**INDEX**

Table of Contents

**Type chapter title (level 1)1**

Type chapter title (level 2)2

Type chapter title (level 3)3

**Type chapter title (level 1)4**

Type chapter title (level 2)5

Type chapter title (level 3)6

**Online Cake Delivery User Manual**

1. **Introduction**

The main objective of this application is to deliver cakes to any corner of the city, also there is no permanent staff. Since deliverer is volunteering by themselves and every function is automated.

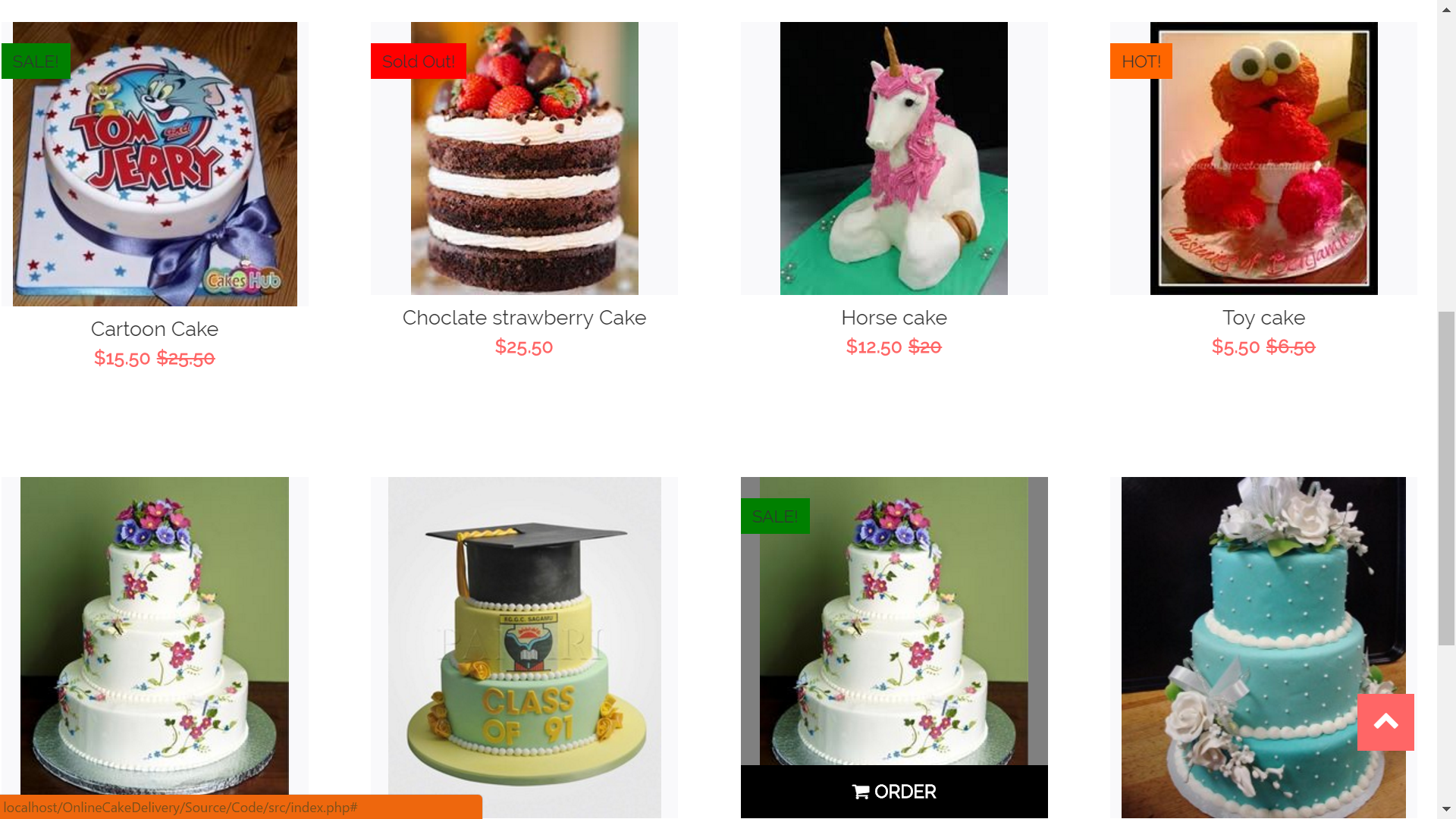
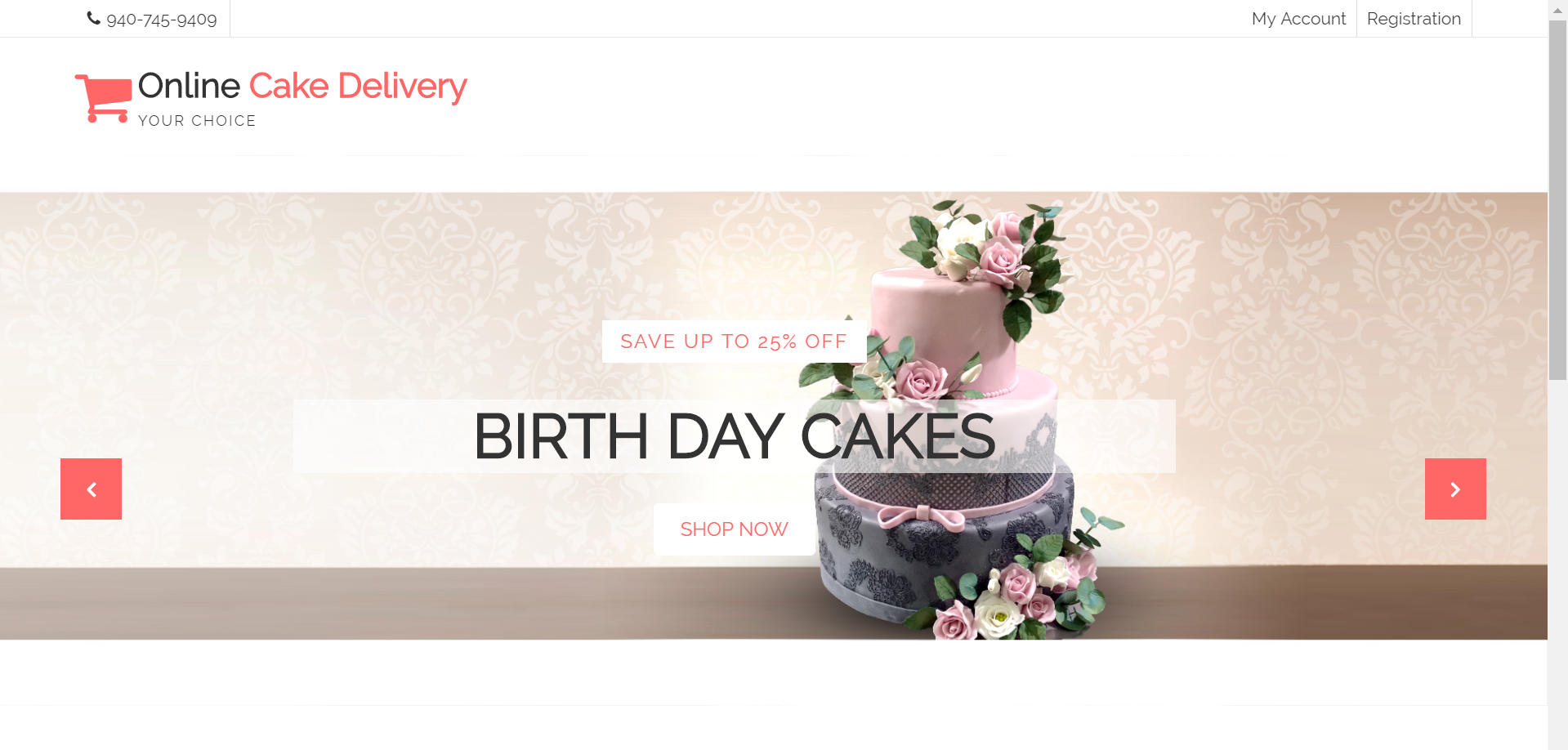
The proposed system will make the lives better by ordering cakes online and benefit the deliverers by providing money with simple tasks.

