Software Engineering

2509ICT

Assignment 2

Lecturer: Larry Wen

Tutor: KB

Workshop: Thursday 10am

Total Errors: 8
Inspection Errors: 3
Testing Errors: 5
Before/After Compile Errors: No compile with php it is all dynamic. Therefore, leaving this blank.
Lines: 505 Function, 384 View, 889 Total

Group Number: 1

Group Leader: Aidan McMillan

Group Members

Student ID	Student Name
S5013937	Cambridge, Jake
S2944951	Chowdhury, Wasif
S2930944	Feng, Haoxuan
S2807480	Graham, Alex
S2946192	McMillan, Aidan

1 CONTENTS

2 PROJECT PLANNING AND DOCUMENTATION

2.1 TIME SCHEDULE

Task		Plan				Actual		
#	**Task Name**	**Student**	**Planed Time**	**Cumulative Time**	**Finished Date**	**Time**	**Cumulative Time**	**Finished Date**
1	Test plan	Haoxuan FENG	3 hours	3 hours	14/10/2016	3 hours	3 hours	14/10/2016
2	Unit and Component Test Cases	Aidan McMillan Haoxuan FENG	3 hours	6 hours	14/10/2016	3 hours	6 hours	14/10/2016
3	Integration and System test cases	Aidan McMillan Haoxuan FENG	3 hours	9 hours	14/10/2016	3 hours	9 hours	14/10/2016
4	Inspection Plan	Alex Graham	5 hours	14 hours	16/10/2016	5 hours	14 hours	16/10/2016
5	Menuitem Implementation	Aidan McMillan	3 hours	17 hours	16/10/2016	3 hours	17 hours	16/10/2016
6	Inspection One	Alex Graham	5 hours	22 hours	16/10/2016	5 hours	22 hours	16/10/2016
7	Order Implementation	Aidan McMillan	3 hours	25 hours	16/10/2016	3 hours	25 hours	16/10/2016
8	Inspection Two	Alex Graham Jaike Cambridge	5 hours	30 hours	16/10/2016	5 hours	30 hours	16/10/2016
9	Unit and Component Testing	Aidan McMillan Haoxuan FENG	4 hours	34 hours	19/10/2016	4 hours	34 hours	19/10/2016
10	Integration and System Testing	Aidan McMillan Haoxuan FENG	4 hours	38 hours	19/10/2016	4 hours	38 hours	19/10/2016
11	Defect Analysis	Aidan McMillan Haoxuan FENG Alex Graham	3 hours	41 hours	20/10/2016	3 hours	41 hours	20/10/2016
12	User Manual	Aidan McMillan	1 hours	42 hours	20/10/2016	1 hours	42 hours	20/10/2016

2.2 Total Working Hours

Student Name	Plan (hours)	Actual (hours)
Cambridge, Jake	10 hours	3 hours
Chowdhury, Wasif	10 hours	0
Feng, Haoxuan	10 hours	20 hour
Graham, Alex	10 hours	20 hours
McMillan, Aidan	10 hours	20 hours
Total Working Hours	50 hours	20 hours
Average Working Hours Per Person	10 hours	11 hours

2.3 Effort and Contribution Table

Student	Effort Level	Contribution Level	Justification
Cambridge, Jake	2	2	Completed assigned tasks. However compared to other members not much contribution.
Chowdhury, Wasif	0	0	No communication or contribution whatsoever.
Feng, Haoxuan	5	5	
Graham, Alex	5	5	
McMillan, Aidan	5	5	
TOTAL	17	16	

3 Introduction

The following document details the processes followed for the inspections and testing procedures. The inspection process will occur during development while testing will occur after completion of the implementation.

The software will be implemented using PHP and SQLite and developed through Cloud 9, an online IDE. The testing process will be based on the requirements of the previous assignment. However, upon inspection and developing the testing plan and testing cases it has been discovered that some additional functionality would benefit the system. As the framework was already there we added this functionality in response to the testing (for ease of use). These assumed Requirements are:

- Customer Details must be able to be edited.
- Orders must be able to be edited.
- The Operator must be able to view the entire menu.
- The Operator must be able to see Orders made.

With the addition of these functionalities the system has become more user friendly and usable. The implementation of them was an easy tradeoff of a small amount of time for a great deal of benefit.

4 TEST PLAN

The process of testing the system can be a major undertaking; even with proper planning. The purpose of a test plan is to ensure maximum efficiency when testing as well as the maximum amount of defects identified. The testing process will proceed as follows.

4.1 TESTING TEAM

The testing team will consist of the developers as well as independent testers. This ensures both an expertise and familiarity regarding the system as well as an independent party able to identify errors overlooked by the developers. As the size of the system is small, the independent testers will consist of the team members who didn't work on the module currently being tested. In the secondary phase of testing actual independent testers will be used.

4.2 TESTING PROCESS

Throughout development of the software frequent unofficial smoke testing was conducted by the developers. This was to ensure basic function of the program without going into an in-depth testing or inspection process. This methodology occurs is almost every software project as programmers strive to debug their code. However, when officially starting the testing process using the testing team and the developed testing cases many encountered defects can be found.

Throughout the testing process black-box testing will be employed. The reasoning behind the exclusive use of black-box testing early on lies in the simplicity of the software undergoing testing. As we are taking a bottom-up approach defects will be simple and isolated to singular functions or components starting out, allowing for the producer to know that a problem exists and identify it themselves in the few lines of programming that it corresponds to.

Testing will occur over the course of an entire day; after which results will be compiled into a comprehensive list of defects. Upon receiving the initial results of the testing the defects will be identified within the program and fixed. After this the testing process will begin again in order to eliminate the possibility of defects in the now-patched program as well as identify any potential new ones. This process will continue until no significant defects are found or the project manager is satisfied with the resulting software.
Testing will follow the high-level process outlined below:

4.3 INTRODUCTION AND ORIENTATION

When testing first commences a short presentation detailing the purpose of the program and outlining the major goals of testing. This presentation should put everyone on the same page and allow for a more efficient testing process with a good consensus on the intended outcome. This will also enable the demonstration of how test cases work and the method that must be followed while completing them.

4.4 Initial Testing Phase

The initial phase of testing will be conducted by independent testers. In the case of such a small scale project as this the “independent testers” will consist of the team members who didn’t work on the module being tested. Testing will take place in a quiet room with at least two people present; the Producer and the “Independent” Tester. The actual testing will be conducted by the Tester, who will record the results of the testing according to the test case. The Producer will be able to provide support for any queries as well as observe a user without knowledge of the system and the way with which they interact with it and expect it to behave. This provides insight beyond merely testing for the Producer. This stage of testing will consist solely of black-box testing; the Tester will be given a process to follow and note the results. This ensures that modules are working correctly and allows for integration to work more smoothly, as well as detecting any defects that would be much harder to identify later on when dealing with a larger portion of the system. The unit/component and integration test cases will be followed at this stage.

4.5 Results

Upon completion of the initial testing phase, the results will be compiled and work will begin to fix any defects found. This process will involve the regular unofficial smoke-testing of the various modules in order to determine that any errors have indeed been fixed. Once the large majority of defects have been remedied testing can begin again.

4.6 Secondary Testing Phase

This phase of testing is optional depending on the requirements of the software and how well the initial testing phase went.
This phase of testing can be much more in-depth and may consist of the developers doing white-box testing alongside the independent testers doing black-box testing. Depending on the results of the last phase of testing, it may not be necessary to conduct white-box testing. If major errors are occurring and the defects can’t be identified, then the developers will conduct white-box testing. If testing is going well then it isn’t entirely necessary for such a small-scale project.

Black box testing will occur regarding the system as an integrated whole using the system test cases. These test cases will vary from being detailed and specific to generally vague. The purpose of vague instruction is to observe how the independent tester naturally interacts with the system. This allows for the quality of various aspects like usability to be determined, as well as the verification of functional requirements being fulfilled.

5 TEST CASES

As the program in development is of a small stature, testing will make use of three kinds of test-cases; unit/component, integration, and system. The reason behind the combination of unit and component is because almost every component within the program is made up of a single unit.

5.1 UNIT/COMBINATION TEST CASES

These test cases will cover singular functions and components in order to identify potentially major flaws that can affect the other modules of the program early. They will take the inspector through a short series of steps laid out to determine both the function of a component as well as the error-handling.

