Development Phase-3 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

11/18/2019

Table of contents

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.

1.5: Image overlapping of toppings over pizzas

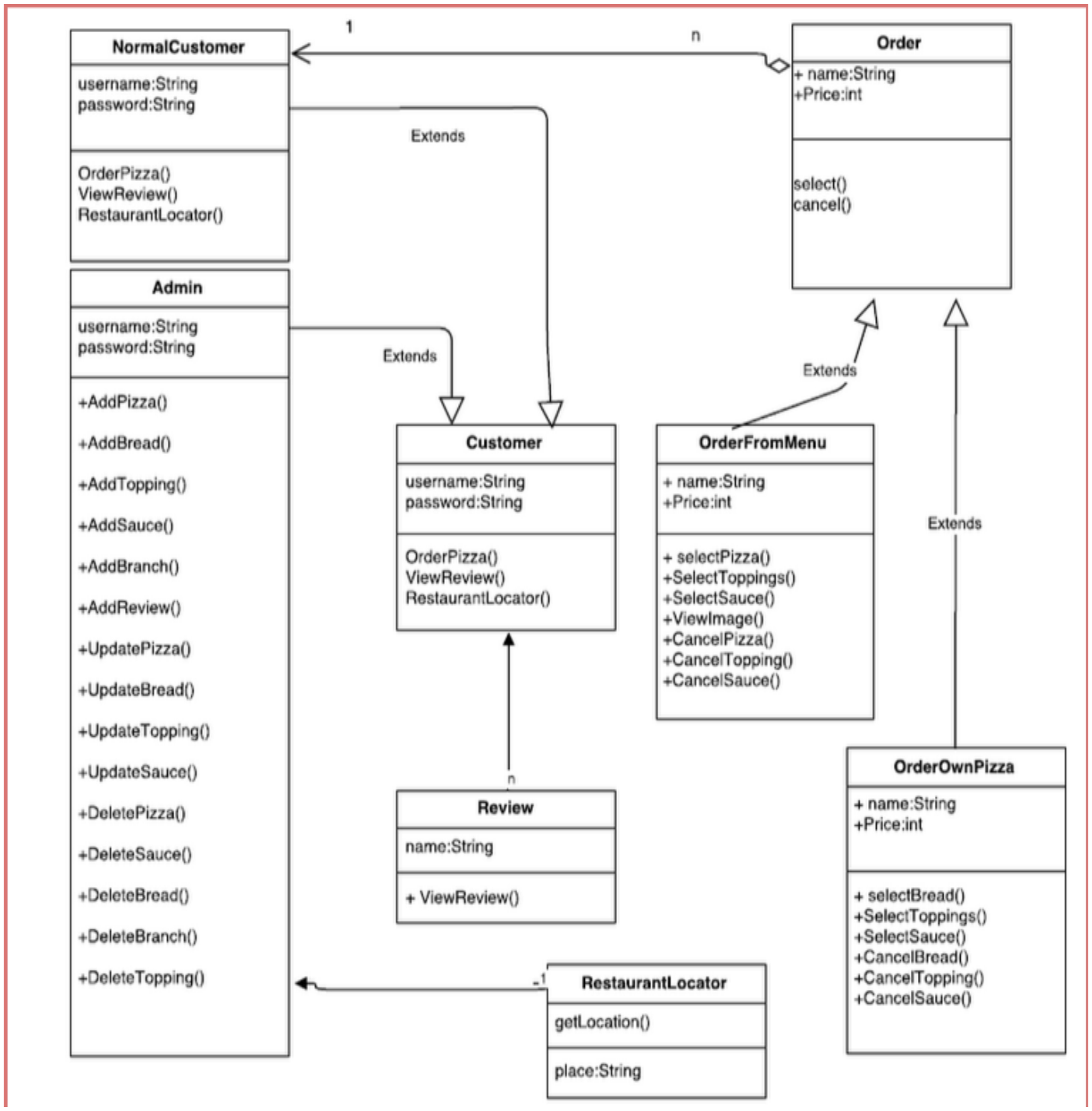
The user can get a view of how his/her pizza is going to look like because of the toppings overlapping feature over the pizzas.

1.6 Dynamic Bill Generation

The bill is generated as soon as the user selects a pizza and adds to his cart.