1. **Using the System**

The online cake delivery has three Users

1. Admin - Has features like upload cakes, view the feedback provided by customers.
2. Customer – Has features like browse available cakes, Place order and check status of the orders.
3. Deliverer – Has features like view placed orders and view orders deliverer should complete.
   1. ***Home Page***

The following is the screenshot of Online cake delivery home page.



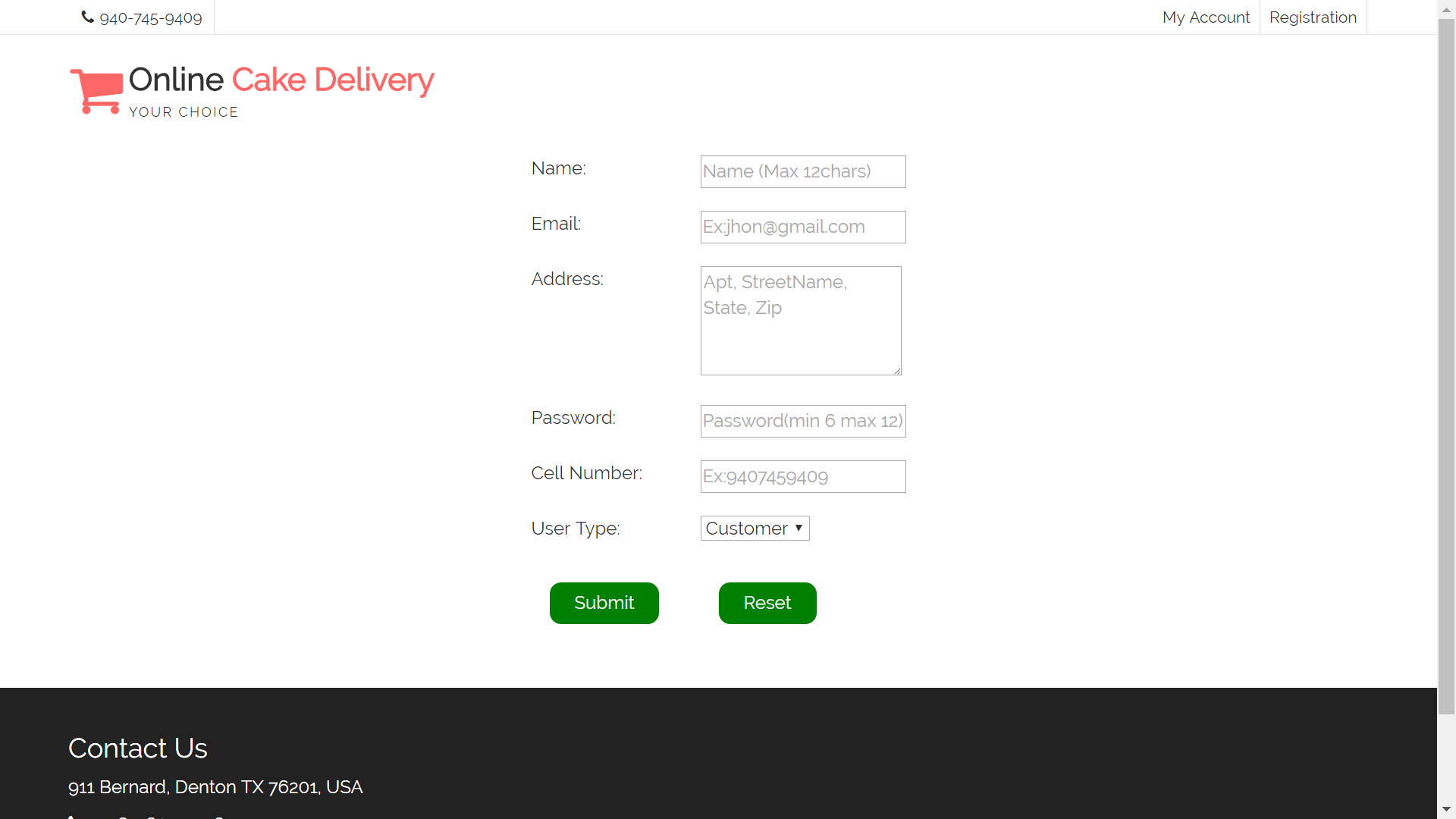
1. Home Page
2. The home page consists of Registration button so that users using which new Customers and Deliverers can register in the system.
3. It also contains (Login button) “My Account” so that the users such as Customers, Deliverers and Admin can login into the website.
4. The home page also display cakes for the users though they are not registered with the application.

* 1. **Registration Page:**

The following is the registration page for the application.

