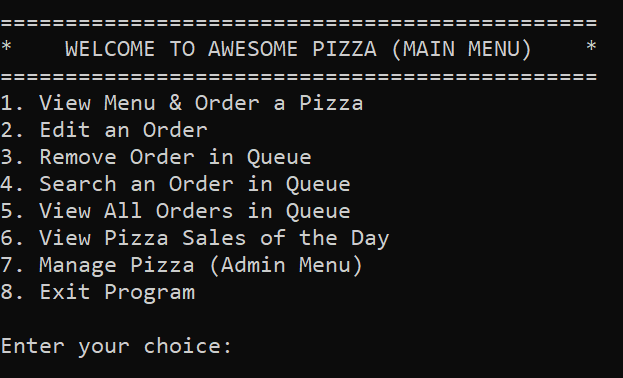
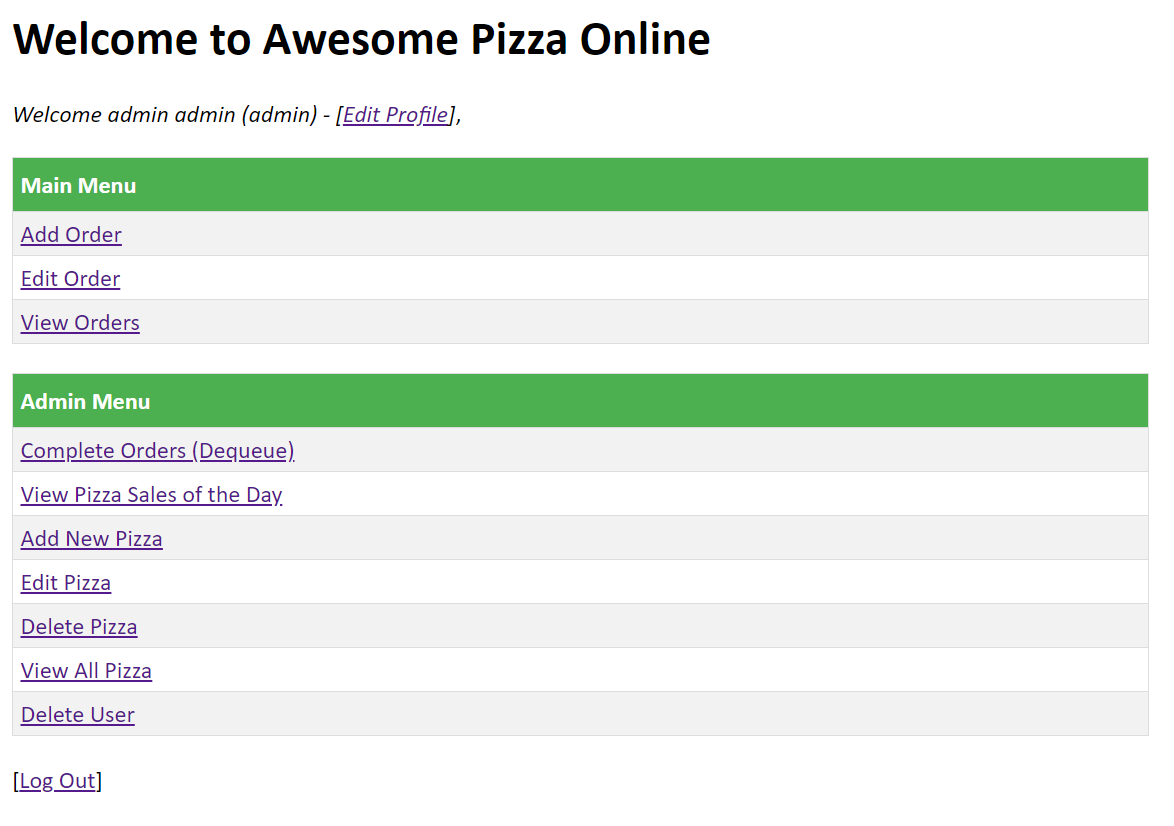
**GO IN ACTION 1 - ASSIGNMENT - WRITE-UP**

1. **Introduction**

I have chosen to use my GoAdvanced Assignment for this current assignment to convert to a client-server setup. The application I have developed is a pizza ordering system. The application is created on the idea of a first-in-first-out (FIFO) concept whereby an order is processed on a first-come-first-serve queue basis i.e. the first order that comes into the system must be processed before the next one can be processed. Thus the code will make use of the Queue data structures to manage orders. To manage the pizzas, I have selected to use the pointer-based LinkedList data structure. The screenshots below show the main menu before in terminal and after conversion to the client-server setup.



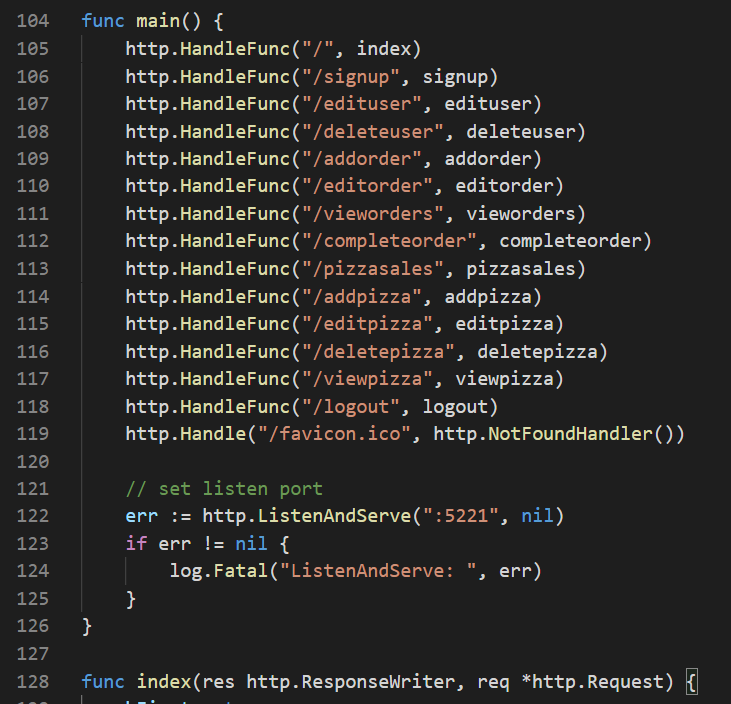
*Before: Main Menu in GoAdvanced Assignment submission*



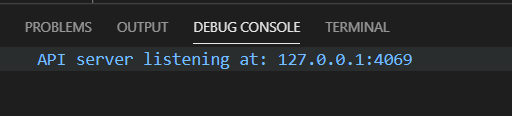
*After: Main Menu in GoInAction1 Assignment submission*

1. **Description of the Client-Server**

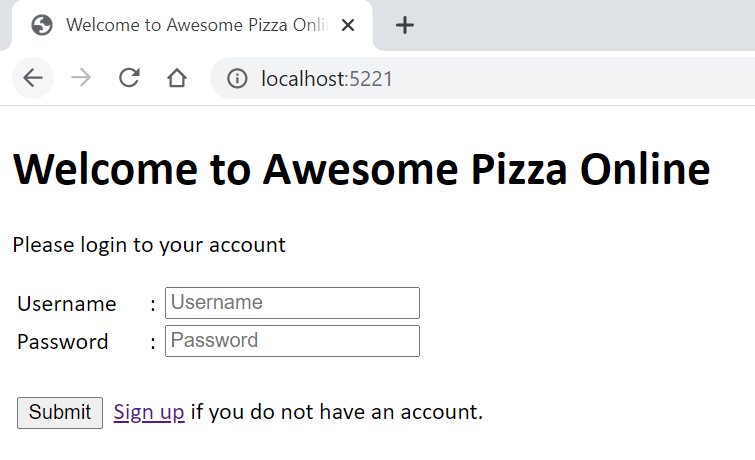
The application uses a simple HTTP server using the HandleFunc which will map the handler function into a default serveMux to a corresponding pattern string (the page in which we want to access). This adapter allows ordinary functions to be used as HTTP handlers and the function names are assigned directly as the handlers.



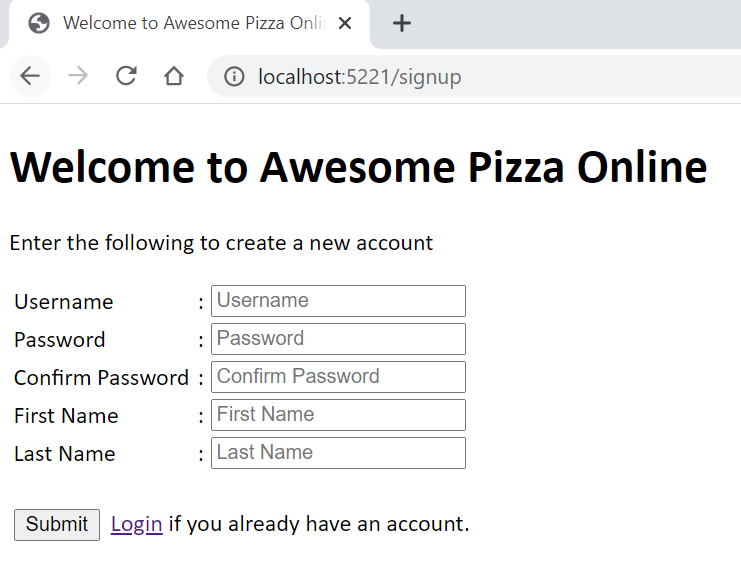
The server will listen and serve on the IP of 127.0.0.1 (localhost) at port 5221. When the server is run (F5), the following will be shown in the debug console of Visual Studio IDE. The server is started and ready.



