Enterprise Framework Project

Document Control

Revision History

Date	Version	Scope of Activity	Prepared	Reviewed	
21/10/2010	1	Create	John McCarthy		
18/12/2010	2	Complete	John McCarthy		

Distribution List

Name	Title	Version
Michael Bradford	Lecturer	2
John McCarthy (10205381)	Student	2
Gary Rutledge (10206931)	Student	2
~~Peter Conway~~	~~Student~~	~~1~~

Related Documents

Title	Comments

Table of Contents

1. Introduction 5
   1.1 Introduction to the project 5
   1.2 Project Goals 5
   1.3 Strategy to Achieve Goals 6
2. Proposed System 7
   2.1 Introduction 7
   2.2 Menu Display and Orders 8
   2.3 Menu Administration 9
   2.4 Store Order Processing 10
   2.5 Promotions and Vouchers 10
   2.6 Customer Signup and Login 11
3. Requirements Analysis 12
   3.1 Problem Definition 12
   3.2 Requirements Capture 12
   3.3 Key Functionality 13
   3.4 Use Case Diagrams 14
   3.5 Class Diagrams 15
   3.6 Requirements Validation 15
4. Architecture 16
   4.1 MVC 16
   4.2 Domain Model 16
   4.3 Data Access Layer 16
4.4 LINQ to SQL 17
   4.5 Dependency Injection (DI) 17
   4.6 Conclusion 18
5. Implementation 19
   5.1 Introduction 19
   5.2 Development Methodology 19
   5.3 Design Patterns 19
      5.3.1 Factory 20
      5.3.2 Repository 20
      5.3.3 State 20
   5.4 Testing 21
   5.5 Security 22

6. Conclusion 23
6.1 Introduction 23
6.2 Problems Encountered 23
6.3 Outstanding Issues 24
References 25
Appendix A - General Site Functionality 27
Appendix A - General Site Functionality 27
Appendix B - Admin Backend Functionality 38
Appendix C – Use Case Diagrams 46

List of Figures

Figure 1 - Sample Online Menu 8
Figure 2 - Sample Shopping Cart 8
Figure 3 - Sample Admin Menu 9
Figure 4 - Sample Login Window 11
Figure 5 - Sample Customer Signup 11
Figure 6 - Main Use Case Diagram 14
Figure 7 - System Class Diagram 15
Figure 8 - Visual Studio Projects 18
Figure 9 - Online Menu 28
Figure 10 - Online Menu (Pizza Category) 29
Figure 11 - Online Menu (Chicken Category) 30
Figure 12 - Shopping Cart 31
Figure 13 - Customer Login 32
Figure 14 - Shopping Cart Deliver Details 33
Figure 15 - Complete Order Display 34
Figure 16 - Order Email Alert 35
Figure 17 - Customer Signup 36
Figure 18 - Display Customer Details 37
Figure 19 - Customer Signup Email 37
Figure 20 - Backend Admin Login 40
Figure 21 - Admin Online Menu 40
Figure 22 - Create Menu Item 41
Figure 23 - New Menu Item Display 41
Figure 24 - Edit Menu Item 42
Figure 25 - Display Menu Item 42
Figure 26 - Admin Account Management 43
Figure 27 - Admin Customer List 43
Figure 28 - Admin Login Page 44
Figure 29 - Admin Order Display 44
Figure 30 - Admin Individual Order Display 45
Figure 31 - Use Case Moderate Menu 46
Figure 32 - Use Case Promotions 46
Figure 33 - Use Case Vouchers 47

1. Introduction

1.1 Introduction to the project

A pizza franchise in Ireland needs to take the business to the next level by creating a new website incorporating on-line sales and a new marketing strategy. There are currently 20 stores in Ireland.

Competitors in the pizza market are boosting their sales through their websites. The shift from the traditional shop sees customers ordering and paying on-line without visiting the shop. This has been noticed through the number of phone orders received, with customers paying the delivery person. With no method to manage this process it is very tedious and frequently prone to human error, with wrong orders going to wrong houses etc. Additional effort must be made to collect cash from delivery drivers and this does not always tally correctly. Most calls received will require a member of staff to go through the menu options with the customer. This can be time consuming and is very unproductive when the shop\restaurant is busy.

A clear mechanism is required to list on-line orders with all the details needed to facilitate all aspects of the web orders. The use of special offers and promotions are used in-store to influence customer purchases. This needs to be added to the functionality of the new site.

1.2 Project Goals

From the introduction above the need for a dynamic enterprise web application is clearly evident. The primary goal for this project is to create a dynamic web site developed in an object orientated fashion. This site may require additional functionality in the near future, so all code written must be easily scalable and reusable.

The site will require all the usual functionality of a typical business website and the requirements listed below will be incorporated into the project.

a) Cater for on-line orders
b) Provide a complete dynamic on-line menu
c) Promote special offers on-line
d) Improve company advertising and marketing
e) Provide structure to the entire delivery process
f) Easy method for staff to update menu content, view on-line orders etc…

1.3 Strategy to Achieve Goals

To complete the above requirements, a system development life-cycle will be loosely followed based of the following development life-cycle:

- Problem Definition
- Feasibility
- Analysis
- System Design
- Detailed Design
- Implementation
- Maintenance

As this is a short project an agile methodology will be used for the system development. From the initial scope the Extreme Programming (XP) Agile Methodology will be used. XP recommends very short iterations and is also a very low ceremony methodology. XP are four core values: communication, simplicity, feedback, and courage. These will form the bases of the teams approach to the project.

A detailed requirements analysis will be conducted to determine the needs for the new system. This process will be contained in the XP Agile methodology.

2. Proposed System

2.1 Introduction

The first task to be completed will be the requirements capture. This information will be gathered from a number of different interviews with the business owner and staff members. UML diagrams will be created to map the business processes and the new functionality of the system.

The database will be created from the requirements and data populated using the existing menu information provided by the business owner. When this section has been completed the system functionality will be divided into three distinct pieces for each team member to take ownership of.

a) Menu Display and Ordering (shopping cart style)
b) Menu administration and store order processing, promotions\vouchers
c) Customer Sign-up, Customer and staff logins, driver print out sheet.

2.2 Menu Display and Orders

Figure 1 shows a sample of the layout of the online menu. The categories are stored in a database table. The Menu items are stored in a separate table. When a required item is clicked it will be added to the order cart.

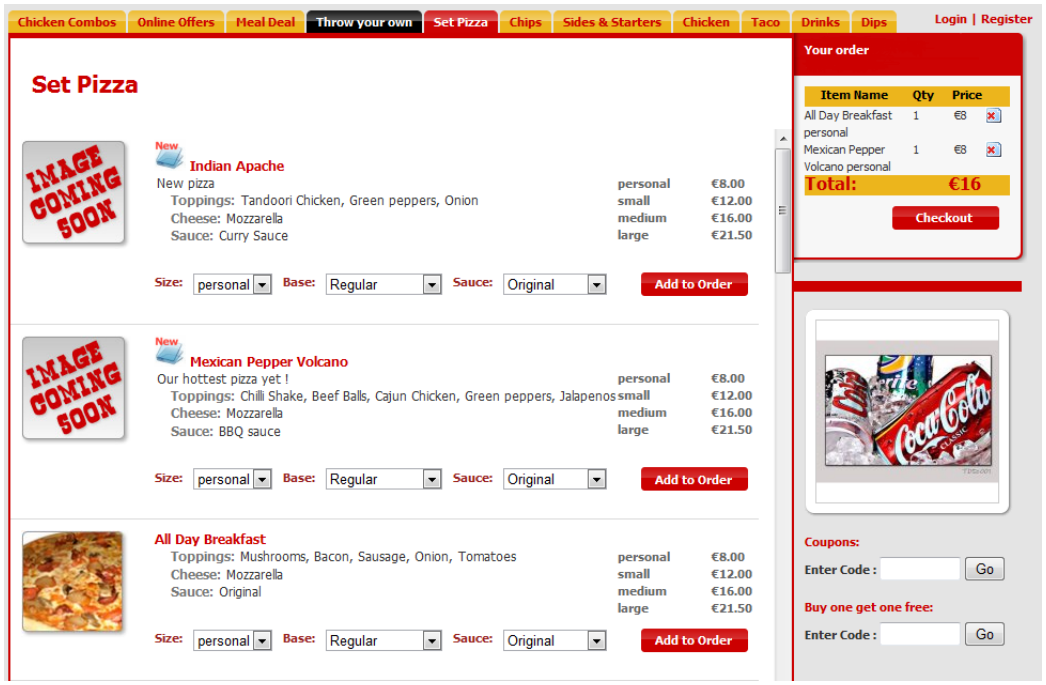

Figure 1 - Sample Online Menu

When the customer has selected all the required items for purchase the Checkout button is pressed. This will display all the items on the order to the customer (Figure 2).

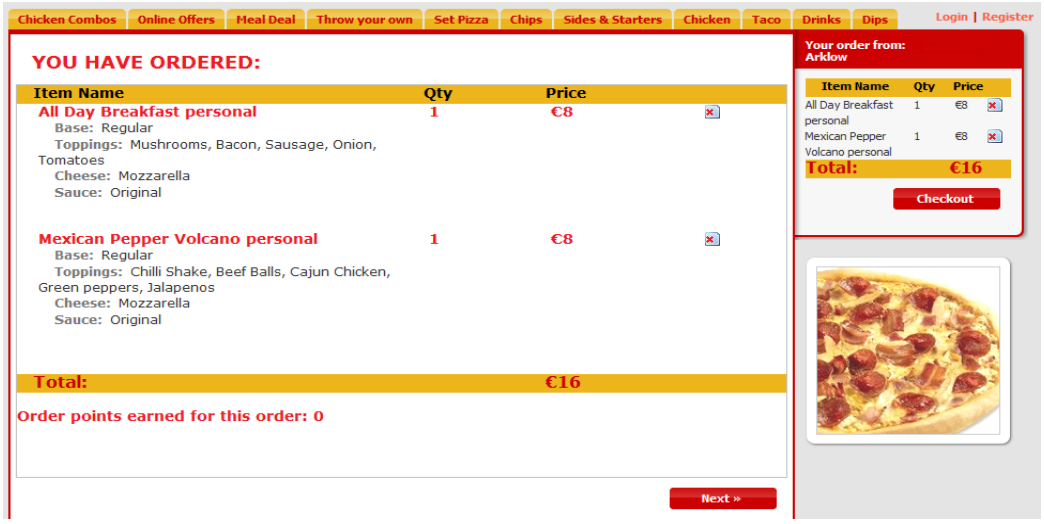

Figure 2 - Sample Shopping Cart

If the customer is happy to proceed with the order, the next button will check if the customer has logged in. If not they will be asked to login, if they have no account they will be directed to the sign-up page. After login the customer will be presented with the order information capture.