Customers and Deliverers must register by providing all the required data in required fields.

Screenshot:

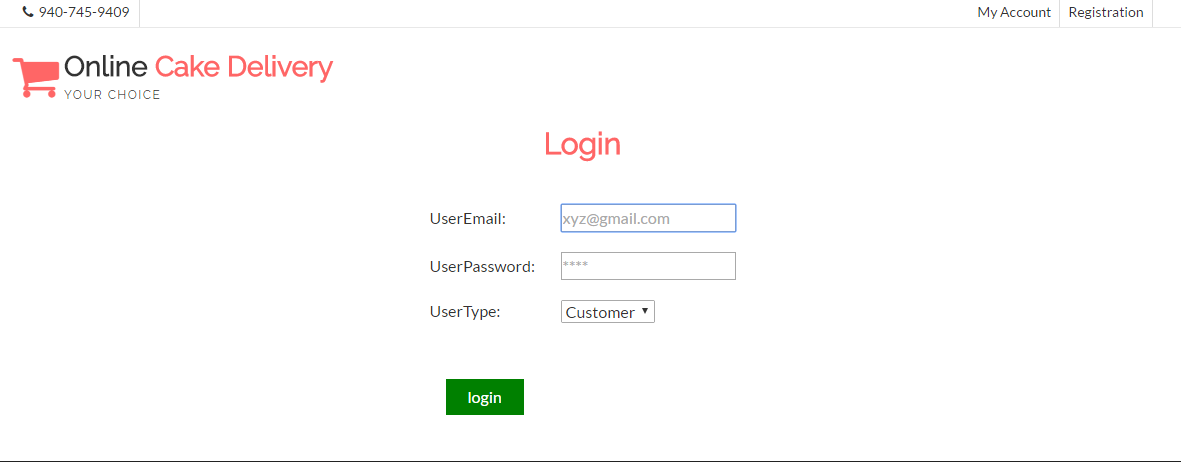


1. Registration Page
   1. **Login page:**

The following screenshot is the login page for the application.

All the users must provide valid credentials to log in into the system.

Screenshot:



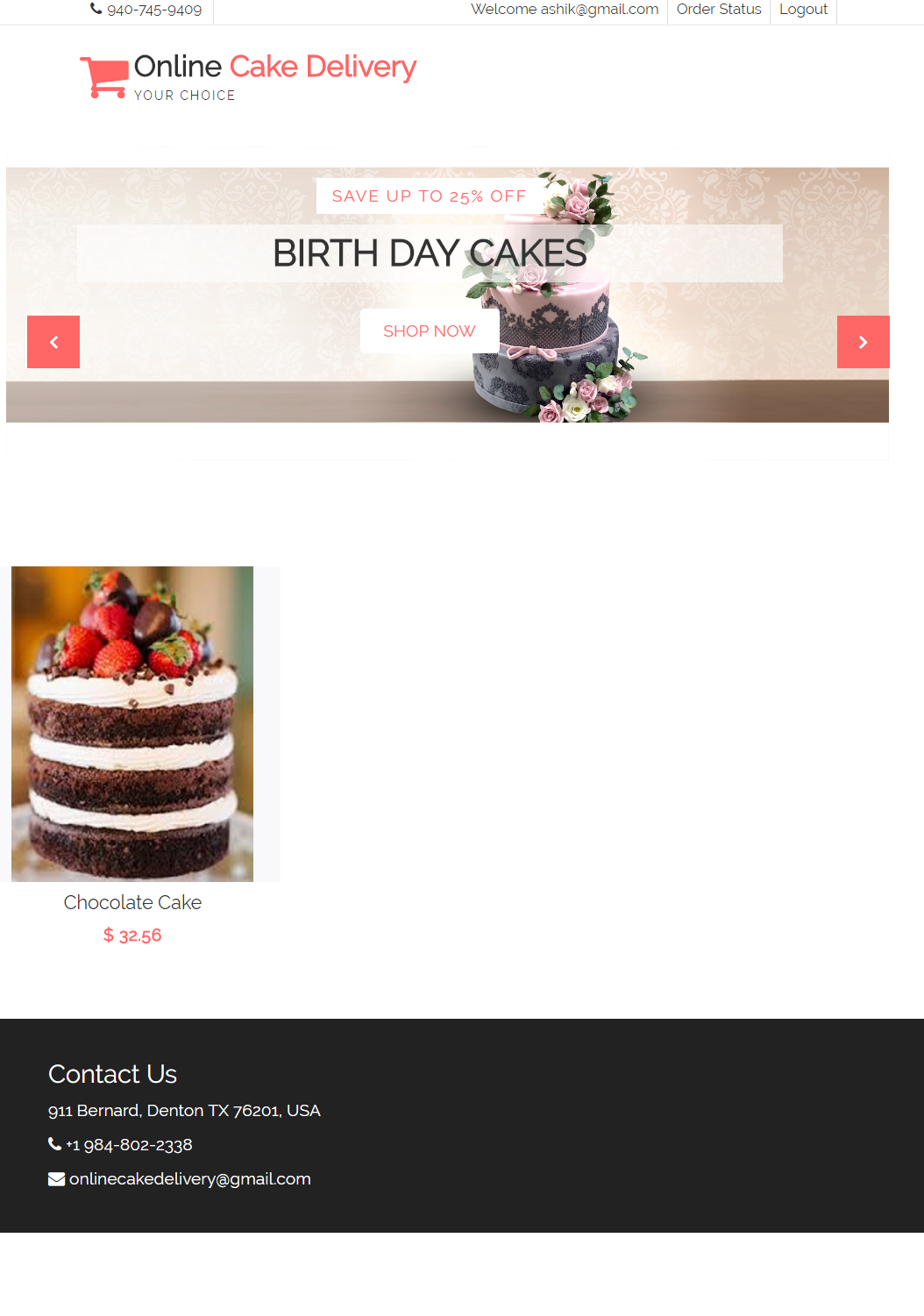
1. Login Page
   1. **Customer Account:**

If the user is registered as customer, then providing his credentials redirects the system to customer account page.

Once the customer is logged in, cakes available for delivery are displayed with its details and order button on hover above every cake.