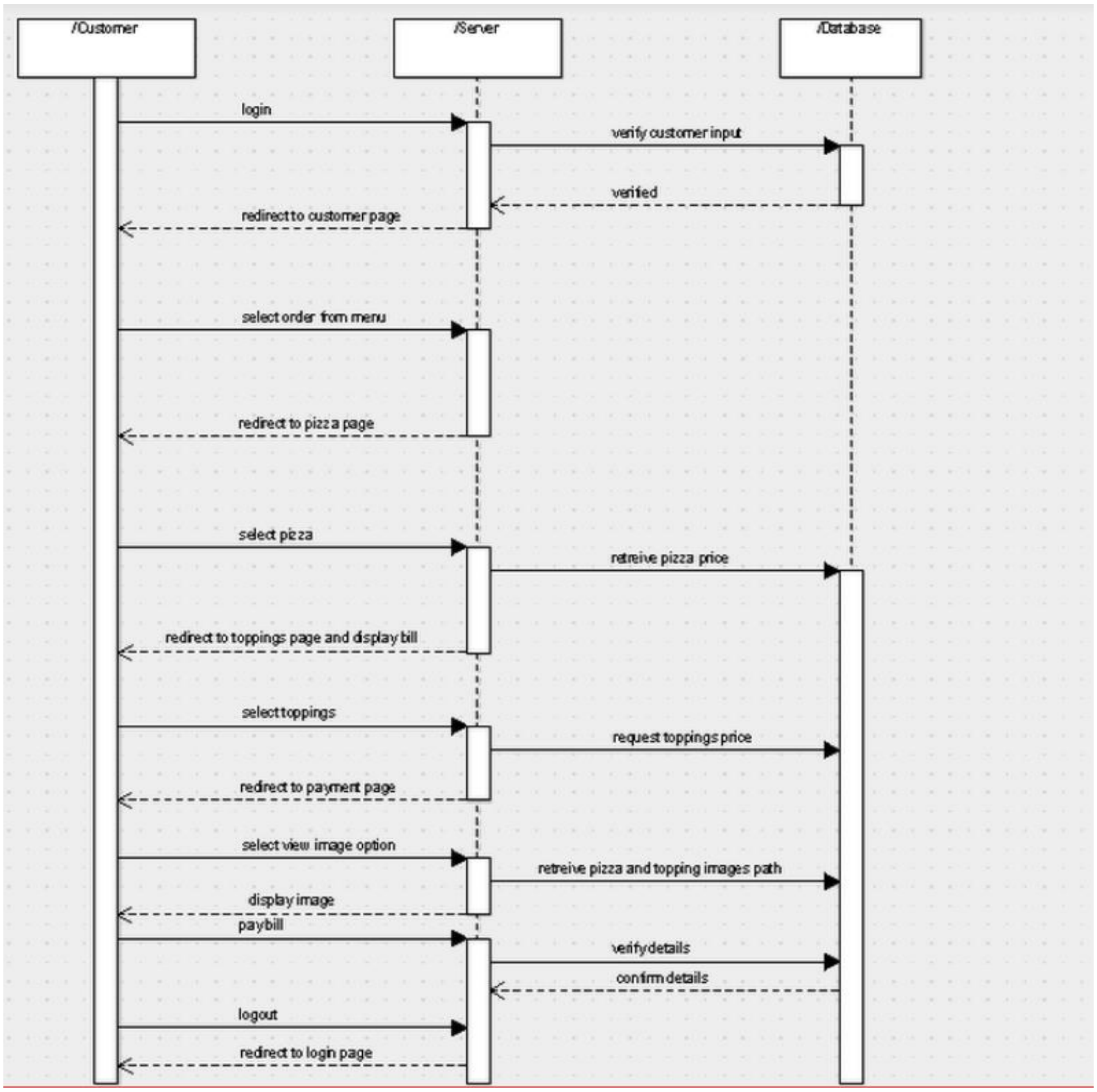
2. UML Design

2.1: Class Diagram

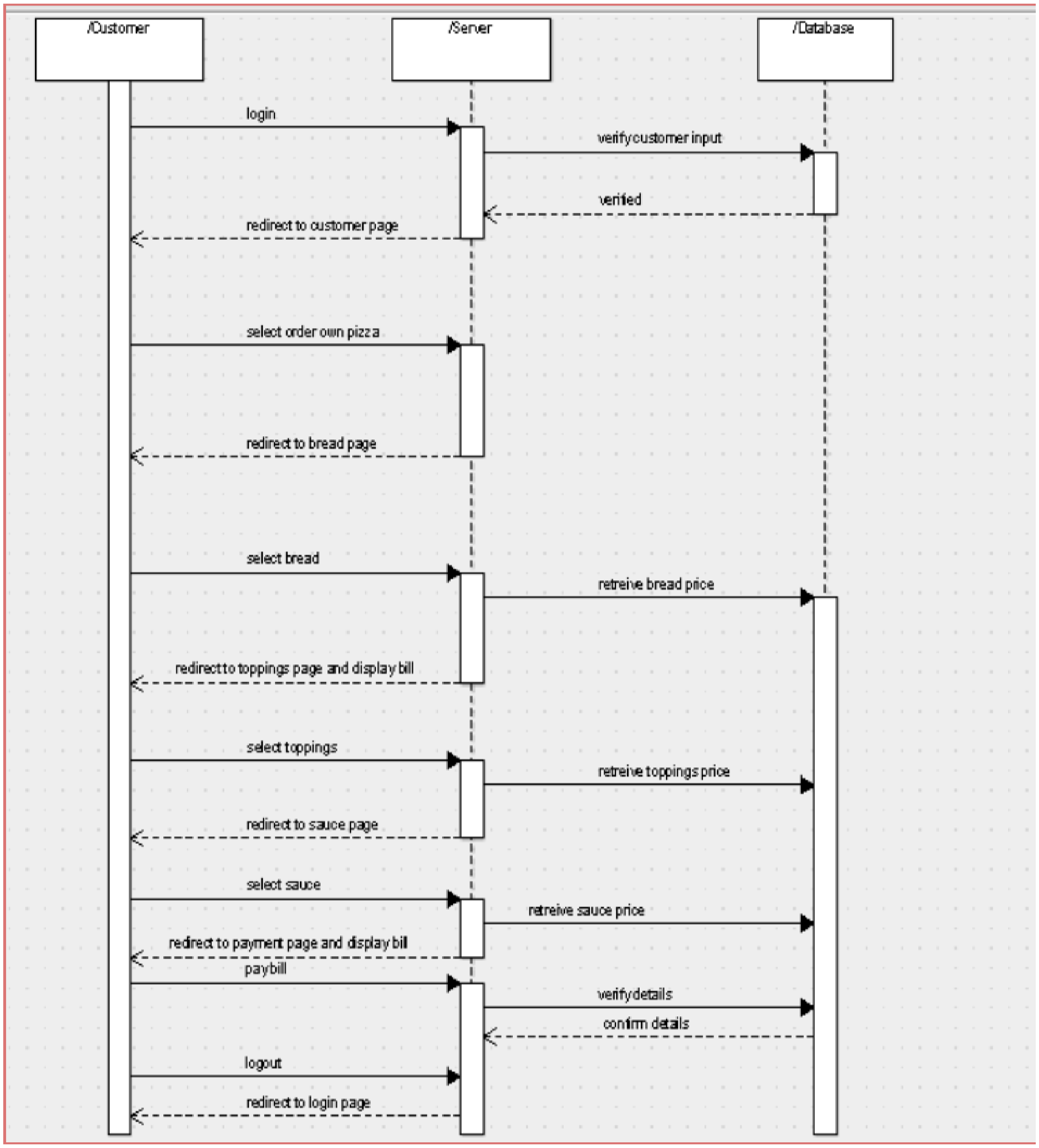


2.2 Sequence Diagram

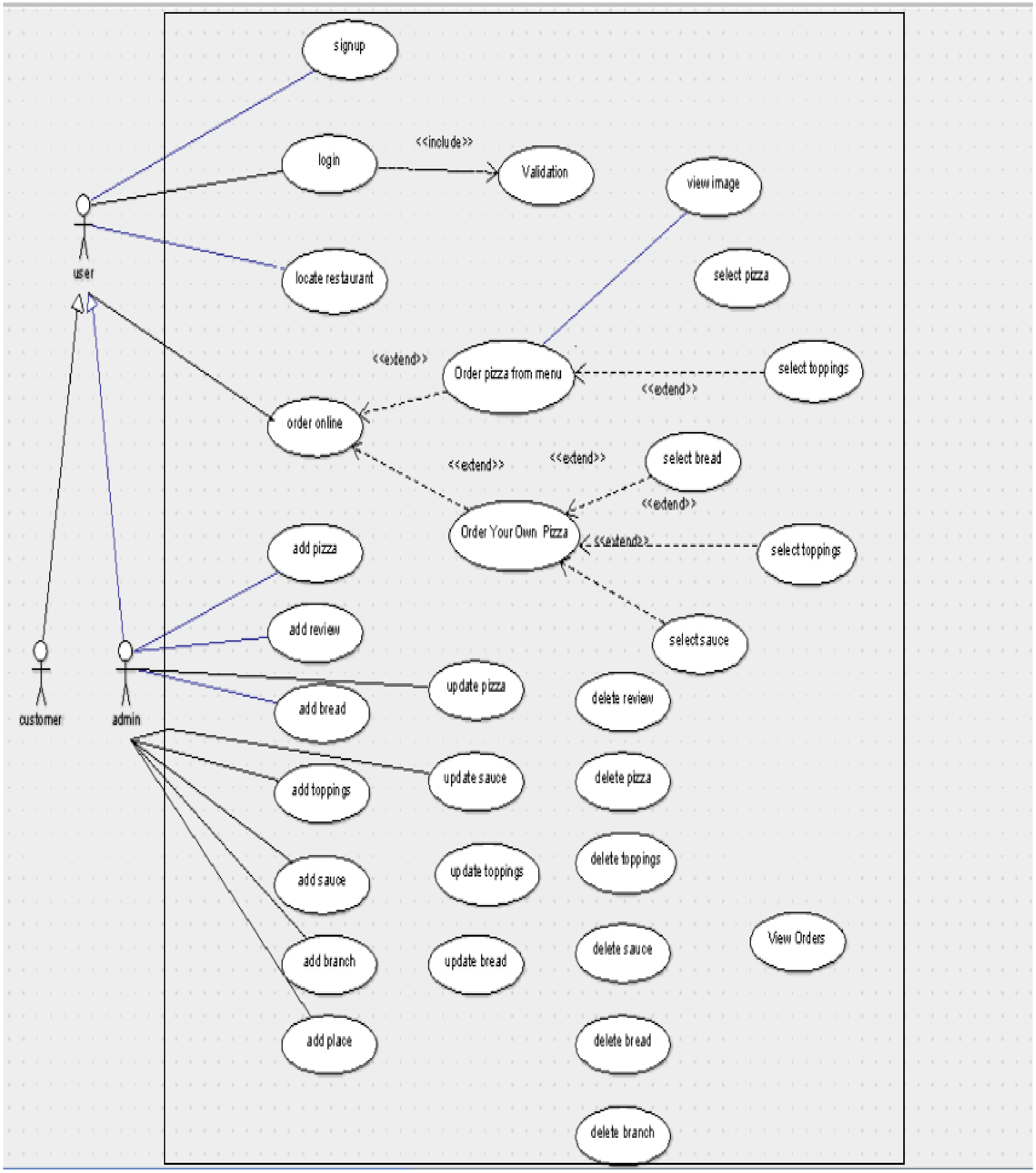
i)Order pizza from menu



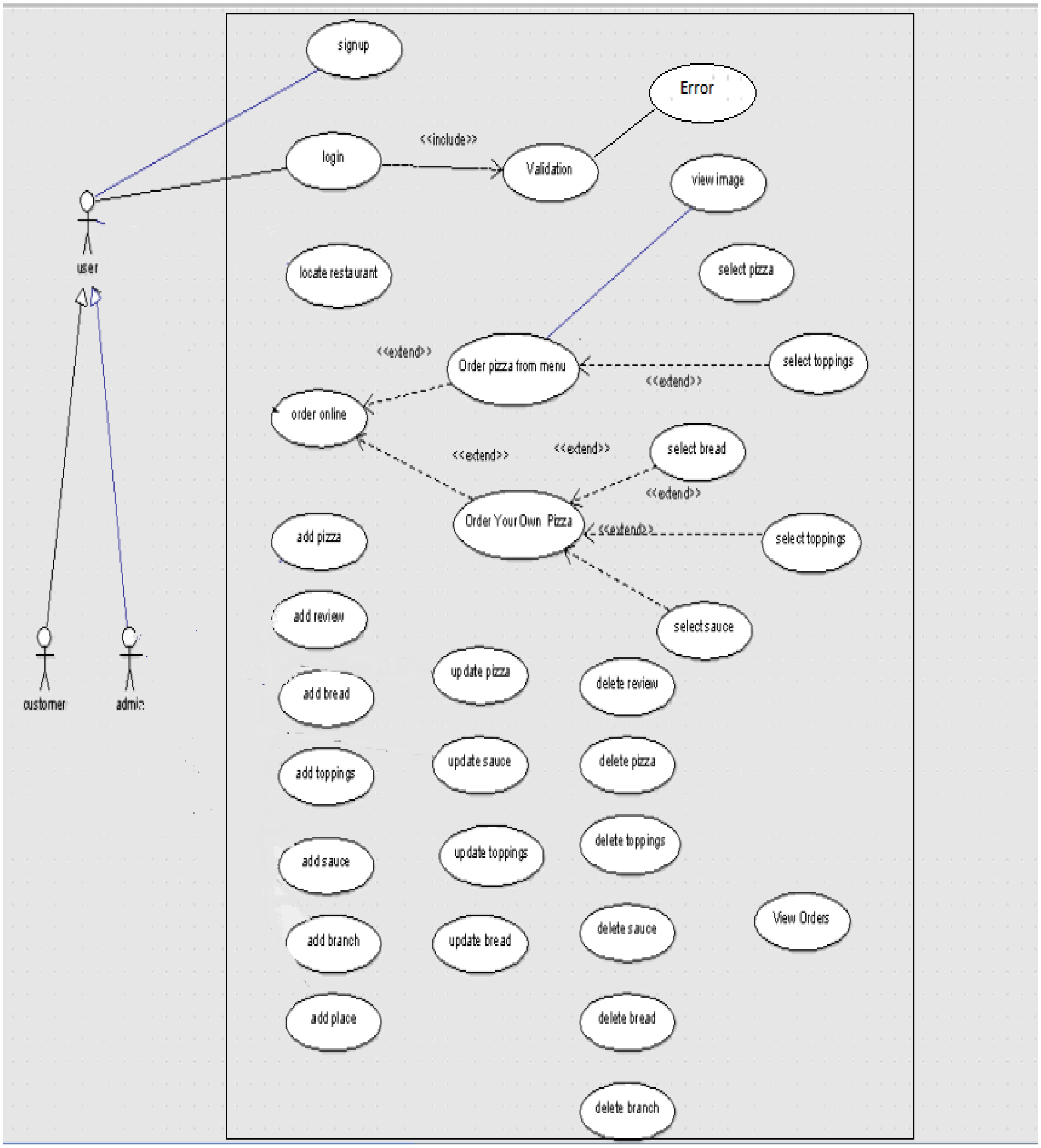
ii)Order customized pizza



2.3Use case diagram working model



2.4 Use case diagram error case



3. Test Cases and System Tests

i) **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();
  }

//Add user test
  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();
  }
 //add pizza
 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();
   }

//add toppings
 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();
   }

//adding bread
  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();
   }
//adding sauce test
  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();
  }
//testing update pizza price
  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();
  }
//update toppings price testing
  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();
  }

//testing update extra price
  public function Test_updateextraprice_submit() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['name']="Mirch";
    $_POST = $input;
    $this->updateextraprice_submit($_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();
   }
  //testing update sauce price
  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();
   }
//testing delete sauce
  public function Test_deletesauce() {
     $this->load->library("unit_test");
     $_SERVER["REQUEST_METHOD"] = "POST";
     $input['name']="Alfredo";
     $_POST = $input;
     $this->deletesauce($_POST);
     $test = count($this->db->select('name')->from('sauce')-
>where('name',$input['name'])->get()->result());
     $expected_result = 0;
     $test_name = "Unit test for updating sauce price";
     $this->unit->run($test, $expected_result, $test_name);
     echo $this->unit->report();
   }
 //testing delete branch
    public function Test_deletebranch() {
     $this->load->library("unit_test");
     $_SERVER["REQUEST_METHOD"] = "POST";
     $input['name']="Himayatnagar";
     $_POST = $input;
     $this->updatesauceprice_submit($_POST);
     $test = count($this->db->select('name')->from('branch')-
>where('name',$input['name'])->get()->result());
     $expected_result = 0;
     $test_name = "Unit test for delete branch";
    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
  }
//testing delete pizza
    public function Test_deletepizza_submit() {
        $this->load->library("unit_test");
        $_SERVER["REQUEST_METHOD"] = "POST";
        $input['name']="Veg Pizza";
        $_POST = $input;
        $this->deletepizza_submit($_POST);
        $test = count($this->db->select('name')->from('pizza')-
>where('name',$input['name'])->get()->result());
        $expected_result = 0;
        $test_name = "Unit test for deleting pizza";
        $this->unit->run($test, $expected_result, $test_name);
        echo $this->unit->report();
      }
//testing delete toppings
  public function Test_deletetoppings_submit() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['name']="onions";
    $_POST = $input;
    $this->deletetoppings_submit($_POST);
    $test = count($this->db->select('name')->from('toppings')-
>where('name',$input['name'])->get()->result());
    $expected_result = 0;
    $test_name = "Unit test for deleting toppings";
    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
  }
// testing delete bread
  public function Test_deletebread_submit() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['name']="Tomato";
    $_POST = $input;
    $this->deletebread_submit($_POST);
    $test = count($this->db->select('name')->from('bread')-
>where('name',$input['name'])->get()->result());
    $expected_result = 0;
    $test_name = "Unit test for delete bread";
    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
  }
//testing delete sauce
```