To access the client, type in the URL: <http://localhost:5221/> in Chrome. The following page will be display. This is the default start up page which takes in a template from index.gohtml.

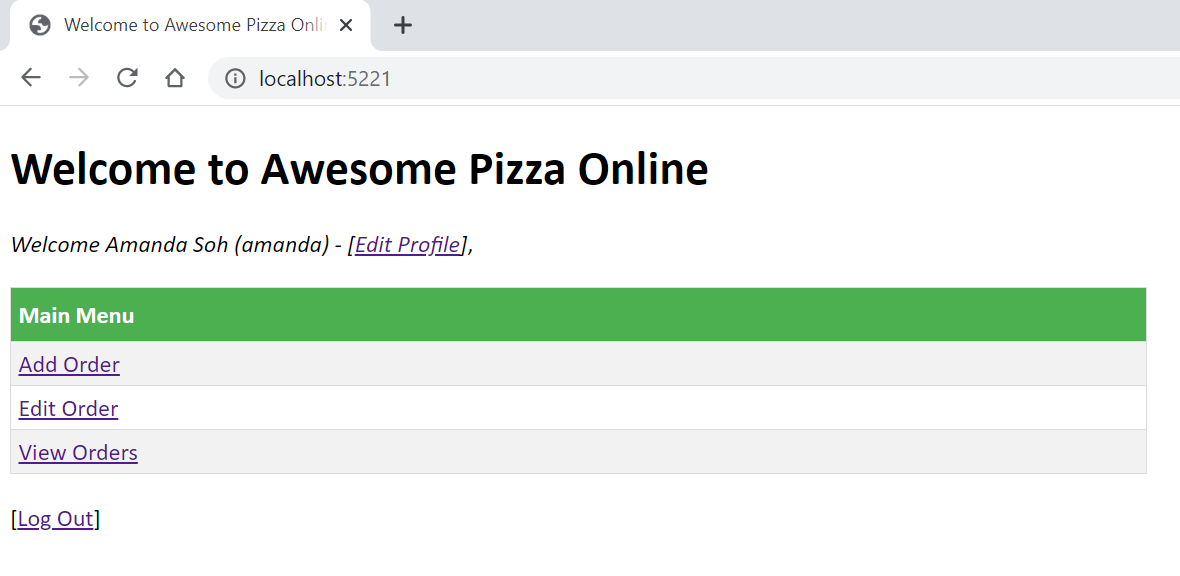


By default, the admin user login has been predefined in the codes with username ‘admin’ and password as ‘password’ (Note that security access features have not been incorporated into this version of the application). A new user can create an account by clicking on the [Sign up] hyperlink which will shows the page below.

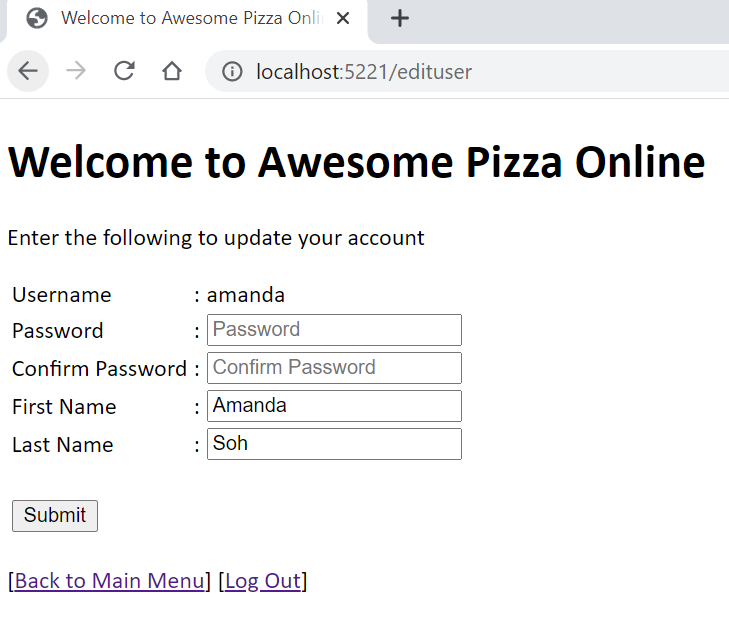


**General Users**

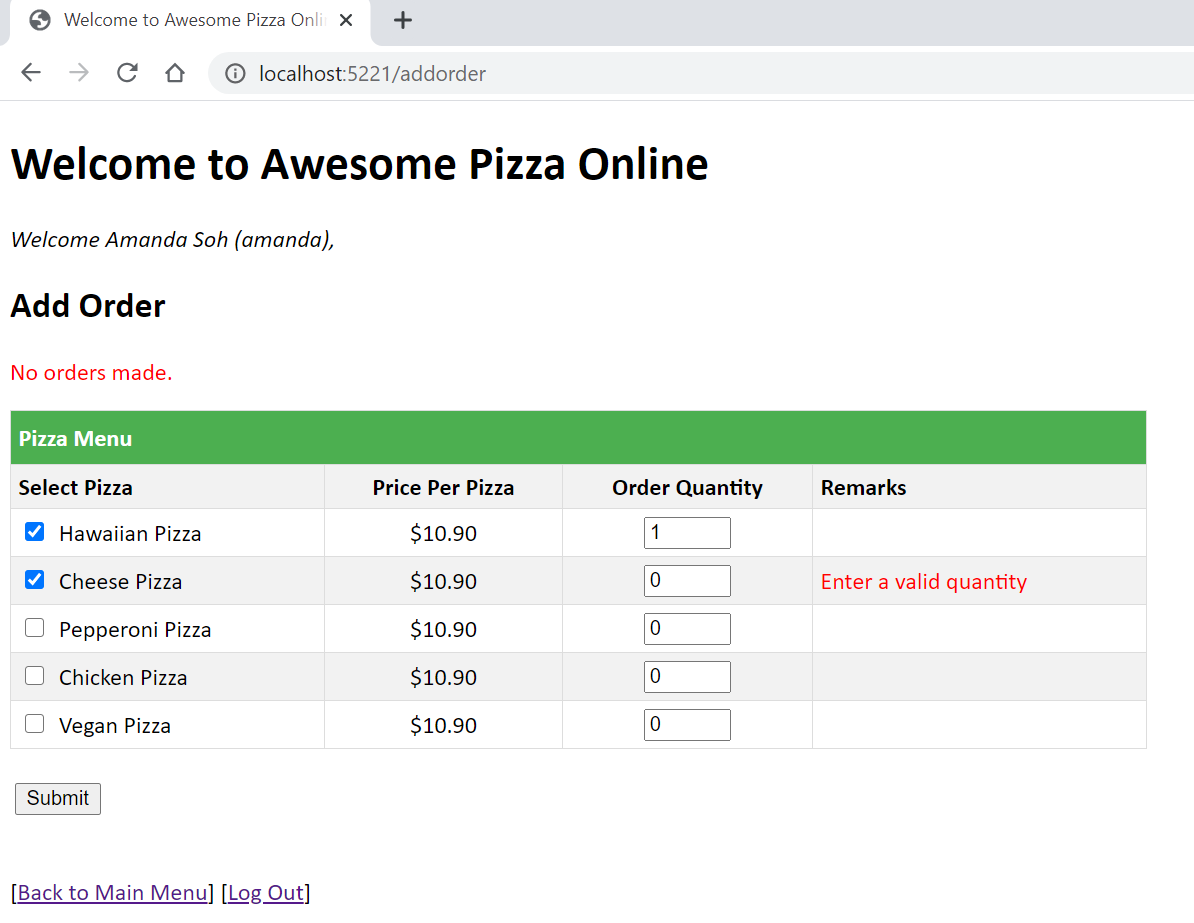
Upon valid submission of the account information, user will be directed back to the default index page now showing the user menu that is accessible by the user. Currently a user can add an order, edit an order and view orders that he/she has added.