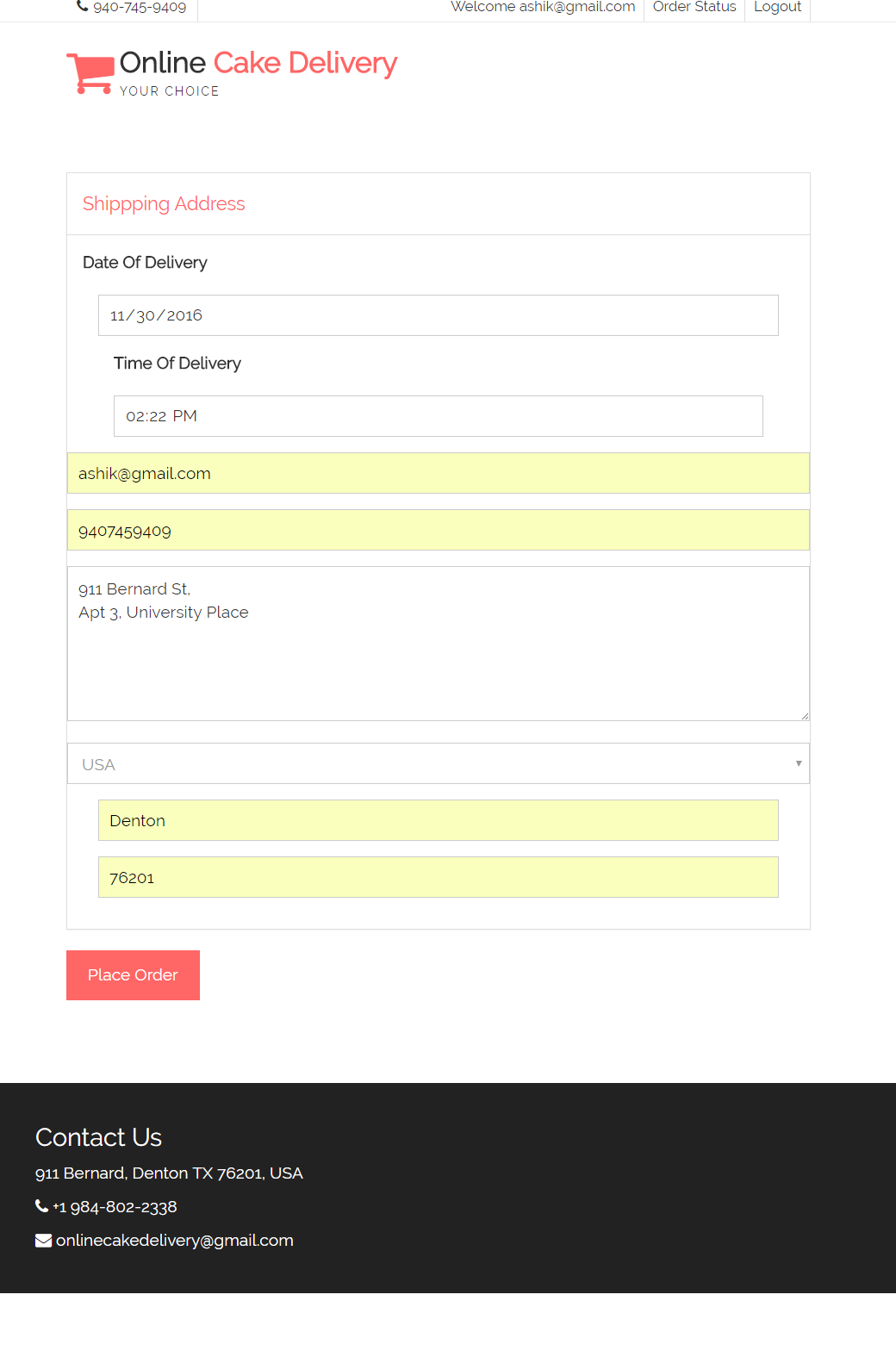
Customer can browse for the cakes he wishes to buy and click on order button to buy it.

The following is the screenshot of customer account page.



1. Customer Home Page.
   * 1. **Check out page:**
2. Clicking on order button would re-direct to check out page where user should fill the details needed to ship the order.
   1. Date of delivery
   2. Time of delivery
   3. Email, phone and address of the customer receiving the delivery should be filled.
3. All the fields are required in this page.

The following is the screenshot of checkout page.