This contains:

- Contact Person
- Conformation of the contact number
- Delivery address
- Total Cost

When this is complete the order must be inserted into the database and display a message to the customer.

2.3 Menu Administration

This is the functionality to allow the owner\general manager to moderate the content for the online menu.

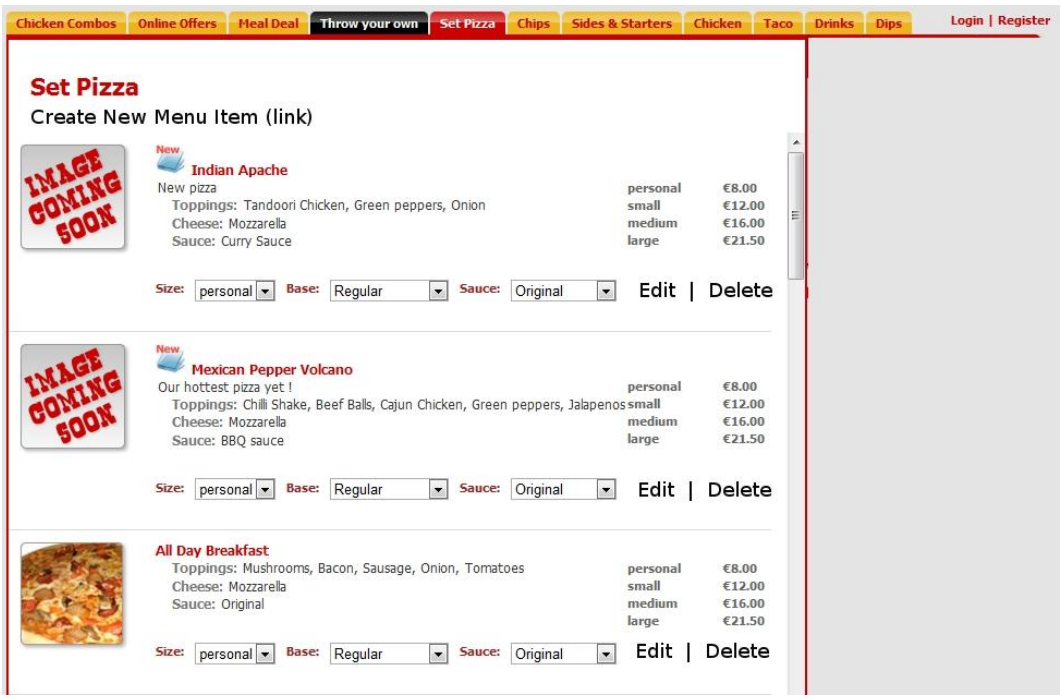

Figure 3 - Sample Admin Menu

The layout and display will be similar to the layout on the main site. Additional links will be provided to facilitate:

- Create New Menu Item
- Edit menu item
- Delete menu item

2.4 Store Order Processing

When an order is placed on the website the store closest to the customer must process the order. A screen will be available in the store to display all online orders to be processed. This will be in a list format to be processed in a sequential manner. The store staff can view all the information pertaining to the order. When the preparation of the order is complete the store staff will mark the order as complete and print a docket for the delivery driver.

2.5 Promotions and Vouchers

Vouchers can be created by the store manager. These can be used to encourage customers to sign up the website by offering a percentage discount to online orders.

Special offers and promotions can be added to the site for specific store locations. The manager of a specific store can create special offers etc and these will be displayed to the user when they have selected their location on the website. This allows specific stores have the flexibility to target their customer base with offers and promotions.

2.6 Customer Signup and Login

Customer Login

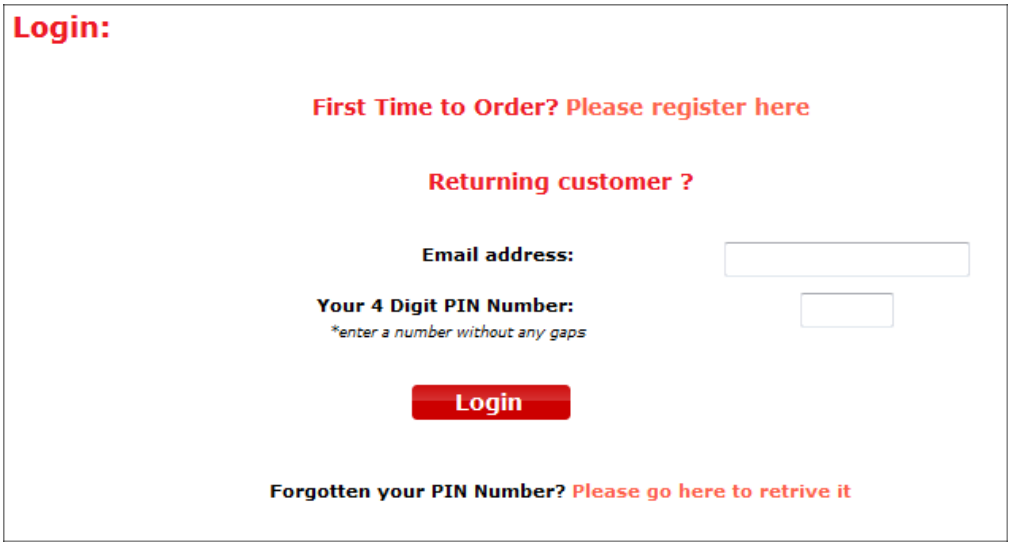

Figure 4 - Sample Login Window

Customer Signup

Figure 5 - Sample Customer Signup

3. Requirements Analysis

3.1 Problem Definition

A pizza franchise in Ireland needs to take the business to the next level by creating a new website incorporating on-line sales and a new marketing strategy. There are currently 20 stores in Ireland.

Competitors in the pizza market are boosting their sales through their websites. The shift from the traditional shop sees customers ordering and paying on-line without visiting the shop. This has been noticed through the number of phone orders received, with customers paying the delivery person. With no method to manage this process it is very tedious and frequently prone to human error, with wrong orders going to wrong houses etc. Additional effort must be made to collect cash from delivery drivers and this does not always tally correctly. Most calls received will require a member of staff to go through the menu options with the customer. This can be time consuming and is very unproductive when the shop\restaurant is busy.

A clear mechanism is required to list on-line orders with all the details needed to facilitate all aspects of the web orders. The use of special offers and promotions are used in-store to influence customer purchases. This needs to be added to the functionality of the new site.

3.2 Requirements Capture

The main methods used for the requirements elicitation was through meetings with the owner and staff of the pizza restaurant. The requirements capture process followed the guidelines of (Fowler & Scott 2000). The systems functionality is not unique and is similar to a number of different competitors' websites. The owner of the company has given a vast quantity of information relating to the project specification. All this information was used in the requirements specification to create the key functionality detailed below.

3.3 Key Functionality

The new site will include a dynamic menu that can be managed by the central management team. The menu will be available in the main site and the items and their prices will be the same for all the 20 stores. This is a requirement as the pricing should be consistent from store to store. Special offers and promotions can be added to the site for specific store locations. The manager of a specific store can create special offers etc and these will be displayed to the user when they have selected their location on the website. This allows specific stores have the flexibility to target their customer base with offers and promotions. The editing of the menus will be done through a secure custom CMS that will be designed for the website. Central management can login and add\update\delete menu items and view useful statics regarding on-line orders through a dashboard style interface.

Customers can browse the site and place orders on-line. A shopping cart style will be applied to this functionality, allowing the customer to select a number of items before placing the order. To place an order a customer must have signed up to the site. This requires entering their name, address, email and phone number. All data captured will be used to target customers with new products etc in the near future.

The system will be used to control the complete business process of all orders. This will ensure orders are completed in a timely fashion and delivered to the customer with no delays or errors. This will be incorporated into the back end site to help achieve these goals.

The following functionality will be incorporated into the new system.

- Dynamic On-line Menu
- Interface for staff to add\update\delete menu items
- Promotions and Special Offers
- On-line Ordering and Payment
- Provide structure to the on-line order business process
- Logins for staff, differing levels etc…
- Logins for customers \ Customer sign-up

3.4 Use Case Diagrams

Figure 6 describes the overall functionality of the complete pizza ordering system. This process was conducted to describe how people interact with the system.

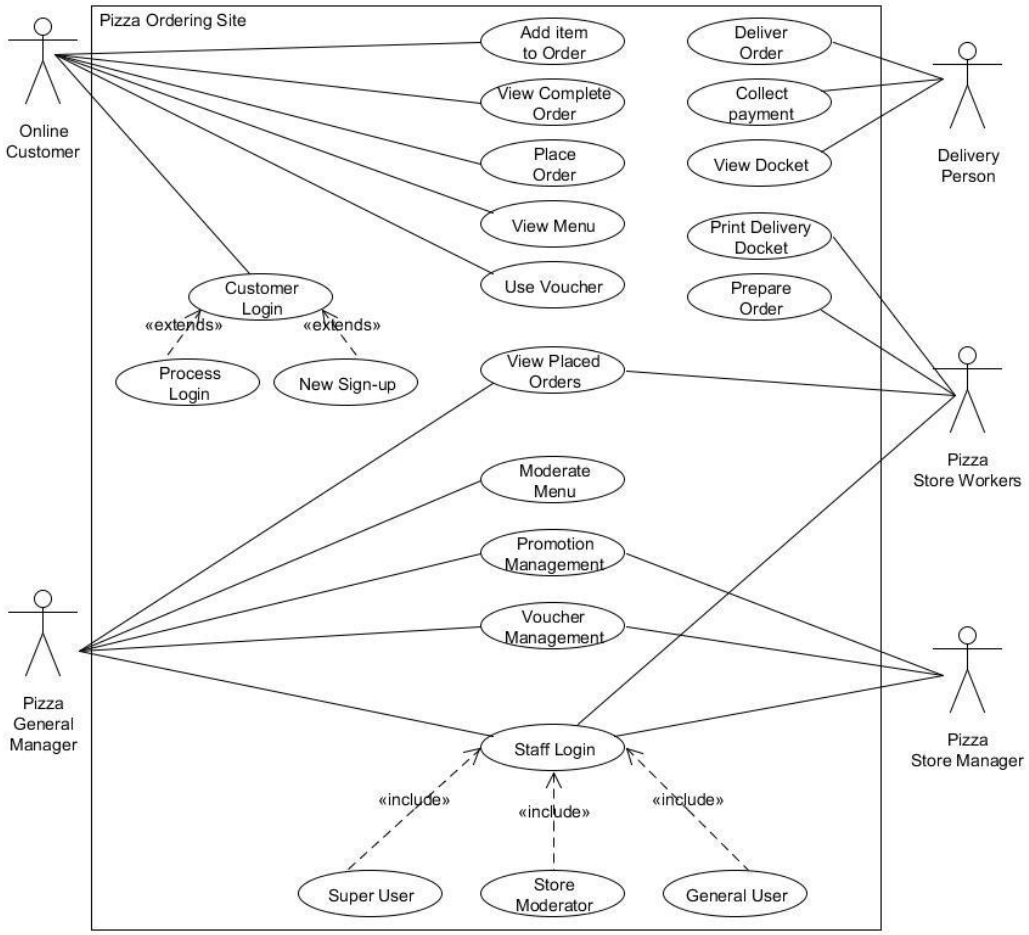

Figure 6 - Main Use Case Diagram

3.5 Class Diagrams

The class diagram was drawn up from the conceptual perspective, which will be used to build up a rigorous vocabulary of the domain.