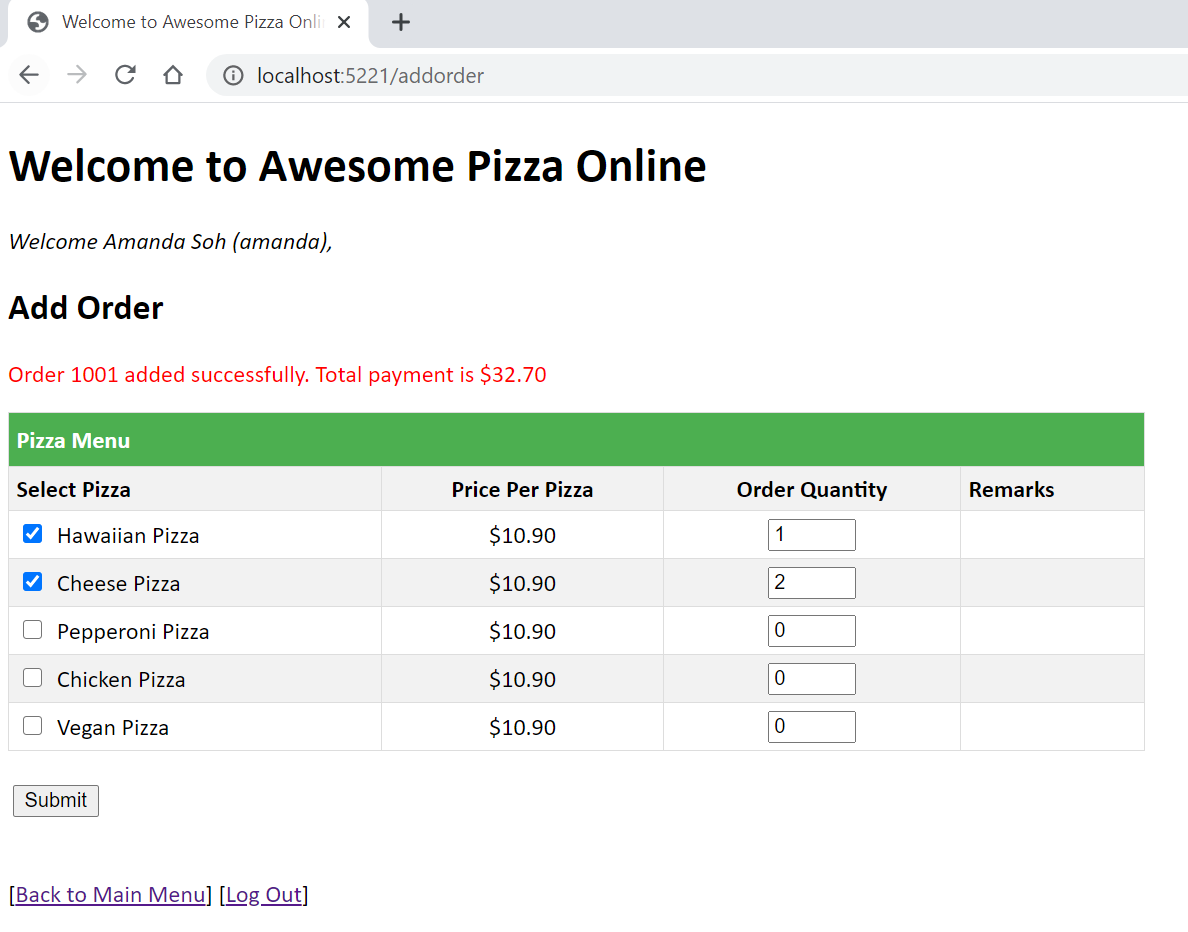
The user is also able to edit his profile by clicking on the [Edit Profile] hyperlink. User is not allowed to change his/her user name but he/she can update their names and change their password. Due to time constraints, the user profile is limited to this information, however for future upgrades, information like addresses, emails, phone numbers can be added to beef up the application.



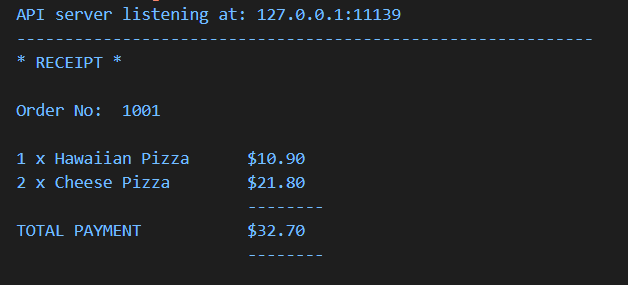
The [Add Order] hyperlink in the main menu will bring user to the order page where user can select the pizza he/she wants to order by checking on the checkbox next to the pizza name and entering a valid quantity. There is currently a limit on the quantity of 5 for each pizza selected. This is set in a global variable and can be changed easily.



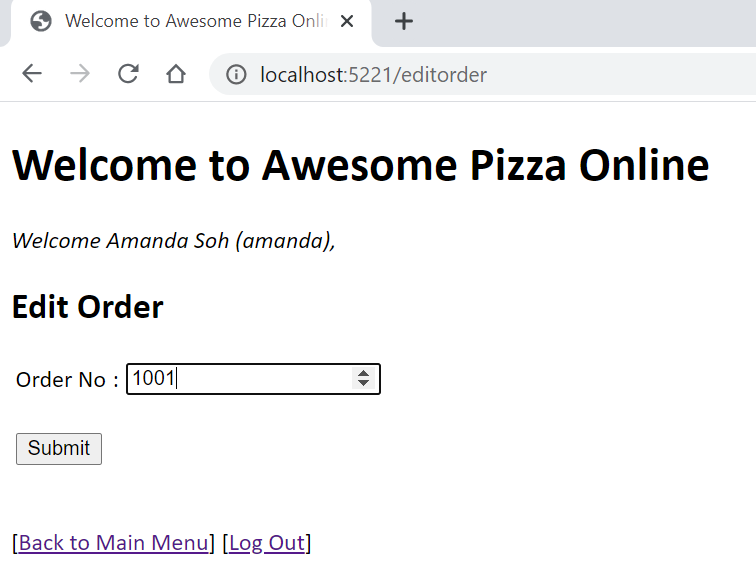
When a valid order is made, user will be prompted with a message on the same page.



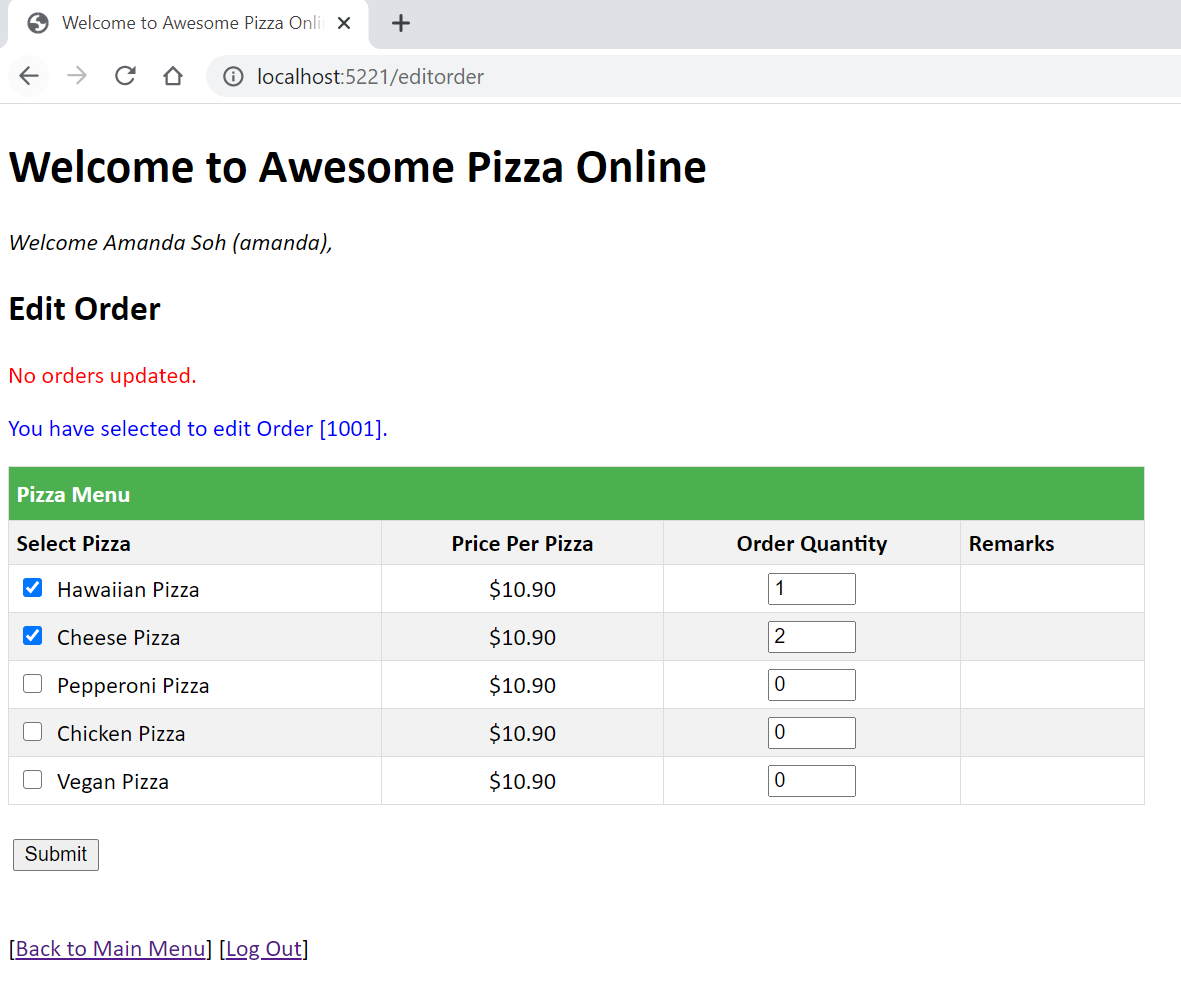
At the same time, a receipt will be printed at the server end to notify of the orders made.