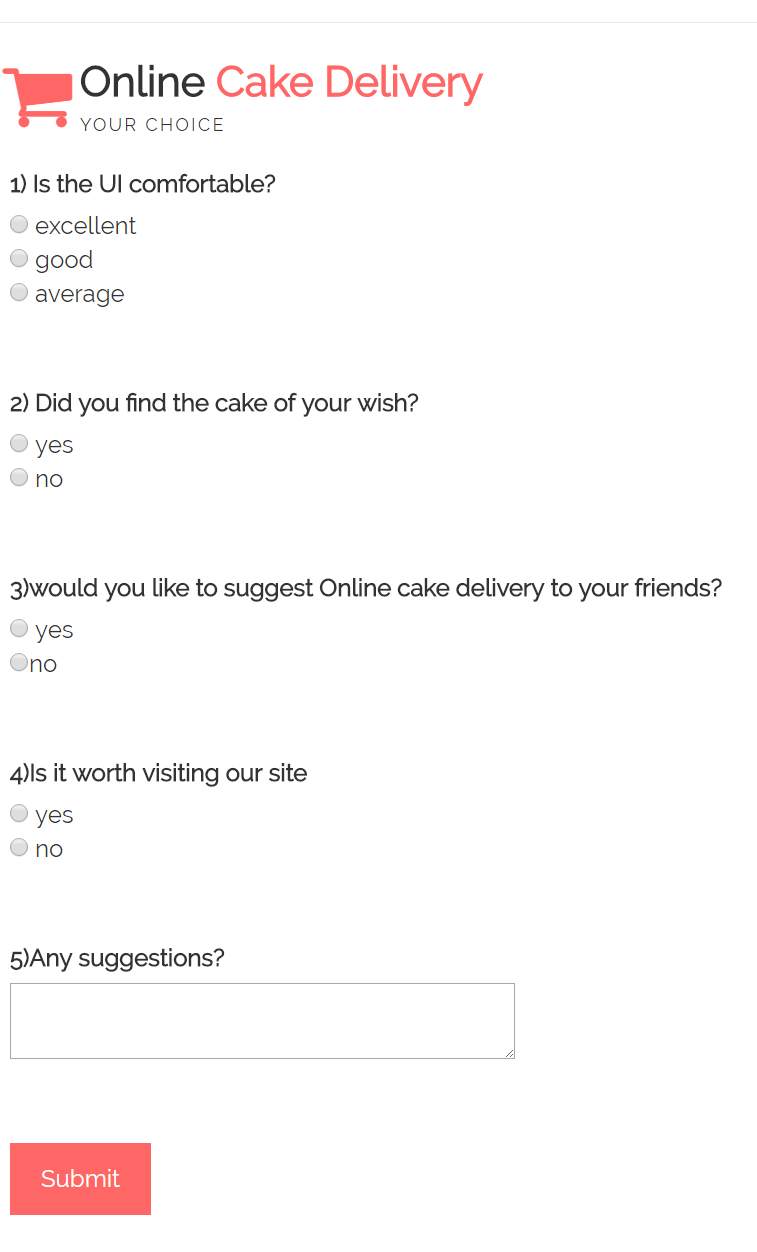


1. Shipping Details Screen
   * 1. **Feedback page:**

By clicking on place order, on successful validation the order is placed and the user is then re-directed to feedback page.

* Feedback is anonymous and no one knows which user has submitted feedback.

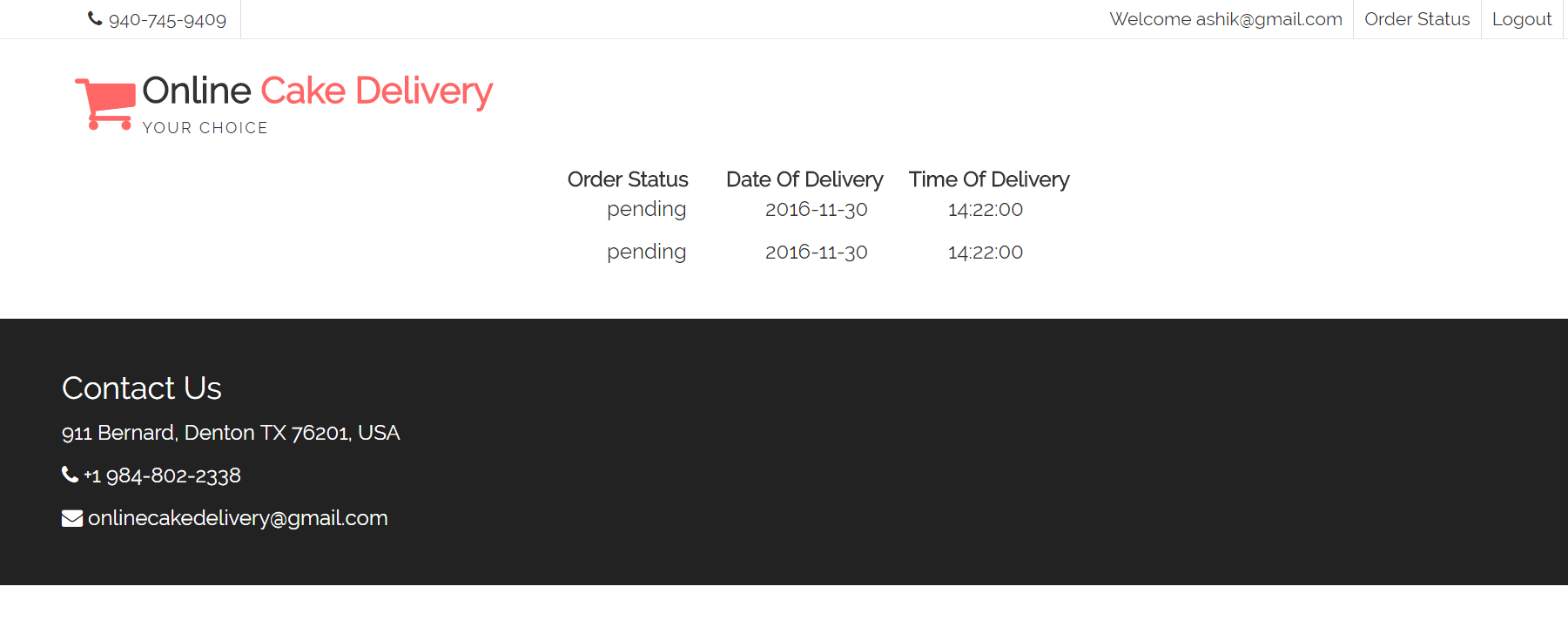
The following is the feedback page screenshot.



1. Feedback Screen
   * 1. **Order Status page:**

The Customer can then check the order status by clicking on the order status button on the top.

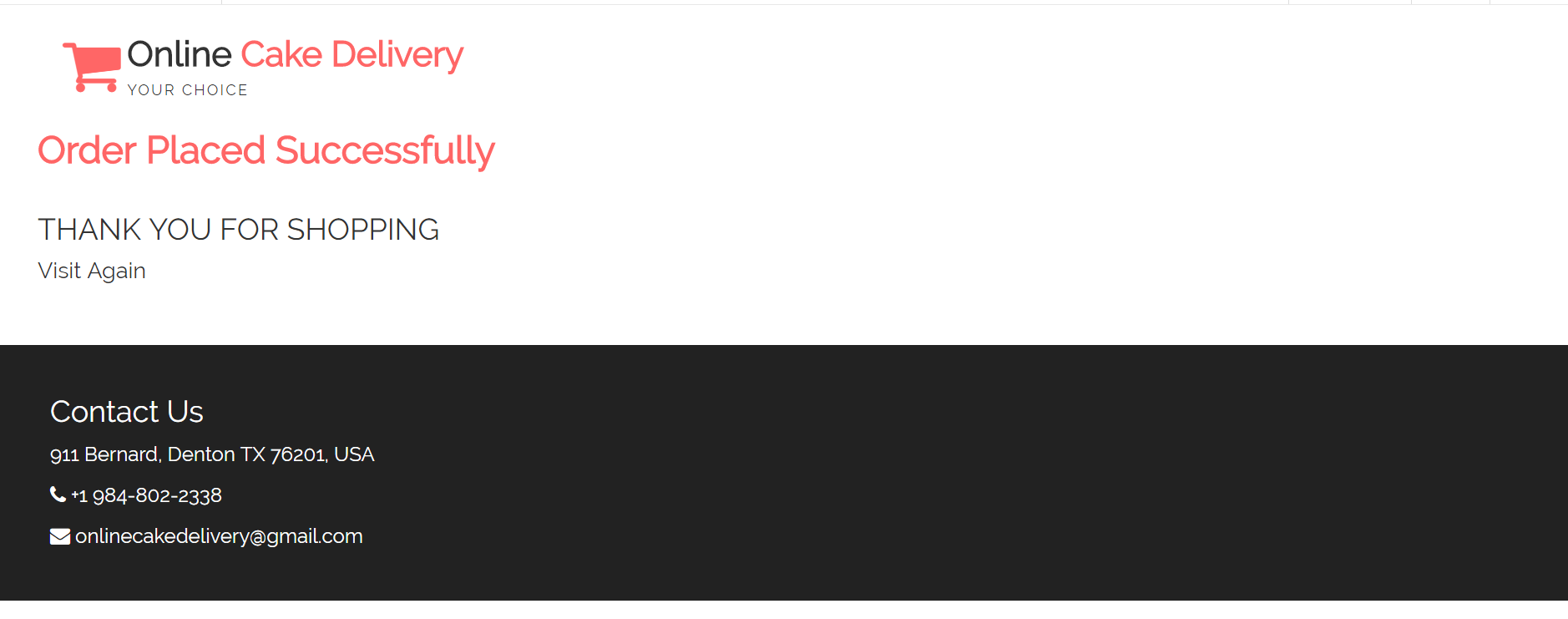
It contains the details about the order status, date of delivery and time of delivery.