```
  public function Test_deletesauce_submit() {
     $this->load->library("unit_test");
     $_SERVER["REQUEST_METHOD"] = "POST";
     $input['name']="Game";
     $_POST = $input;
     $this->deletesauce_submit($_POST);
     $test = count($this->db->select('name')->from('sauce')-
>where('name',$input['name'])->get()->result());
     $expected_result = 0;
     $test_name = "Unit test for deleting sauce price";
     $this->unit->run($test, $expected_result, $test_name);
     echo $this->unit->report();
  }
//testing delete branch
  public function Test_deletebranch_submit() {
     $this->load->library("unit_test");
     $_SERVER["REQUEST_METHOD"] = "POST";
     $input['name']="Alfredo";
     $_POST = $input;
     $this->deletebranch_submit($_POST);
     $test = count($this->db->select('name')->from('branch')-
>where('name',$input['name'])->get()->result());
     $expected_result = 0;
     $test_name = "Unit test Delete Branch";
     $this->unit->run($test, $expected_result, $test_name);
     echo $this->unit->report();
  }
//testing delete p
  public function Test_delete_p() {
     $this->load->library("unit_test");
     $_SERVER["REQUEST_METHOD"] = "POST";
     $input['name']="Delete_P";
     $_POST = $input;
     $this->deletebranch_submit($_POST);
     $test = count($this->db->select('name')->from('pizza')-
>where('name',$input['name'])->get()->result());
     $expected_result = 0;
     $test_name = "Unit test delete Pizza";
     $this->unit->run($test, $expected_result, $test_name);
     echo $this->unit->report();
  }
//testing delete p
  public function fTest_delete_p() {
     $this->load->library("unit_test");
     $_SERVER["REQUEST_METHOD"] = "POST";
     $input['name']="Delete_T";
```

```
    $_POST = $input;
    $this->deletebranch_submit($_POST);
    $test = count($this->db->select('name')->from('pizza')-
>where('name',$input['name'])->get()->result());
    $expected_result = 0;
    $test_name = "Unit test delete Toppings";
    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
  }

  //testing own toppings
  public function Test_own_toppings_click() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['name']="Toppings";
    $_POST = $input;
    $this->own_toppings_click($_POST);
    $test = count($this->db->select('name')->from('toppings')-
>where('name',$input['name'])->get()->result());
    $expected_result = 0;
    $test_name = "Unit test delete Toppings";
    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
  }

  //testing own delivered
  public function Test_owndelivered() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['name']="Alfredo";
    $_POST = $input;
    $this->deletebranch_submit($_POST);
    $test = count($this->db->select('uid')->from('temp_o')->where('uid',$input['uid'])-
>get()->result());
    $expected_result = 0;
    $test_name = "Unit test own delivered";
    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
  }

  //testing bread_click
  public function Test_bread_click() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['name']="Alfredo";
    $_POST = $input;
    $this->deletebranch_submit($_POST);
    $test = count($this->db->select('name')->from('bread')->where('uid',$input['uid'])-
>get()->result());
    $expected_result = 0;
    $test_name = "Unit test bread click";
    $this->unit->run($test, $expected_result, $test_name);
    echo $this->unit->report();
  }

  //testing toppings click
  public function Test_toppings_click() {
    $this->load->library("unit_test");
    $_SERVER["REQUEST_METHOD"] = "POST";
    $input['name']="Alfredo";
    $_POST = $input;
    $this->deletebranch_submit($_POST);
    $test = count($this->db->select('name')->from('toppings')->get()->result());
    $expected_result = 0;
    $test_name = "Unit test toppings delivered";
    $this->unit->run($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');
 }