5.2 INTEGRATION TEST CASES

When conducting integrated testing the bottom-up method will be employed. The reasons behind choosing bottom-up rather than top-down are as follows:

- Test conditions are easier to create.
- Observation of test results is easier, therefore identification of flaws more likely.
- If major flaws occur toward the “bottom” of the program using bottom-up detects early

While top-down also offers its advantages, it doesn’t suit the nature of the program as well as bottom-up.

5.3 SYSTEM TEST CASES

The purpose of system test cases is to test a system as a complete and integrated whole. They not only test functional requirements but system requirement as well.
System testing will be done using the test cases below to test the following aspects of the complete system:

- GUI
- Usability
- Performance
- Reliability

Test Cases Over the Page

5.3.1 Test Case 1

PROJECT :		ordSystem		
MODULE:		UNIT/COMPONENT: menuItem database insertion		
REQUIREMENT:		R8a		
TEST CASE ID:		001		
TEST OBJECTIVE:		Verify ability to create a menuItem and insert it into the database.		
TEST DATE and TIME				
MODERATOR		Aidan McMillan, Haoxuan Feng		
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**
1	Enter the data into the form and click “Make Order”.	Custom ID: “ ” Name: “ “ Details: “ “ Price: “ “	Validation Returns Error for all fields except description: “[field] is required”	
2	Enter the data into the form and click “Make Order”.	Custom ID: “fr” Name: “ 45“ Details: “3f5g “ Price: “ gdf“	Validation Errors for Custom ID: “[field] needs to be a number. Validation Error for Price: “[fields] needs to be a number.	
3	Enter the data into the form and click “Make Order”.	Custom ID: “01” Name: “testName“ Details: “testDetail“ Price: “12.65“	Successful creation and insertion of menuItem into database.	
4	Enter the data into the form and click “Make Order”.	Custom ID: “01” Name: “testName“ Details: “testDetail“ Price: “.5“	Validation Error for Custom ID: ‘[field] must be unique’	
5	Enter the data into the form and click “Make Order”.	Custom ID: “02” Name: “testName“ Details: “testDetail“ Price: “12 “	Successful creation and insertion of menuItem into database.	

5.3.2 Test Case 2

PROJECT :	ordSystem
MODULE:	UNIT/COMPONENT: menuItem database update
REQUIREMENT:	R8b
TEST CASE ID:	002
TEST OBJECTIVE:	Verify ability to update a menuItem in the database
TEST DATE and TIME	
MODERATOR	Aidan McMillan, Haoxuan Feng

Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**
1	Enter the data into the form and click “Make Order”.	Name: “ “ Details: “ “ Price: “ “	Validation Returns Error for all fields except description: “[field] is required”	
2	Enter the data into the form and click “Make Order”.	Name: “ 45“ Details: “3f5g “ Price: “ gdf“	Validation Error for Price: “[fields] needs to be a number.	
3	Enter the data into the form and click “Make Order”.	Name: “testName“ Details: “testDetail“ Price: “12.65“	Successful creation and insertion of menuItem into database.	
5	Enter the data into the form and click “Make Order”.	Name: “testName“ Details: “testDetail“ Price: “12.65“	Successful creation and insertion of menuItem into database.	

5.3.3 Test Case 3

PROJECT :		ordSystem		
MODULE:		UNIT/COMPONENT: menuItem database deletion		
REQUIREMENT:		R8c		
TEST CASE ID:		003		
TEST OBJECTIVE:		Verify ability to delete a menuItem from the database.		
TEST DATE and TIME				
MODERATOR		Aidan McMillan, Haoxuan Feng		
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**
1	Click Delete Item		Item is deleted from database and not displaying.	

5.3.4 Test Case 4

PROJECT :		ordSystem		
MODULE:		UNIT/COMPONENT: customer database insertion		
REQUIREMENT:		R3		
TEST CASE ID:		004		
TEST OBJECTIVE:		Verify ability to add a customer to the database.		
TEST DATE and TIME				
MODERATOR		Aidan McMillan, Haoxuan Feng		
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**
1	Enter the data into the form and click “Make Order”.	Name: “ “ Address: “ “ Phone Number “ “ Card Number: “ “ Expiry Date: “ “ Cardholder Name: “ “ Security Code: “ “	Validation Returns Error for all fields except description: “[field] is required”	
2	Enter the data into the form and click “Make Order”.	Name: “12“ Address: “ 1 test PDE“ Phone Number “four“ Card Number: “ewr “ Expiry Date: “ewe“ Cardholder Name:“43“ Security Code: “er “	“Name must be a String” “Phone Number must be a number” “Card Number must be number. Card Number must have 16 digits” “Expiry Date must have 5 letters” “Must be a string” “Security Code must be a number. Must have 3 digits”	
3	Enter the data into the form and click “Make Order”.	Name: “Bob Smith“ Address: “ 1 test PDE“ Phone Number “04123456“ Card Number: “1234567812345678 “ Expiry Date: “12/10“ Cardholder Name: “Bob Smith“ Security Code: “123 “	Successfully creates customer and inserts into database.	

5.3.5 Test Case 5

PROJECT :		ordSystem		
MODULE:		UNIT/COMPONENT: order creation and database insertion		
REQUIREMENT:		R4		
TEST CASE ID:		005		
TEST OBJECTIVE:		Verify ability to open order and add to database.		
TEST DATE and TIME				
MODERATOR		Aidan McMillan, Haoxuan Feng		
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**
1	Enter the data into the form and click “Create”.	Additional Details: “ “ Delivery	Successful Order creation and insertion into database.	
2	Enter the data into the form and click “Create”.	Additional Details: “ fwf4wfwe“ Take Away	Successful Order creation and insertion into database.	

5.3.6 Test Case 6

PROJECT :		ordSystem		
MODULE:		UNIT/COMPONENT: adding item to order		
REQUIREMENT:		R4		
TEST CASE ID:		006		
TEST OBJECTIVE:		Verify ability to add an item and quantity to an order.		
TEST DATE and TIME				
MODERATOR		Aidan McMillan, Haoxuan Feng		
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**
1	Enter the data into the form and click “Add item”.	Item Number: “ “ Quantity: “1”	Validation Error: “Item Number cannot be null.”	
2	Enter the data into the form and click “Add item”.	Item Number: “ 65634“ Quantity: “1”	Validation Error: “Item doesn’t exist.”	
3	Enter the data into the form and click “Add item”.	Item Number: “ fsd“ Quantity: “1”	Validation Error: “Must be an integer”	
4	Enter the data into the form and click “Add item”.	Item Number: “1“ Quantity: “1”	Successful Insertion into database.	
5	Enter the data into the form and click “Add item”.	Item Number: “1“ Quantity: “1”	Validation Error: Item already in order.	
6	Enter the data into the form and click “Add item”.	Item Number: “2“ Quantity: “10”	Successful Insertion into database.	

5.3.7 Test Case 7

PROJECT :	ordSystem			
MODULE:	UNIT/COMPONENT: deleting item from order			
REQUIREMENT:	R11			
TEST CASE ID:	007			
TEST OBJECTIVE:	Verify ability to delete an item from the order.			
TEST DATE and TIME				
MODERATOR	Aidan McMillan, Haoxuan Feng			
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**
1	Click Delete Item		Item successfully deleted from the database.	

5.3.8 Test Case 8

PROJECT :	ordSystem			
MODULE:	UNIT/COMPONENT: deleting order			
REQUIREMENT:	R12			
TEST CASE ID:	008			
TEST OBJECTIVE:	Verify ability to cancel an order (delete from database).			
TEST DATE and TIME				
MODERATOR	Aidan McMillan, Haoxuan Feng			
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**
1	Click Cancel Order		Order successfully deleted from the database.	

5.3.9 Test Case 9

PROJECT :		ordSystem		
MODULE:		UNIT/COMPONENT: Calculate Daily Takings		
REQUIREMENT:		R7		
TEST CASE ID:		009		
TEST OBJECTIVE:		Verify function to calculate daily takings.		
TEST DATE and TIME				
MODERATOR		Aidan McMillan, Haoxuan Feng		
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**
1	Click calculate	No orders	Calculation outputs 0.	
2	Click calculate	Multiple orders with varying totals.	Calculation outputs correct result.	