1. Order Status Page
   * 1. **Success page:**

After submitting the feedback success page is displayed with thank you note and link to visit again.

The following is the screenshot for success page.

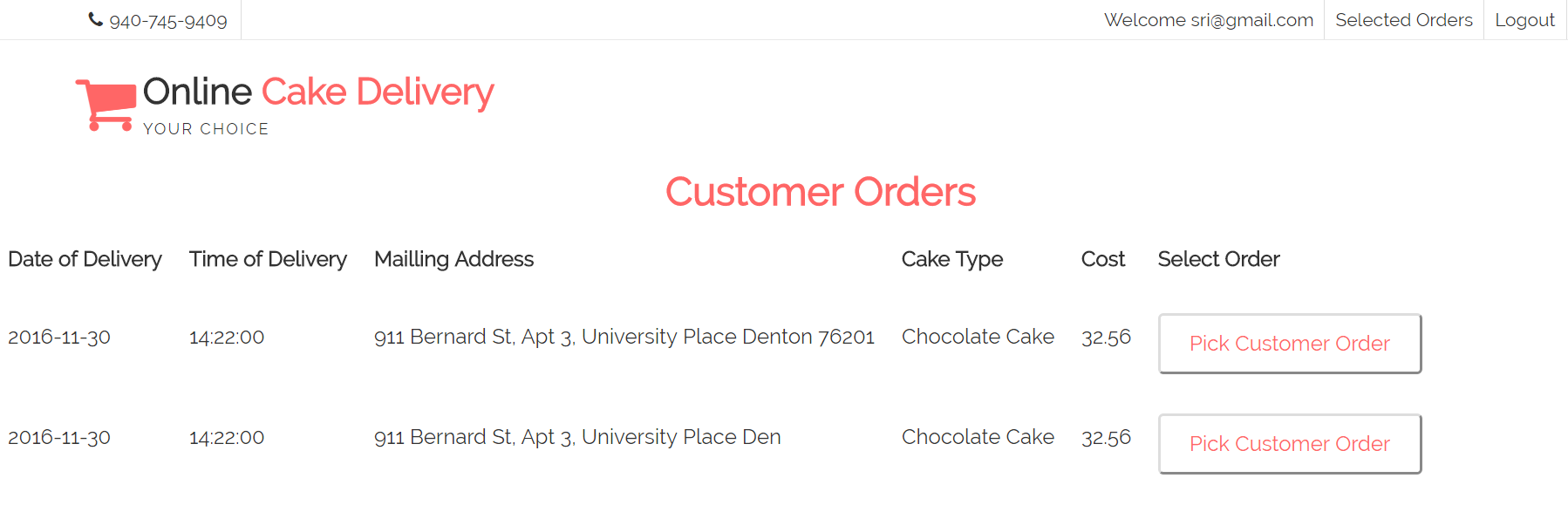


1. Success Page.
   1. **Deliverer Account:**

If a user register’s as deliverer, then providing his credentials redirects the system to deliverer account page.

It has all the unselected orders with details.

The following is the screenshot of deliverer account.



1. Customer Orders

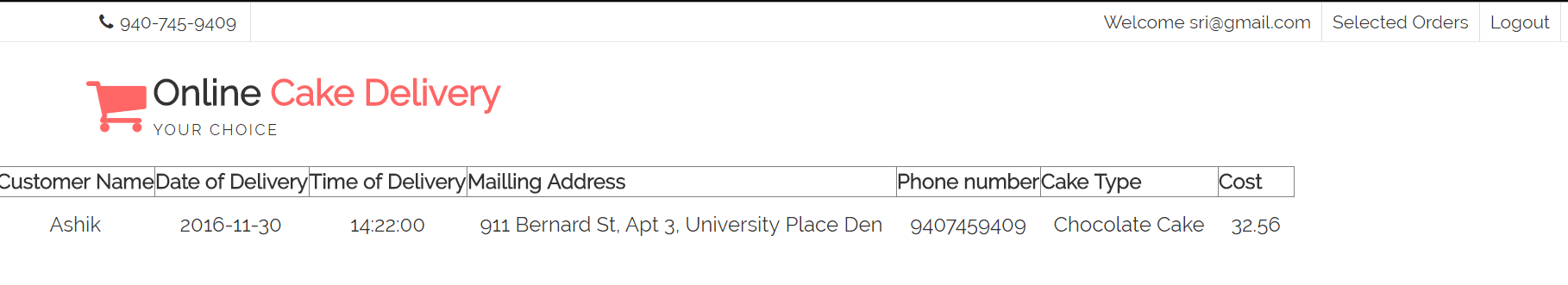
**2.5.1 Selected Order Page:**

If the user picks an order, then it is redirected to this page where all the previously selected orders are displayed with currently picked order.

Can also navigate to this page by clicking on the selected orders button.

It also displays the address where he must deliver the cake.

The following is the screenshot of deliverer selected order page.



