

Development Phase-1 requirements report

For

Pizza Crush

Version 1.0

Prepared by: Group 8 (Sourab Reddy Pailla, Gowtham Kesa, Nagendra Beesabathuni, Rishi Reddy Kolanu)

University of North Texas

10/21/2019

Table of contents

1. Pizza Crush Requirements.....	3
1.1 Add, delete and update products.....	3
1.2 Mark as Delivered.....	3
1.3 Order pizza from menu.....	3
1.4 Order customized pizza.....	3
2. UML Design.....	4
2.1 Class Diagram.....	4
2.2 Sequence Diagram.....	5
2.3 Use Case Diagram.....	7
2.4 Use Case Diagram error case.....	8
3. Test Cases.....	9
4. Contributions.....	12
5. User Manual.....	13
6. Installation instructions.....	20
7. Peer review feedback.....	21

1. Pizza Crush Requirements

1.1: Add, delete and update products

Admin can add or delete new products to the database. He can also update the product price. Products may include items like

- i) Pizza
- ii) Toppings
- iii) Sauce
- iv) Bread

1.2: Mark as Delivered

Admin can mark the orders as delivered.

1.3: Order pizza from menu

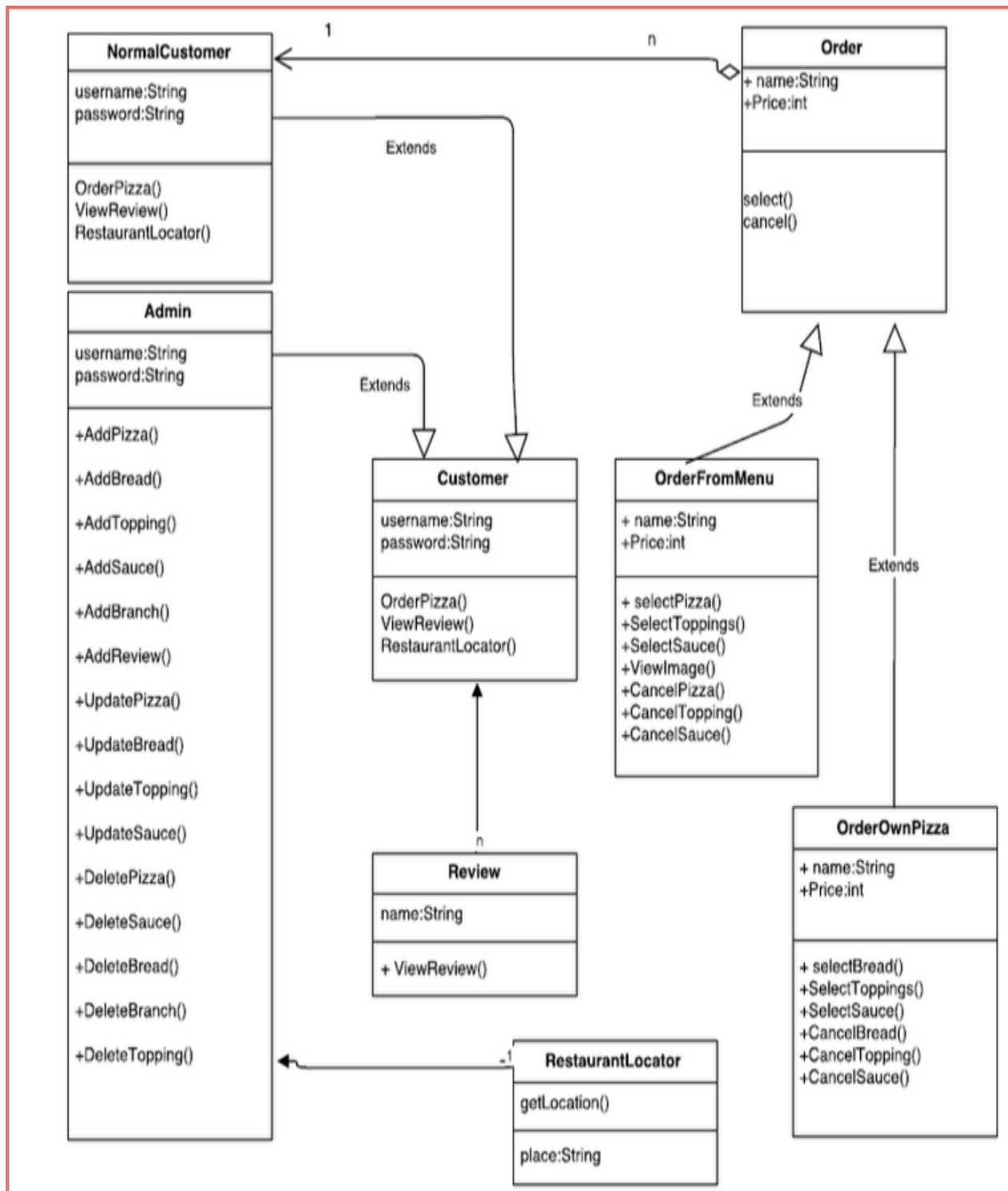
- i) Order.selectPizza: Add selected pizzas to the cart which are selected from the list of pizzas provided.
- ii) Order.selectTopping: Add selected toppings to the cart which are selected from the list of toppings provided.
- iii) Order.selectSauce: Add selected sauces to the cart which are selected from the list of sauces provided.
- iv) Order.CancelPizza: Remove the selected pizza from the cart.
- v) Order.CancelTopping: Remove the selected topping from the cart.
- vi) Order.CacelSauce: Remove the selected sauce from the cart.

1.4: Order customized pizza

- i) Order.selectBread: Add selected Bread to the cart which is selected from the list of pizzas provided.
- ii) Order.selectTopping: Add selected toppings to the cart which are selected from the list of toppings provided.
- iii) Order.selectSauce: Add selected sauces to the cart which are selected from the list of sauces provided.
- iv) Order.CancelBread: Remove the selected pizza from the cart.
- v) Order.CancelTopping: Remove the selected topping from the cart.
- vi) Order.CancelSauce: Remove the selected sauce from the cart.
- vii) Order.AnotherPizza: Decides whether to order another customized pizza or not.