5.3.10 Test Case 10

PROJECT :	ordSystem
MODULE:	UNIT/COMPONENT: Search Customer Number
REQUIREMENT:	R1
TEST CASE ID:	010
TEST OBJECTIVE:	Verify Search of customer's number.
TEST DATE and TIME	
MODERATOR	Aidan McMillan, Haoxuan Feng

Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**
1	Enter number. Click Search	NULL	Validation Error: "Field Required"	
2	Enter number. Click Search	"12"	Validation Error: "Must have 8 to 10 digits"	
3	Enter number. Click Search	"qwqweqw"	Validation Error: "Must be a number"	
4	Enter number. Click Search	"1h12h4"	Validation Error: "Must be a number"	
5	Enter number. Click Search	"04123456"	Returns existing.	
6	Enter number. Click Search	"00010001"	Returns Not Existing.	

5.3.11 Test Case 11

PROJECT :		ordSystem		
MODULE:		SYSTEM: GUI, Performance and Usability Testing		
REQUIREMENT:				
TEST CASE ID:		011		
TEST OBJECTIVE:		Verify appropriateness of GUI and usability of system as a whole. No exact steps as this process is for observing the user intuitively determining how to accomplish a given task.		
TEST DATE and TIME				
MODERATOR		Aidan McMillan, Haoxuan Feng		
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**
1	Navigate to Modify Menu	NULL		
2	Add a menu item.	Name: “testName“ Details: “testDetail“ Price: “12.65“	Successful creation and insertion of menuItem into database. Easy process.	

3	Delete a menu item		Easy and successful Deletion	
4	Navigate Back		Easily Completed.	
5	Search a number	“00010002”	Returns Not Existing	
6	Create a Customer	Name: “Bob Smith“ Address: “ 1 test PDE“ Phone Number “00010002“ Card Number: “1234567812345678 “ Expiry Date: “12/10“ Cardholder Name: “Bob Smith“ Security Code: “123 “	Successful Creation	
7	Make a new Order	Additional Details: “ fwf4wfwe“ Take Away	Successful Order creation and insertion into database and go to order open state.	
8	Add item to order	Number: “1” Quantity: “1”	Successful addition of item and return to order open state.	
9	Delete Item from Order		Successful Deletion and return to order open state.	
10	Complete Order		Successful Completion and return to ready state	
	Repeat steps: 5 and 7. Then cancel order.		Successful Cancelation and return to ready state.	

5.3.12 Test Case 12

PROJECT :		ordSystem		
MODULE:		SYSTEM: Reliability and Stress Testing		
REQUIREMENT:				
TEST CASE ID:				
TEST OBJECTIVE:		Verifying validation and inputting data in the wrong format to ensure the system can handle it.		
TEST DATE and TIME				
MODERATOR		Aidan McMillan, Haoxuan Feng		
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**
1	Navigate to Modify Menu	NULL		
2	Add a menu item.	Name: “testName“ Details: “testDetail“ Price: “12.65“	Successful creation and insertion of menuItem into database. Easy process.	

3	Delete a menu item		Easy and successful Deletion	
4	Navigate Back		Easily Completed.	
5	Search a number	“00010002”	Returns Not Existing	
6	Create a Customer	Name: “Bob Smith“ Address: “ 1 test PDE“ Phone Number “00010002“ Card Number: “1234567812345678 “ Expiry Date: “12/10“ Cardholder Name: “Bob Smith“ Security Code: “123 “	Successful Creation	
7	Make a new Order	Additional Details: “ fwf4wfwe“ Take Away	Successful Order creation and insertion into database and go to order open state.	
8	Add item to order	Number: “1” Quantity: “1”	Successful addition of item and return to order open state.	
9	Delete Item from Order		Successful Deletion and return to order open state.	
10	Complete Order		Successful Completion and return to ready state	
	Repeat steps: 5 and 7. Then cancel order.		Succesful Cancelation and return to ready state.	

6 INSPECTION

6.1 INSPECTION PLAN

In order to conduct a successful inspection phase for the system, the inspection plan will follow the well-known inspection process called the Fagan inspection. The Fagan Inspection is comprised of six steps, Planning, Overview, Preparation, Meeting, Rework and Follow up. Within the project there will be 4 roles assigned for the duration of the process, these being: Moderator/Recorder, Inspector, Reader and Producer. It is widely known from previous cases that using this Inspection process effectively will guarantee software and product quality. The Fagan Inspection six step plan for our system is as follows:

6.1.1 Planning

In the first stage of the plan we will detail the Inspective Objectives, Entry and Exit Criteria, Arranging of participants and finally the time and place of the Inspection.

- **Inspective Objectives:** This will detail the objectives aiming to be obtained from the Overall Inspection.

- **Entry and Exit Criteria:** This will detail the criteria in which the source code will be assessed. This usually contains a rule followed by a code value. The code value will be referenced in the Inspection Data collection process in respect to which rule is being defied in the segment of code.

- **Arranging of Participants:** This is where the participants will be notified and arranged to meet at a set time and date.

- **Place of Inspection:** Once the place the inspection will take place in is decided it will be detailed here.

6.1.2 Overview

The Overview is the second stage in the process in which the Group presentation will be planned and the Roles of the Participants will be designated accordingly.

- **Group Presentation:** The purpose of the presentation is to bring all the participants up to speed with the project they will be testing and the order of events and times processes will be conducted in the Inspection session.

- **Roles of Participants:** The roles of each of the participants will also be assigned to the respective participant at this stage. Arranging roles prior to the day will assist in choosing the most suitable participant for their corresponding roles.

6.1.3 Preparation

This stage is mainly for the Inspector to take some time to understand the software source code and the process in which they will be inspecting it, and then perform the inspection. The events that will conclude are as follows:

- Check the type of Software they will be inspecting
- Review the exit criteria
- Begin reviewing/recording all defects found
- Study and assign rankings to these found defects
- Record the time taken

6.1.4 Meeting

This is the most critical stage of the Inspection plan as this is where the results from the Inspection and discussed, classified and formally recorded. In this case the moderator will be responsible for recording the findings from these results into the Inspection Data Collection Questionnaire, and also making sure all the inspectors are well prepared.

6.1.5 Rework and Follow up

In the final stage the producer will take into account the defects found and begin rectifying the identified defects. They will also be assigned a moderator or overseer to report back to about the progress of the rework taking place. If applicable are inspection date will be discussed at this point, if it's not urgent it can be organized at a later date.

Inspections Over the Page

6.2 INSPECTION ONE

Planning Stage

Objectives: The overall objective of this inspection meeting is to ensure the correct function of the Menu Modification module of the Order System.

Entry Criteria: The entry criteria for this inspection will be as follows:

ID	Criteria Description
1	The code must conform to pro forma standards, ie Line numbers.
2	The code must pass error free automated checks such as: • Spelling • Logical • Complexity • Compilation
3	High level design document of the code must be provided.

Exit Criteria: The exit criteria for this inspection will be as follows:

ID	Criteria Description
ID-01	All declared identifiers should be utilized
V-01	There should be no variables that are used in an expression without previously being assigned
V-02	Variables should be initialized as close as possible to where they are used
V-03	Each variable should only be used for a single purpose
ID-02	Consistent case-sensitive naming conventions should be used for variables, constants procedure calls, etc.
E-01	The use of raw numbers in the body of program should be avoided
G-01	There should be no redundant logical tests
A-01	There should be no redundant assignments
L-01	Every Loop should make progress in all its guarded components
M-01	Function calls should be properly documented
IF-01	Every branch of an if-statement must be reachable

Arranging of Participants: There will be three people participating for the first inspection; Aidan McMillan, Alex Graham and Haoxuan Feng. They will be notified via face-to-face interaction of the meeting time at 3pm Monday 17th October.

Place of Inspection: The first inspection meeting will take place at Griffith University in 2.17 G23.

Overview

Group Presentation: The group presentation will be conducted at 3:10pm to ensure that all participants had time to arrive and settle in. The presentation will go for approximately 30 minutes. All participants will be made aware of both the Objectives of the inspection and the entry and exit criteria. A brief background of the project will be given to allow the participants to understand the intended purpose of the software as well as the reasoning behind the architecture of the software. Questions will be taken at the end of the presentation.