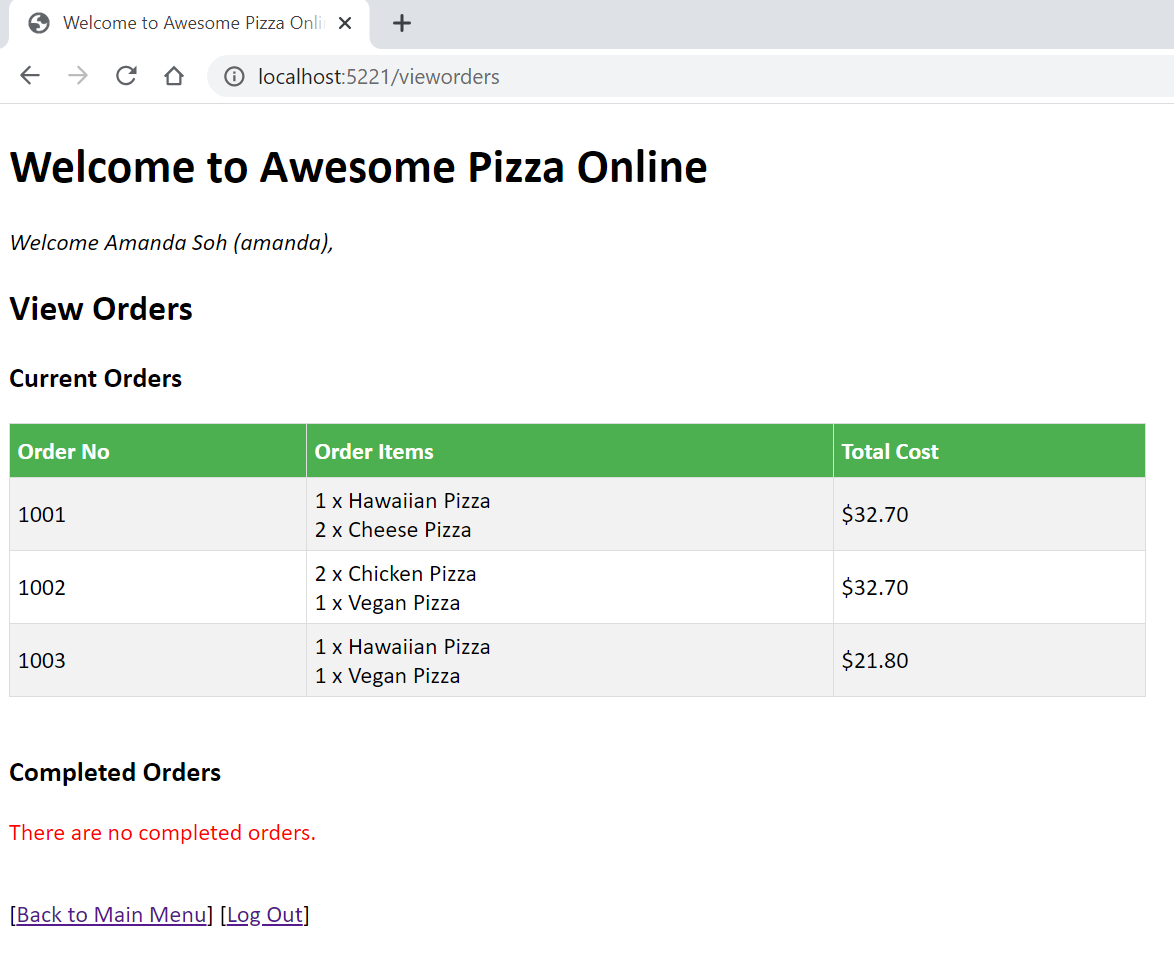
The [Edit Order] hyperlink in the main menu brings user to the edit order page where user has to enter the order number that he/she wants to edit. User is only allowed to edit orders that he/she had made.



When a valid order is found, the same page will be displayed showing a similar page as add order for user to amend the order information. A new receipt will also be printed at the server.

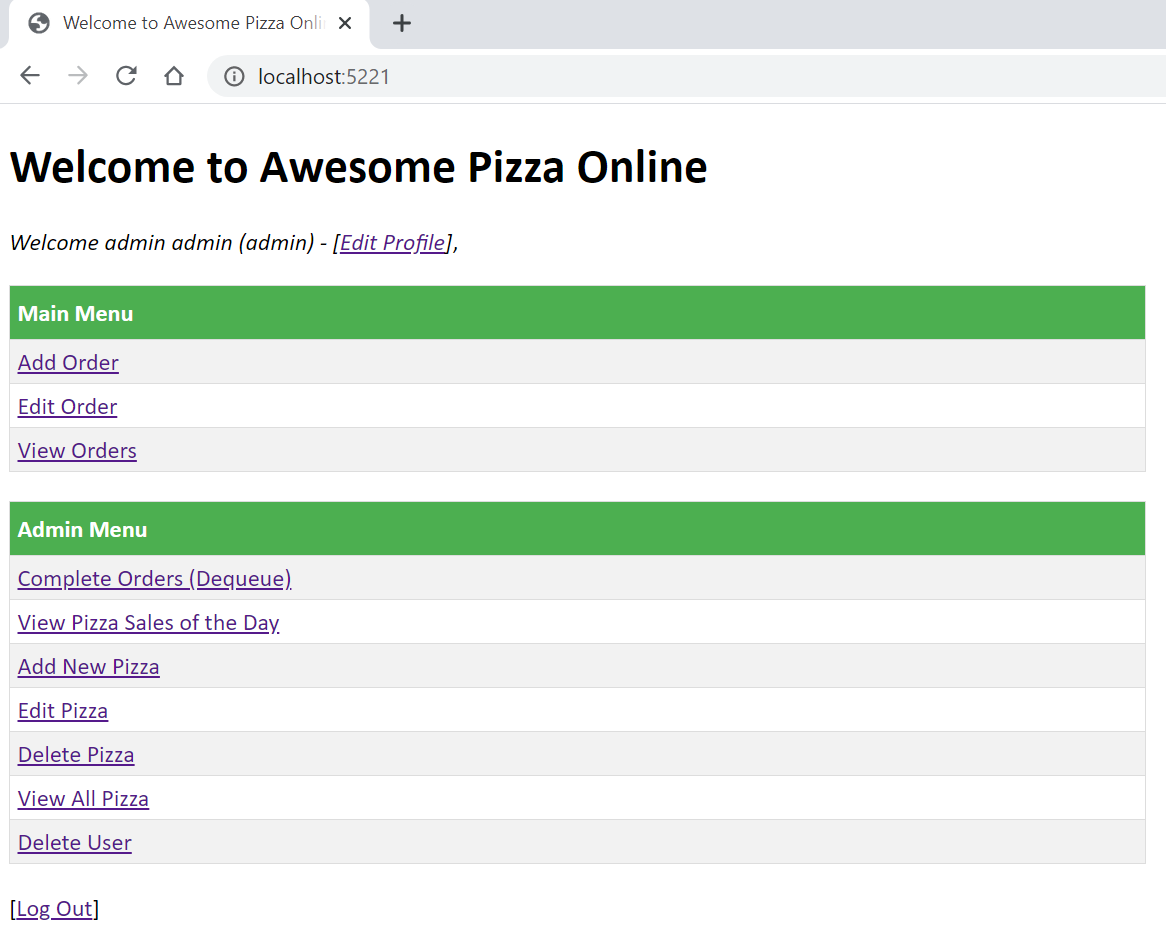


The last menu option for user is the [View Orders] hyperlink. This page will show the current orders as well as completed orders for the user (completed orders will be explained in admin user menu).