}

?>
```

ii) **System testing:** We have tested our website by integrating all the components and have also tested in different Operating systems to check the compatibility.

4.Contributions

Filename	Developer
TestingController.php	Gowtham
Showorders.php	Nagendra
Ownorders.php	Nagendra
Ownpizza.php	Nagendra
SignInUp.php	Rishi
Sauce_display.php	Sourab
Own_pizza_application.php	Sourab
Own_toppings_display.php	Sourab

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

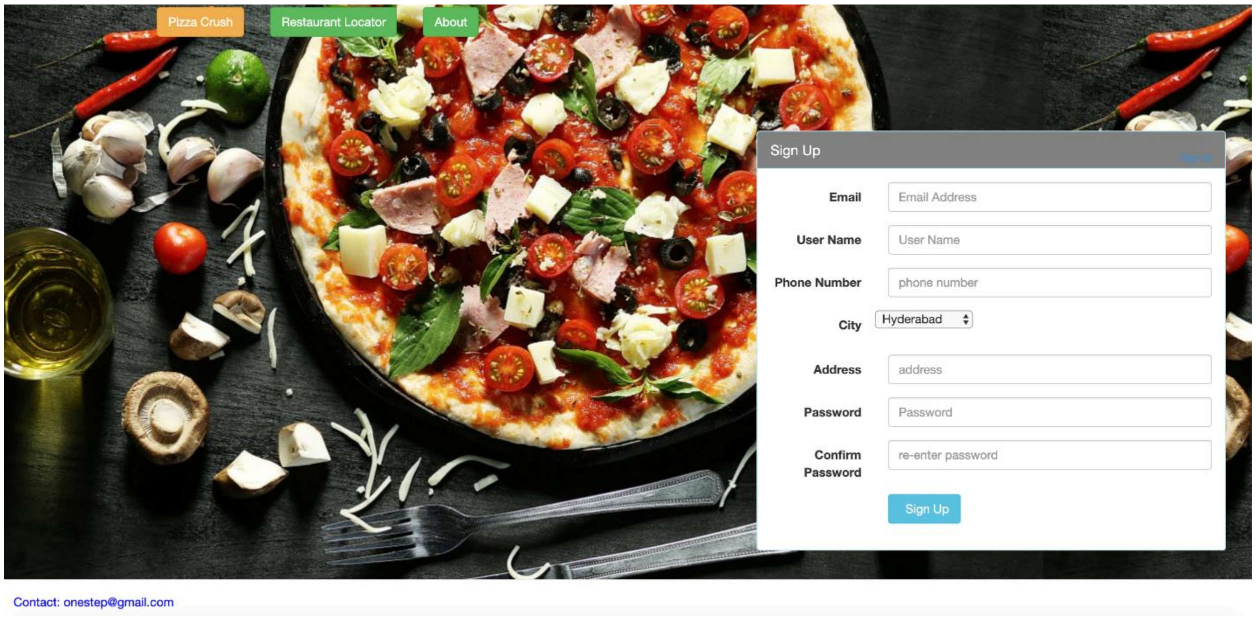
This is the Home page which appears when the customer enters the website's URL. To get inside our website, the users have two options either to sign-in or sign-up. For that, we have 2 pages.

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.



Email	Email Address
User Name	User Name
Phone Number	phone number
City	Hyderabad
Address	address
Password	Password
Confirm Password	re-enter password

Contact: onestep@gmail.com

3. Customer with user account:

3.1. Home Page

3.1.1. Please refer to section 2.1 for details.

3.2. Login

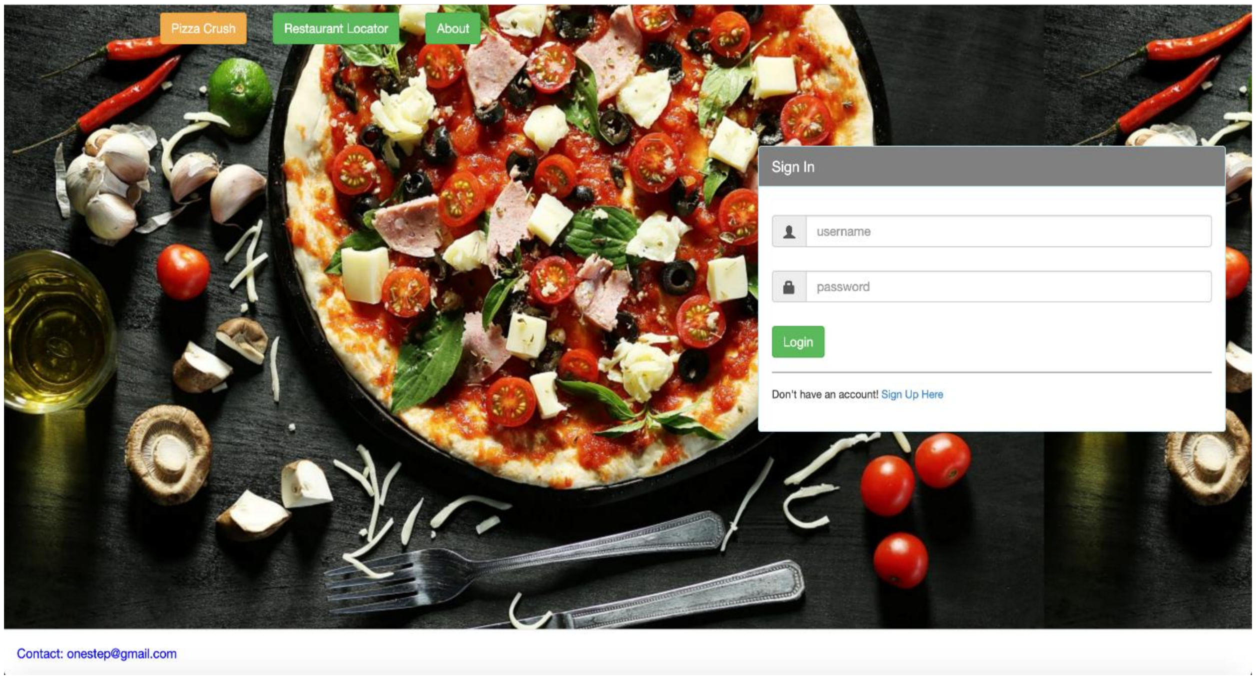
3.2.1. A Customer who has registered an account with the system can Login to the system by using the "Login" button located in the home page.

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.

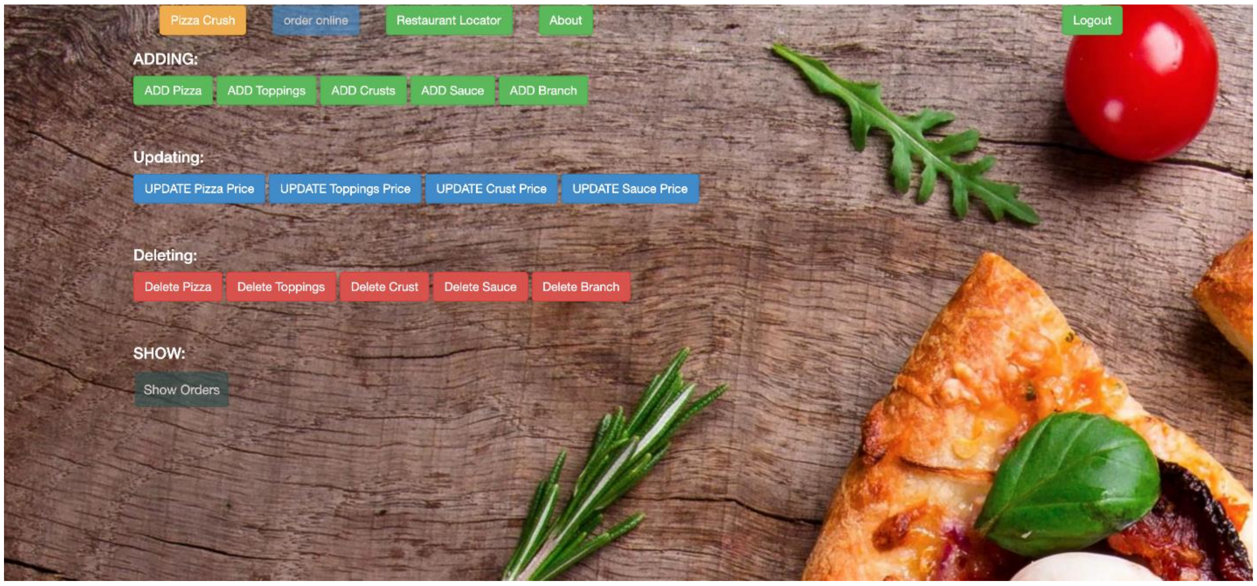
3.2.5. If customer is identified as an admin, he will be redirected to admin page.



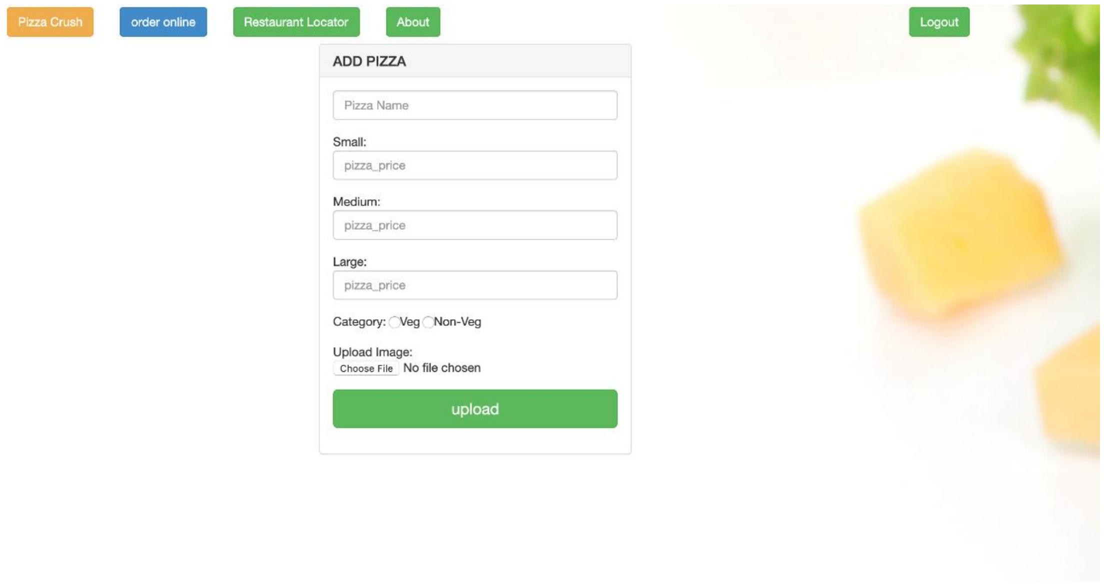
4. **Admin**

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.



ADD PIZZA

Pizza Name

Small: pizza_price

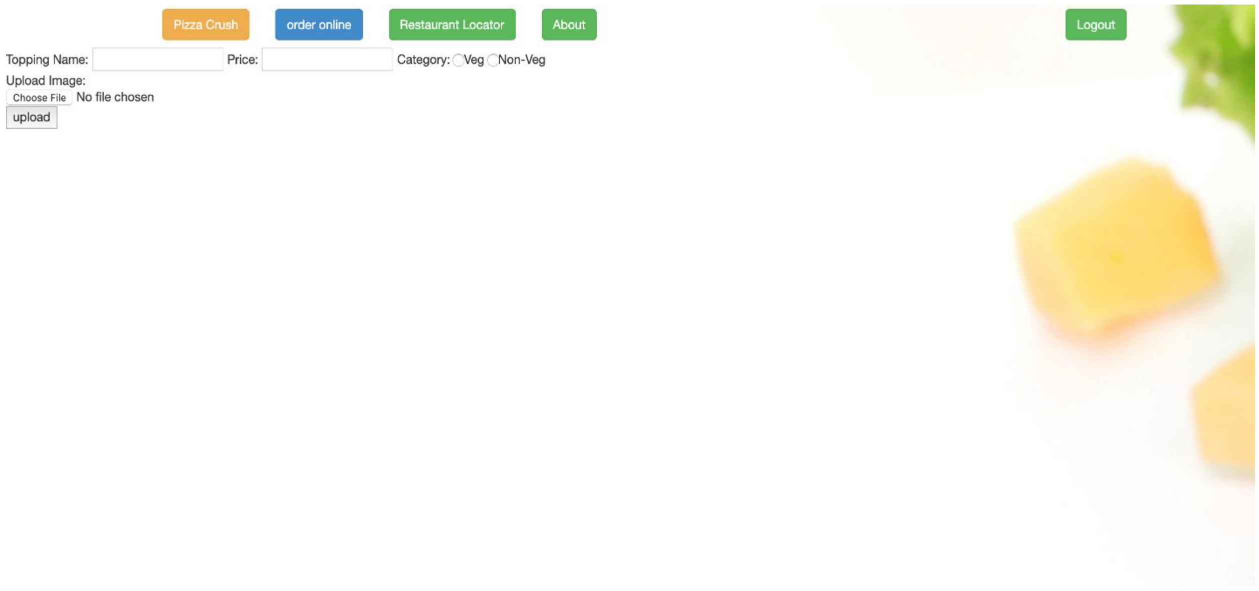
Medium: pizza_price

Large: pizza_price

Category: Veg Non-Veg

Upload Image: Choose File No file chosen

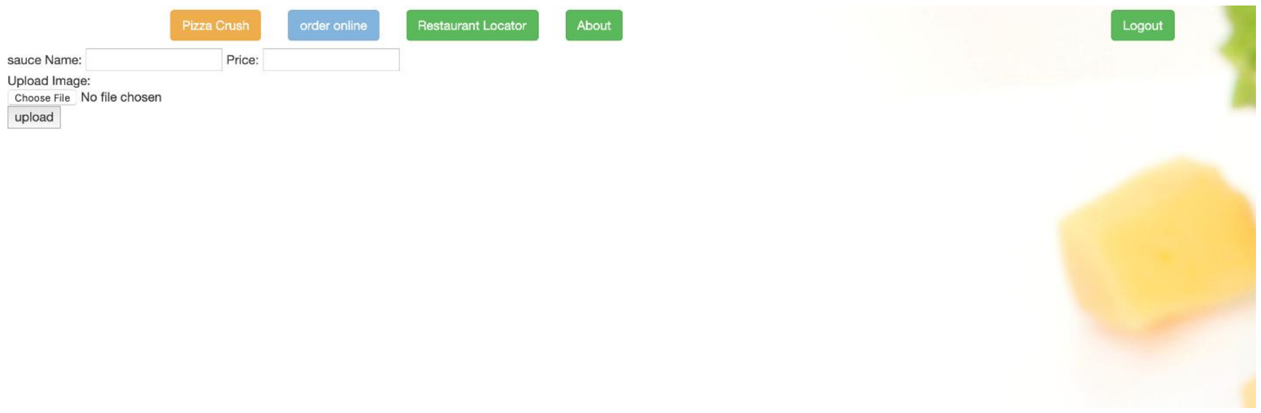