Roles of the Participants:

- Moderator/Inspector: Alex Graham
- Producer/Reader: Aidan McMillan
- Recorder: Haoxuan Feng

Preparation

The inspector in this case, Alex Graham, will sit down with Aidan McMillan and begin the inspection process. Initially as the reader Aidan will paraphrase though each of the sections of code. Alex will then read over and approve of the logic and statements being used in each of the sections. Alex is then responsible for taking note of any errors or improvements needing to be acted on, to be further analyzed later on.

Meeting

After the inspection is complete, the notes taken by Alex in the previous section will be discussed, clarified and formalized between himself and Haoxuan and ported into the official Data Collection Questionnaire as seen over the page.

Questionnaire Over the Page

Inspection 1 Data Collection Questionnaire

Date of Meeting:	13/10/2016
Date of Rework Complete:	N/A
Inspection Type:	Initial
Product / Function ID:	Menu Item Controller
Moderator:	Alex Graham
Recorder:	Haoxuan Feng
Inspector:	Alex Graham
Reader:	Aidan McMillan
Producer:	Aidan McMillan
Overview:	Yes
Preparation Time (Duration):	2.5 hours
Meeting Time (Duration):	4 hours

Defect Identification:

No.	Line No.	Type	Description	High Level Attribute
1				
2				

Notes:
No defects were found within the code.

Rework and Follow Up

The Data Collection Questionnaire will now be handed to the producer as a guide on which parts of the source code need acting upon. As the moderator, Alex will be assigned to assist Aidan in any further questions regarding the Data Collection and Oversee any updates done to the inspected source code. A new Inspection and follow up has also planned for the 18/10/2016.

Source Code

Source Code Over the Page

```php
<?php
class MenuitemController extends \BaseController {  //CONTROLLER FOR ALL MENUITEM FUNCTIONALITY

    //Function to make a view for menuitems ORDSYSTEM [MODIFY MENU] state
    public function index()
    {
        $menuitems = Menuitem::all();
        return View::make('modifyMenu', compact('menuitems'));
    }

    //Function to make a view for creating new menuitems
    public function create()
    {
        return View::make('createMenuitem');
    }

    //Function for storing menu items in the database
    public function store()
    {
        $input = Input::all();
        $v = Validator::make($input, Menuitem::$rules);
        if ($v->passes())
        {
        $menuitem = new Menuitem();
        $menuitem->name = $input['name'];
        $menuitem->details = $input['details'];
        $menuitem->price= $input['price'];
        $menuitem->save();
        return Redirect::route('menuitem.index');
        }
        else{
            return Redirect::back()->withErrors($v);
        }
    }

    //Function for making a view to update a menu item
    public function edit($id)
    {
        $menuitem = Menuitem::find($id);
        return View::make('editMenuitem')->withMenuitem($menuitem);
    }

    //Function to update a menu item
    public function update($id)
    {
        $input = Input::all();
        $v = Validator::make($input, Menuitem::$rules);
        if ($v->passes())
        {
        $menuitem = Menuitem::find($id);
        $menuitem->name = $input['name'];
        $menuitem->details = $input['details'];
        $menuitem->price= $input['price'];
        $menuitem->save();
        return Redirect::route('menuitem.index');
        }
        else{
            return Redirect::back()->withErrors($v);
        }
    }

    //Function to delete a menu item from the database
    public function destroy($id)
    {
        $menuitem = Menuitem::find($id);
        $menuitem->delete();
        return Redirect::route('menuitem.index');
    }
```

6.3 INSPECTION TWO

6.3.1 Planning Stage

Objectives: The overall objective of this inspection meeting is to ensure the correct function of the Order module of the Order System, and to approve the finalization of the previous inspection update.

Entry Criteria: The entry criteria for this inspection will be as follows:

ID	Criteria Description
1	The code must conform to pro forma standards, ie Line numbers.
2	The code must pass error free automated checks such as: Spelling Logical Complexity Compilation
3	High level design document of the code must be provided.

Exit Criteria: The exit criteria for this inspection will be as follows:

ID	Criteria Description
ID-01	All declared identifiers should be utilized
V-01	There should be no variables that are used in an expression without previously being assigned
V-02	Variables should be initialized as close as possible to where they are used
V-03	Each variable should only be used for a single purpose
ID-02	Consistent case-sensitive naming conventions should be used for variables, constants procedure calls, etc.
E-01	The use of raw numbers in the body of program should be avoided
G-01	There should be no redundant logical tests
A-01	There should be no redundant assignments
L-01	Every Loop should make progress in all its guarded components
M-01	Function calls should be properly documented
IF-01	Every branch of an if-statement must be reachable

Arranging of Participants: There will be three people participating for the first inspection; Aidan McMillan, Alex Graham and Haoxuan Feng. They will be notified via face-to-face interaction of the meeting time at 3pm Tuesday 18th October.

Place of Inspection: The second inspection meeting will take place at Griffith University in 2.17 G23.

6.3.2 Overview

Group Presentation: The group presentation will be conducted at 3:10pm to ensure that all participants had time to arrive and settle in. The presentation will go for approximately 10 minutes. All participants will be made aware of both the Objectives of the inspection and the entry and exit criteria. As this is the second meeting a very brief background of the project will be given to allow the participants to understand the intended purpose of the software as well as the reasoning behind the architecture of the software. The presentation will most likely focus on the architecture of the module being inspected. Questions will be taken at the end of the presentation.

Roles of the Participants:

- Moderator/Inspector: Alex Graham
- Producer/Reader: Aidan McMillan
- Recorder: Haoxuan Feng

6.3.3 Preparation

The inspector in this case, Alex Graham, will sit down with Aidan McMillan and begin the inspection process. Initially Alex will check the modifications to the previous inspection and sign off them as complete once they are at an approvable standard. Then Aidan will paraphrase though each of the sections of code relevant to the module being inspected. Alex will then read over and approve of the logic and statements being used in each of the sections. Alex is then responsible for taking note of any errors or improvements needing to be acted on, to be further analyzed later on.

6.3.4 Meeting

After the inspection is complete, the notes taken by Alex in the previous section will be discussed, clarified and formalized between himself and Haoxuan and ported into the official Data Collection Questionnaire as seen over the page.

Inspection 2 Data Collection Questionnaire

Date of Meeting:	20/10/2016
Date of Rework Complete:	19/10/2016
Inspection Type:	Re-Inspection
Product / Function ID:	Order Controller
Moderator:	Alex Graham
Recorder:	Haoxuan Feng
Inspector:	Alex Graham
Reader:	Aidan McMillan
Producer:	Aidan McMillan
Overview:	Yes
Preparation Time (Duration):	1.5 hours
Meeting Time (Duration):	5 hours

Defect Identification:

No	Line No.	Type	Description	High Level Attribute
1	M-01 Inspection 2 Line 47	Function not documented	No comment describing function.	Comment Description
2	M-01 Inspection 2 Line 53	Function not documented	No comment describing function.	Comment Description
3	ID-01 Inspection 2 Line 55	Unnecessary comment	Code commented out	Delete Comment

Notes:
The previous inspection has been signed off on and was changed in accordance to the previous inspection. Once again aside from the technical aspects of code layout/style the code has been developed greatly.

6.3.5 Rework and Follow Up

- The Data Collection Questionnaire will now be handed to the producer as a guide on which parts of the source code need acting upon. As the moderator, Alex will be assigned to assist Aidan in any further questions regarding the Data Collection and Oversee any updates done to the inspected source code. As this is the final component to be assessed no further inspection meetings will be organized. However, Alex will still sign off on the updated code at a later date.