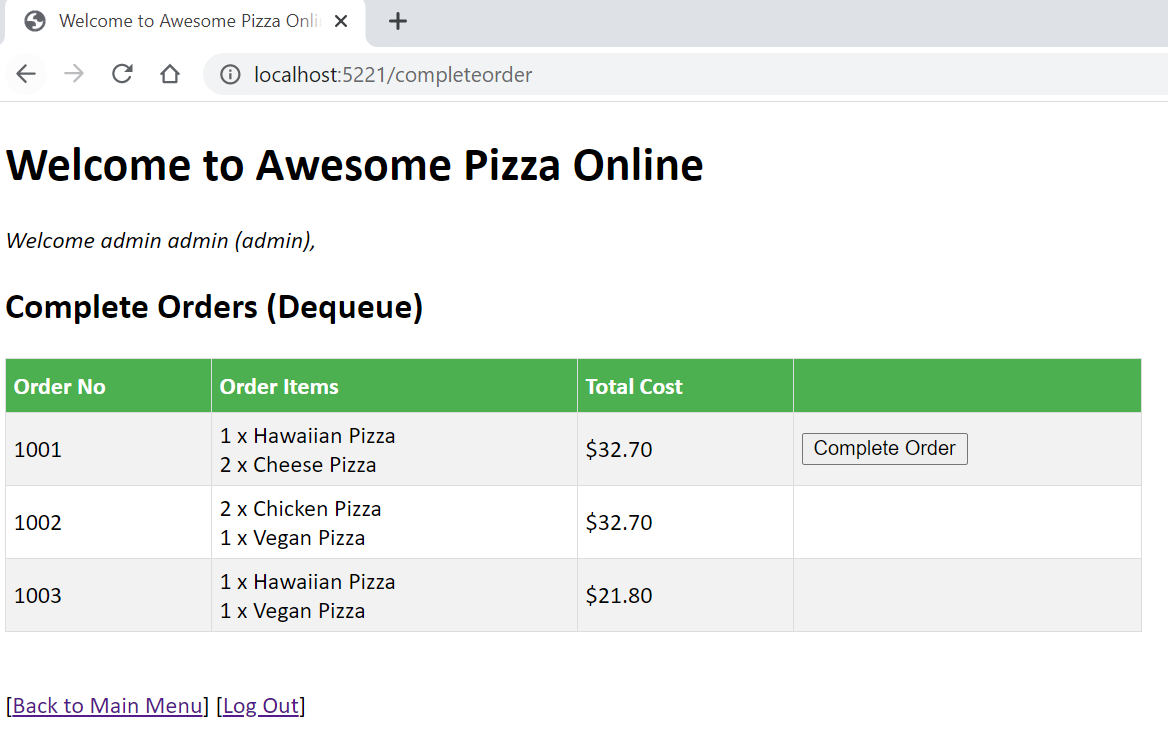


**Admin User**

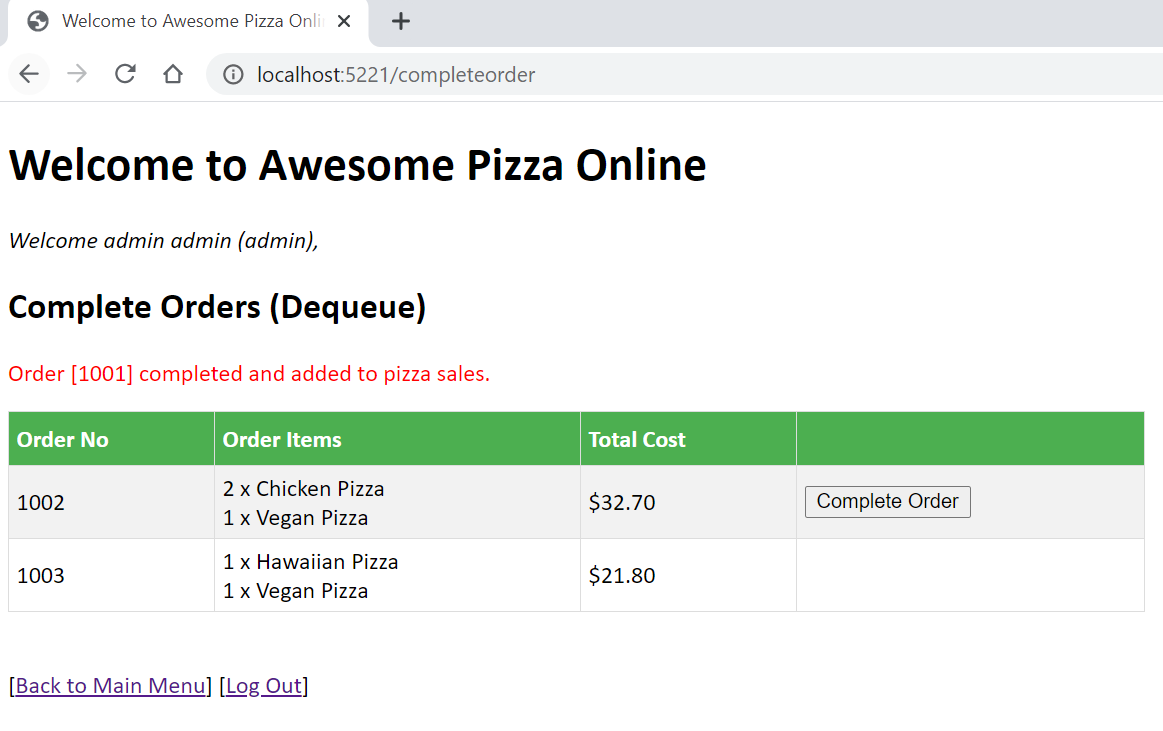
The following shows menu options when the admin is logged in. Admin is able to edit and view orders of all users. Besides the Main Menu which is the same as what a typical user sees, the admin user will be able to see the Admin Menu as well.



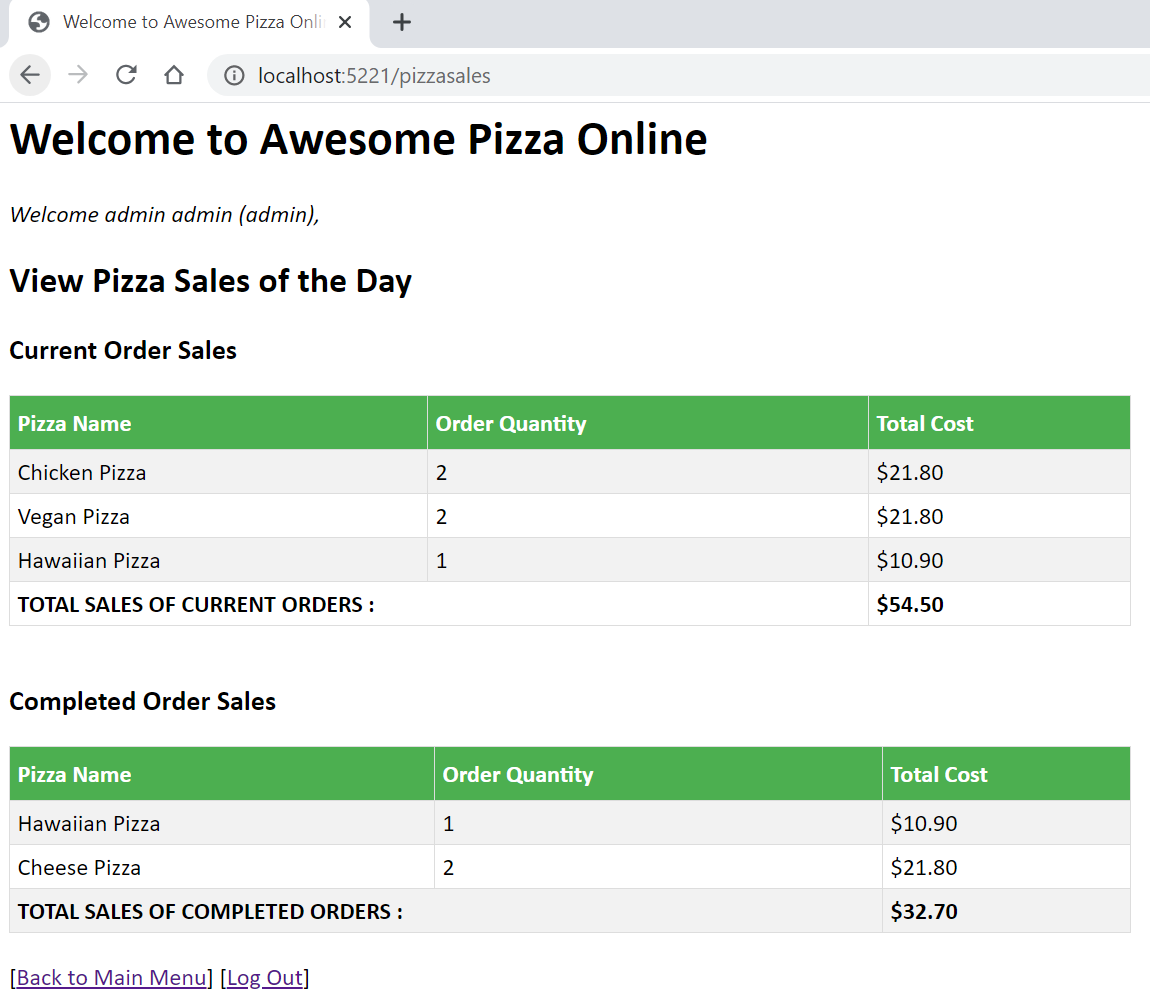
The [Complete Orders (Dequeue)] hyperlink allows the admin to remove an order from the queue using the Queue data structure. As mentioned in the introduction, a queue having a first-in-first-out concept will have the first order being dequeued one-by-one thus there is no need to enter the order no.