upload



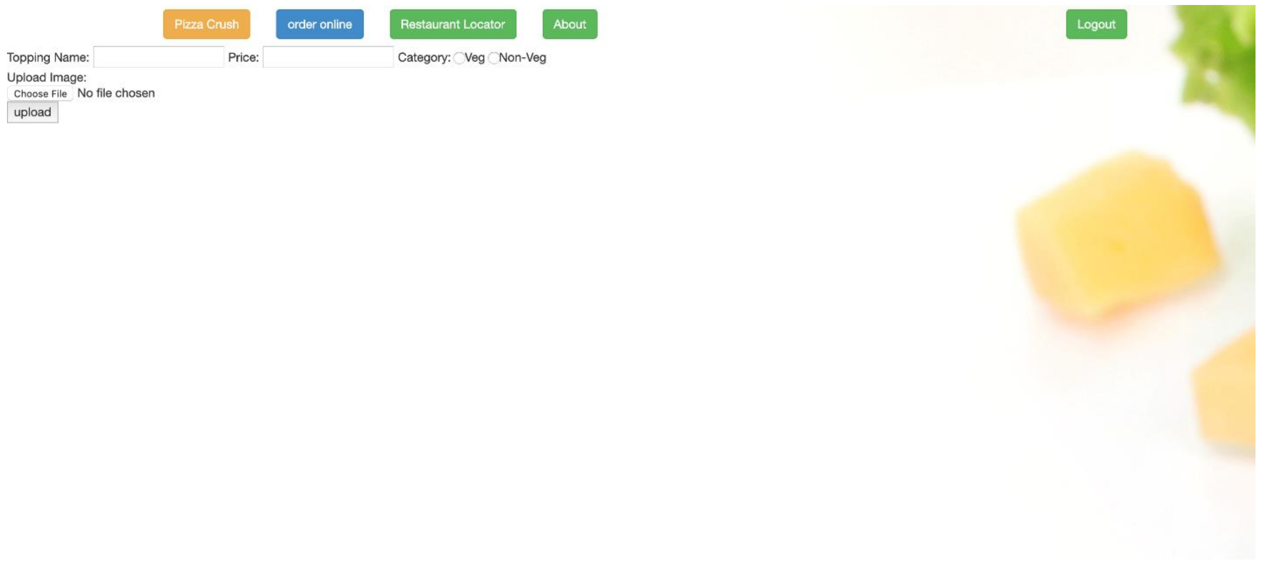
Topping Name: Price: Category: Veg Non-Veg

Upload Image: Choose File No file chosen

upload

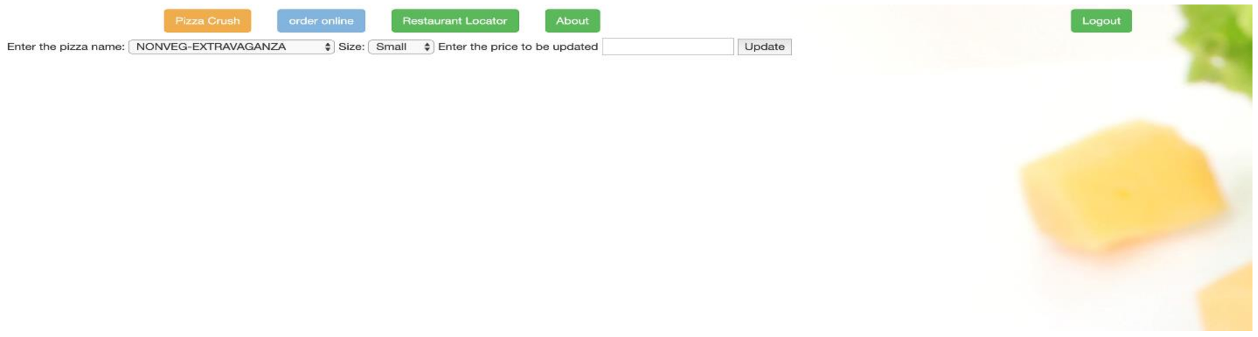


sauce Name: Price:
Upload Image:
No file chosen

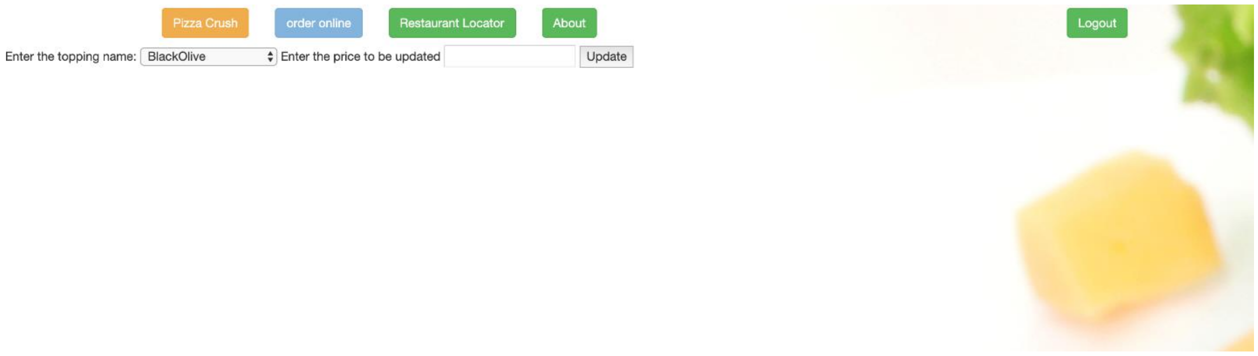


Topping Name: Price: Category: Veg Non-Veg
Upload Image:
No file chosen

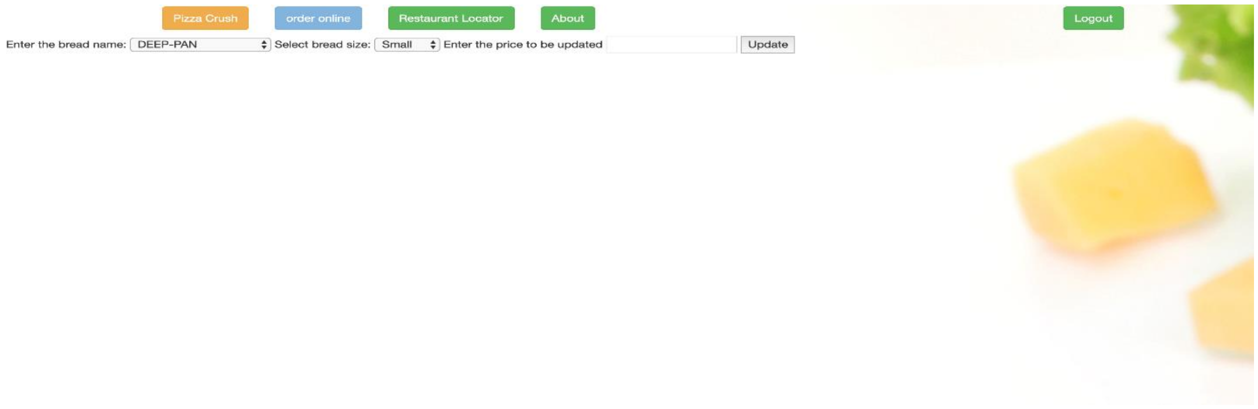
4.3.Update page: This page is used by the admin to update toppings, sauces, breads, stores info and prices.



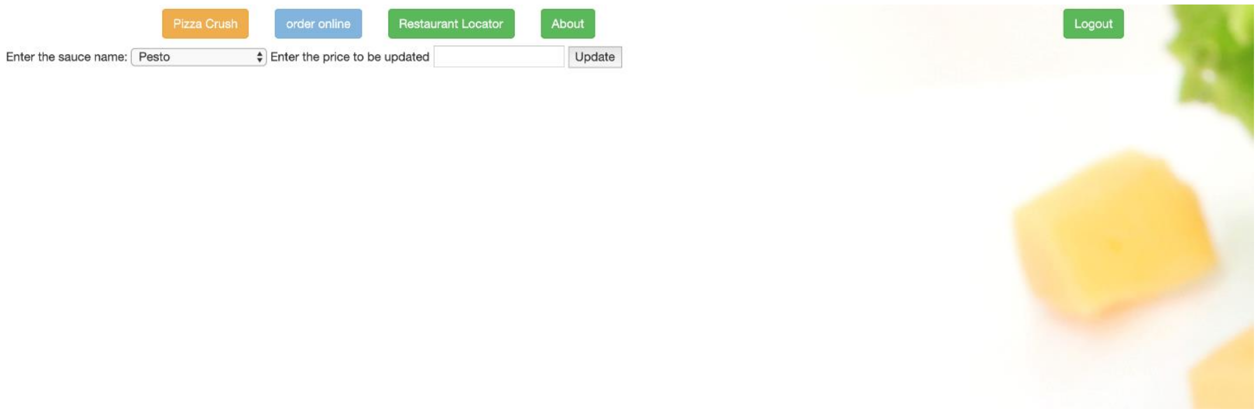
Enter the pizza name: NONVEG-EXTRAVAGANZA Size: Small Enter the price to be updated



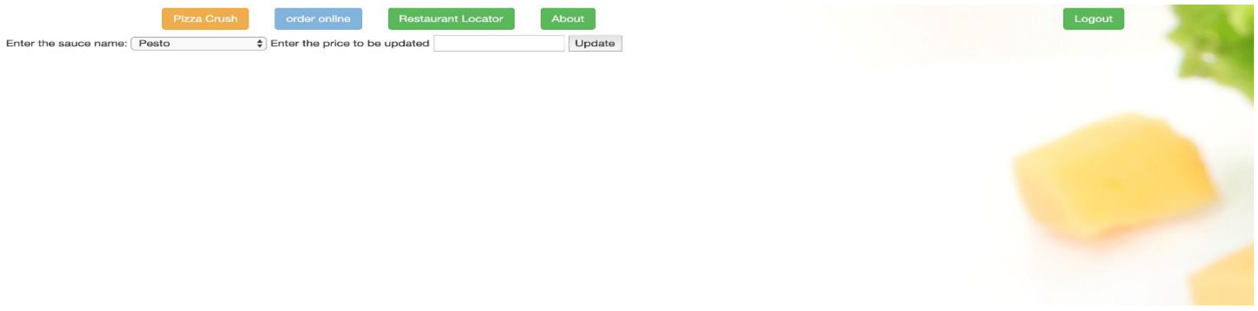
Enter the topping name: BlackOlive Enter the price to be updated



Enter the bread name: DEEP-PAN Select bread size: Small Enter the price to be updated

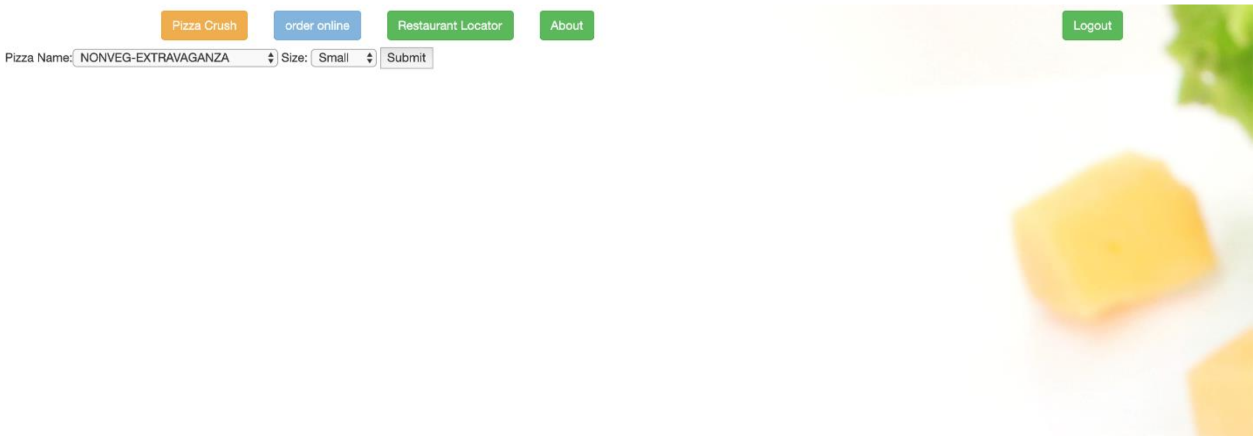


Enter the sauce name: Pesto Enter the price to be updated

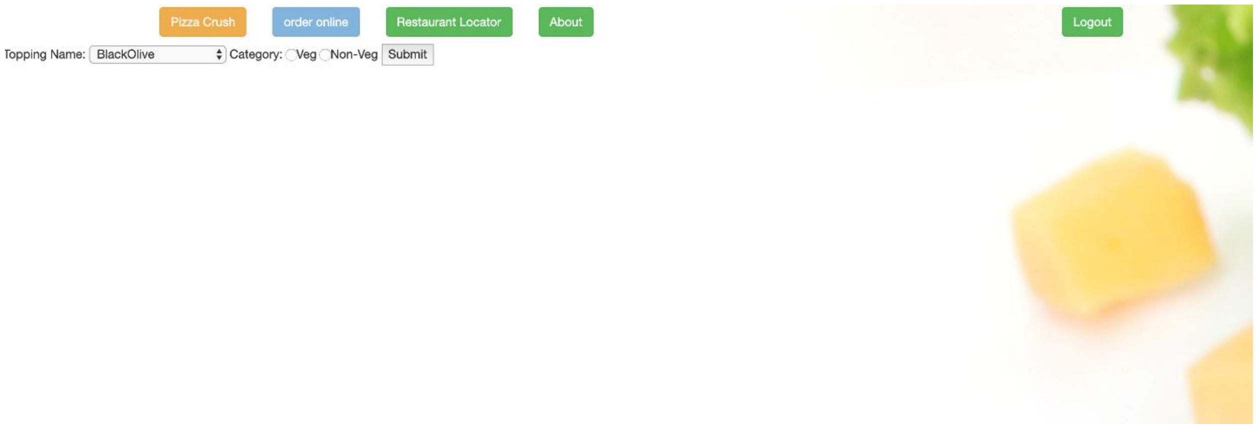


Enter the sauce name: Pesto Enter the price to be updated

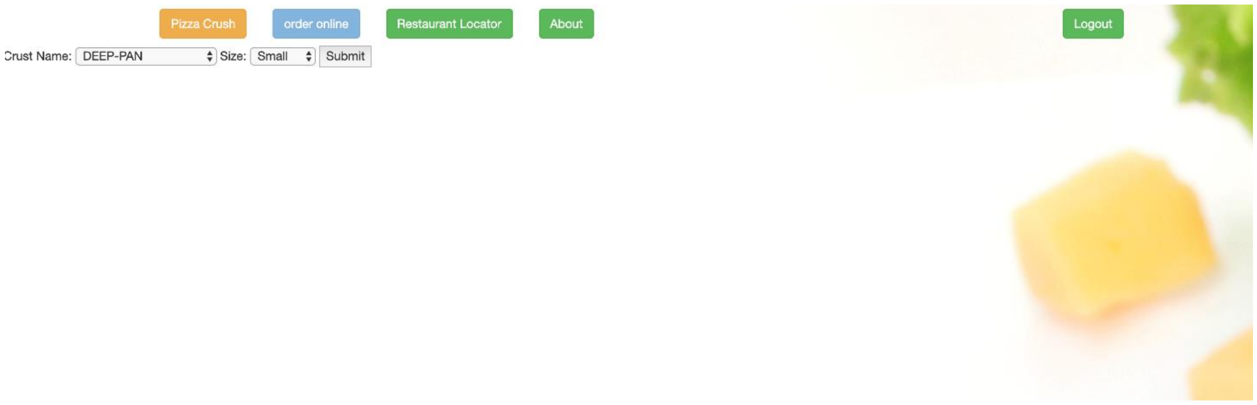
4.4. Delete page: This page is used by the admin to delete toppings, sauces, breads, store-info and prices.



Pizza Name: NONVEG-EXTRAVAGANZA Size: Small

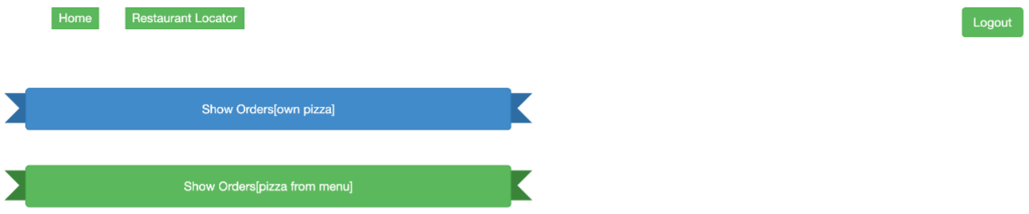


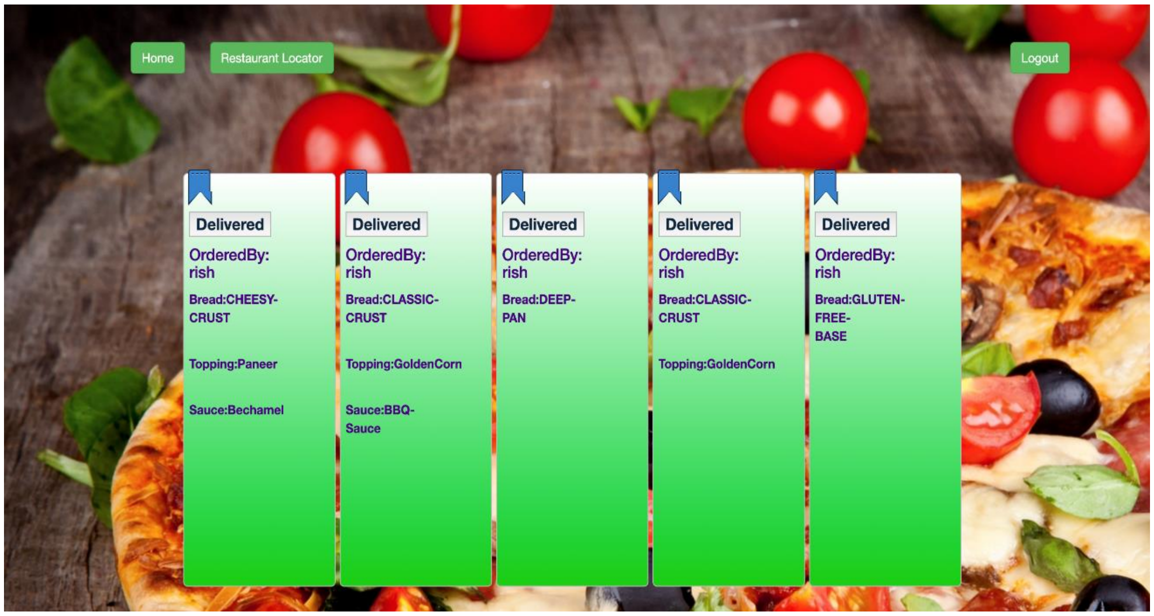
Topping Name: BlackOlive Category: Veg Non-Veg



Crust Name: DEEP-PAN Size: Small

4.5. Show orders: In this page the admin can see all the orders placed by the user and mark them as delivered.





Delivered
OrderedBy: rish
Bread:CHEESY-CRUST
Topping:Paneer
Sauce:Bechamel

Delivered
OrderedBy: rish
Bread:CLASSIC-CRUST
Topping:GoldenCorn
Sauce:BBQ-Sauce

Delivered
OrderedBy: rish
Bread:DEEP-PAN

Delivered
