza

C7.1) Description : Information about a base pizza, without any toppings.

C7.2) Attributes :

Name	Description
<i>Size</i>	: The size of a pizza: small, median, or large.
<i>NumberOfSlices</i>	: The number of slices contained in this size of pizza.

C7.3) Methods: None.

██████████

C8) Class: Toppings

C8.1) Description : Information about a pizza topping.

C8.2) Attributes :

Name	Description
<i>Coverage</i>	: Indicates whether the topping is to cover the entire pizza, just the left half, or just the right half.

C8.3) Methods: None.

3.2 External Interface Requirements

3.2.1 User Interfaces

This section describes the interface between the *Pizza Order* application and its users.

Invocation

UI.1 : *Invocation*

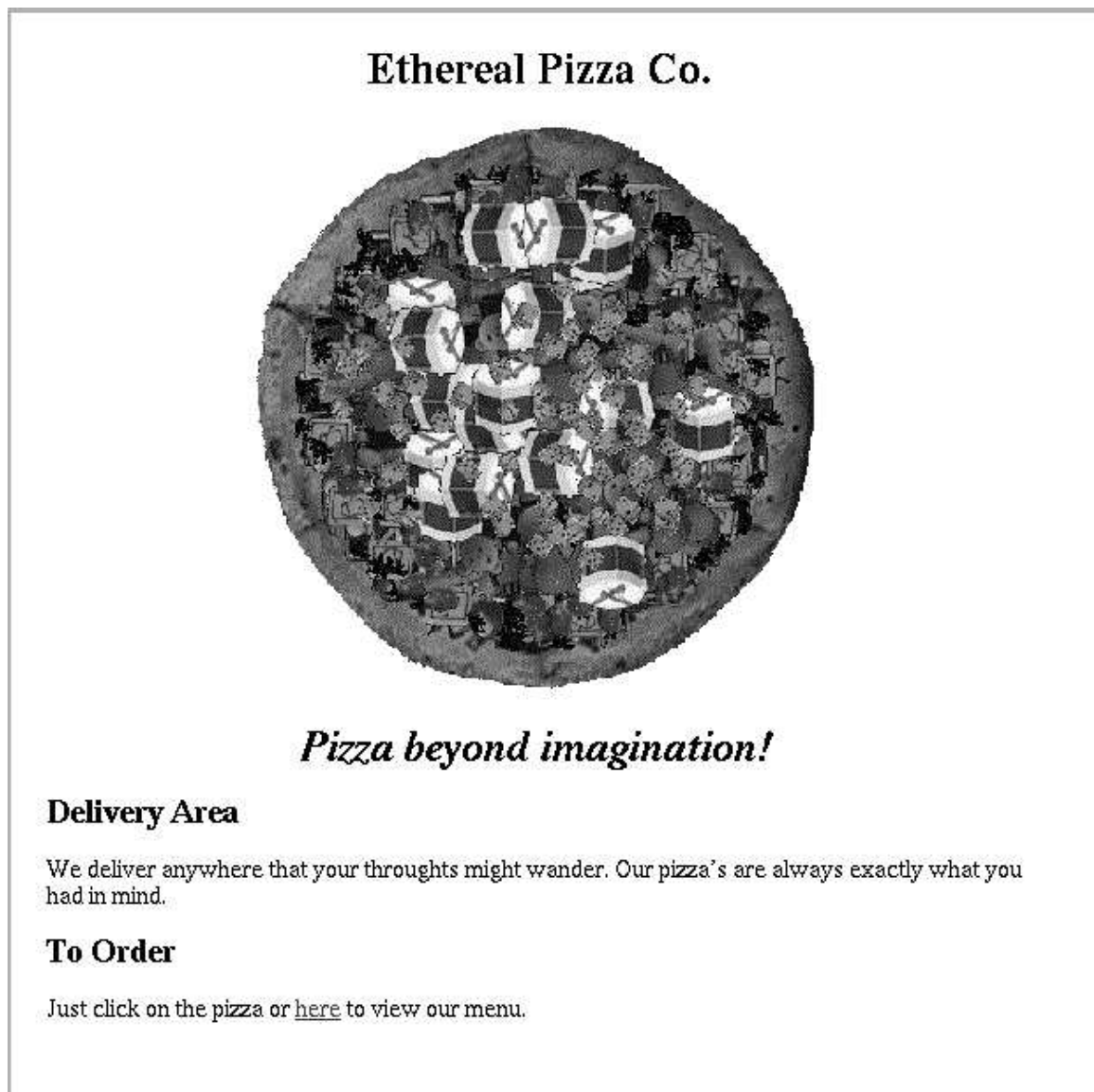


Figure 3.4: Ethereal Pizza's Home Page

- UI.1.1 :** The *Pizza Order* application will be invoked from the restaurant's home page. The home page is shown in Figure 3.4.
- UI.1.2 :** The user may click on the image of the pizza to invoke the application.
- UI.1.3 :** At the bottom of the home page there is a sentence " Just click on the pizza or *here* to view our menu." The user can click on the word "here" to invoke the application.

Windows

UI.2 : *GUI_OrderForm Window*

- UI.2.1 :** When the application is invoked the GUI_OrderForm window will be raised. Figure 3.5 shows this window.
- UI.2.2 :** *Pizza Size Pull Down Menu*
 - UI.2.2.1 :** The GUI_OrderForm has a pull down menu that enables a user to select the size of pizza to order.
 - UI.2.2.2 :** This menu has a menu item for each size of pizza offered by the restaurant. Each menu item lists the (A) pizza size, (B) number of slices, and the (C) base cost of that size of pizza.
 - UI.2.2.3 :** When a user selects a pizza size, the cost of the order will be updated in the *cost of order* field. (See Requirement UI.2.4).
 - UI.2.2.4 :** Initially, there is no size of pizza selected.
- UI.2.3 :** *Toppings*
 - UI.2.3.1 :** Each topping offered by the restaurant is listed on the order form.
 - UI.2.3.2 :** The user can select a topping by clicking on a radio button associated with the desired topping.