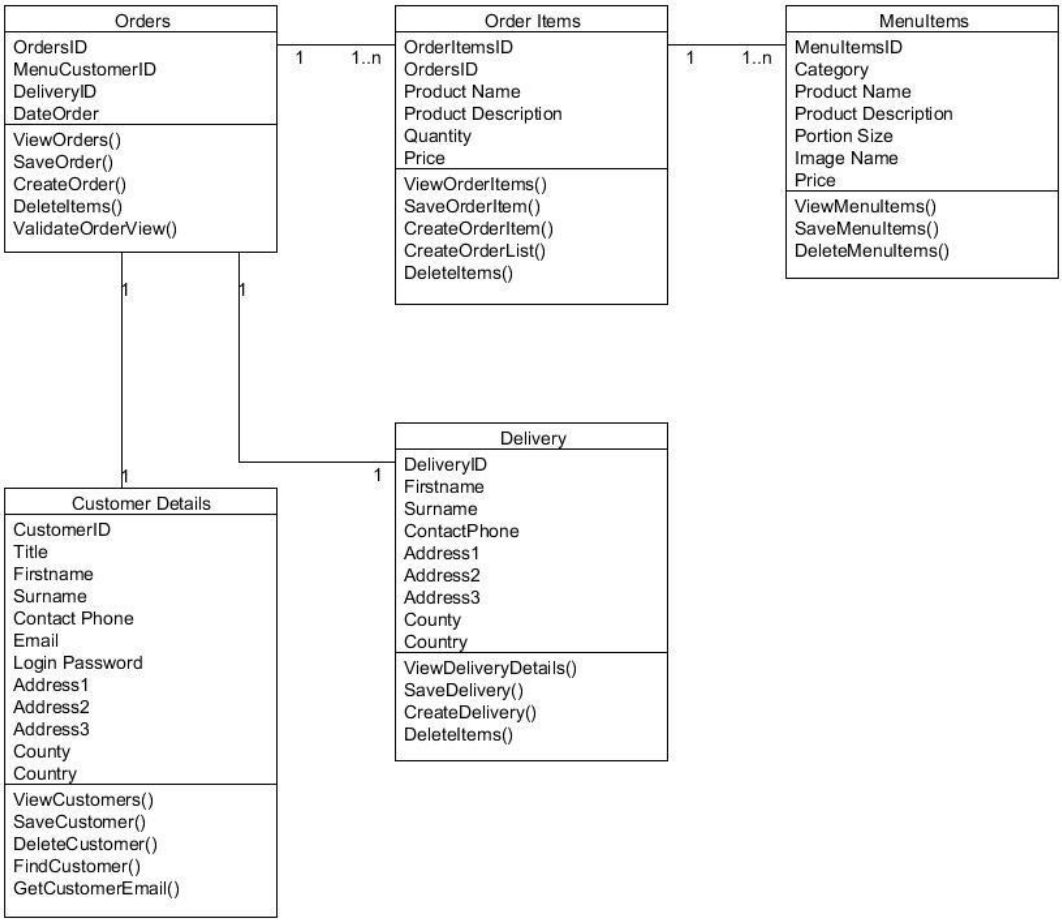

Figure 7 - System Class Diagram

3.6 Requirements Validation

The activity of requirements analysis involves trying to figure out what the users and customers of a software effort want the system to do (Fowler & Scott 2000). All of the information from the requirements capture was presented to the client to ensure the system meets all of its functionality requirements. For the numerous iterations on the requirements process, the most important element was our communication with the client and their staff.

4. Architecture

4.1 MVC

The Model-View-Controller (MVC) architectural pattern separates an application into three main components: the model, the view, and the controller (ASP.NET 2010). MVC is a standard design pattern and this web application will benefit from the MVC framework.

The MVC framework includes the following components (ASP.NET 2010):

- Model objects are the parts of the application that implement the logic for the applications data domain.
- Views are the components that display the application s user interface (UI).
- Controllers are the components that handle user interaction, work with the model, and ultimately select a view to render that displays UI.

4.2 Domain Model

For this project a separate domain model will be used to encapsulate the business logic. This technique helps decouple the domain objects from the other functions of the system (S. Sanderson & D. Sanderson 2010). From the start of the project it was decided to use the domain model to provide this separation.

Following the information gained in the requirements capture the domain model was constructed. The UML class diagram shown in Figure 7 was used to model the classes for the business objects. Any functionality related to the business logic will be kept in the domain model, these will be discussed later in this section.

4.3 Data Access Layer

A data access layer is used for a system to store and retrieve information from a database. The domain model will host the data access functionality and repositories

will be created to provide this functionality. The repositories are nothing more than object-oriented representations of the underlying database acting as a façade over the real implementation (ASP.NET 2010). A separate repository was created for each of the objects in the project.

4.4 LINQ to SQL

LINQ to SQL is an ORM tool designed to provide strongly typed views of database information (Microsoft MSDN 2007). It was decided to use this in the project as it quick to implement and the project can deliver quicker for a small scale application. The requirements analysis showed there was no complicated database relations needed to provide the client with all the functionality they required. Entity Framework was investigated as a possible strong contender, but LINQ to SQL was favoured for its ease of use and short learning curve.

4.5 Dependency Injection (DI)

Dependency Injection (DI) is a technique shown in Martin Fowler's article Inversion of Control Containers and the Dependency Injection Pattern (Fowler 2004). Dependency Injection (DI) facilitates the injection of objects into a class rather than relying on the class to create the object itself. The use of a factory class is one common way to implement DI. Dependency Injection aims to reduce the amount of boilerplate wiring and infrastructure code that you must write (MSDN Magazine 2005).

Ninject is a software component that can be added to a ASP.NET MVC project to provide Dependency Injection. “Ninject is a lightning-fast, ultra-lightweight dependency injector for .NET applications. It helps you split your application into a collection of loosely-coupled, highly-cohesive pieces, and then glue them back together in a flexible manner. By using Ninject to support your software's architecture, your code will become easier to write, reuse, test, and modify” (Ninject 2010).

4.6 Conclusion

The techniques listed above were used to setup the initial project infrastructure. Figure 8 shows the project created and inside this three additional projects were added. The PizzaStore.Domain is the domain model to contain all the business logic for the application. The PizzaStore.UnitTests is used for the unit testing of the application, this will be covered in detain in the next section. The PizzaStore.WebUI project contains the MVC architecture for the project website. Using three separate projects facilitates the decoupling of the objects in the system, this will be demonstrated in the next section.

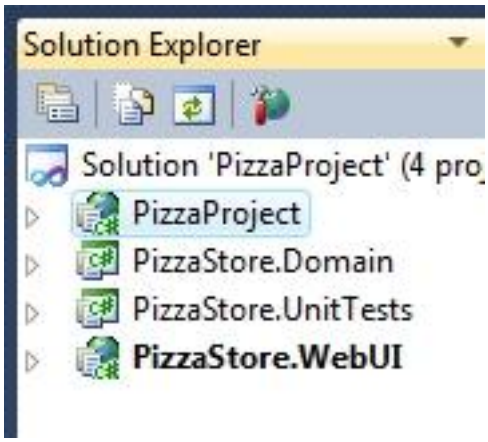

Figure 8 - Visual Studio Projects

5. Implementation

5.1 Introduction

This section will describe the details and features of the application.

5.2 Development Methodology

This project used the Extreme Programming (XP) development methodology. XP puts high value on customer satisfaction. Instead of delivering everything you could possibly want on some date far in the future this process delivers the software you need as you need it (XP 2010). For a small team of two members, XP suited the team as it encourages teamwork.

Extreme Programming improves a software project in five essential ways:

- communication
- simplicity
- feedback
- respect
- courage

Constant communication, testing and feedback provide the ideal methodology to develop the application in a structured and efficient process.

5.3 Design Patterns

From the start of the project the team was careful only to apply a design pattern if it matched the problem to be solved. The design patterns that have been used in the application are:

- Dependency Injection
- Repository
- Factory
- State

5.3.1 Factory

Simply referencing the Ninject assembly doesn't make anything new happen. Ninject needs to be hooked into the MVC Framework's pipeline. ASP.NET MVC will not be allowed to instantiate controller classes directly, and make it start requesting them from the DI container. This will allow the DI container to resolve any dependencies those controllers may have. This is accomplished by creating a custom controller factory (which is what the MVC Framework uses to instantiate controller classes) by deriving a subclass from ASP.NET MVC's built-in DefaultControllerFactory (S. Sanderson & D. Sanderson 2010).

5.3.2 Repository

The Data Access Layer (DAL) is setup using the repository pattern, in which an object-oriented representation of a data store acts as a façade on top of the database.

The abstract factory pattern is used to ensure the model isn't coupled to any concrete implementation of a data repository. The repositories use .NET interfaces or abstract base classes making the model totally decoupled from the type of database.

5.3.3 State

The shopping cart used to process online orders follows the state design pattern. The concept of this is the customer cannot revert to a previous stem in the lifeline of an order.

5.4 Testing

NUnit is a unit-testing framework for all .Net languages (NUnit 2010). This was chosen for the unit testing as Visual Studio does not come with unit testing functionality. Using an open source tool allowed all team members to use the same testing software.