OrderedBy: rish
Bread:CLASSIC-CRUST
Topping:GoldenCorn

Delivered
OrderedBy: rish
Bread:GLUTEN-FREE-BASE

Section f:

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

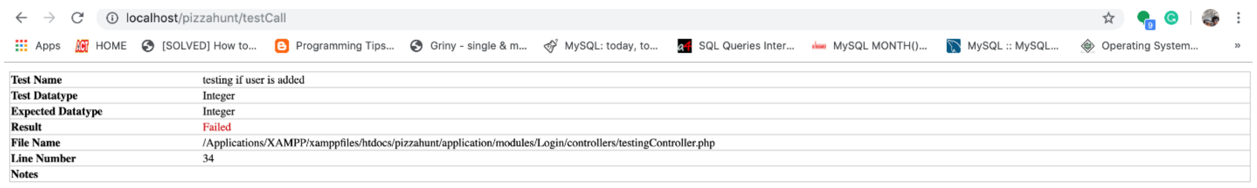
Password: hiiamrishi

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

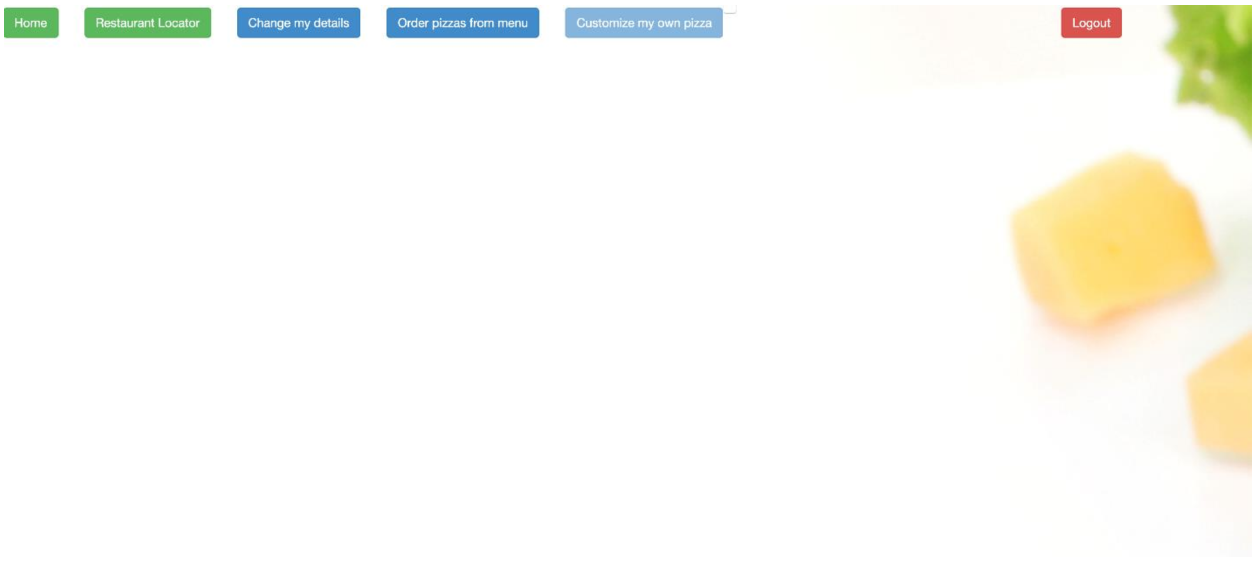
- Open browser type "localhost/pizzacrush/testmethod-name" to run the test cases. You will be able to see the output of test case.



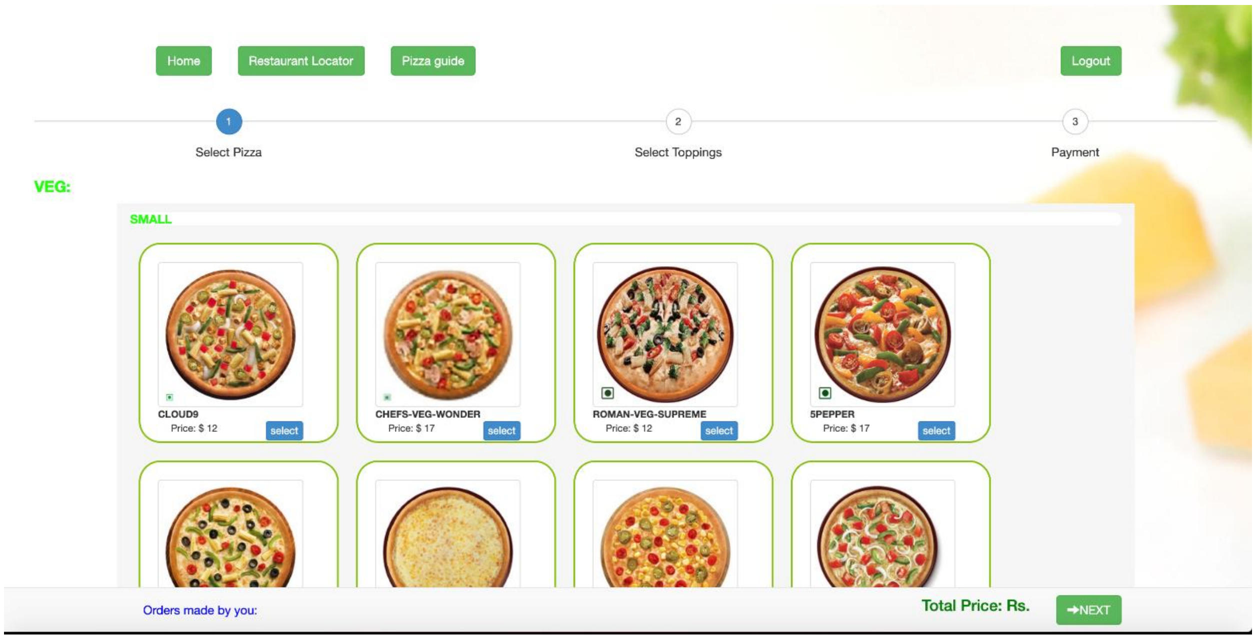
Test Name	testing if user is added
Test Datatype	Integer
Expected Datatype	Integer
Result	Failed
File Name	/Applications/XAMPP/xamppfiles/htdocs/pizzahunt/application/modules/Login/controllers/testingController.php
Line Number	34
Notes	

5. **Order your own pizza**

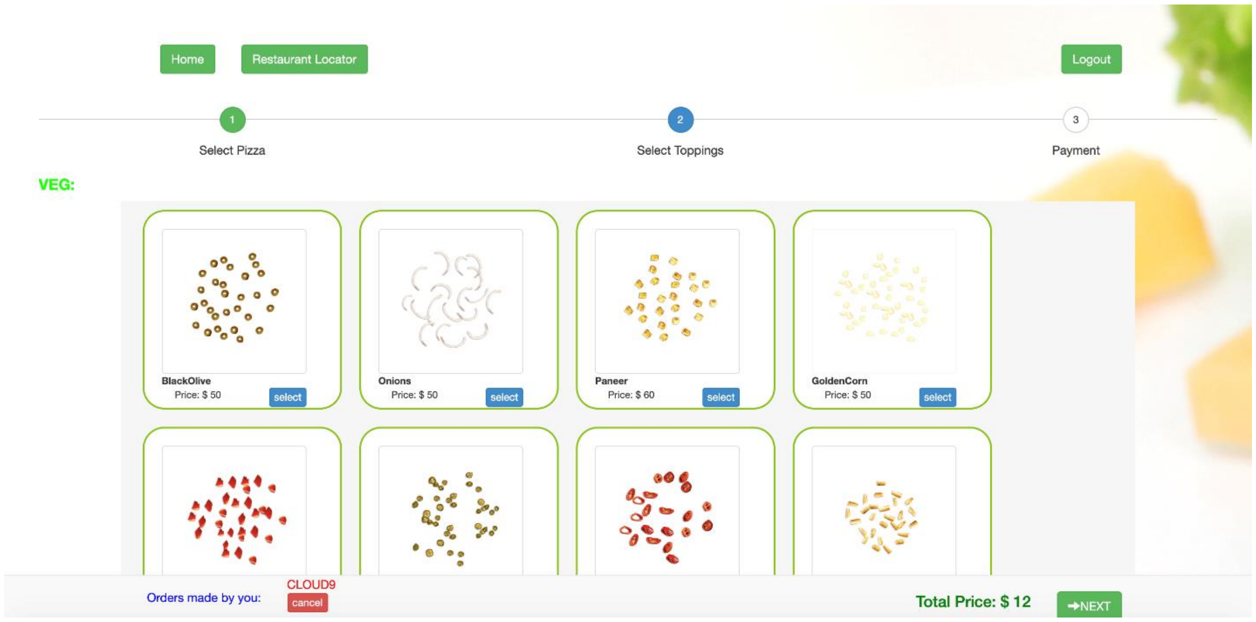
    5.1. Once the user login with his credentials, he will be redirected this page.



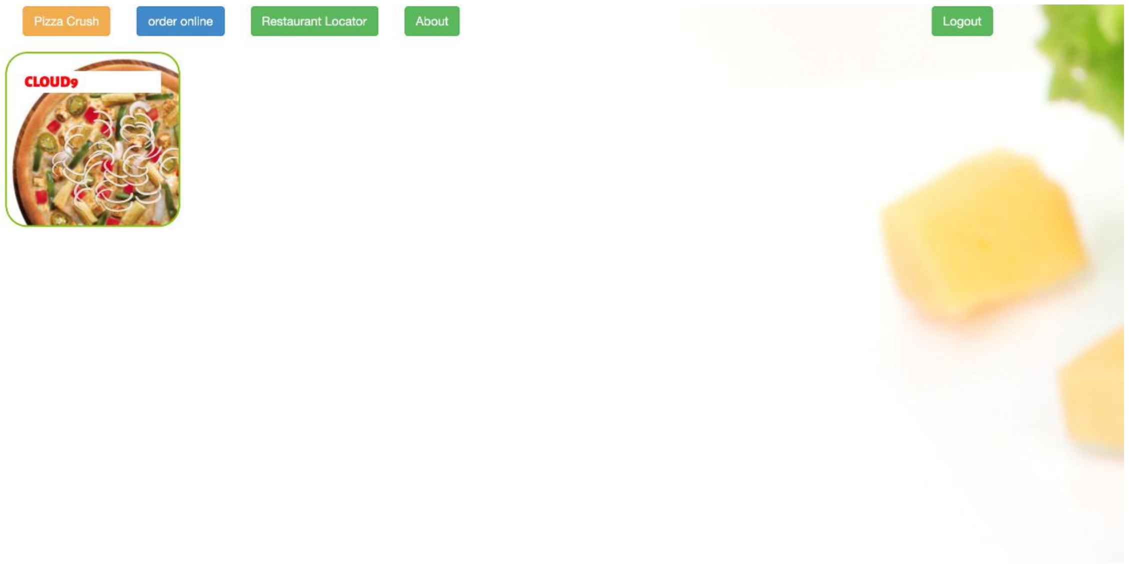
5.2. Select the tab Order pizza from menu, the user will be redirected to the next page which consist of all different kinds of pizzas.



5.3. Additional toppings can also be selected for the pizza from the available list of toppings.

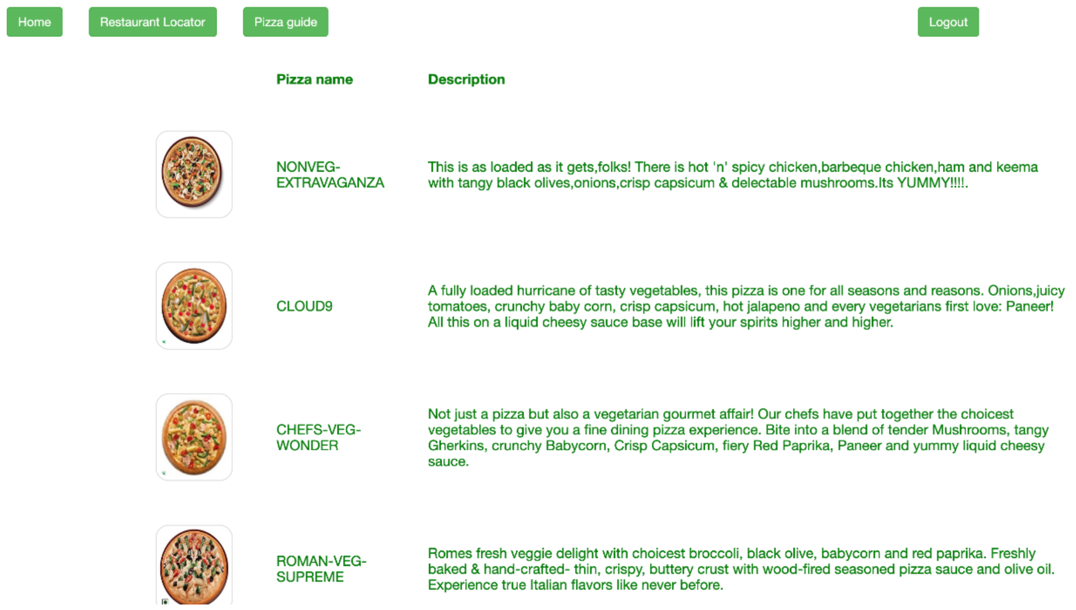


6. **Image Overlapping of toppings over pizza**
   6.1. The toppings selected will be overlapped with the selected pizza. The resultant pizza image is displayed.



7. **Pizza Guide**

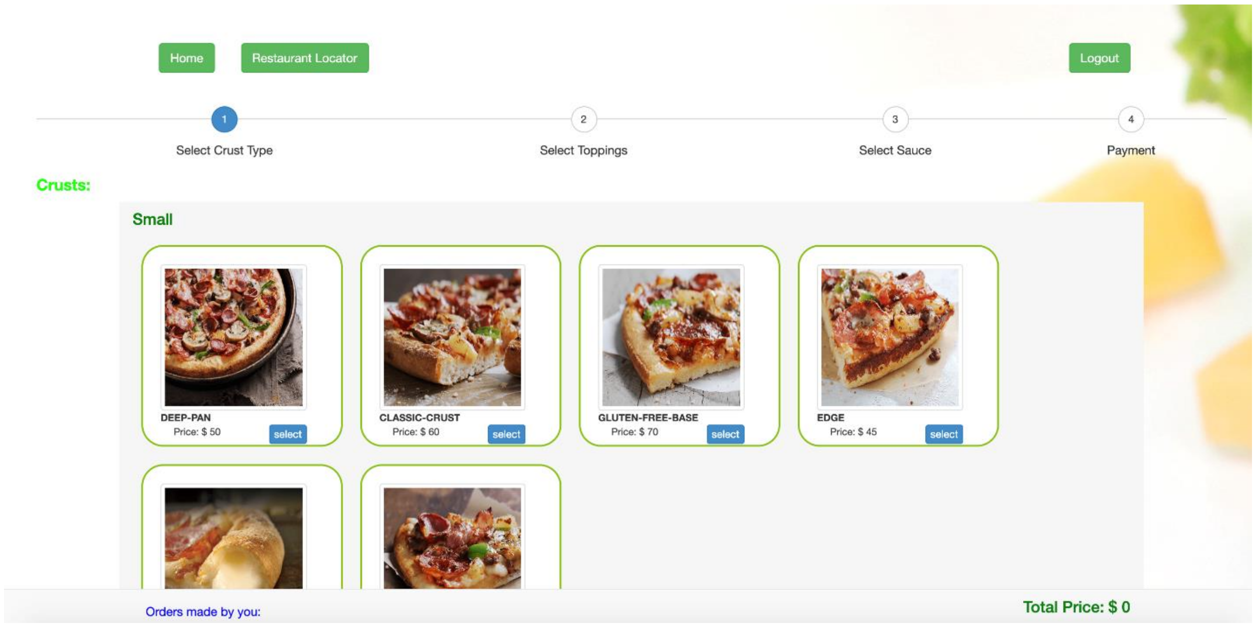
7.1. There is a tab "Pizza Guide", on selecting this you can see the details and description of all the pizzas which are sold.



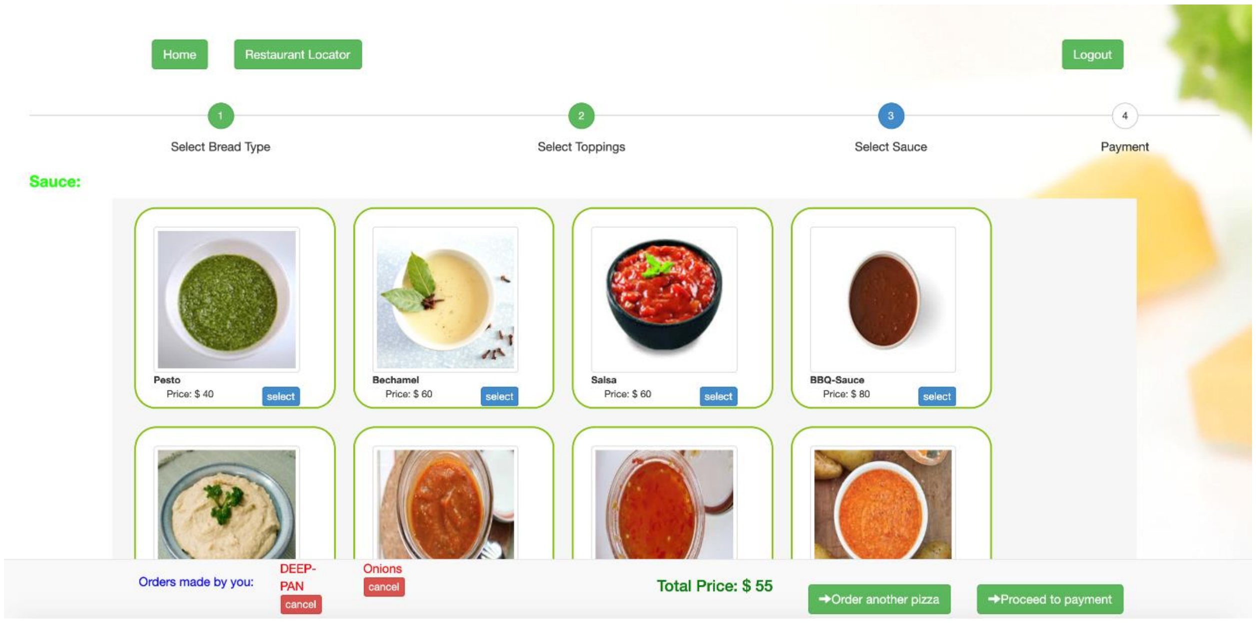
Pizza name	Description
NONVEG-EXTRAVAGANZA	This is as loaded as it gets,folks! There is hot 'n' spicy chicken,barbeque chicken,ham and keema with tangy black olives,onions,crisp capsicum & delectable mushrooms.Its YUMMY!!!!.