1. Selected Order

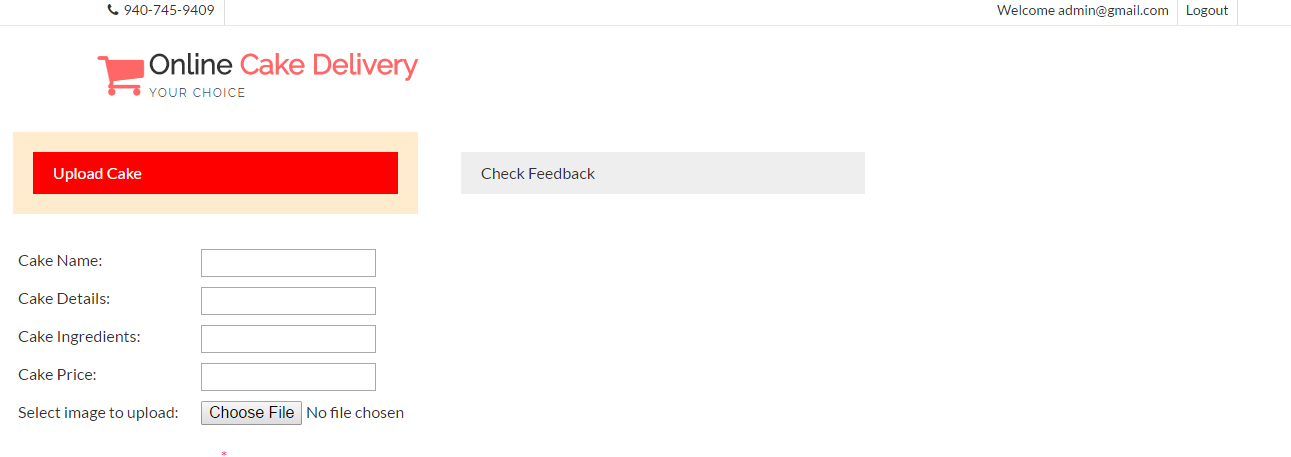
* 1. **Admin Account:**

If the user is registered as admin then providing his credentials redirects the system to admin account page.

**2.6.1 Adding cakes**

Admin can add cakes into the databases so that they can be viewed by customers to buy.

The following is the screenshot for adding cakes.

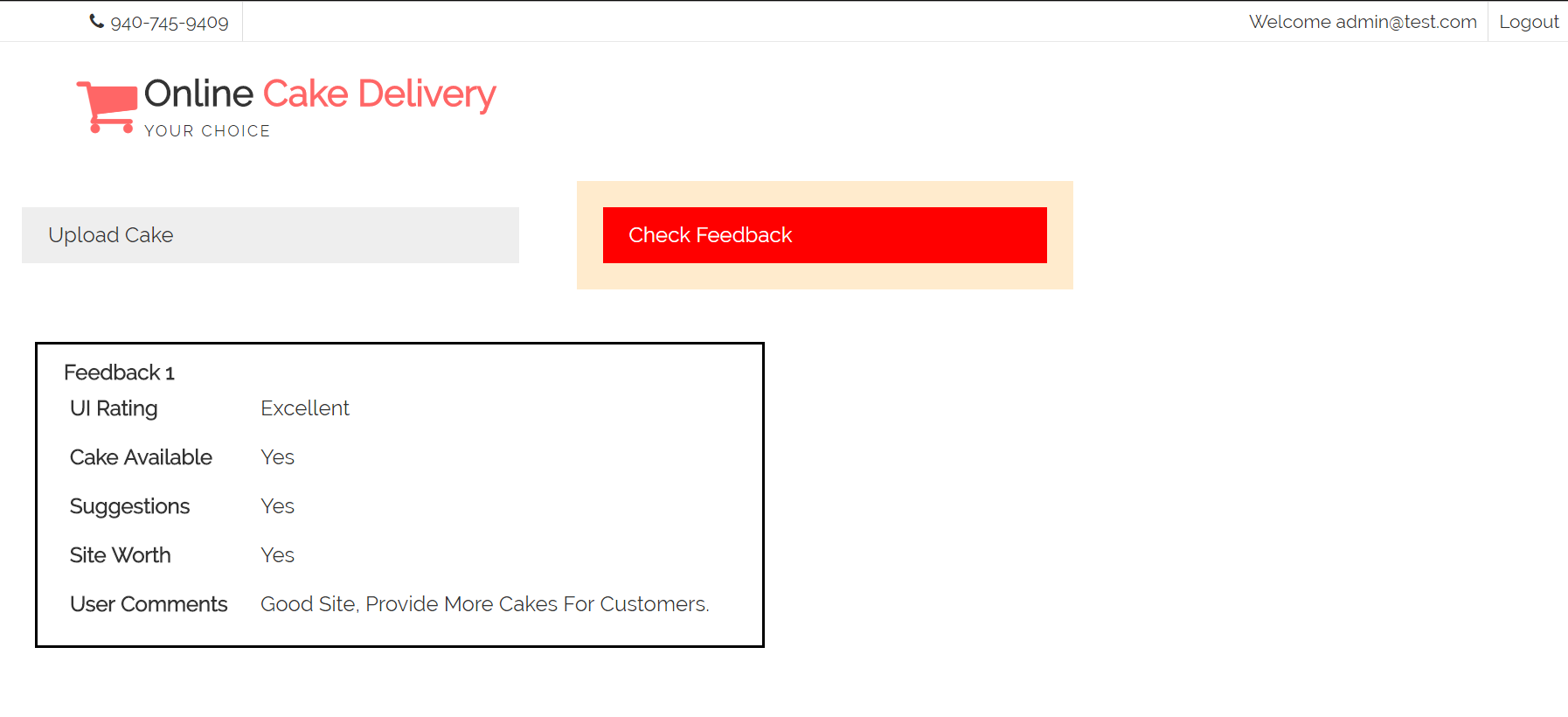


1. Admin Adding Cake.

**2.6.2 View Feedback**

Admin can view the feedback given by the customers by clicking on the feedback tab.

The following is the screenshot for viewing feedback.



1. Feedback Page.
2. **Instructions**

This Section explains how to run and compile the program and its test cases

**3.1 Instructions to run Program**

We need a platform of PHP and MySQL and any browser to run the project.

Step1:

Install Xampp and open Xampp control panel.

Step2:

Then click on start for Apache and MySQL to run it.

Step3:

Go to Localhost:82/phpmyadmin in Browser. Here 82 is my port number similarly you have to give your port number .Generally it will be 80 for all systems unless if there is any conflict with port number.

Step4:

Click on “New” to create a new database named “onlinecakedelivery”.