Moq is a mocking library for .NET developed from scratch to take full advantage of .NET 3.5. It supports mocking interfaces as well as classes. Its API is extremely simple and straightforward, and doesn't require any prior knowledge or experience with mocking concepts (Moq 2010).

The first part of the project to be developed was the online menu, allowing for all the items to be displayed on the site and the option to sort them by category. Unit testing and mock data were used in the coding of this functionality. The unit tests allowed for certain functional requirements to be added to the tests, and these will not pass if the functionality is not correctly coded into the project. The menu items required data from the database, Moq was initially used to hard code some details to test the online menu. When everything was working correctly, the Moq data was replaced with the database repository.

Unit testing and test driven development was only used for the Online menu piece. It involves a fair bit of extra code, which was impacting on the time available to the team to complete the project. It must be noted all team members noted the benefit of unit testing and would have preferred to use it for the remainder of the project if time was not an issue. Unit tests are a very fast, focused, and precise way to define specific behaviors and then verify that your implementation matches them (NUnit 2010).

5.5 Security

There were a couple of security concerns to be addressed in the project. The main problems related to the online orders and customer details. If an order is successfully completed online the customer can view the completed order in their browser. The order display page uses the order id in the url to retrieve the correct order for the customer. The system cannot allow a user to view any order that was not placed by them. The login process was used to hold the customers id in a session variable. This allows the application to validate if a users id is linked to a specific order if the customer tries entering different order id variables in the url.

This was also a problem for the customer signup functionality. When a customer signs up to the website, an email is sent to their specified email account with a welcome message and a link allowing them to edit their details. This link contains the customer id, which could be changed to an id of a different customer. This has been protected by checking if the customer logged in matches the customer id to be edited.

When the site is to go love the order processing and customer signup will be secured using https, protecting the customers data from unauthorized sniffing or packet capture.

Cross Cutting Concerns (XSS) have been addressed by encoding all input fields on the website.

6. Conclusion

6.1 Introduction

Following the initial requirements capture the team is satisfied with the developed application and its associated functionality. The developed system displays qualities of an enterprise application allowing users of the site to view a complete online menu and use the shopping cart to purchase items from the pizza restaurant. Customers have the facility to sign up to the website and can edit their own details using the link sent to them in the signup activation email.

The backend system allows the owner to control content in the online menu. The admin online menu was developed with its look and feel very similar to the sites online menu, providing a familiar environment for the owner to update the menu content.

All orders placed on through the website are visible on the Admin system for the store users. This allows them to process orders and mark them as complete when they have been sent out with the delivery driver.

Overall the system has delivered based on the systems requirements. There are a couple of outstanding pieces, this will be addressed later in this section.

6.2 Problems Encountered

The main problem encountered by the team was the lack of manpower. There were two pieces of functionality that had to be omitted from the project at an early stage after loosing a team member. These omissions will be discussed in the next section.

6.3 Outstanding Issues

From the original requirements the owner of the pizza store needed the functionality to create promotions for display on the site and create discount vouchers for use on the online ordering.

This was not completed and removed from the site at an early stage. The requirements for the project were created for three team member, but the loss of a team member severely impacted the teams' goals and deliverables.

References

ASP.NET, 2010. ASP.NET MVC Overview: The Official Microsoft ASP.NET Site. Available at: http://www.asp.net/mvc/tutorials/asp-net-mvc-overview-cs [Accessed December 19, 2010].

Fowler, 2004. Inversion of Control Containers and the Dependency Injection pattern. Available at: http://martinfowler.com/articles/injection.html [Accessed December 19, 2010].

Fowler, M. & Scott, K., 2000. *UML distilled: a brief guide to the standard object modeling language*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Microsoft MSDN, 2007. LINQ to SQL: .NET Language-Integrated Query for Relational Data. Available at: http://msdn.microsoft.com/en-us/library/bb425822.aspx#linqtosql_topic1 [Accessed December 19, 2010].

Moq, 2010. moq - Project Hosting on Google Code. Available at: http://code.google.com/p/moq/ [Accessed December 19, 2010].

MSDN Magazine, 2005. Design Patterns: Dependency Injection. Available at: http://msdn.microsoft.com/en-us/magazine/cc163739.aspx [Accessed December 19, 2010].

Ninject, 2010. Open source dependency injector for .NET - Ninject. Available at: http://ninject.org/ [Accessed

December 18, 2010].

NUnit, 2010. NUnit - Home. Available at: http://www.nunit.org/ [Accessed December 19, 2010].

Sanderson, S. & Sanderson, D., 2010. *Pro Asp. net Mvc V2 Framework*, Apress.

XP, 2010. Extreme Programming: A Gentle Introduction. Available at: http://www.extremeprogramming.org/ [Accessed December 19, 2010].

Appendix A - General Site Functionality

The main landing page for the Piza Pizza website is the Online Menu page. The main functionality of this website is to facilitate online orders for Customers. Figure 9 shows a screenshot of the Menu page.

The Online Menu lists all the items Piza Pizza offer for sale. The menu item contains the name, description, price, portion size and an image of the item. The menu items are divided into specific categories, allowing customers on the website to quickly sort items. Figure 10 shows the items in the Pizzas category and Figure 11 shows the items in the Chicken category.

The menu is linked to an online shopping cart to provide structure to the customers ordering process. To add an item just click the Add to cart button for the menu item required. The cart supports multiple items with multiple quantities. When a customer adds an item the content of the current cart will be displayed. The customer can click Continue Shopping to return to the menu to add more items. When the customer has all the required items in the cart (Figure 12) they must click the check out now button.

There is a requirement that the customer must have signed up to the website before they can place an online order. The sign up page is shown in Figure 9, requiring the customer to enter their personal details and email address. A confirmation email will be sent after a successful signup. The email (Figure 19) will contain a link allowing the customer to view or update their own personal details (Figure 18). When the customer has clicked the check out now button they will be prompted to Sign In (Figure 13) if they have not already done so. If the customer does not have an account they can sign up and complete their order.

The customer must enter the delivery address and their contact details to complete the order process. This is a requirement from the owner of Piza Pizza as this has caused difficulties with previous orders. Certain customers will order pizzas when they are not at their home address, so automatically using their home address was not an option. Figure 6 shows the delivery details page.

When the Complete Order button is clicked the order is saved a confirmation email is sent to the customer (Figure 16). The complete order with the contact details and delivery address is displayed (Figure 15).

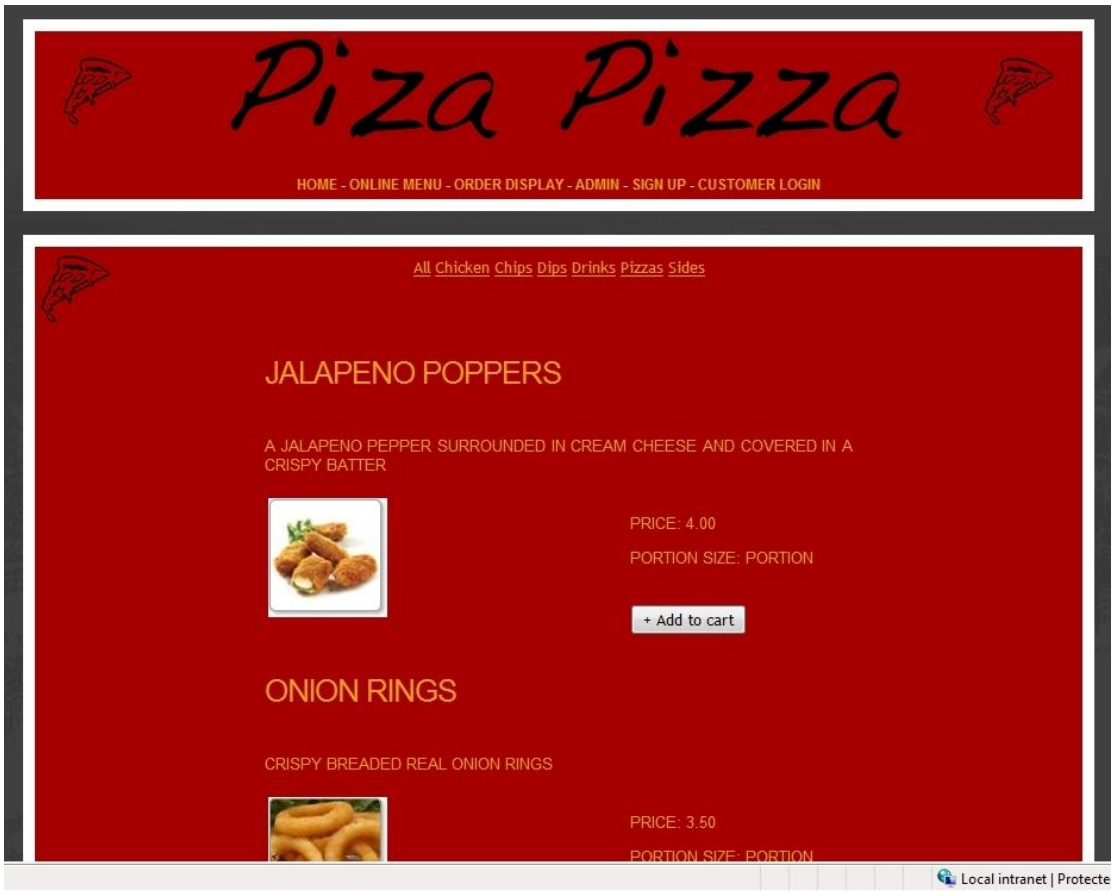

Figure 9 - Online Menu

Figure 10 - Online Menu (Pizza Category)