CLOUD9	A fully loaded hurricane of tasty vegetables, this pizza is one for all seasons and reasons. Onions,juicy tomatoes, crunchy baby corn, crisp capsicum, hot jalapeno and every vegetarians first love: Paneer! All this on a liquid cheesy sauce base will lift your spirits higher and higher.
CHEFS-VEG-WONDER	Not just a pizza but also a vegetarian gourmet affair! Our chefs have put together the choicest vegetables to give you a fine dining pizza experience. Bite into a blend of tender Mushrooms, tangy Gherkins, crunchy Babycorn, Crisp Capsicum, fiery Red Paprika, Paneer and yummy liquid cheesy sauce.
ROMAN-VEG-SUPREME	Romes fresh veggie delight with choicest broccoli, black olive, babycorn and red paprika. Freshly baked & hand-crafted- thin, crispy, buttery crust with wood-fired seasoned pizza sauce and olive oil. Experience true Italian flavors like never before.

8. Customize Pizza

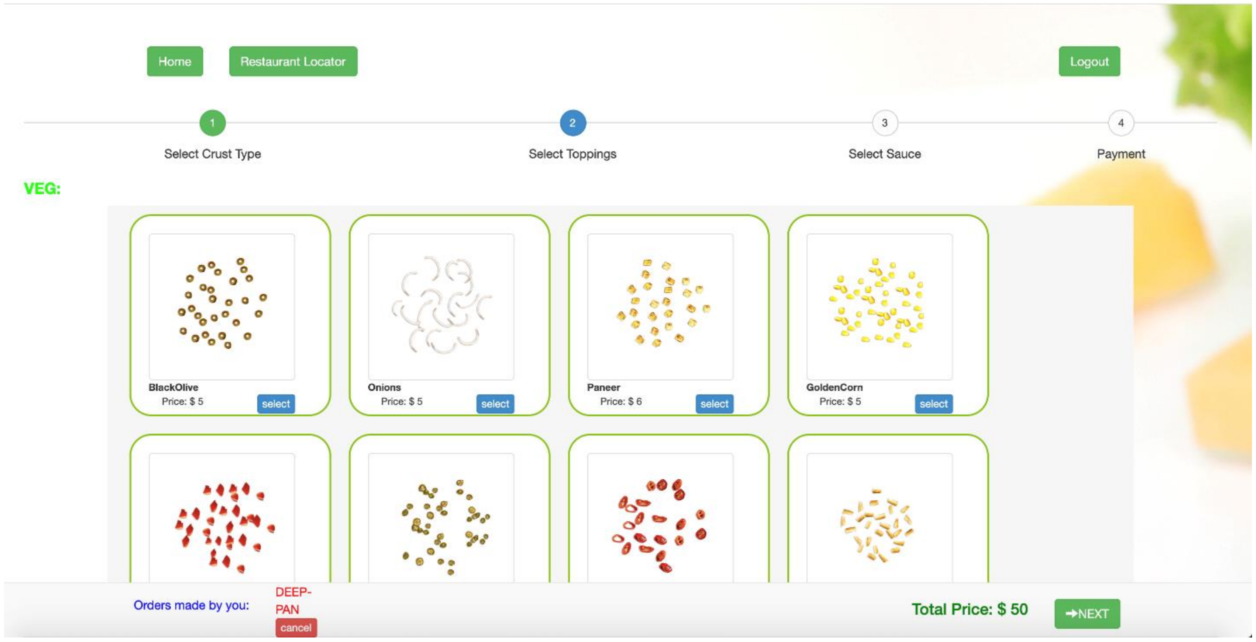
8.1. Choose crust: The user can choose his own crust



8.2. Choose Sauce: The user can choose his own sauce.

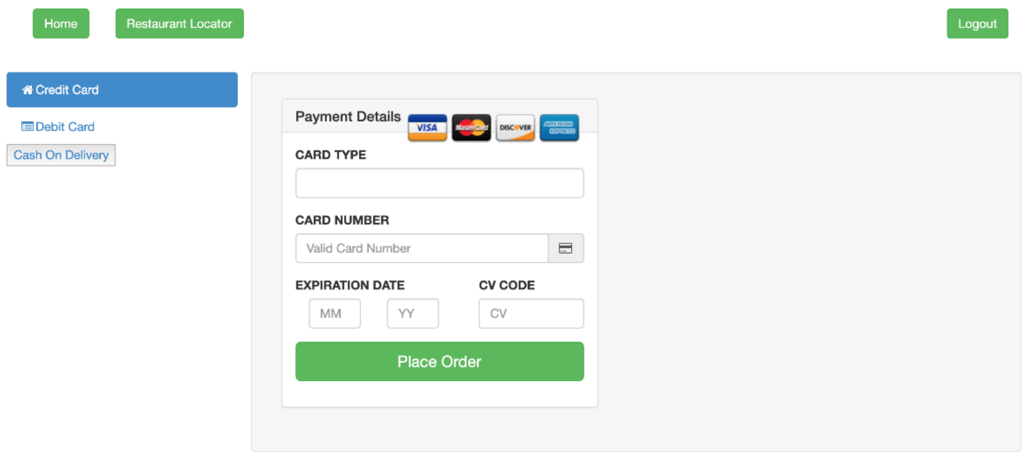


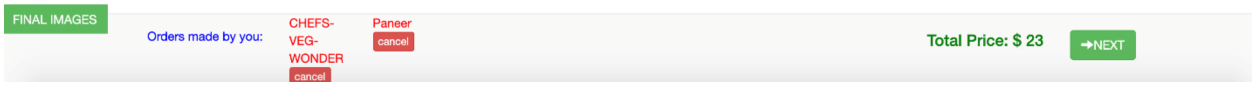
8.3. Choose toppings: The user can choose his desired toppings from the list of available toppings.



9. Dynamic Bill Generation

a. The price for the pizza is generated based on the pizza base price and also the additional toppings selected. Followed by the payment page.





b. Once the payment is done, the user is prompted with the "Thank You" message.

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

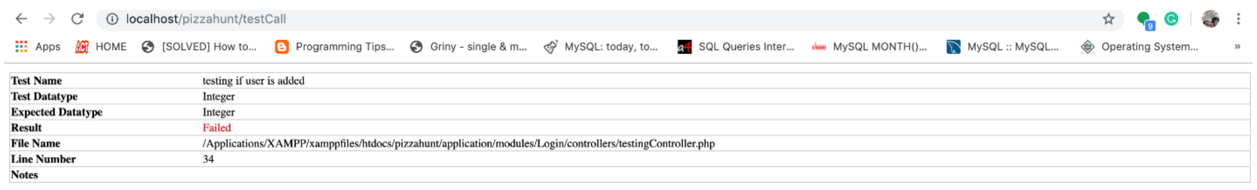
Password: hiiamrishi