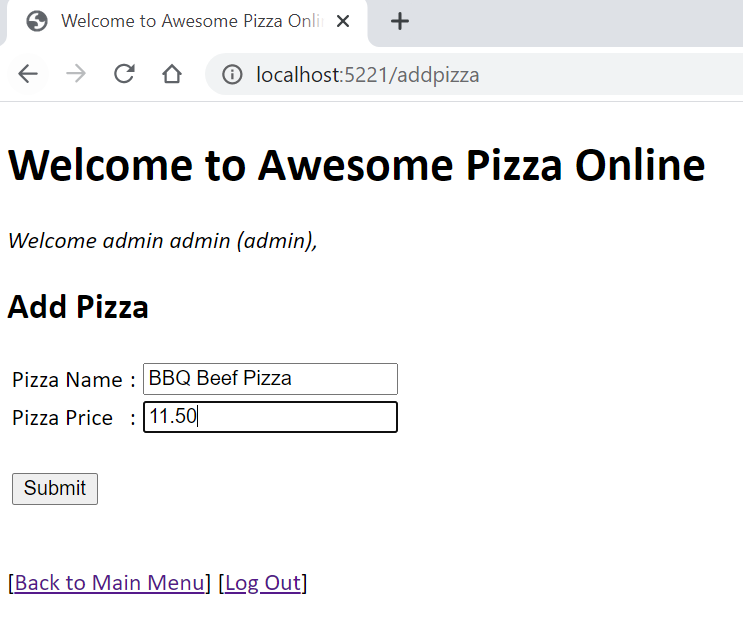
The [Complete Order] button will always be positioned at the next earliest order in the the queue.

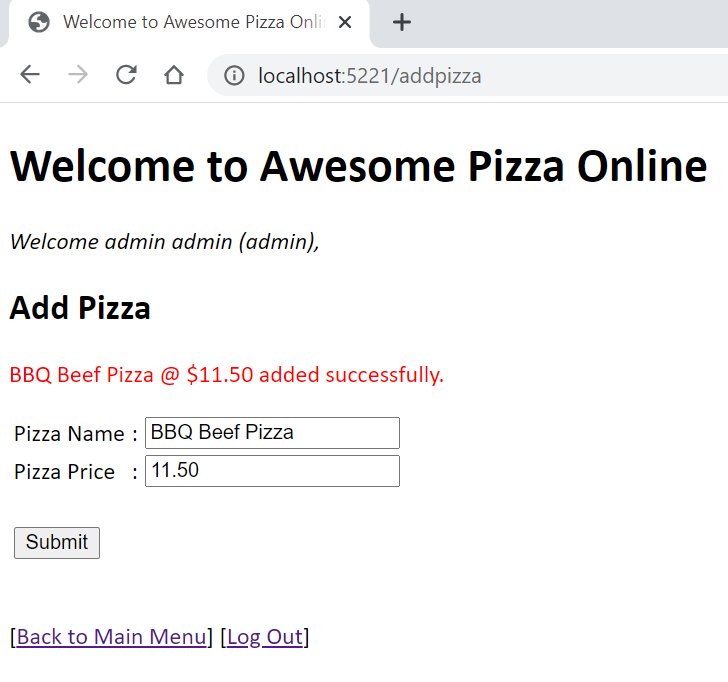


Once an order is dequeued, the pizzas ordered will be added to the pizza sales of the day. Thus in the [View Pizza Sales of the Day] hyperlink, the admin is able to view how many pizzas were purchased for the day and their total sales. In the page, the admin will be able to view pizzas that are currently in orders and those that were completed too.

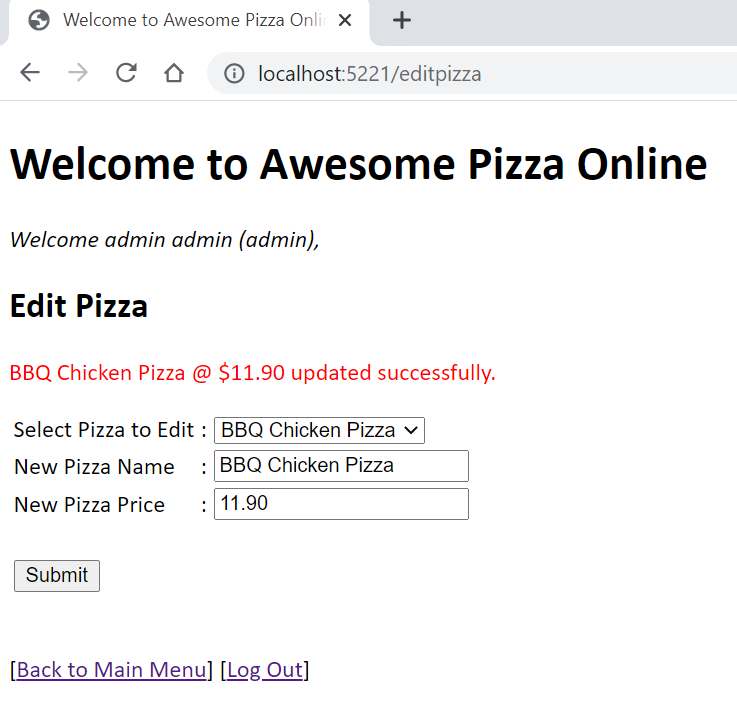


Next, the admin is able to add more pizza types to the menu through the [Add Pizza] hyperlink in the Admin Menu. During application runtime, I have chosen to initialize 5 standard pizzas to the menu. The standard price of each pizza is set at $10.90 in a global variable of the package main to facilitate the start of the application.

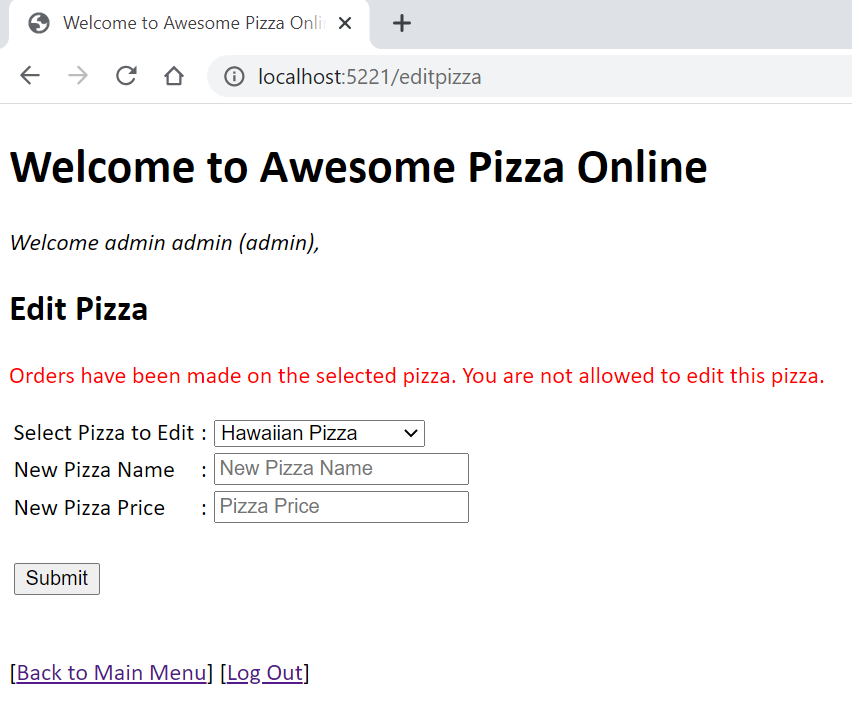




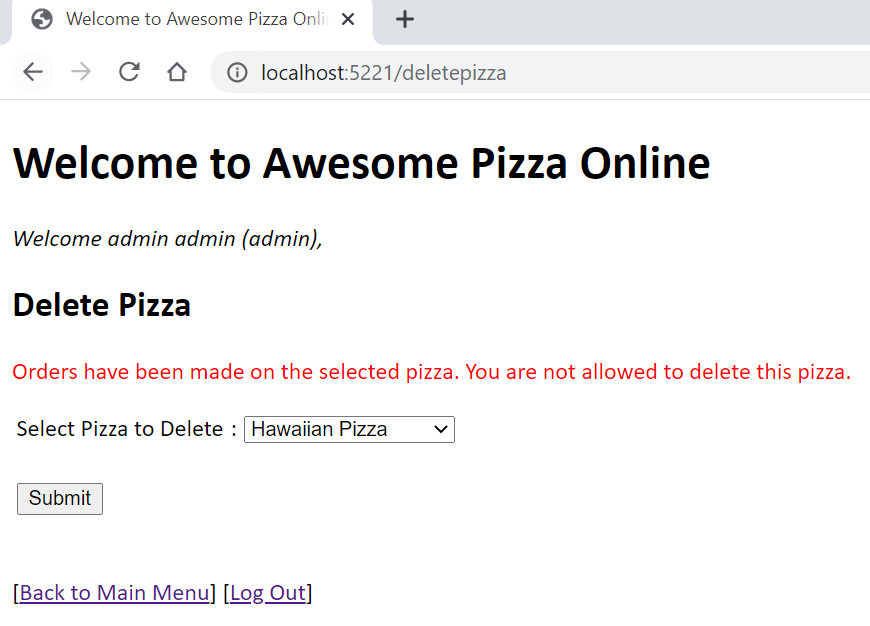
The [Edit Pizza] hyperlink in the Admin Menu allows the admin to edit a pizza which can be selected in a dropdown list. Admin can choose to edit just the pizza name or pizza price as well.