6.3.6 Source Code

```
47      public function edit($id)
48      {
49          $order = Order::find($id);
50          return View::make('editOrder')->withOrder($order);
51      }
52
53      public function update($id)
54      {
55          //$input = Input::all();
56          $order = Order::find($id);
57          $order->complete = true;
58          $order->save();
59          return Redirect::route('order.index');
60      }
61
62      //Function to delete an order from the database
63      public function destroy($id)
64      {
65          $order = Order::find($id);
66          $order->menuitems()->detach();  //Also deletes all items in order
67          $order->delete();
68          return Redirect::route('order.index');
69      }
70
71      //Function to add an item to an order
72      public function addItem($id)
73      {
74          $input = Input::all();
75          $v = Validator::make($input, Order::$itemRules);
76          $order = Order::find($id);
77          if ($v->passes())
78          {
79          $menuitem_id = $input['menuitem_id'];
80          $quantity = $input['quantity'];
81          $menuitem = Menuitem::find($menuitem_id);   //Finds correct menuitem
82          $price = $menuitem->price;
83          $totalPrice = $price * $quantity;       //Generates total price of item depending on quantity
84          $order->menuitems()->sync([$menuitem_id, ['quantity' => $quantity, 'totalPrice' => $totalPrice]]);
85          $order->ordTotal += $totalPrice;
86          $order->save();
87          return Redirect::route('order.show', array($id));
88          }
89          else{
90              return View::make('ordOpen')->withOrder($order)->withErrors($v);
91          }
92      }
93
94      //Function to remove an item from the database
95      public function removeItem($id)
96      {
97          $input = Input::all();
98          $order = Order::find($id);
99          $menuitem_id = $input['menuitem_id'];
100         $menuitem = Menuitem::find($menuitem_id);
101         $totalPrice = $input['totalPrice'];
102         $order->menuitems()->detach($menuitem->id);     //Removes item corresponding to order
103         $order->ordTotal = $order->ordTotal - $totalPrice;
104         $order->save();
105         return Redirect::route('order.show', array($id));
106     }
107
```

7 TESTING RESULTS

7.1.1 Test Result 1

PROJECT :		ordSystem			
MODULE:		UNIT/COMPONENT: menuItem database insertion			
REQUIREMENT:		R8a			
TEST CASE ID:		001			
TEST OBJECTIVE:		Verify ability to create a menuItem and insert it into the database.			
TEST DATE and TIME		19/10/16			
MODERATOR		Haoxuan Feng			
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	**PASS**
1	Enter the data into the form and click "Make Order".	Name: " " Details: " " Price: " "	Validation Returns Error for all fields except details: "[field] is required"	Validation Returns Error for all fields except details: "[field] is required"	YES
2	Enter the data into the form and click "Make Order".	Name: " 45" Details: "3f5g " Price: " gdf"	Validation Error for Price: "[fields] needs to be a number.	Validation Error for Price: "[fields] needs to be a number.	YES
3	Enter the data into the form and click "Make Order".	Name: "testName" Details: "testDetail" Price: "12.65"	Successful creation and insertion of menuItem into database.	Successful creation and insertion of menuItem into database.	YES
4	Enter the data into the form and click "Make Order".	Name: "testName" Details: "testDetail" Price: "12 "	Successful creation and insertion of menuItem into database.	Successful creation and insertion of menuItem into database.	YES

7.1.2 Test Case 2

PROJECT :	ordSystem
MODULE:	UNIT/COMPONENT: menuItem database update
REQUIREMENT:	R8b
TEST CASE ID:	002
TEST OBJECTIVE:	Verify ability to update a menuItem in the database. Verify ability to add multiple same/similar items.
TEST DATE and TIME	19/10/16
MODERATOR	Haoxuan Feng

Step No	Steps	Data	Expected Results	Actual Results	PASS
1	Enter the data into the form and click “Make Order”.	Name: “ “ Details: “ “ Price: “ “	Validation Returns Error for all fields except description: “[field] is required”	Validation Returns Error for all fields except description: “[field] is required”	YES
2	Enter the data into the form and click “Make Order”.	Name: “ 45“ Details: “3f5g “ Price: “ gdf“	Validation Error for Price: “[fields] needs to be a number.	Validation Error for Price: “[fields] needs to be a number.	YES
3	Enter the data into the form and click “Make Order”.	Name: “testName“ Details: “testDetail“ Price: “12.65“	Successful creation and insertion of menuItem into database.	Successful creation and insertion of menuItem into database.	YES
5	Enter the data into the form and click “Make Order”.	Name: “testName“ Details: “testDetail“ Price: “12.65“	Successful creation and insertion of menuItem into database.	Successful creation and insertion of menuItem into database.	YES

7.1.3 Test Result 3

PROJECT :		ordSystem			
MODULE:		UNIT/COMPONENT: menuItem database deletion			
REQUIREMENT:		R8c			
TEST CASE ID:		003			
TEST OBJECTIVE:		Verify ability to delete a menuItem from the database.			
TEST DATE and TIME		19/10/16			
MODERATOR		Aidan McMillan			
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	**PASS**
1	Click Delete Item		Item is deleted from database and not displaying.	Item is deleted from database and not displaying.	YES

7.1.4 Test Result 4

PROJECT :	ordSystem				
MODULE:	UNIT/COMPONENT: customer database insertion				
REQUIREMENT:	R3				
TEST CASE ID:	004				
TEST OBJECTIVE:	Verify ability to add a customer to the database.				
TEST DATE and TIME	19/10/16				
MODERATOR	Haoxuan Feng				
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	**PASS**
1	Enter the data into the form and click “Make Order”.	Name: “ “ Address: “ “ Phone Number “ “ Card Number: “ “ Expiry Date: “ “ Cardholder Name: “ “ Security Code: “ “	Validation Returns Error for all fields except description: “[field] is required”		YES
2	Enter the data into the form and click “Make Order”.	Name: “12“ Address: “ 1 test PDE“ Phone Number “four“ Card Number: “ewr “ Expiry Date: “ewe“ Cardholder Name:“43“ Security Code: “er “	“Name can only have characters” “Phone Number must be a number” “Card Number must be number. Card Number must have 16 digits” “Expiry Date must have 5 letters” “Must be a string” “Security Code must be a number. Must have 3 digits”	“Name can only have characters” “Phone Number must be numerical” “Card Number must be number. Card Number must have 16 digits” “Expiry Date must have 5 letters” “Can only have characters “Security Code must be a number. Must have 3 digits”	YES
3	Enter the data into the form and click “Make Order”.	Name: “Bob Smith“ Address: “ 1 test PDE“ Phone Number “04123456“ Card Number: “1234567812345678 “ Expiry Date: “12/10“ Cardholder Name: “Bob Smith“ Security Code: “123 “	Successfully creates customer and inserts into database.	Successfully creates customer and inserts into database.	YES

7.1.5 Test Result 5

PROJECT :		ordSystem			
MODULE:		UNIT/COMPONENT: order creation and database insertion			
REQUIREMENT:		R4			
TEST CASE ID:		005			
TEST OBJECTIVE:		Verify ability to open order and add to database.			
TEST DATE and TIME		19/10/16			
MODERATOR		Alex Graham			
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	**PASS**
1	Enter the data into the form and click “Create”.	Additional Details: “ “ Delivery	Successful Order creation and insertion into database.	Successful Order creation and insertion into database.	YES
2	Enter the data into the form and click “Create”.	Additional Details: “fwf4wfwe“ Take Away	Successful Order creation and insertion into database.	Successful Order creation and insertion into database.	YES

7.1.6 Test Result 6

PROJECT :	ordSystem
MODULE:	UNIT/COMPONENT: adding item to order
REQUIREMENT:	R4
TEST CASE ID:	006
TEST OBJECTIVE:	Verify ability to add an item and quantity to an order.
TEST DATE and TIME	19/10/16
MODERATOR	Alex Graham

Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	**PASS**
1	Enter the data into the form and click “Add item”.	Item Number: “ “ Quantity: “1”	Validation Error: “Item Number cannot be null.”	Validation Error: “Item Number cannot be null.”	YES
2	Enter the data into the form and click “Add item”.	Item Number: “ 65634“ Quantity: “1”	Validation Error: “Item doesn’t exist.”	Validation Error: “Item isn’t valid”	FAIL
3	Enter the data into the form and click “Add item”.	Item Number: “ fsd“ Quantity: “1”	Validation Error: “Must be an integer”	Validation Error: “Must be an integer”	YES
4	Enter the data into the form and click “Add item”.	Item Number: “1“ Quantity: “1”	Successful Insertion into database.	Successful Insertion into database.	YES
5	Enter the data into the form and click “Add item”.	Item Number: “1“ Quantity: “1”	Validation Error: Item already in order.	Order updates – removes old item adds new.	FAIL
6	Enter the data into the form and click “Add item”.	Item Number: “2“ Quantity: “10”	Successful Insertion into database.	Successful Insertion into database.	YES