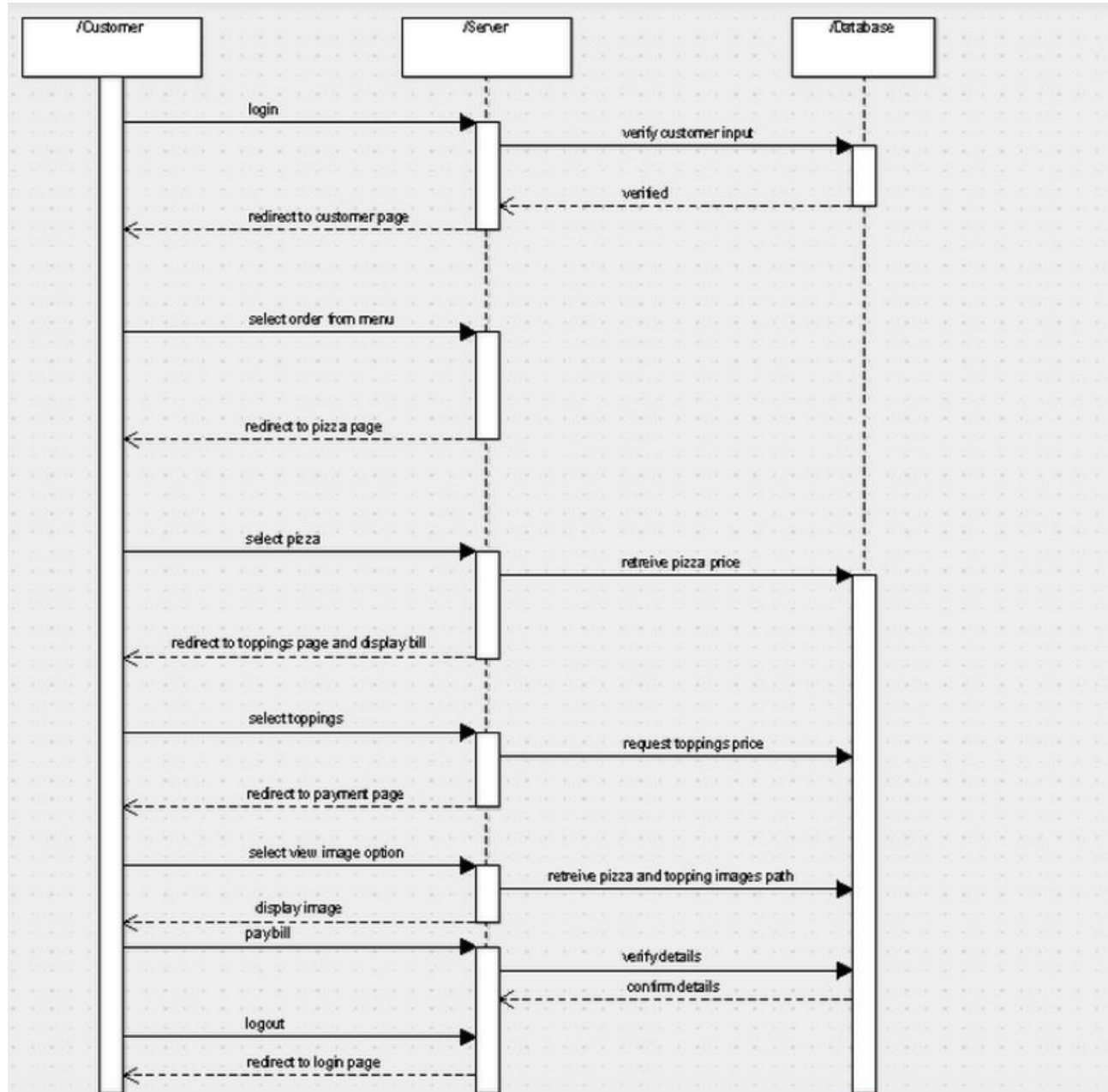
2. UML Design

2.1: Class Diagram

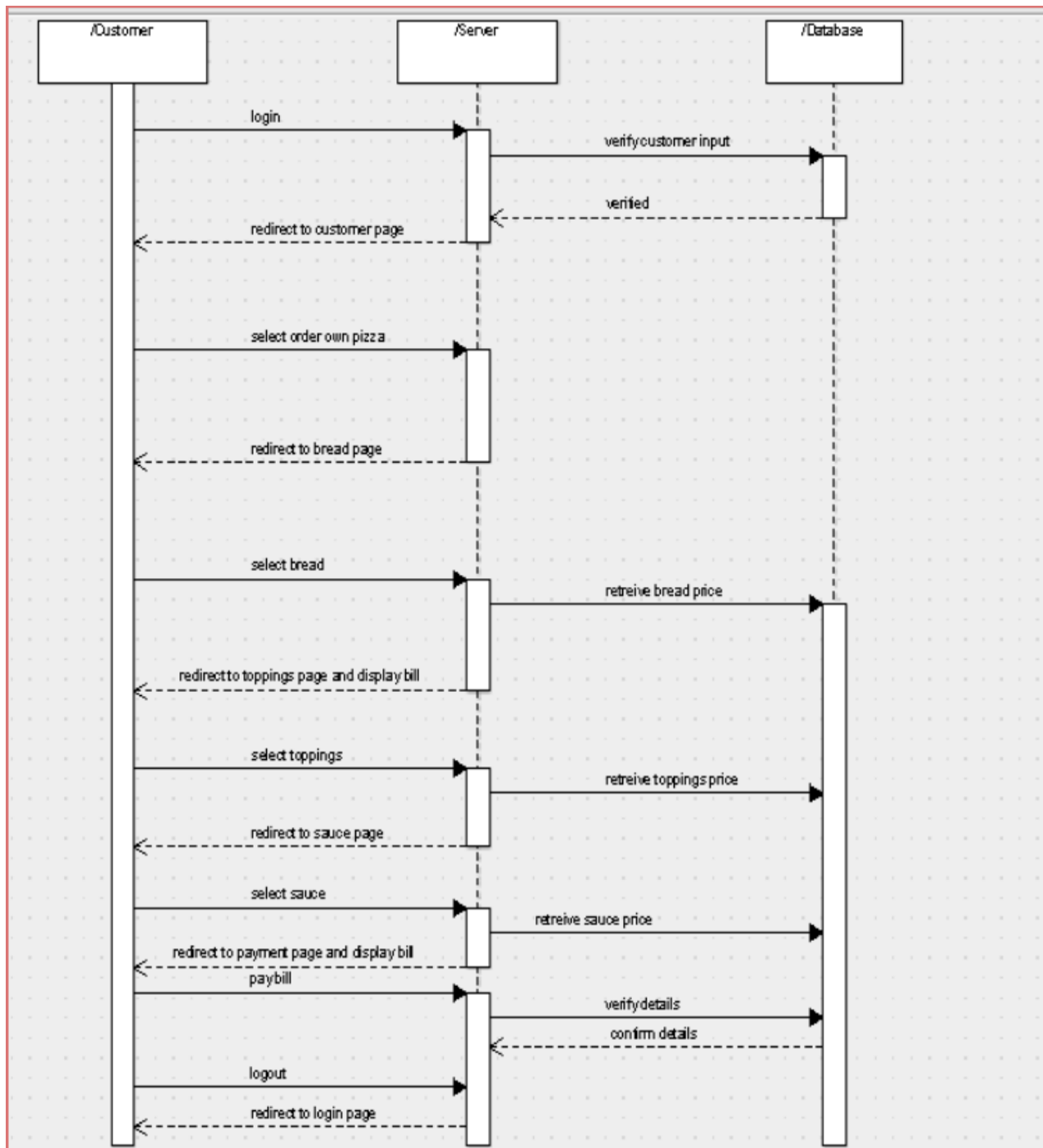


2.2 Sequence Diagram

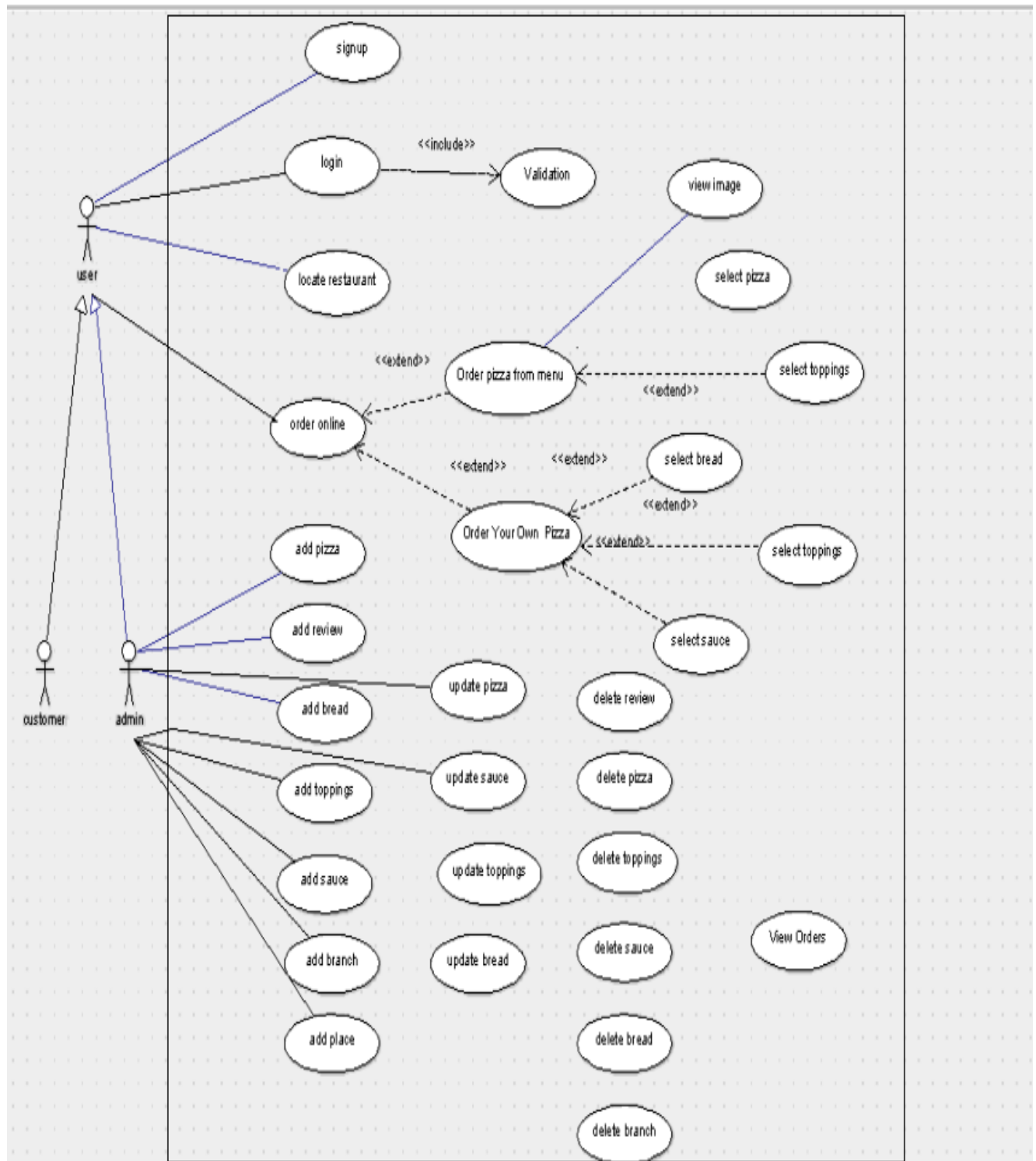
i) Order pizza from menu



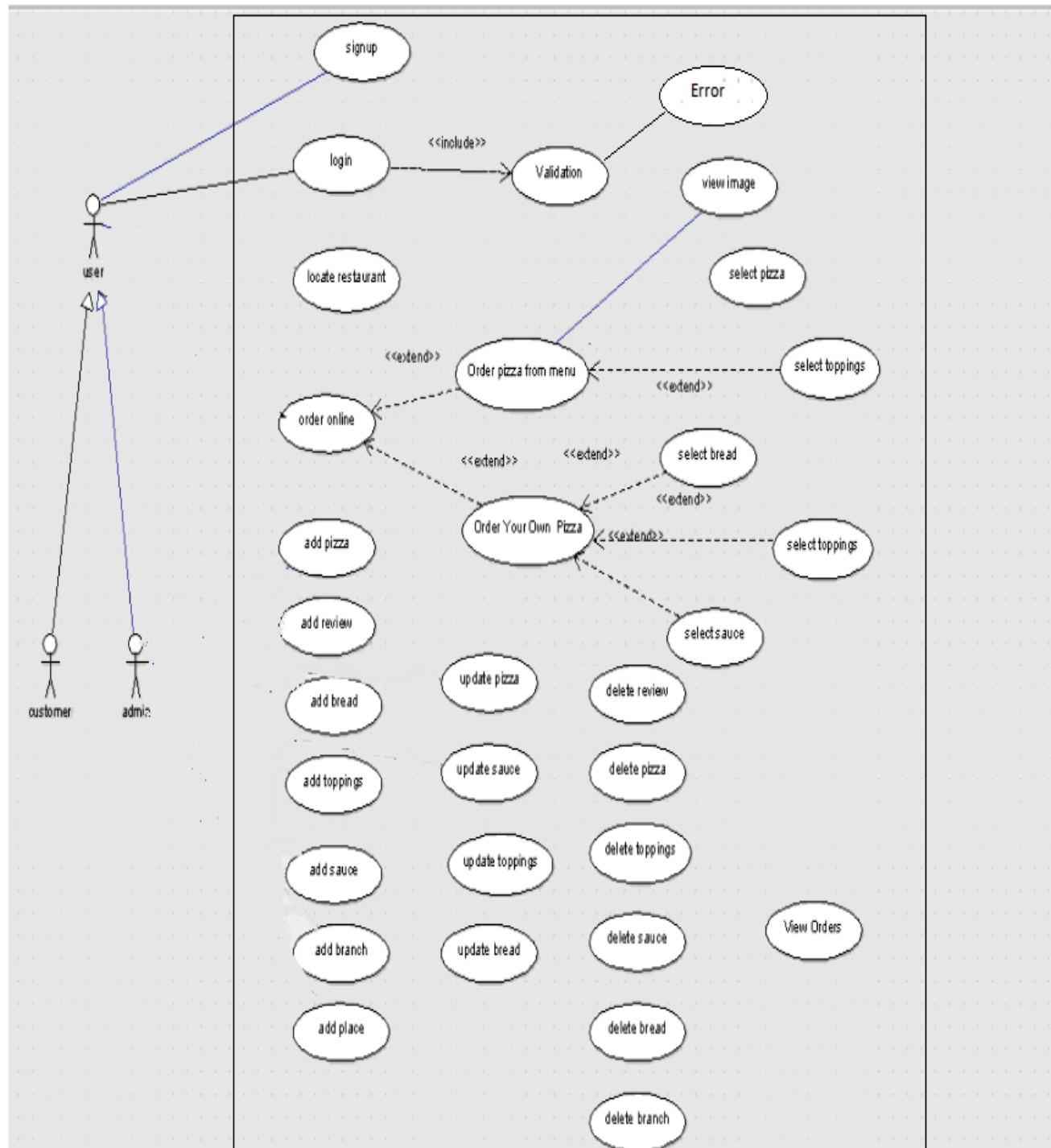
ii) Order customized pizza



2.3 Use case diagram working model



2.4 Use case diagram error case



3. Test Cases

```

<?php
function sum($a,$b){
    return $a+$b;
}

function multiply($a,$b){
    return $a*$b;
}

class testingController extends MX_Controller
{

    public function __construct(){
        parent::__construct();
    }

    function Test_AddUser()
    public function Test_AddUser() {
        $this->load->library("unit_test");
        $_SERVER["REQUEST_METHOD"] = "POST";
        $input['username']="testUser1";
        $input['password']=sha1("testPassword");
        $input['confirm_password']=sha1("testPassword");
        $input['emailid']=sha1("test@gmail.com");
        $input['phonenumber']=sha1("2432387");
        $input['city']=sha1("cityTest");
        $input['address']=sha1("adressTest");
        $_POST = $input;
        $this->signup_submit($_POST);
        $test = count($this->db->select('id')->from('user')-