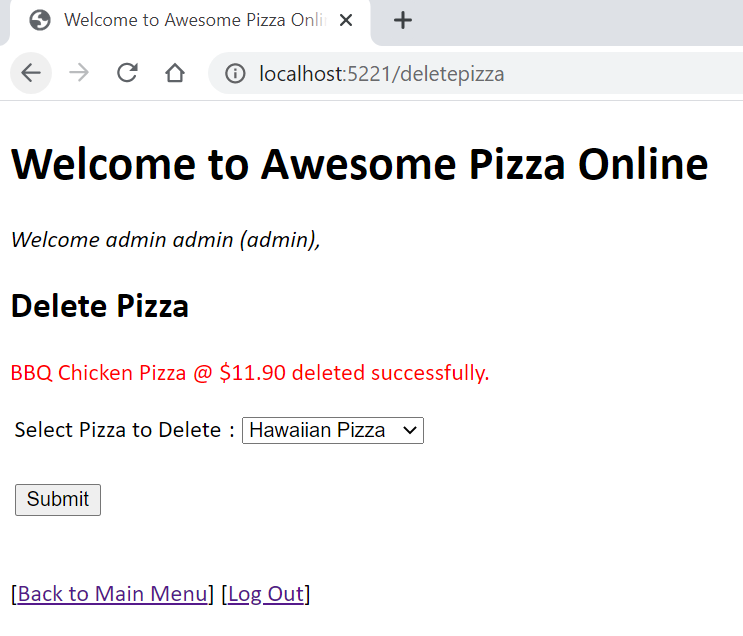
Note that if an order with the selected pizza has been created, admin is not allowed to edit this pizza.



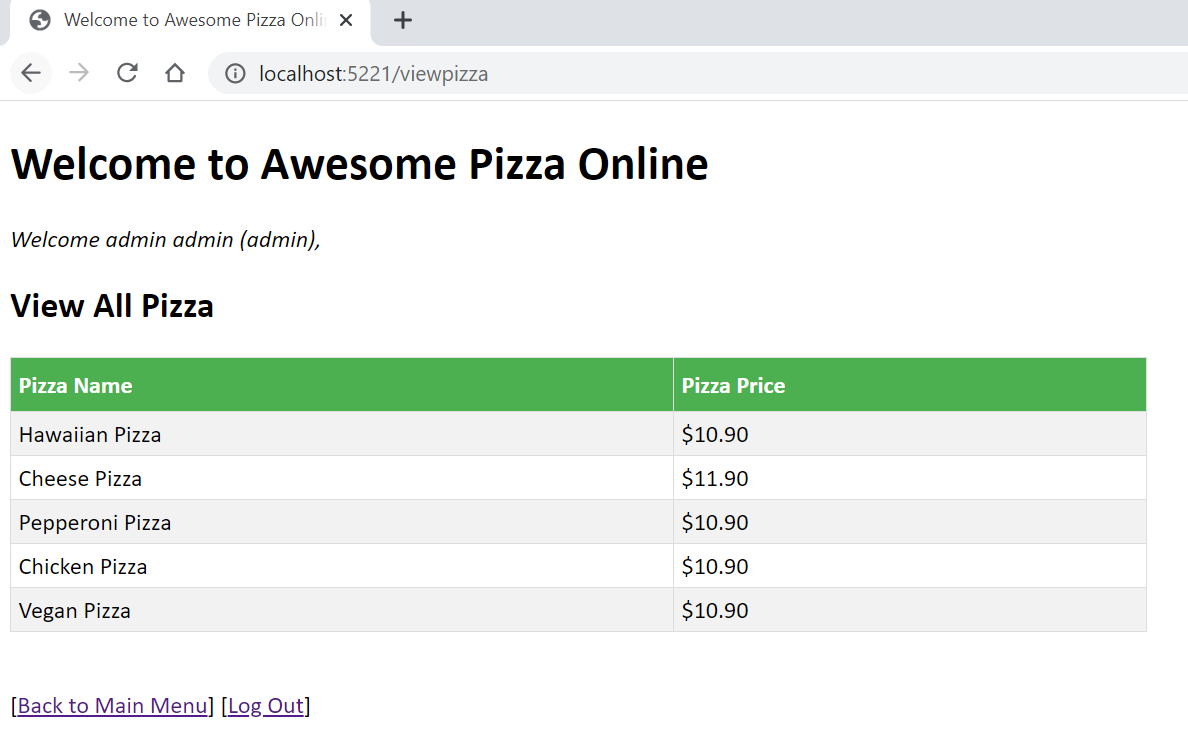
The [Delete Pizza] hyperlink in the Admin Menu allows the admin to delete a pizza. As with edit pizza, the selected pizza cannot be deleted if an order exists.



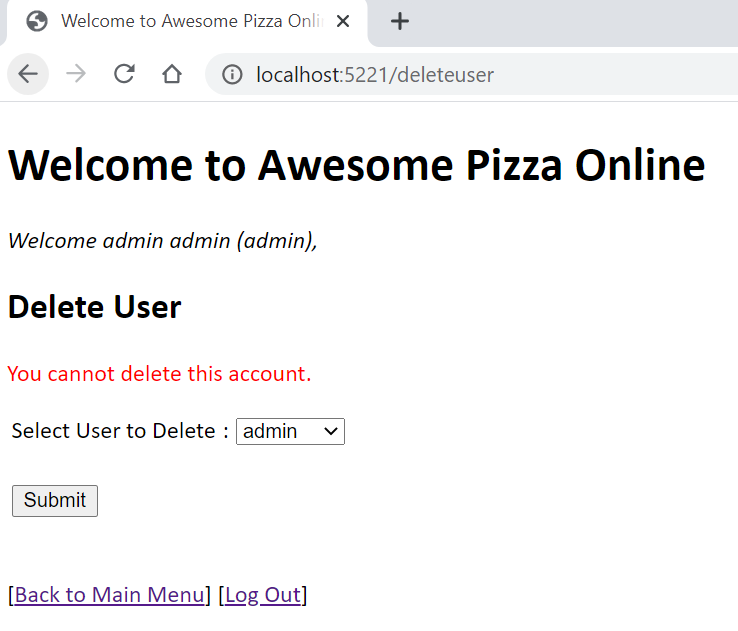
Otherwise, a message will be displayed to indicate that the pizza has been deleted.



Next, admin can choose to view all pizza by clicking on the [View All Pizza] hyperlink in the Admin Menu.