7.1.7 Test Result 7

PROJECT :		ordSystem			
MODULE:		UNIT/COMPONENT: deleting item from order			
REQUIREMENT:		R11			
TEST CASE ID:		007			
TEST OBJECTIVE:		Verify ability to delete an item from the order.			
TEST DATE and TIME		19/10/16			
MODERATOR		Haoxuan Feng			
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	
1	Click Delete Item		Item successfully deleted from the database.	Item successfully deleted from the database.	YES

7.1.8 Test Result 8

PROJECT :		ordSystem			
MODULE:		UNIT/COMPONENT: deleting order			
REQUIREMENT:		R12			
TEST CASE ID:		008			
TEST OBJECTIVE:		Verify ability to cancel an order (delete from database).			
TEST DATE and TIME		19/10/16			
MODERATOR		Alex Graham			
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	**PASS**
1	Click Cancel Order		Order successfully deleted from the database.	Order successfully deleted from the database.	YES

7.1.9 Test Result 9

PROJECT :		ordSystem			
MODULE:		UNIT/COMPONENT: Calculate Daily Takings			
REQUIREMENT:		R7			
TEST CASE ID:		009			
TEST OBJECTIVE:		Verify function to calculate daily takings.			
TEST DATE and TIME		19/10/16			
MODERATOR		Aidan McMillan			
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	**PASS**
1	Click calculate	No orders	Calculation outputs 0.	No Result.	FAIL
2	Click calculate	Multiple orders with varying totals.	Calculation outputs correct result.	No Result.	FAIL

7.1.10 Test Result 10

PROJECT :	ordSystem
MODULE:	UNIT/COMPONENT: Search Customer Number
REQUIREMENT:	R1
TEST CASE ID:	010
TEST OBJECTIVE:	Verify Search of customer's number.
TEST DATE and TIME	19/10/16
MODERATOR	Alex Graham

Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	**PASS**
1	Enter number. Click Search	NULL	Validation Error: "Field Required"	Validation Error: "Field Required"	YES
2	Enter number. Click Search	"12"	Validation Error: "Must have 8 to 10 digits"	Validation Error: "Must have 8 to 10 digits"	YES
3	Enter number. Click Search	"qwqweqw"	Validation Error: "Must be a number"	Validation Error: "Must be a number"	YES
4	Enter number. Click Search	"1h12h4"	Validation Error: "Must be a number"	Validation Error: "Must be a number"	YES
5	Enter number. Click Search	"04123456"	Returns existing.	Returns existing.	YES
6	Enter number. Click Search	"00010001"	Returns Not Existing.	Returns Not Existing.	YES

7.1.11 Test Result 11

PROJECT :		ordSystem			
MODULE:		SYSTEM/Integrated: GUI, Performance and Usability Testing			
REQUIREMENT:					
TEST CASE ID:		011			
TEST OBJECTIVE:		Verify appropriateness of GUI and usability of system as a whole. No exact steps as this process is for observing the user intuitively determining how to accomplish a given task.			
TEST DATE and TIME		19/10/16			
MODERATOR		Aidan McMillan			
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	**PASS**
1	Navigate to Modify Menu	NULL			YES
2	Add a menu item.	Name: “testName“ Details: “testDetail“ Price: “12.65“	Successful creation and insertion of menuItem into database. Easy process.	Successful creation and insertion of menuItem into database. Easy process.	YES
3	Delete a menu item		Easy and successful Deletion	Easy and successful Deletion	YES
4	Navigate Back		Easily Completed.	Easily Completed.	YES

5	Search a number	“00010002”	Returns Not Existing	Returns Not Existing	YES
6	Create a Customer	Name: “Bob Smith“ Address: “ 1 test PDE“ Phone Number “00010002“ Card Number: “1234567812345678 “ Expiry Date: “12/10“ Cardholder Name: “Bob Smith“ Security Code: “123 “	Successful Creation	Successful Creation – Very easy	YES
7	Make a new Order	Additional Details: “ fwf4wfwe“ Take Away	Successful Order creation and insertion into database and go to order open state.	Successful Order creation and insertion into database and go to order open state.	YES
8	Add item to order	Number: “1” Quantity: “1”	Successful addition of item and return to order open state.	Successful addition of item and return to order open state.	YES
9	Delete Item from Order		Successful Deletion and return to order open state.	Successful Deletion and return to order open state.	YES
10	Complete Order		Successful Completion and return to ready state	Successful Completion and return to ready state	YES
	Repeat steps: 5 and 7. Then cancel order.		Successful Cancelation and return to ready state.	Successful Cancelation and return to ready state.	YES

7.2 DEFECT ANALYSIS

Abnormal Result	Defect	Nature	Method to Fix
M-01 Inspection 2 Line 47	Function not documented		Comment Description
M-01 Inspection 2 Line 53	Function not documented		Comment Description
ID-01 Inspection 2 Line 55	Unnecessary comment	Code commented out	Delete Comment
Test Case 009 isn't calculating takings.	Not implemented or integrated properly.	Not integrated properly.	
Test Case 006 Step 2 validation error doesn't indicate why error.	Error doesn't indicate problem	Small oversight	Add custom error message.
Test Case 006 Step 5 ordTotal not calculating properly	Defect in the way ordTotal is calculated. Doesn't integrate with database.	Improper integration/implementation.	Reassess method to calculate ordTotal – maybe from many-to-many table.
Test Case 009 Daily Takings not calculated properly.	Defect is in the way ordTotal is calculated.	Improper integration/implementation.	Reassess method to calculate ordTotal – maybe from many-to-many table.
Test Case 009 Daily Takings not integrated properly.	Displaying of Daily Takings not working.		Integrate Daily Takings function into system.

7.3 ASSUMPTIONS, CHANGES AND PROBLEMS

During the implementation of the system it was discovered that functions that could benefit the system greatly with minimal overhead (cost/time) could be implemented. These function facilitate basic operations on the various objects (order, customer, menu item) and allow the system to work at a much higher level. These include:

- Ability to edit order after creation.
- Display of orders not finished in order to see details and items.
- Ability to edit an order after creation.
- Ability to "complete" a finished order
- Ability to edit customer details (if they change payment/address).

These functions were implemented under the assumption that the user of the system would require them. They make up the major changes that occurred between the planning stage and the implementation stage.

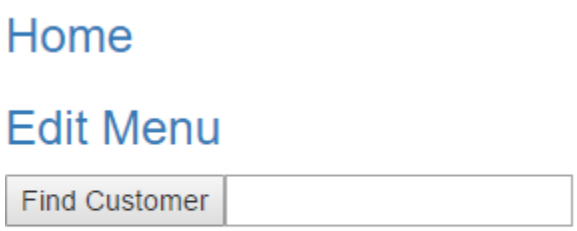
Problems

Only one major problem was encountered in the creation of the system. That was implementing a takings function. The way it was originally working was discovered to have major errors during the testing phase of the project. This was documented and eventually a fix was found. Other than this there were no major problems with the system.

8 APPENDICES

8.1 APPENDIX A – USER MANUAL

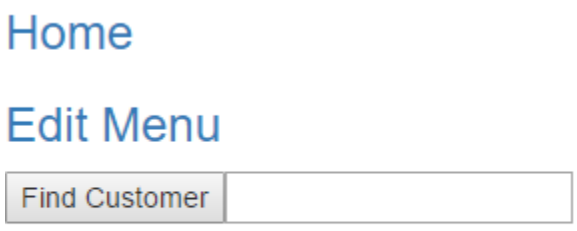
Landing Page



Instructions

- "Home" will always take you back to the landing page.
- "Edit Menu" allows you to modify the menu ([ModifyMenu] state).
- Search bar allows you to enter a phone number.
- Orders will be displayed on the landing page.
- You can View and Edit order by clicking "View Order". See ORDER OPEN for details.