>where('emailid',$input['emailid'])->get()->result());
        $expected_result = 1;
        $test_name = "testing if user is added";
        $this->unit->run($test, $expected_result, $test_name);
        echo $this->unit->report();
    }

    function Test_do_addpizza()
    public function Test_do_addpizza() {
        $this->load->library("unit_test");
        $_SERVER["REQUEST_METHOD"] = "POST";
        $input['pizza_name']="Mean Green Pizza";
        $input['category']="Non-veg";
        $_POST = $input;
        $this->signin_submit($_POST);
        $test = count($this->db->select('pizza_name')->from('pizzas')-
>where('pizza_name',$input['pizza_name'])->get()->result());
        $expected_result = 1;
        $test_name = "Unit test for checking successful creation of pizza";
    }
}

```

```

    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
}

Function Test_do_addtoppings()
public function Test_do_addtoppings() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['topping_name']="Jalapeno";
    $input['category']="Average";
    $_POST = $input;
    $this->do_addtoppings($_POST);
    $test = count($this->db->select('topping_name')->from('toppings')-
>where('topping_name',$input['topping_name'])->get()->result());
    $expected_result = 1;
    $test_name = "Unit test for checking successful addition of
toppings";
    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
}

Function Test_do_addbread()
public function Test_do_addbread() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['name']="Wheat";
    $input['size']="Large";
    $_POST = $input;
    $this->do_addbread($_POST);
    $test = count($this->db->select('name')->from('breads')-
>where('name',$input['name'])->get()->result());
    $expected_result = 1;
    $test_name = "Unit test for checking successful addition of breads";
    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
}

Function Test_do_addsauce()
public function Test_do_addsauce() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['name']="Tomato";
    $_POST = $input;
    $this->do_addsauce($_POST);
    $test = count($this->db->select('name')->from('sauce')-
>where('name',$input['name'])->get()->result());
    $expected_result = 1;
    $test_name = "Unit test for checking successful addition of Sauces";
    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
}

Function Test_updatepizzaprice_submit()
public function Test_updatepizzaprice_submit() {