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.Successful features and Peer review feedback

Successful features:

- Add, update and delete products
- Order pizza from menu
- Order customized pizza
- Image overlapping of toppings over pizzas
- Dynamic bill generation

Unsuccessful features:

- Email generation

Plans for next phase:

- Payment Gateway
- Email generation.

Feedback received during code inspection session:

Suggestion to add comments for the code, author tags missing and need to do code refactoring.

Changes/actions taken based on the feedback:

Based on the feedback that we received, we have added comments wherever necessary and mentioned about the authors.

8.Member Contribution Table

Member name	Contribution description	Overall Contribution (%)	Note(if applicable)
Sourab Reddy Pailla	Addbread.php Addpizza.php Addsauce.php Addtoppings.php Addbranch.php Orderonline.php Toppings_display.php Sauce_display.php Own_pizza_application.php Own_toppings_display.php User Manual: • Sequence diagram • User Manual	25	Group Leader and Developer
Gowtham Kesa	Deletebreadprice.php Deletepizzaprice.php Deletesauceprice.php Deletetoppingsprice.php Testingcontroller.php User Manual: • Class diagram • Test cases • Requirements • Peer feedback	25	Developer and Tester
Nagendra Beesabathuni	Updatebreadprice.php Updatepizzaprice.php Updatesauceprice.php Updatetoppingsprice.php Paid.php Readymade.php Ownorders.php Showorders.php Ownpizza.php User Manual: • Use case diagram • Testcases • Contribution tables	25	Scrum Master and Developer

	• Requirements		
Rishi Reddy Kolanu	About.php Afterlogin.php Changelocation.php Afteradmin.php Locator.php Login_view.php Header.php Signinup.php User Manual: • Requirements • Installation instructions	25	Architect and Developer