Search Bar



Instructions

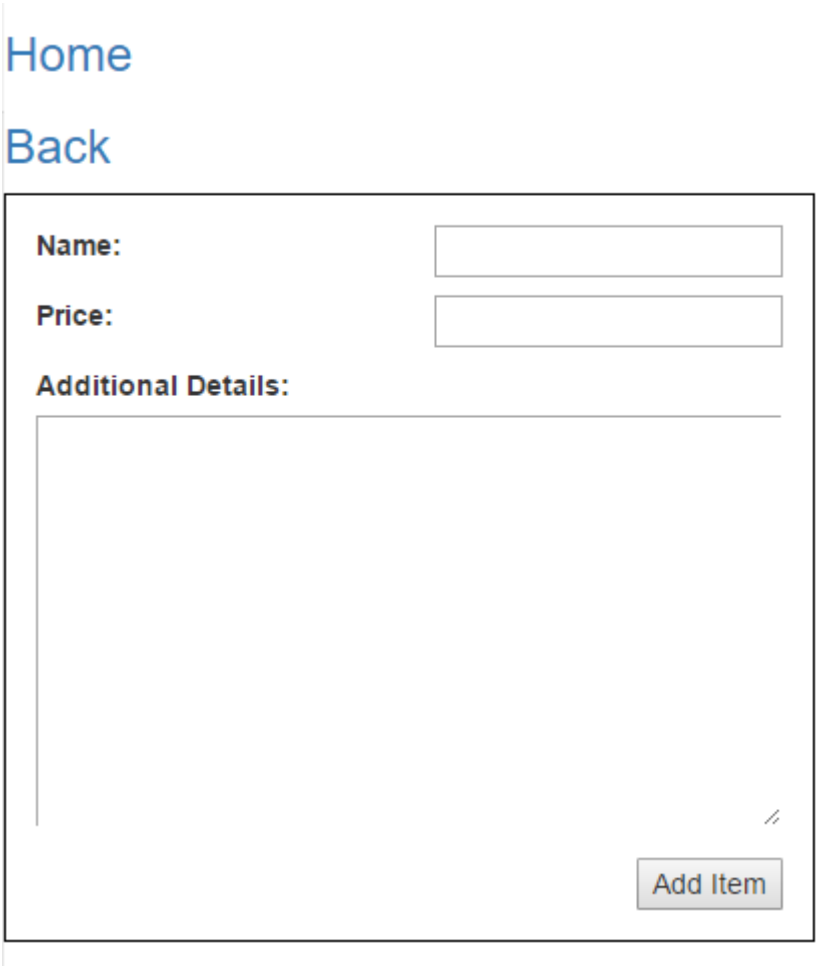
- Query MUST be between 8-10 DIGITS.
- If the number exists, it will take you to a Create Order page immediately.
- If the number doesn't exist, it will take you to the customer creation page.

Modify Menu

Instructions

- Home link will take you back to the landing page.
- “Add Item” allows you to add a menuItem.
- menuItems stored in the database will be displayed here.
- You are able to edit menuItems.

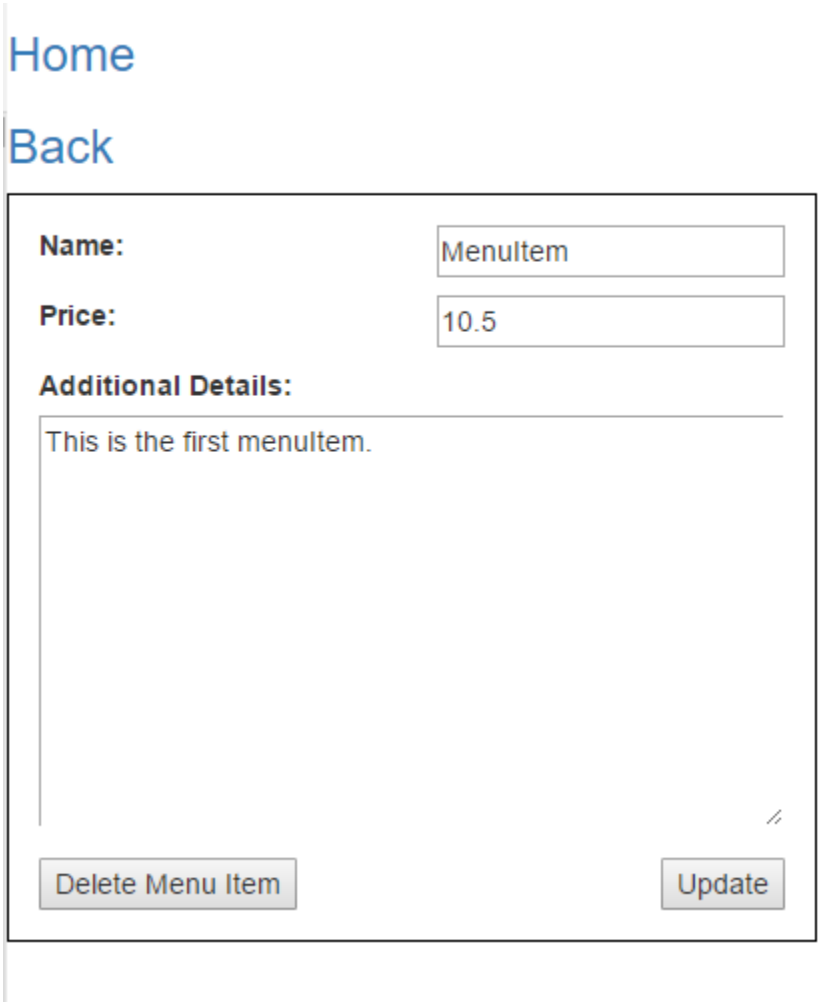
Add menuItem



Instructions

- Home link will take you back to landing page.
- Back link will take you back to Modify Menu.
- You are able to add a new item here, some rules apply:
    - Name and Price REQUIRED
    - Price MUST BE A NUMERIC VALUE
- Press “Add Item” to create new menuItem in database.

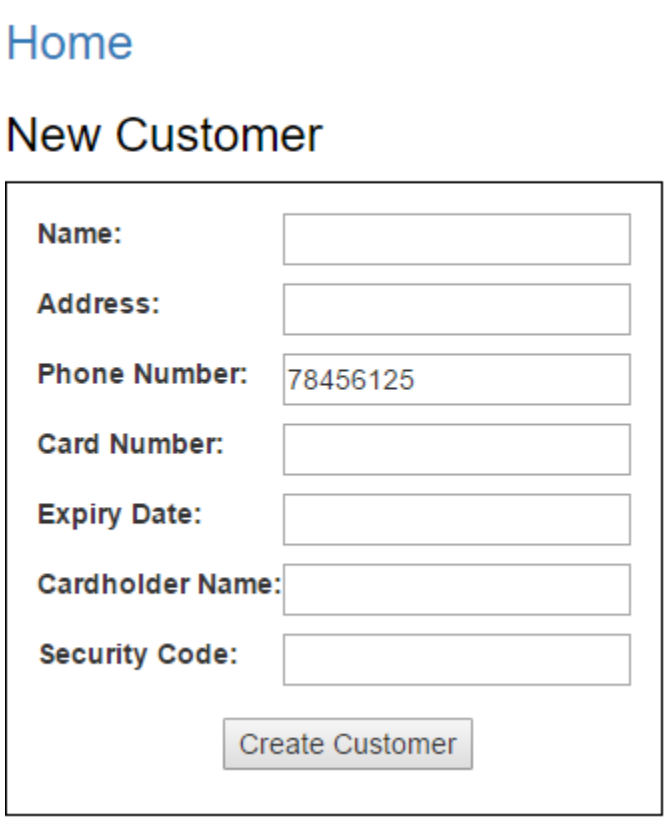
Edit Menu Item



Instructions

- Home link will take you back to landing page.
- Back link will take you back to Modify Menu.
- You are able to update a menuItem here, some rules apply:
    - Name and Price REQUIRED
    - Price MUST BE A NUMERIC VALUE
- Press “Add Item” to update menuItem in database.
- Press “Delete Menu Item” to delete the menuItem from the database.