```

```

$this->load->library("unit_test");
$_SERVER["REQUEST_METHOD"] = "POST";
$input['name']="Tomato";
$_POST = $input;
$this->updatepizzaprice_submit($_POST);
$test = count($this->db->select('name')->from('pizzas')-
>where('name',$input['name'])->get()->result());
$expected_result = 1;
$test_name = "Unit test for updating pizzas price";
$this->unit->run($test, $expected_result, $test_name);
echo $this->unit->report();
}

Function Test_updatetoppingsprice_submit()
public function Test_updatetoppingsprice_submit() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['name']="Tomato";
    $_POST = $input;
    $this->updatetoppingsprice_submit($_POST);
    $test = count($this->db->select('name')->from('toppings')-
>where('name',$input['toppingsname'])->get()->result());
    $expected_result = 1;
    $test_name = "Unit test for updating toppings price";
    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
}

function Test_updateextrasprice_submit()
public function Test_updateextrasprice_submit() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['name']="Mirch";
    $_POST = $input;
    $this->updateextrasprice_submit($_POST);
    $test = count($this->db->select('name')->from('extras')-
>where('name',$input['name'])->get()->result());
    $expected_result = 1;
    $test_name = "Unit test for updating extra price";
    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
}

function Test_updatesauceprice_submit()
public function Test_updatesauceprice_submit() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['name']="Alfredo";
    $_POST = $input;
    $this->updatesauceprice_submit($_POST);
    $test = count($this->db->select('name')->from('sauce')-
>where('name',$input['name'])->get()->result());
    $expected_result = 1;
    $test_name = "Unit test for updating sauce price";
    $this->unit->run($test, $expected_result, $test_name);
}

```

```

        echo $this->unit->report();
    }

    public function testing()
    {
        return "hello world";
    }

    public function signup_submit($input){

        return 1;
        //$this->load->view('signinup.php');
    }

}

?>

```

4.Contributions

Filename	Developer
About.php	Rishi
Afterlogin.php	Rishi
Changelocation.php	Rishi
Afteradmin.php	Rishi
Locator.php	Rishi
Login_view.php	Rishi
Header.php	Rishi
Updatebreadprice.php	Nagendra
Updatepizzaprice.php	Nagendra
Updatesauceprice.php	Nagendra
Updatetoppingsprice.php	Nagendra
Deletebreadprice.php	Gowtham
Deletepizzaprice.php	Gowtham
Deletesauceprice.php	Gowtham

Deletetoppingsprice.php	Gowtham
Addbread.php	Sourab
Addpizza.php	Sourab
Addsauce.php	Sourab
Addtoppings.php	Sourab
Addbranch.php	Sourab
Signup.php	All

5. User Manual

1. Summary

Pizza Crush is a web application that allows customers to order pizzas online providing many customizable features. The existing online pizza ordering applications allow customers to select pizzas and toppings which are present in their menu.

Our web application provides many services to the customers like ordering pizzas from the menu as in existing system, Customize their own pizzas by selecting products in sequential manner and thus enjoy a new delicious pizza which they like, displaying the final image where toppings and pizza images are overlapped after selecting pizza and toppings and dynamically generating bill as soon as customer selects an item.

This document contains detailed steps indicating its reader on how to use this application.

2. Customer with no account:

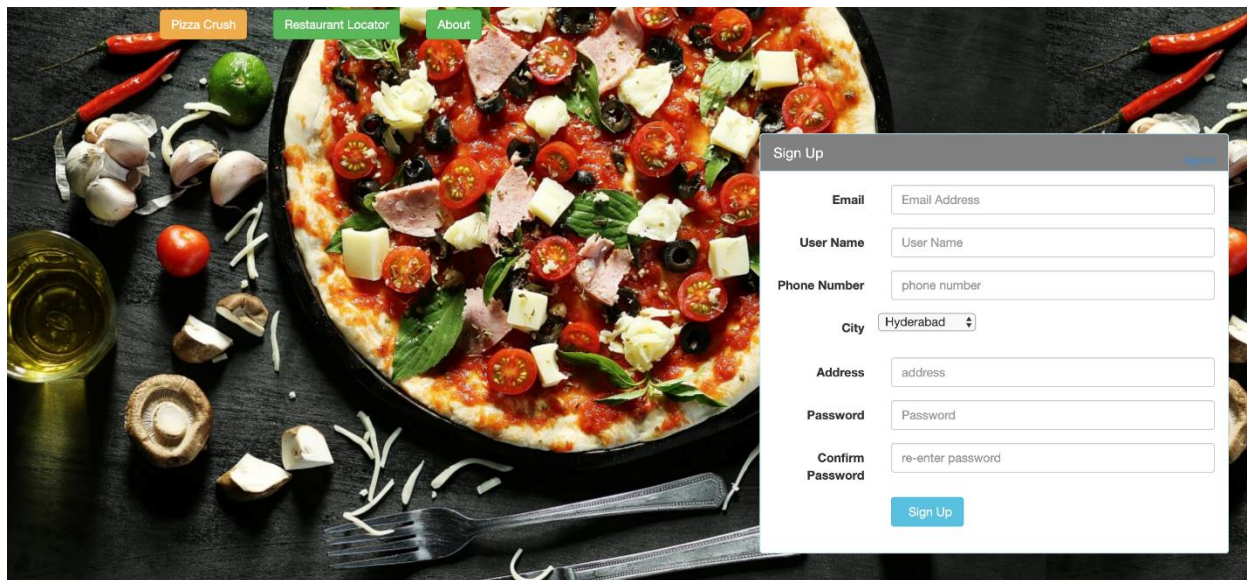
A user has to register himself in the system in case if he/she wants to order a pizza.

2.1. Home Page

This is the Home page which appears when the customer enters the website's URL. To get inside our website, the users have two options either to sign-in or sign-up. For that, we have 2 pages.

2.2. Sign-up

- 2.2.1. A new user can register for our website by filling up his email, username, phone number and password fields
- 2.2.2. Once the user fills in all the details, click on the "Sign up" button to create an account.
- 2.2.3. On successful registration the user is directly redirected to the Login page.