 - UI.2.3.2 :** For each selected topping, a user can choose to (A) have the topping cover the entire pizza, or (B) just the right half, or (C) the left half of the pizza. The degree of cover is selected by clicking on the appropriate radio button.
 - UI.2.3.3 :** Each time a topping is selected, the cost of the order will be updated in the *cost of order* field. (See Requirement UI.2.4).
 - UI.2.3.4 :** Initially, an order has no selected toppings.
- UI.2.4 :** *Cost of Order Field*
 - UI.2.4.1 :** Any time a user selects a new pizza size or a topping the cost of the order will be calculated displayed in this field.
 - UI.2.4.2 :** Initially, the cost of an order is \$0.0.
- UI.2.5 :** *Customer Information*
 - UI.2.5.1 :** Your Name Field - The user must enter the name of the customer in this field, before placing an order.
 - UI.2.5.2 :** Street Address Field - The user must enter the delivery address in this field, before placing an order.

Pizza Order Form

Warning: Applet Window

Pizza Order Form :

Create Your Own Pizza:

<i>Size</i>	<i>Number of Slices</i>	<i>Base Cost</i>
<input style="width: 100%;" type="text" value="What would you like to have ?"/>		

Select Your Toppings :

	<i>Cover Whole</i>	<i>Left Half</i>	<i>Right Half</i>
Pepperoni	◇ whole	◇ left	◇ right
Jalapeno	◇ whole	◇ left	◇ right
Cheese	◇ whole	◇ left	◇ right
Salami	◇ whole	◇ left	◇ right
Mushroom	◇ whole	◇ left	◇ right
Green Peppers	◇ whole	◇ left	◇ right
Ground Beef	◇ whole	◇ left	◇ right
Italian Sausage	◇ whole	◇ left	◇ right
Fresh Tomatoes	◇ whole	◇ left	◇ right
Onions	◇ whole	◇ left	◇ right

Cost Of Order: \$

Customer Information

Your Name :

Street Address :

Apt #

Nearest Cross Street

Phone No

Payment

Cash
 Check
 Credit
 Visa
 MasterCard
 Discover

Figure 3.5: GUI_OrderForm Window

UI.2.5.3 : Apt # Field - If the delivery address is an apartment, the apartment number must be entered in this field.

UI.2.5.4 : Nearest Cross Street Field - The user should enter the name of a major cross street that is near the delivery address.

UI.2.5.5 : Phone No. Field - The user must enter a phone number where the customer can be reached.

UI.2.6 : *Payment Information*

UI.2.6.1 : The user must select a method of payment. Valid forms of payment are: (A) cash, (B) check, or (C) credit card.