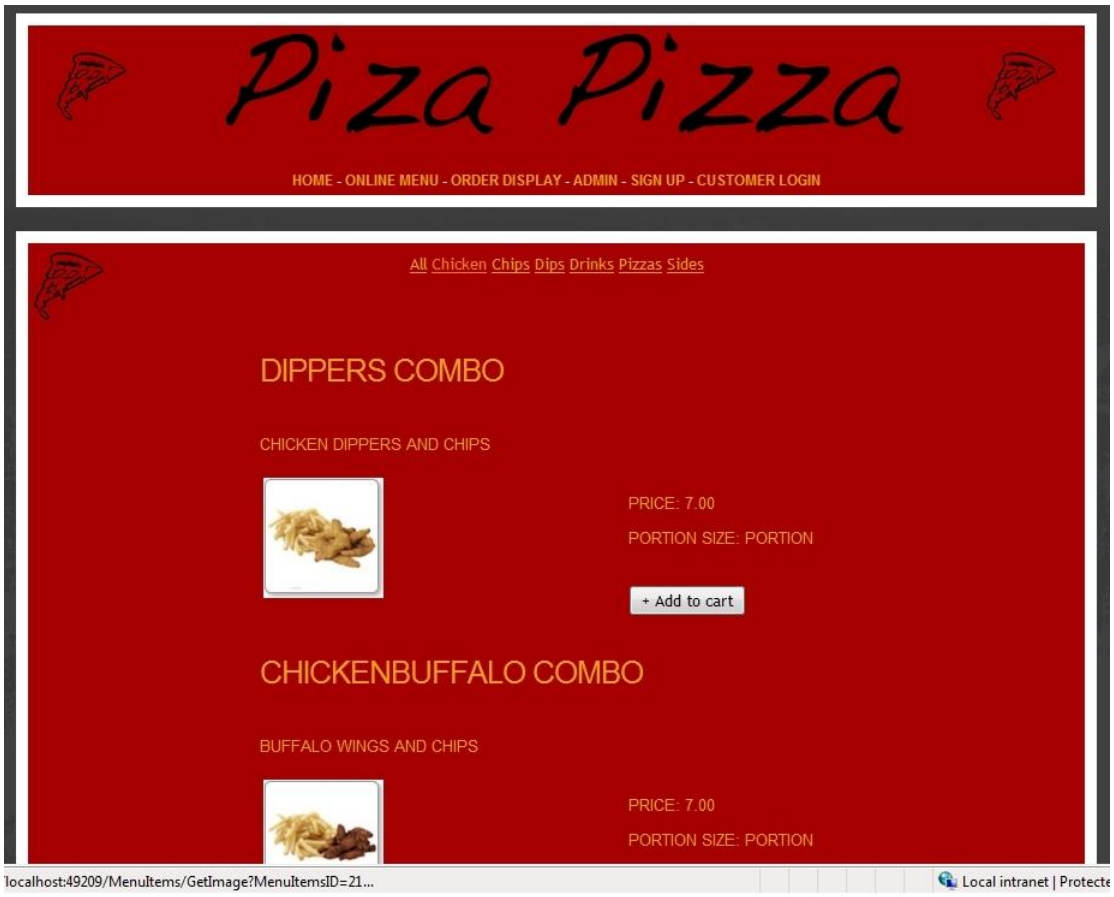

Figure 11 - Online Menu (Chicken Category)