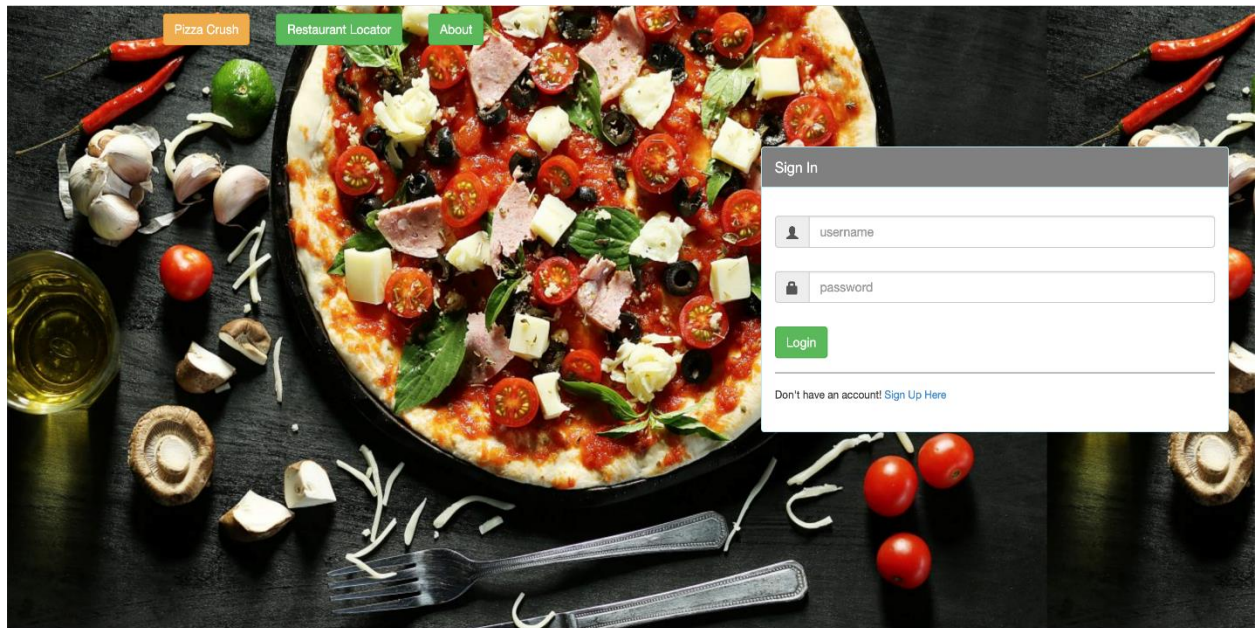
3. Customer with user account:

3.1. Home Page

- 3.1.1. Please refer to section 2.1 for details.

3.2. Login

- 3.2.1. A Customer who has registered an account with the system can Login to the system by using the “Login” button located in the home page.
- 3.2.2. The Customer must enter their credentials and click on “Login”. If the user is validated, they are redirected to the “Home”.
- 3.2.3. Else and error message is displayed.
- 3.2.4. Additionally, this page also provides an option to “Sign up here” to register an account if they do not have one.
- 3.2.5. If customer is identified as an admin, he will be redirected to admin page.