UI.2.6.2 : If the user chooses to pay by credit card, one of the following types of cards must be selected: (A) Visa, (B) MasterCard, or (C) Discover.

UI.2.6.3 : Initially, the method of payment selected is *cash*.

UI.2.7 : *Buttons*

UI.2.7.1 : When the user is ready to record the order, the button is clicked. This button will raise the GUI_Confirmation Window. (See Requirement UI.3).

UI.2.7.2 : If the user wants to reset the order form to its initial values, the button is clicked.

UI.2.7.3 : When the user is ready to exit the application, the button is clicked. This button will terminate the execution of the application and return the user to the restaurant's home page.

UI.3 : *GUI_Confirmation Window*



Figure 3.6: GUI_Confirmation Window

UI.3.1 : When the user clicks on the button in the GUI_OrderForm window, the GUI_Confirmation window (See Requirement U.2) will be raised. This window is shown in Figure 3.6.

UI.3.2 : To cancel the order, the user can click on the button.

UI.3.3 : If the user does cancel the order, the GUI_Confirmation window is lowered and the order is not recorded.

UI.3.4 : To place the order, the user can click on the **Confirm** button.

UI.3.5 : If the user does confirm the order, the order information will be saved to a file (See Section 3.4). In addition, this window will be lowered.

UI.4 : *GUI_Error Window*

Note: individual error messages are covered in the next section.



Figure 3.7: GUI_Error Window

UI.4.1 : When the application must report an error message to the user, it will pop up the GUI_Error window shown in Figure 3.7.

UI.4.2 : To lower the window the user must click on the **Dismiss** button.

Error Messages

The application will detect and report the following error situations.

Error : 1
Message : Customer name is missing!
Caused By : Before the application can record the order, the customer *name* field in the GUI_OrderForm window must contain a value.

Error : 2
Message : Customer address is missing!
Caused By : Before the application can record the order, the customer *address* field in the GUI_OrderForm window must contain a value.

Error : 3
Message : Customer information is incomplete!
Caused By : Before the application can record the order, the *cross street* and *phone number* field in the GUI_OrderForm window must contain a value

3.2.2 Hardware Interfaces

Not applicable.

3.2.3 Software Interfaces

The *Pizza Order* application does not have any explicit interface with any other software system. However, it does have two implicit interfaces.

- SI.1 :** The application must execute within the environment provided by the web browser. Thus the application will interact with the browser environment according to standard protocols.
- SI.2 :** The application is invoked from a home page. The home page itself is written in HTML. The Return Home button on the GULOrderForm window will enable the application to terminate and transfer back to the home page.

3.3 Non-Functional Requirements

The following non-functional requirements apply to the *Pizza Order* application.

NF.1 : *Implementation Language.*

The application must be implemented in Java.

NF.2 : *Conformance to Standards.*

The implementation of the application must satisfy the relevant coding standard enforced by the developer's organization.

NF.3 : *Maintainability.*

The source code must be written to support understandability and ease of modification.

NF.4 : *Reliability.*

The implementation of the application must have an estimated minimum Mean-Time-Between-Failure of six months.

3.4 Database Requirements

The application will not make use of a DBMS. Individual orders will be written to separate files. An *order file* will contain the following information:

1. Customer Name
2. Delivery Address
3. Nearest Major Cross Street
4. Customer's Phone Number
5. Method of Payment
6. Credit Card Type (if paying by credit card)
7. Time Order Was Placed
8. Total Cost of Order
9. Size of Pizza
10. List of Selected Toppings. For each topping the following information is given:
 - (a) Name of Topping
 - (b) Whether to cover left half, right half, or entire pizza.

Chapter 4

Possible Product Evolution

This chapter provides a list of features that are expected to either change or be added to the application in the future. The application's design should be structured to help reduce the effort required to integrate these features when they are incorporated into the application.

1. Enable a customer to request more than one pizza per order.
2. Provide the capability to request drinks to be delivered with the order.
3. Recognize and accept a broader range of credit cards.
4. Provide support for electronic cash payment.
5. Provide the ability to verify that a customer's delivery address is within the client's delivery area.
6. Provide the ability for a customer to check the status of an order.
7. Provide the ability for a customer to cancel an order after it has been confirmed, but before delivery occurs.
8. Provide the ability to order frozen pizzas for delivery anywhere in the USA via an overnight delivery service.