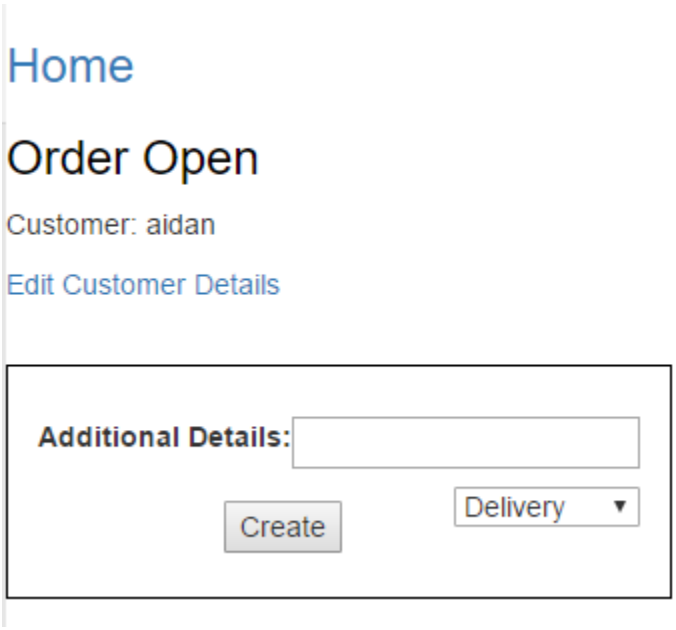
New Customer



Instructions

- Home link will take you back to landing page.
- You are able to create a customer here, some rules apply:
    - Name MUST be in CHARACTERS
    - Phone Number MUST be BETWEEN 8-10 DIGITS
    - Card Number MUST be 16 DIGITS
    - Expiry Date MUST be <= 5 length
    - Cardholder Name MUST be in CHARACTERS
    - Security Code MUST be 3 DIGITS
- Click “Create Customer” to proceed

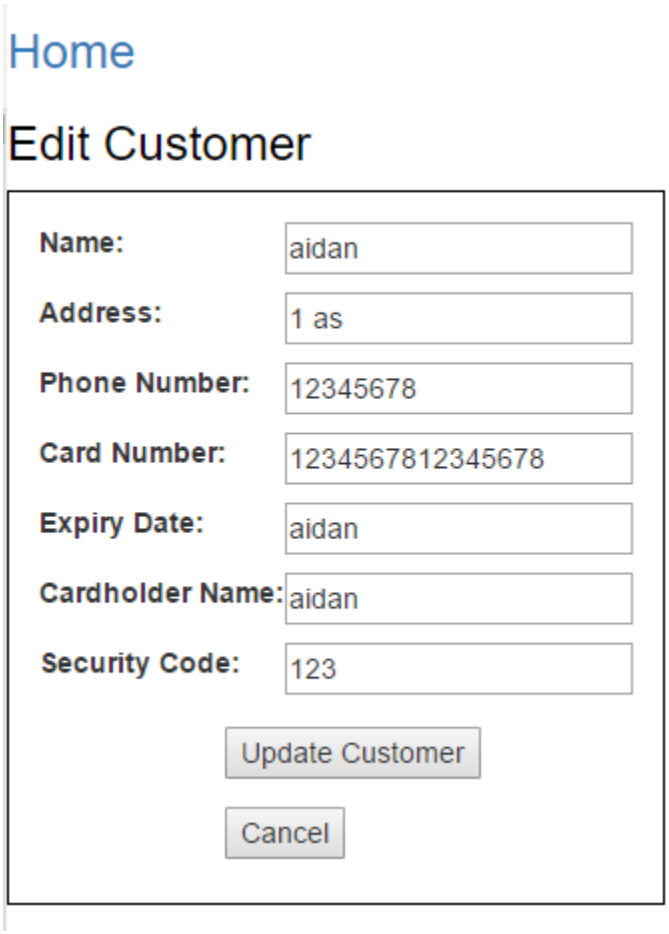
Create Order



Instructions

- Home link will take you back to landing page.
- The name of the customer is displayed.
- You can Edit the customers' details from here.
- You can create an order by clicking "Create". There are no rules.

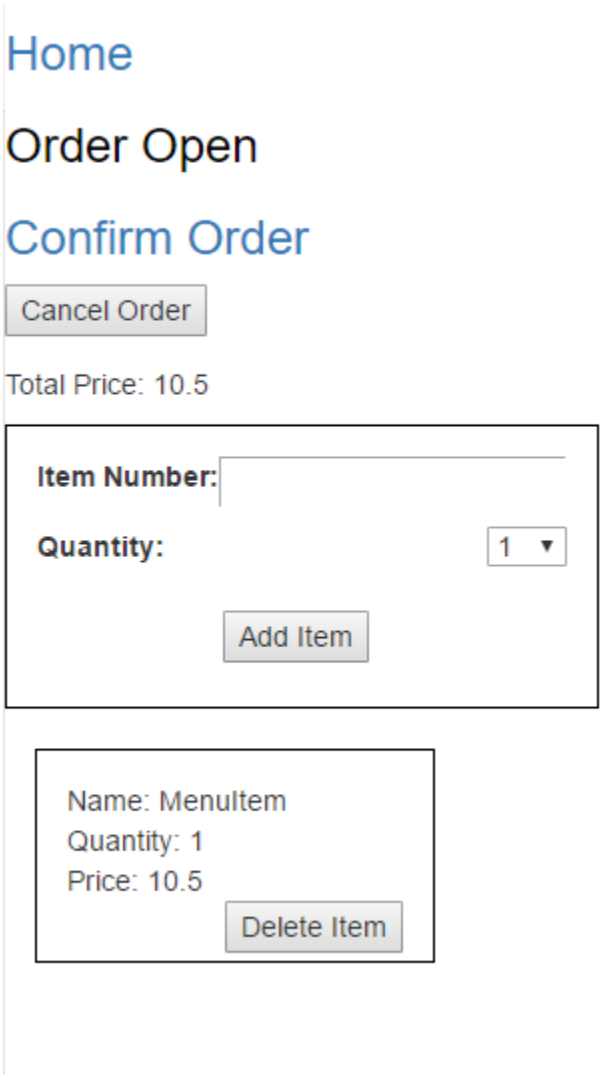
Edit Customer Details



Instructions

- Home link will take you back to landing page.
- All the current details will be filled in for you.
- See New Customer for validation rules.
- “Update Customer” will take you back to Create Order and update customer details.
- “Cancel” will take you back Create Order without changing details.

Order Open



Instructions

- Home link will take you back to landing page.
- "Confirm Order" will confirm order and will take you back to landing page.
- "Cancel Order" will delete order and take you back to landing page.
- You can add an item to the order by entering its ID Number (can be found in Modify Menu Page)
    - MUST be valid id.
    - Quantity from 1-10
- Click "Add Item" to add item to the order.
- If there are items in the order, click "Delete Item" to remove from order and database.
- If you cancel order it will also remove everything related to order.

8.2 APPENDIX 2 – SOURCE CODE LOCATED EXTERNALLY IN ZIP FILE

There are four folders locating the code used to implement the software. This includes all programming done by the team – all other code not included in the zip file is used for the management of the IDE and templating. The software can be observed in action using a cloud9 account or displayed during a face-to-face meeting. “Routes” is a file dictating the navigation structure of the program (15 lines)

The folders included are:

Controllers
Contain the main implementation of the software. Each file corresponds to functions used by the object.

- BaseController – 18 lines
- CustomerController – 121 lines
- HomeController – 9 lines
- MenuitemController – 72 lines
- OrderController – 121 lines

Database
Contains the initialization (and structure) of the various database tables.

- customers_table – 28 lines
- menuItem_table – 24 lines
- orders_table – 27 lines
- menuitem_order_table (many to many relationship) – 24 lines

Models
Contains the models for each object. These define the validation rules as well as the relationships between objects.

- Customer.php - 26 lines
- Menuitem.php – 13 lines
- Order.php – 22 lines

Views
Contains all the various views for the different actions taken. Generally, the purpose of a view can be discerned from its name.

- layouts/master (is parent template inherited by all children) – 39 lines
- createCustomer – 45 lines
- createMenuitem – 29 lines
- createOrder – 35 lines
- editCustomer – 57 lines
- editMenuitem – 35 lines
- editOrder – 48 lines
- index (landing page) – 43 lines
- modifyMenu (menu modification subsection) – 25 lines
- ordOpen (view when order is open) – 54 lines
- viewTakings – 13 lines