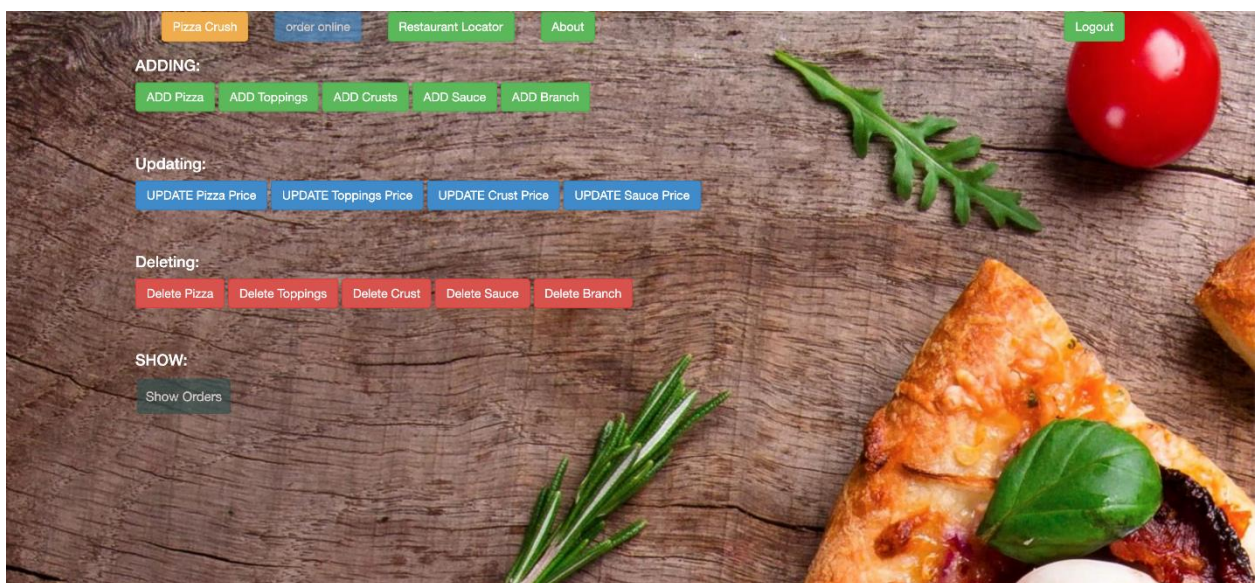


Contact: onestep@gmail.com

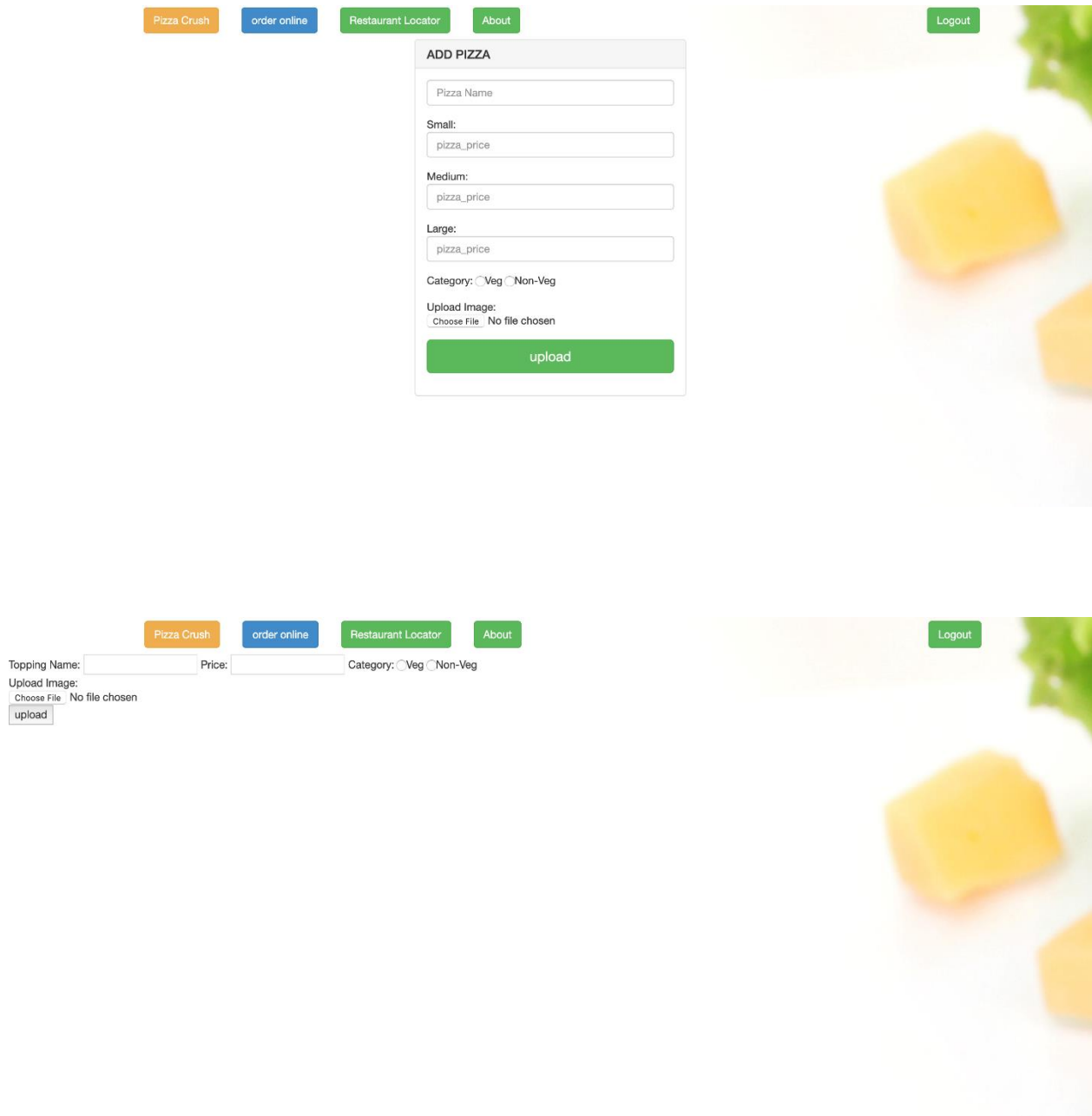
4. Admin

Admin views a similar website as the Customer but has the following additional options.

- 4.1. This page has options for adding, updating and deleting of products from the product list. The products include toppings, bread, sauce, pizzas, restaurant locator. The admin page can proceed to add page, update page and delete page.



4.2. Add page: This page is used by the admin to add toppings, sauces, breads, stores info and prices.



Pizza Crush order online Restaurant Locator About Logout

ADD PIZZA

Pizza Name

Small:

Medium:

Large:

Category: ☐ Veg ☐ Non-Veg

Upload Image: No file chosen

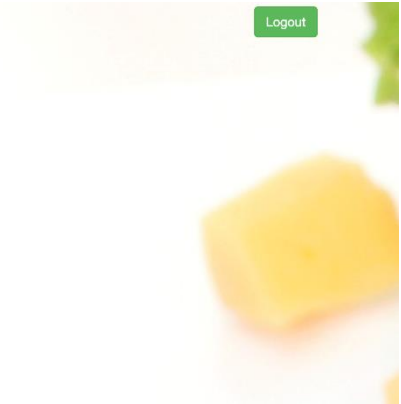
Topping Name: Price: Category: ☐ Veg ☐ Non-Veg

Upload Image: No file chosen

[Pizza Crush](#) [order online](#) [Restaurant Locator](#) [About](#)

sauce Name: Price:

Upload Image:
 No file chosen



[Pizza Crush](#) [order online](#) [Restaurant Locator](#) [About](#) [Logout](#)

Topping Name: Price: Category: ☐ Veg ☐ Non-Veg

Upload Image:
 No file chosen



4.3. Update page: This page is used by the admin to update toppings, sauces, breads, stores info and prices.

Pizza Crush

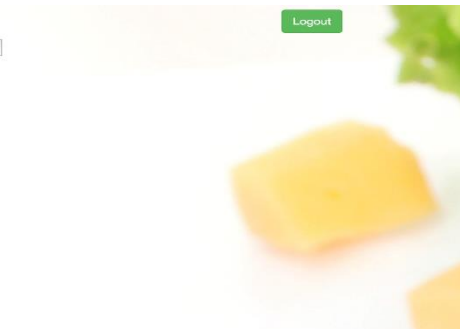
order online

Restaurant Locator

About

Enter the pizza name: Size: Enter the price to be updated

Logout



Pizza Crush

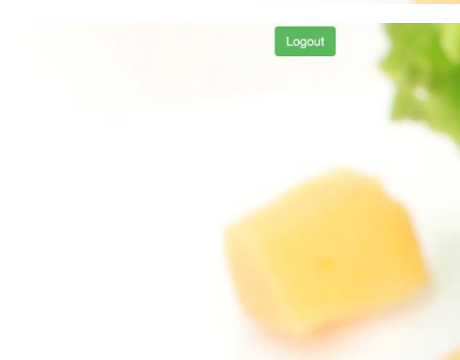
order online

Restaurant Locator

About

Enter the topping name: Enter the price to be updated

Logout



Pizza Crush

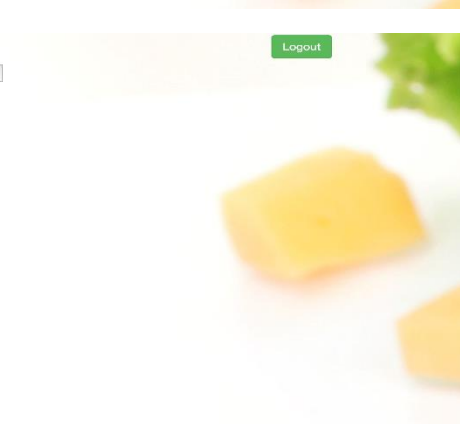
order online

Restaurant Locator

About

Enter the bread name: Select bread size: Enter the price to be updated

Logout



Pizza Crush

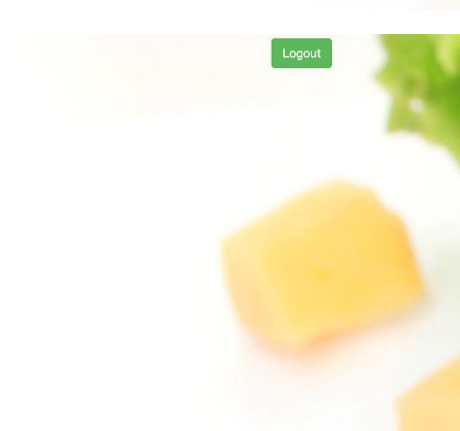
order online

Restaurant Locator

About

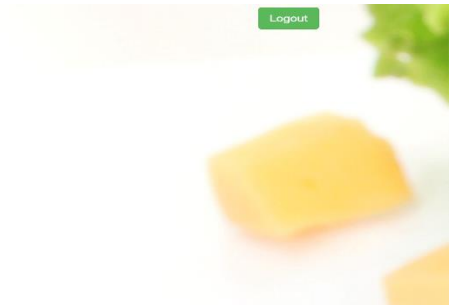
Enter the sauce name: Enter the price to be updated