Figure 12 - Shopping Cart

Figure 13 - Customer Login

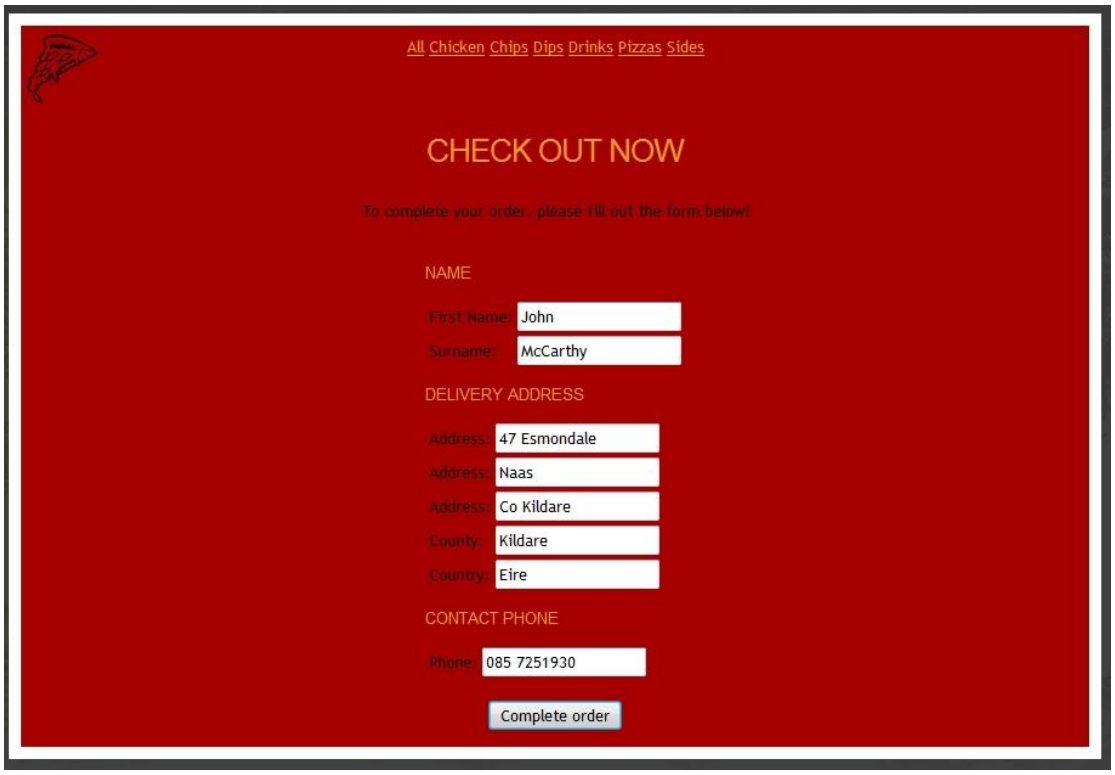

Figure 14 - Shopping Cart Deliver Details

Figure 15 - Complete Order Display

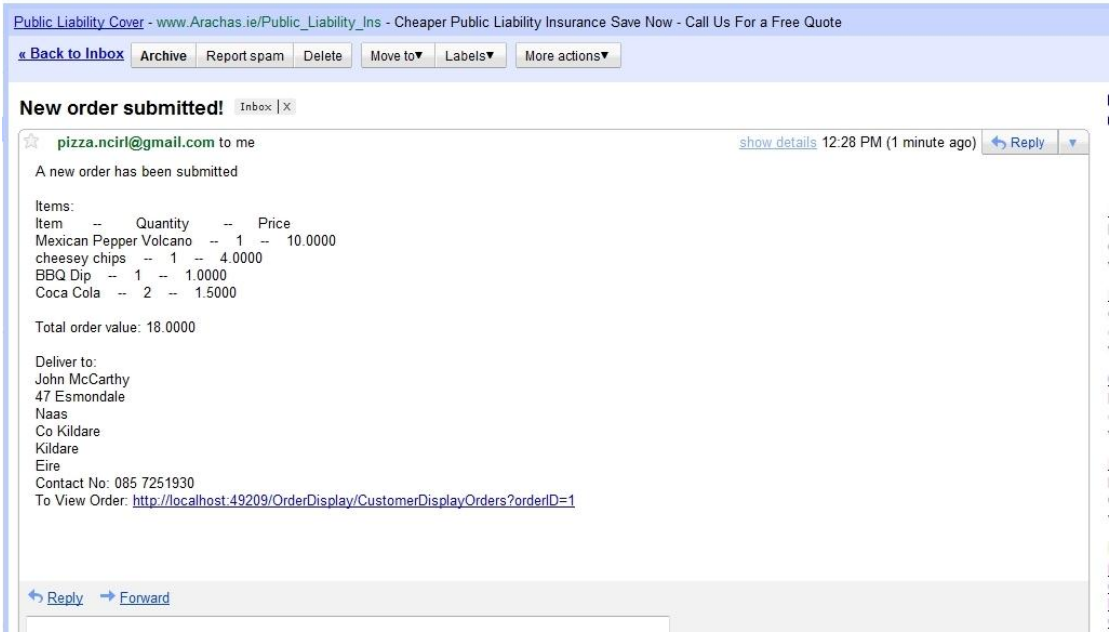

Figure 16 - Order Email Alert

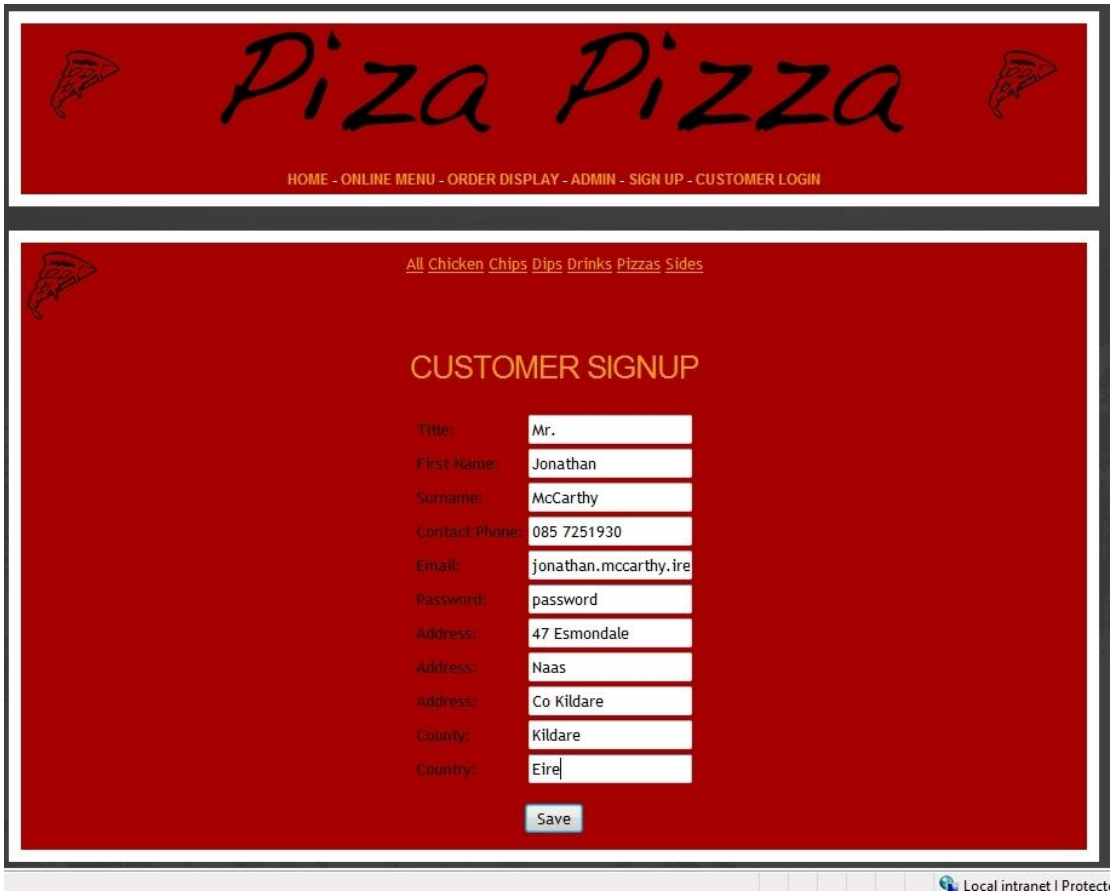

Figure 17 - Customer Signup

Figure 18 - Display Customer Details

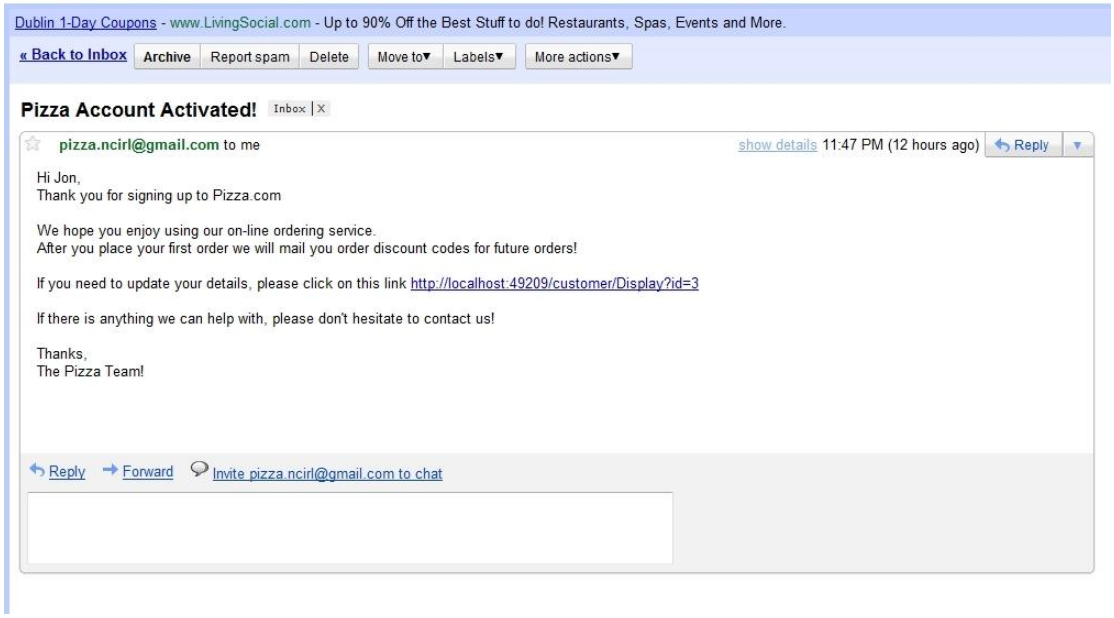

Figure 19 - Customer Signup Email

Appendix B - Admin Backend Functionality

The owner of Piza Pizza needs to be able to update the content of the Online Menu whenever it is needed. A back end system was created to facilitate this process. The second requirement is for the Restaurant to view any orders placed and the option to mark the order as complete when it has been sent out. This functionality is protected with a login process, with different login accounts for the owner and the store users.

Figure 20 shows the login window for the back end system. The edit menu functionality will be only accessible using the admin password, which will be set for the owner. Following a successful login, the owner will be shown the admin menu system (Figure 21). The admin menu system follows the same format of the Online Menu to provide a familiar environment to the user. All items are listed, and the category links can be used to view items by their assigned category. Each menu item has create, edit and delete links.

The create new menu item page is shown in Figure 22.

To create a new item, the user must input the following:

- Category
- Name
- Description
- Portion Size
- Price
- Image (Optional)

If no image is added, a generic placeholder will be used. Figure 23 shows the newly created item in the menu list.

The edit menu item option is similar to the create functionality and follows the same rules. Figure 24 and Figure 25 shows the pages for the update and its associated output.

If a menu item is no longer required, it can be removed by clicking the delete link.

If additional login accounts are required, the owner can create them (Figure 26).

All the Customers who signed up to the site can be viewed and updated if needed.

These details will be used as part of the site promotions in the next phase of the website. Vouchers and promotions will be sent via email to customers.

The restaurant needs to be able to view all orders made on the website. Figure 27 shows the Order Display page listing all unprocessed order from the website. The restaurant workers can view the specifics of the order by clicking the view link. This will show all details related to the order (Figure 30). When the order has been prepared and sent for delivery the workers can mark the order as completed on the system.