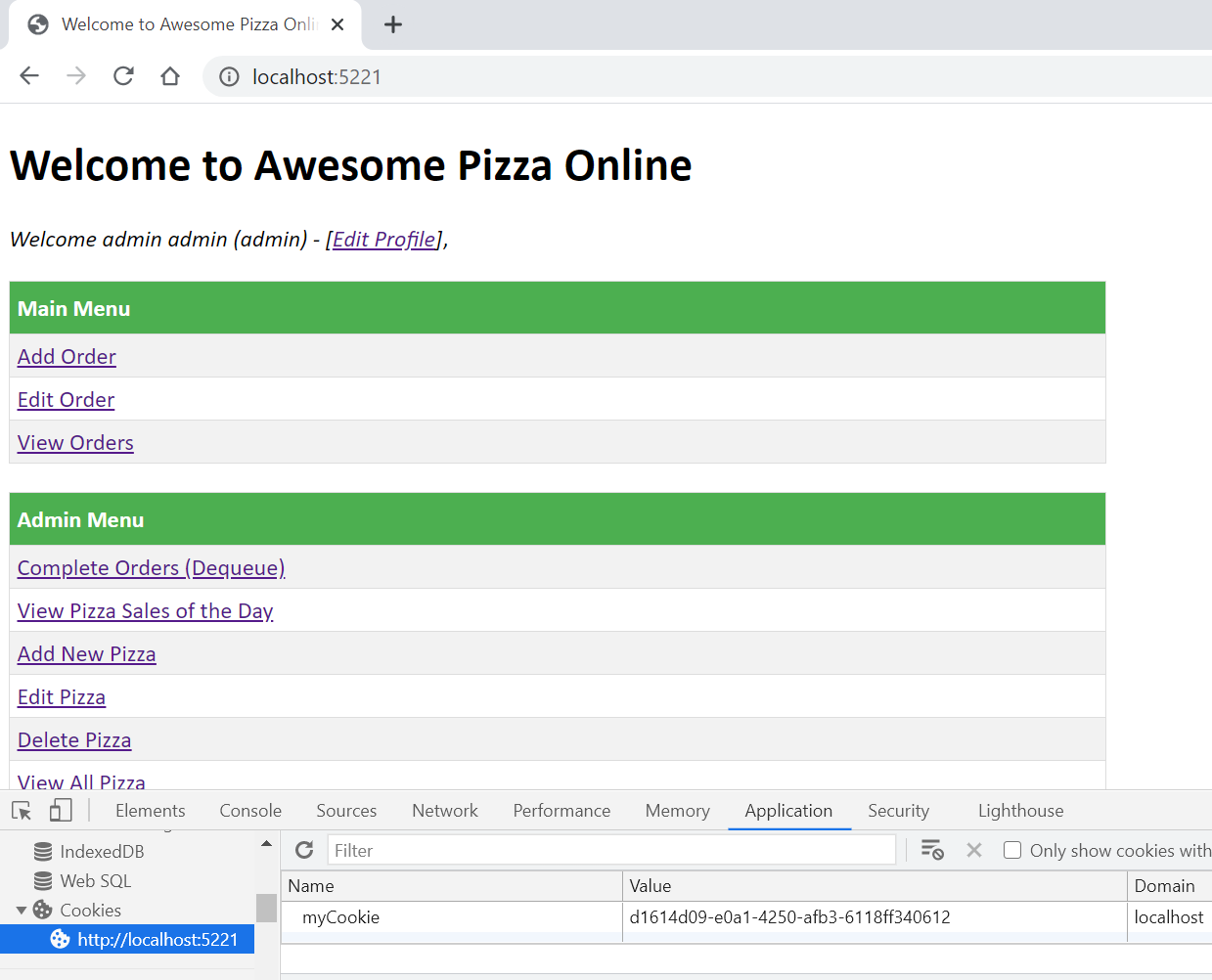


Lastly, an admin can delete user account (except itself) by selecting the [Delete User] hyperlink in the Admin Menu.



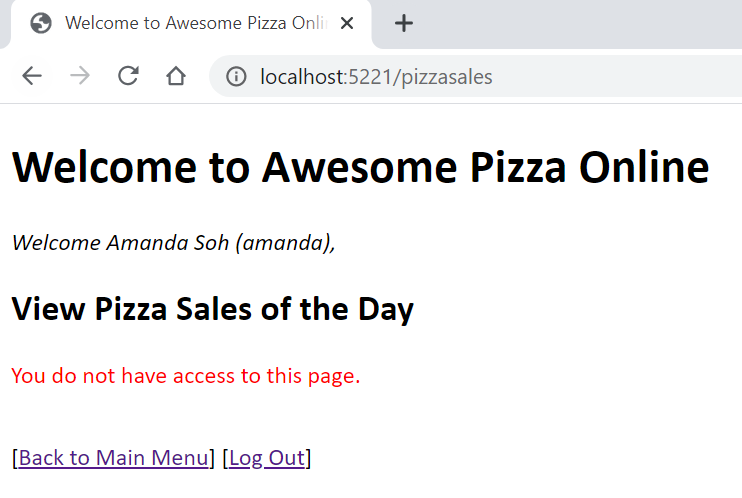
1. **Description of Important Main Features**
2. **Cookies**

The application uses cookies to maintain the state data of users login to the system. It is encrypted during transport from server to client. The cookie name given is “myCookie” for this application. The cookie is created with the use of a unique ID generated using the Universally Unique Identifier (UUID) through the package: import uuid "github.com/satori/go.uuid" which is a 128-bit number and it is set upon user first access to the index page. The cookie is deleted when user logout and a new cookie is created and set for the next login.



1. **Session**

The application keeps track of session with the value of the cookie that uses the unique UUID. The server will compares the session ID and retrieves the user name, which is then used to extract the user information. On every http.HandleFunc(), the user information will be retrieved and checked upon whether user has already login. If user has not login, he/she will be directed to the index page to prompt for login. With the user information, I am able to restrict user access on the information displayed by the templates e.g. showing Admin Menu only to the admin user. And when a non-admin user tries to access a admin user page for instance: <http://localhost:5221/pizzasales,> they will be denied access.



1. **Templates**

The application uses .gohtml templates files to personalize the results I want to display for the website to be shown to the client. The page display is written in HTML inside the templates whereby in the main package, the directory of files are read using: tpl = template.Must(template.ParseGlob("templates/\*")) in the func init() so that the files are executed only once. The output is parsed using template.ExecuteTemplate in each http.HandleFunc(). By using templates, it is easy to apply HTML codes to improve usability. CSS styles can also be applied to enhance the layout to create better aesthetics aspects to the users.

1. **Description of Error Handling and Concurrency Mechanism**
2. **Log.Fatal**

In the main(), log.Fatal is used when listening to the port 5221 using err := http.ListenAndServe(":5221", nil). It is called when listen failed and err is returned, triggering log.Fatal("ListenAndServe: ", err). The fatal function is similar to the print function with the difference that it calls upon os.Exit(1) at the end.

1. **Errors.New**

Error handling is managed in most of the functions used in the packages I have created by incorporating errors.New() and returning the error to the calling program. Errors will be printed for the user if it is not nil.

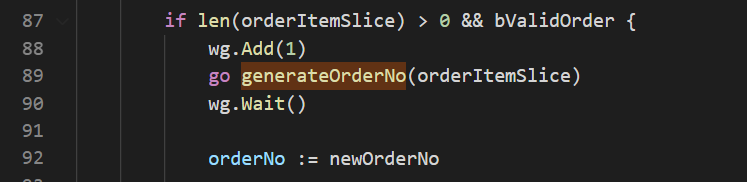
1. **Panics**

Capturing of panics runtime errors have been added into some functions in the packages I have created to handle the runtime error that might occurs when nil cannot be assigned to a currentNode. To demonstrate that panics have been managed properly, I have printed the panic errors into the server side terminal and the client is still able to run without the program collapsing.

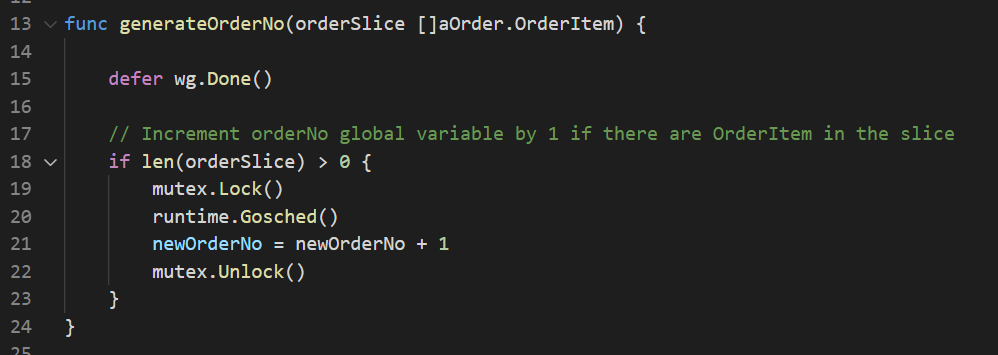


1. **Concurrency**

To handle concurrency, I have decided to use WaitGroup and mutex in the func addorder(res http.ResponseWriter, req \*http.Request) {} of the order.go file. A goroutine is created in the func at the following lines:



Then I added a mutex.Lock() and mutex.Unlock() when newOrderNo is incremented. This is done to prevent concurrency issues and the same newOrderNo is used concurrently.



1. **Instructions on How to Run Application**

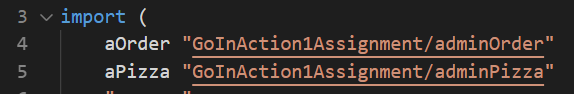
* For information, my program is currently running Go version - Go1.16.3
* My project files resides in my local C:\Users\Amanda\go\src\GoInAction1Assignment
* To run the program, run in cmd prompt: go install in the respective folders of the 2 packages that I have created as follows.
* Install package in GoInAction1Assignment/adminPizza



* Install package in GoInAction1Assignment/adminOrder



* It is important to use the same folder name structure because the import of the packages in order.go and admin.go takes the following format.



* When the packages are installed successfully, run F5 in Visual Studio IDE on main.go file. Alternative you can also run in cmd prompt: go run. (because the package main has been split into 4 different files - admin.go, main.go, order.go and user.go)

Remember to input the period “.”



* The server will start running.
* Open the browser in Chrome and type in <http://localhost:5221/> to start the client.