Logout



[Pizza Crush](#) [order online](#) [Restaurant Locator](#) [About](#)

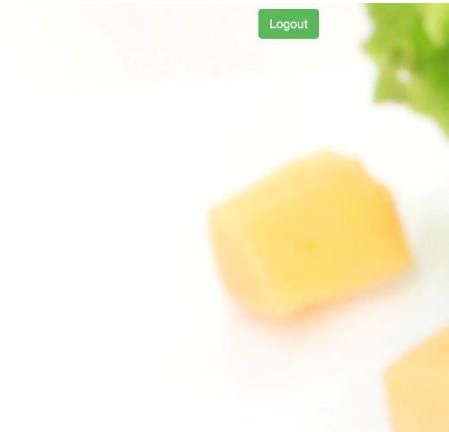
Enter the sauce name: Enter the price to be updated:



4.4. Delete page: This page is used by the admin to delete toppings, sauces, breads, store-info and prices.

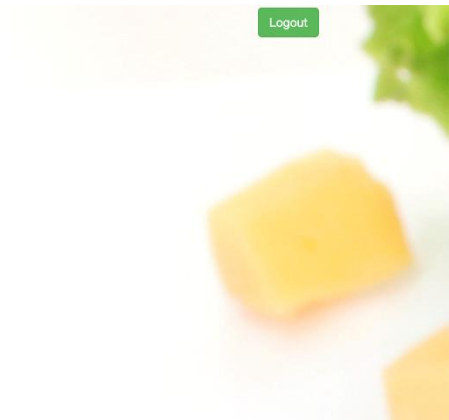
[Pizza Crush](#) [order online](#) [Restaurant Locator](#) [About](#)

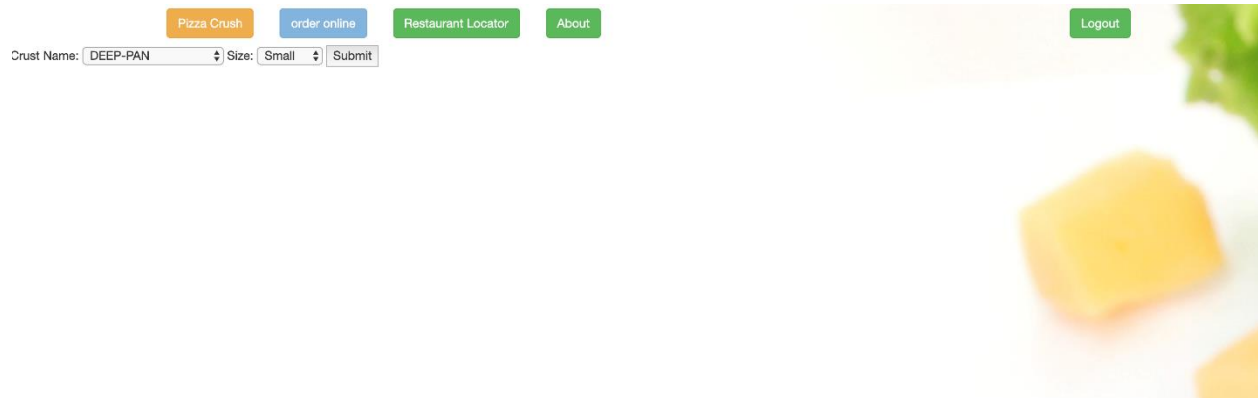
Pizza Name: Size:



[Pizza Crush](#) [order online](#) [Restaurant Locator](#) [About](#)

Topping Name: Category: ☐ Veg ☐ Non-Veg





6.Installation Instructions

To compile/run the program and test cases.

- Install PHP.
- Install XAMPP for server and database.
- Place the mysql database file in localhost/phpMyAdmin.
- Copy the project folder “pizzacrush” to “htdocs” folder in XAMPP.
- Open browser, type localhost/pizzacrush.

Sample login credentials:

Customer

Email: testuser1@gmail.com

Password: venu123\$

Administrator

Email: rishireddykolanu@gmail.com

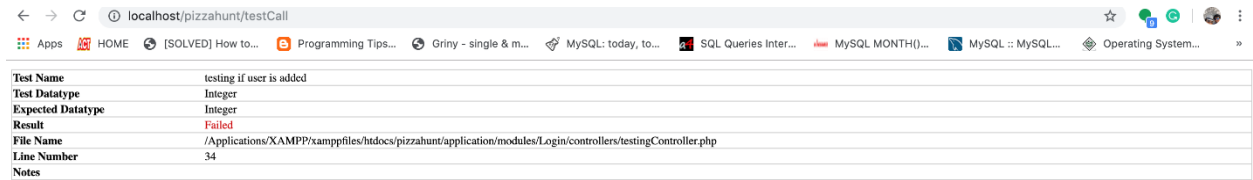
Password: hiiamrishi

Compile/Run the program

- To run the application, you need to refresh the browser after successfully doing the above mentioned steps.

Compile/Run the test cases

- Open browser type “localhost/pizzacrush/testmethod-name” to run the test cases. You will be able to see the output of test case.



Test Name	testing if user is added
Test Datatype	Integer
Expected Datatype	Integer
Result	Failed
File Name	/Applications/XAMPP/xamppfiles/htdocs/pizzahunt/application/modules/Login/controllers/testingController.php
Line Number	34
Notes	

7. Peer review feedback

Feedback received during peer review session:

Suggestion to add review component where customer can submit his review

Changes/actions taken based on the feedback:

As we are working on admin module currently, there is no scope to add this component now. Will be going to work on customer module and implement the review component where customers can submit their review.