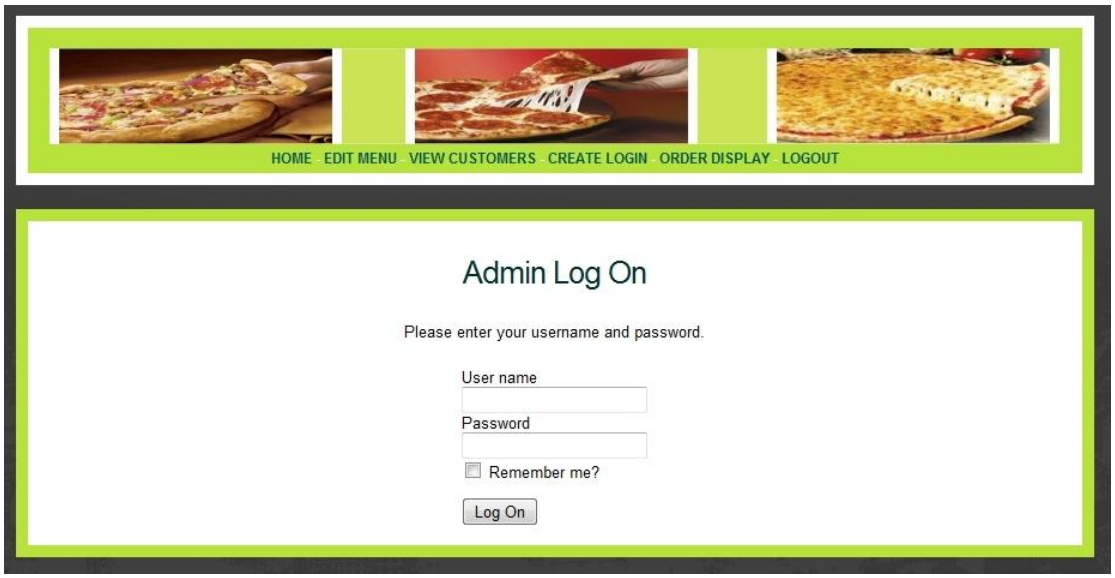

Figure 20 - Backend Admin Login

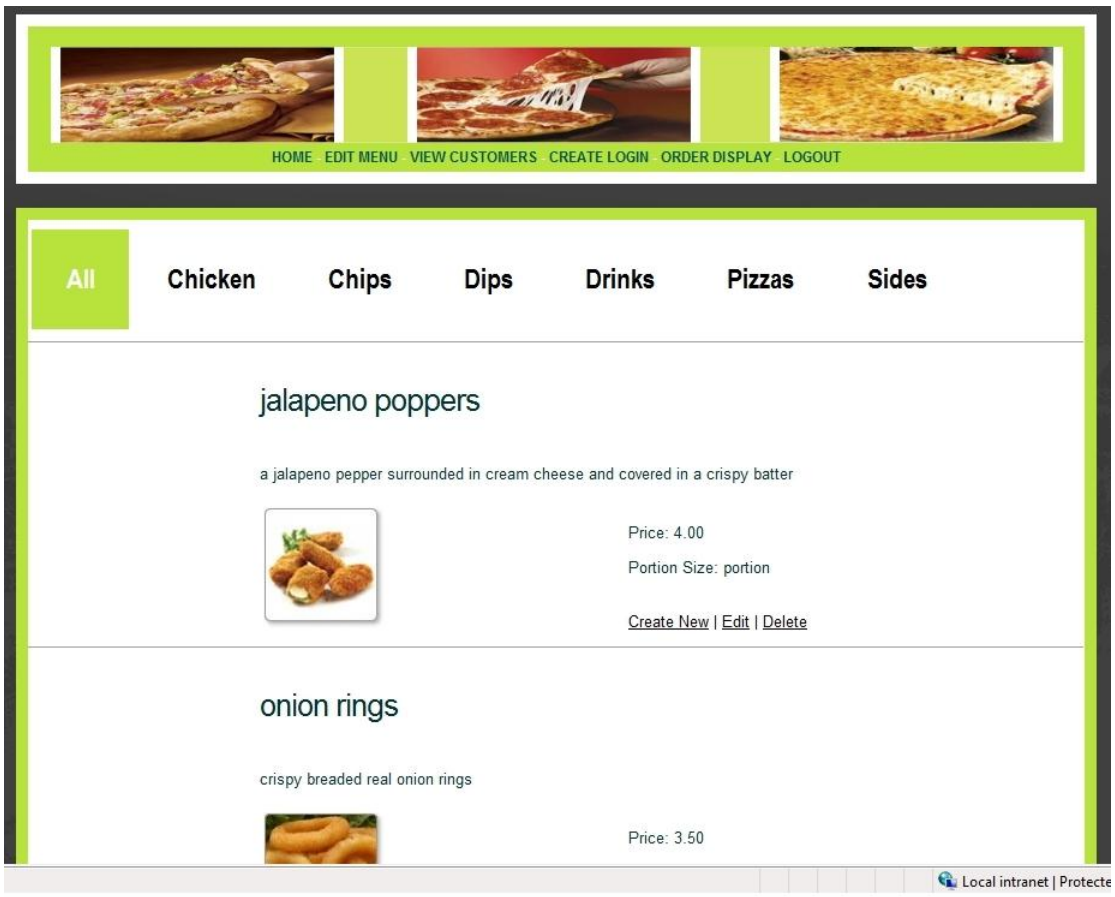

Figure 21 - Admin Online Menu

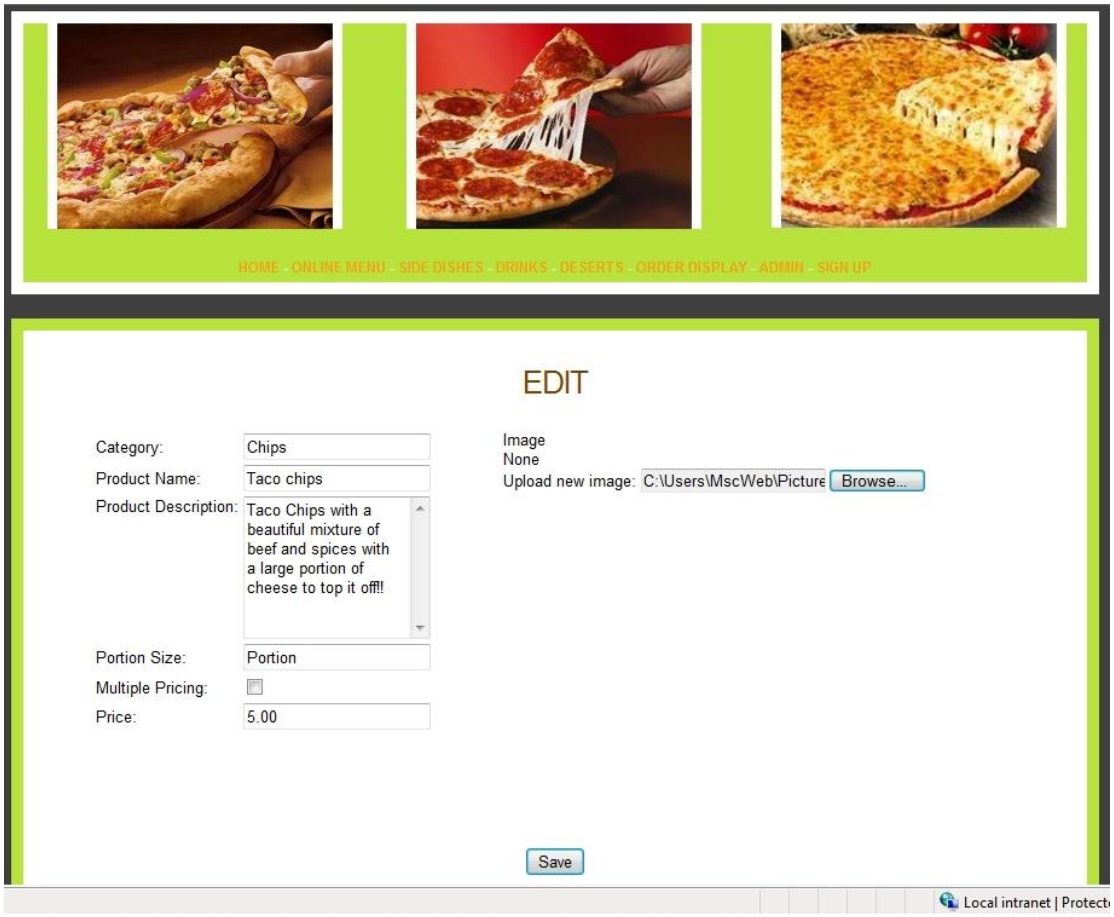

Figure 22 - Create Menu Item

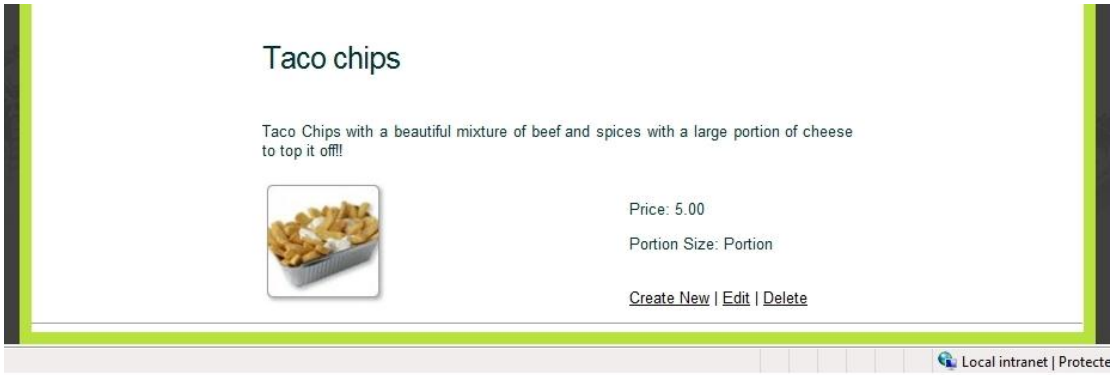

Figure 23 - New Menu Item Display

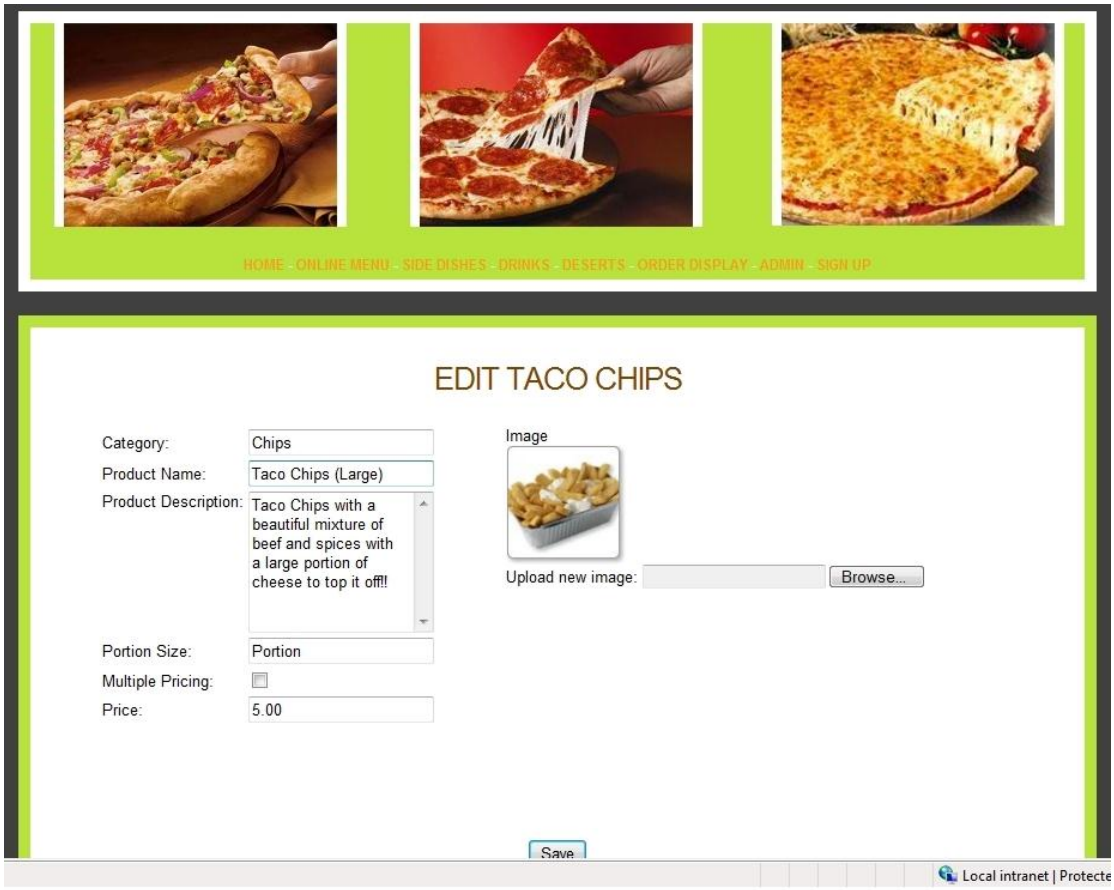

Figure 24 - Edit Menu Item

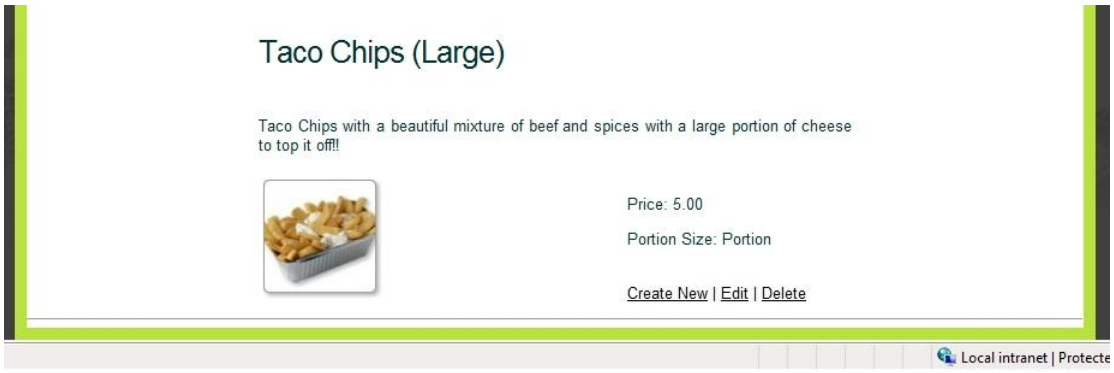

Figure 25 - Display Menu Item

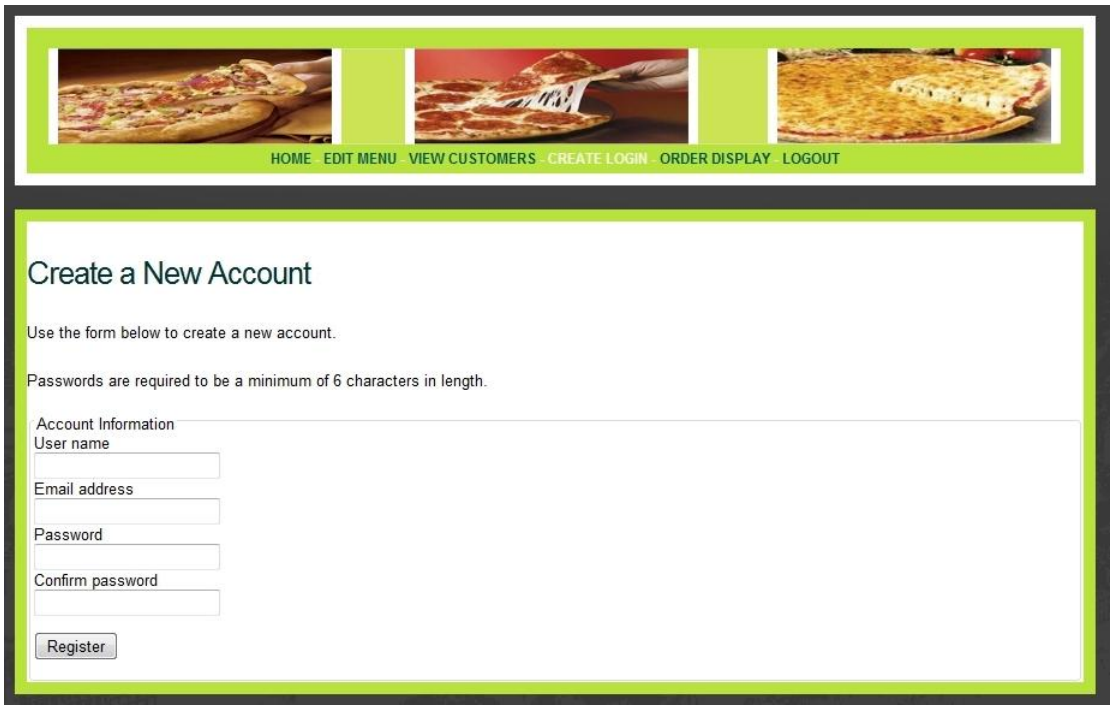

Figure 26 - Admin Account Management

Customer ID	Firstname:	Surname:	Address:	Address:	Address:	Email:	Password:	County:	Country	Edit
1	John	McCarthy	Address 1	Address 2	Address 3	jon@email.com	password	Kildare	Ireland	Edit
2	John	McCarthy	asdasd	asdasd	asdasd	jonmcire1@gmail.com	password	asdasdd	asdasdd	Edit
3	Jonathan	McCarthy	47 Esmondale	Naas	Co Kildare	jonathan.mccarthy.ire@gmail.com	password	Kildare	Eire	Edit

Figure 27 - Admin Customer List

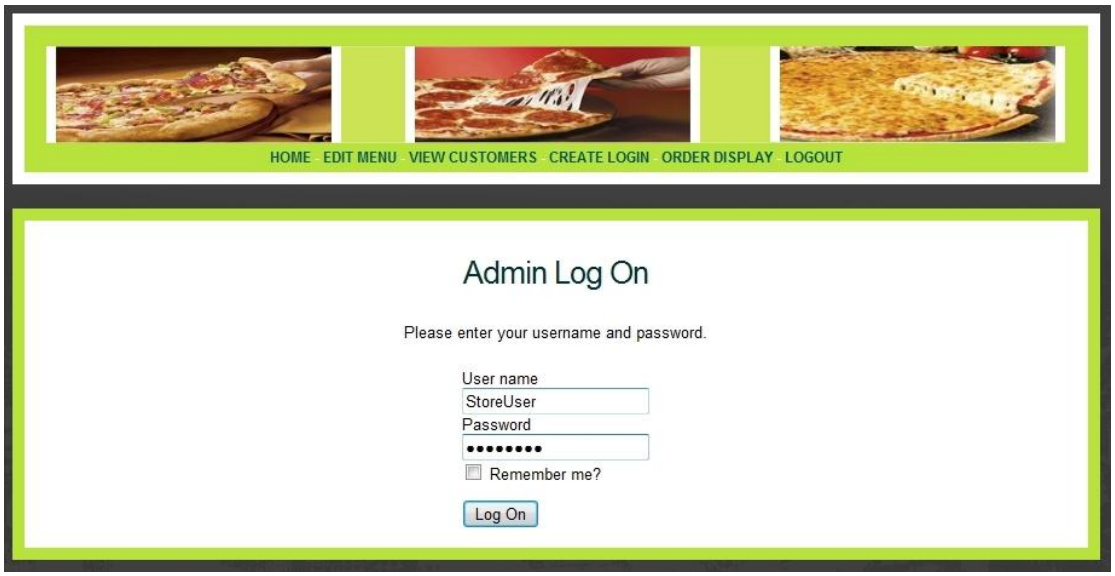

Figure 28 - Admin Login Page

Figure 29 - Admin Order Display

Figure 30 - Admin Individual Order Display

Appendix C – Use Case Diagrams

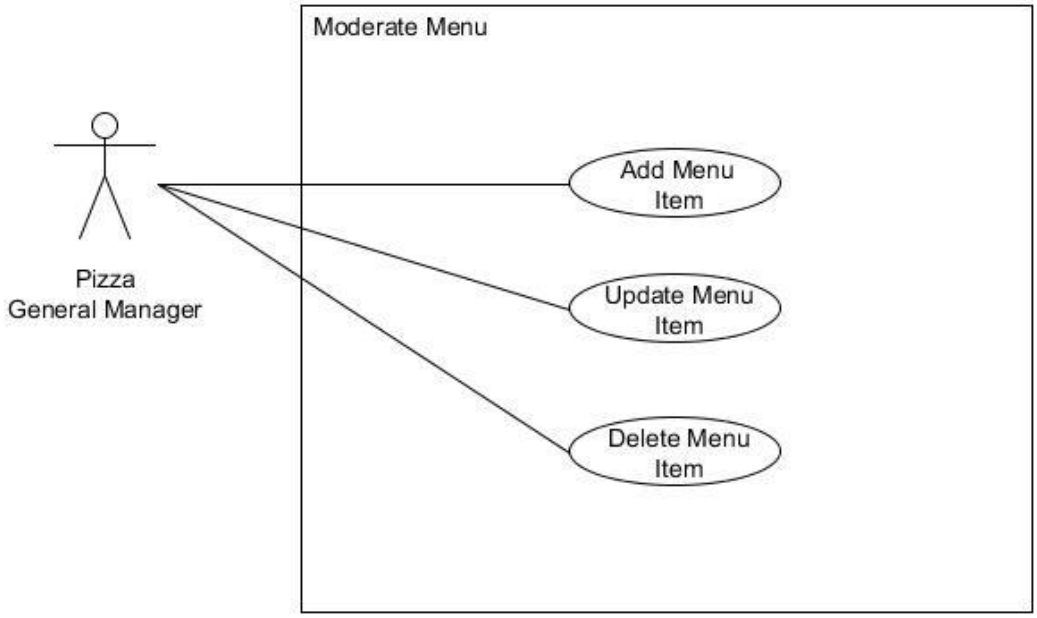

Figure 31 - Use Case Moderate Menu

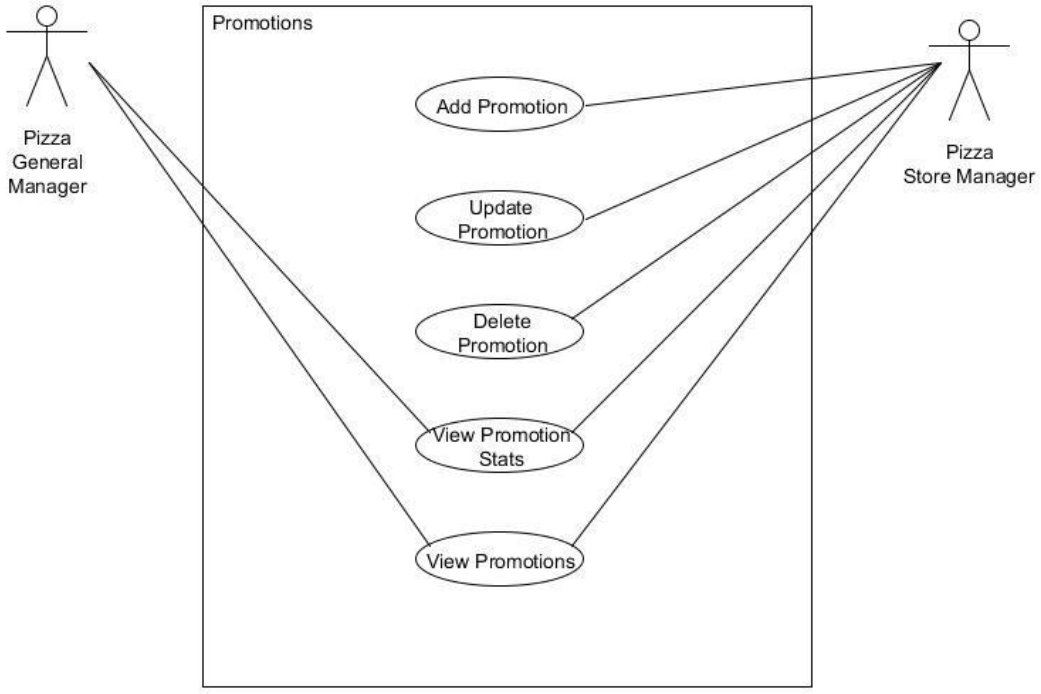

Figure 32 - Use Case Promotions

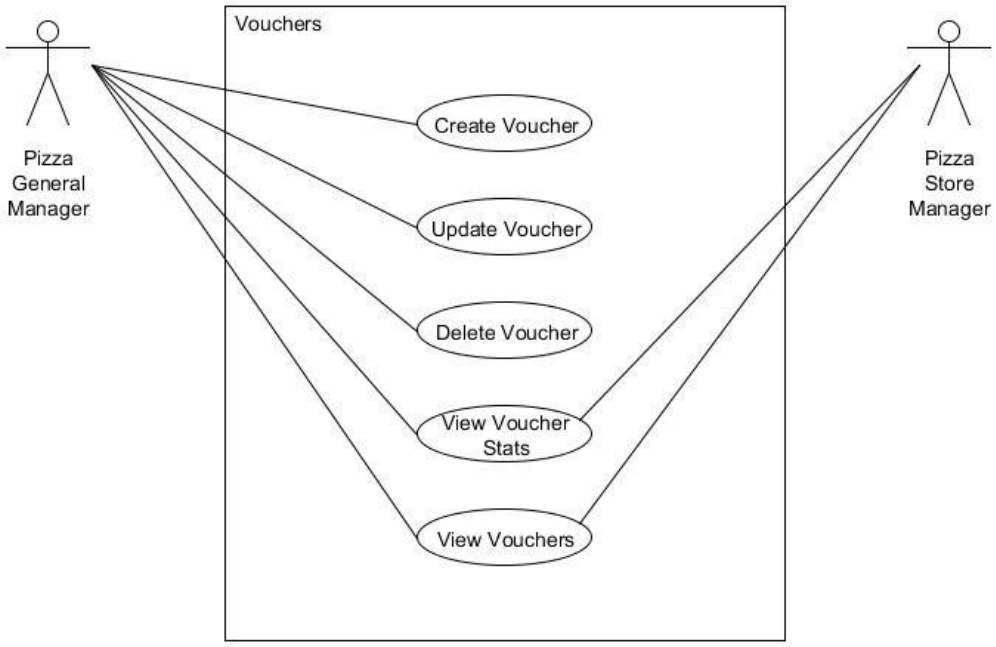

Figure 33 - Use Case Vouchers