Step5:

Click on import button and browse the system folders for “OnlineCakeDelivery\Source\Code\database\_scripts/onlinecakedelivery.sql” file then click on go to run the script file of database.

Step6:

Now go to the path where Xampp is installed and place project directory in ‘htdocs’ folder.

Step7:

Open browser and type localhost/OnlineCakeDelivery/Source/Code/src/index.php. This will display home page of the application.

Now you should be able to perform the tasks as in the user manual.

* 1. **Instructions to run Test cases:**

Step1:

Open Xampp control panel and start apache and MySQL

Step2:

In project directory got to Source/code/test for the test cases.

Step3:

Open command prompt and type the command cd C:\xampp\htdocs\OnlineCakeDelivery\Source\Code\test

Now type: phpunit <space> filename.php This will run the test case.

**4.Test Cases and problems encountered**

This sections has test cases and problems encountered during the testing phase

**4.1TestCases:**

RegistrationTests.php

<?php

require '../src/Registration.php';

//Insert query should be executed before running these test cases

class RegistrationTests extends PHPUnit\_Framework\_TestCase

{

/\*Testing whether given username and username in database is same or not.

In this testcase we are giving username as "test1" and sending the value as "test" so that this testcase would fails \*/

public function test\_uname1()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "test1",

"email" => "test1@gmail.com",

"address" => "ss",

"pwd" => "Sri\_1993",

"mobile" => "0987654321",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

$this->assertEquals("test",$a->getUname());

}

/\*Testing whether given username and username in database is same or not.

In this testcase we are giving username as "" and sending the value as "test" so that this testcase would fails \*/

public function test\_uname2()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "",

"email" => "test1@gmail.com",

"address" => "ss",

"pwd" => "Sri\_1993",

"mobile" => "0987654321",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

$this->assertEquals("test",$a->getUname());

}

/\*Testing whether given username and username in database is same or not.

In this testcase we are giving username as "test" and sending the value as "test" so that this testcase would pass \*/

public function test\_uname3()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "test",

"email" => "test1@gmail.com",

"address" => "ss",

"pwd" => "Sri\_1993",

"mobile" => "0987654321",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

$this->assertEquals("test",$a->getUname());

}

/\*Testing whether given emailid and emailid in database is same or not.

In this testcase we are giving emailid as "" and sending the value as "test@gmail.com" so that this testcase would fails \*/

public function test\_email1()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "test",

"email" => "",

"address" => "ss",

"pwd" => "Sri\_1993",

"mobile" => "0987654321",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

$this->assertEquals("test@gmail.com",$a->getEmail());

}

/\*Testing whether given emailid and emailid in database is same or not.

In this testcase we are giving emailid as "test1@gmail.com" and sending the value as "test@gmail.com" so that this testcase would fails \*/

public function test\_email2()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "test",

"email" => "test1@gmail.com",

"address" => "ss",

"pwd" => "Sri\_1993",

"mobile" => "0987654321",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

$this->assertEquals("test@gmail.com",$a->getEmail());

}

/\*Testing whether given emailid and emailid in database is same or not.

In this testcase we are giving emailid as "test@gmail.com" and sending the value as "test@gmail.com" so that this testcase would pass \*/

public function test\_email3()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "test",

"email" => "test@gmail.com",

"address" => "ss",

"pwd" => "Sri\_1993",

"mobile" => "0987654321",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

$this->assertEquals("test@gmail.com",$a->getEmail());

}

/\*Testing whether given address and address in database is same or not.

In this testcase we are giving address as "Bernard street" and sending the value as "Bernard street" so that this testcase would pass \*/

public function test\_address()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "test",

"email" => "test1@gmail.com",

"address" => "Bernard street",

"pwd" => "Sri\_1993",

"mobile" => "0987654321",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

$this->assertEquals("Bernard street",$a->getAddress());

}

/\*Testing whether given password and password in database is same or not.

In this testcase we are giving password as "" and sending the value as "Sri\_1993" so that this testcase would fails \*/

public function test\_password1()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "test",

"email" => "test1@gmail.com",

"address" => "Bernard street",

"pwd" => "",

"mobile" => "0987654321",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

$this->assertEquals("Sri\_1993",$a->getPassword());

}

/\*Testing whether given password and password in database is same or not.

In this testcase we are giving password as "Sri1993" and sending the value as "Sri\_1993" so that this testcase would fails \*/

public function test\_password2()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "test",

"email" => "test1@gmail.com",

"address" => "Bernard street",

"pwd" => "Sri1993",

"mobile" => "0987654321",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

$this->assertEquals("Sri\_1993",$a->getPassword());

}

/\*Testing whether given password and password in database is same or not.

In this testcase we are giving password as "Sri\_1993" and sending the value as "Sri\_1993" so that this testcase would pass \*/

public function test\_password3()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "test",

"email" => "test1@gmail.com",

"address" => "Bernard street",

"pwd" => "Sri\_1993",

"mobile" => "0987654321",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

$this->assertEquals("Sri\_1993",$a->getPassword());

}

/\*Testing whether given phone number and phone number in database is same or not.

In this testcase we are giving phone number as "" and sending the value as "0987654321" so that this testcase would fails \*/

public function test\_mobilenum1()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "test",

"email" => "test1@gmail.com",

"address" => "Bernard street",

"pwd" => "Sri\_1993",

"mobile" => "",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

$this->assertEquals("0987654321",$a->getMobileNumber());

}

/\*Testing whether given phone number and phone number in database is same or not.

In this testcase we are giving phone number as "1987651452" and sending the value as "0987654321" so that this testcase would fails \*/

public function test\_mobilenum2()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "test",

"email" => "test1@gmail.com",

"address" => "Bernard street",

"pwd" => "Sri\_1993",

"mobile" => "1987651452",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

$this->assertEquals("0987654321",$a->getMobileNumber());

}

/\*Testing whether given phone number and phone number in database is same or not.

In this testcase we are giving phone number as "0987654321" and sending the value as "0987654321" so that this testcase would pass \*/

public function test\_mobilenum3()

{

// Arrange

$a = new Registration();

$testValue = Array (

"uname" => "test",

"email" => "test1@gmail.com",

"address" => "Bernard street",

"pwd" => "Sri\_1993",

"mobile" => "0987654321",

"type" => "customer" );

//Set the form details.

$a->setFormDetails($testValue);

//execute the query.

$a->executeInsertQuery();

//assert if the errormessage is empty or not.

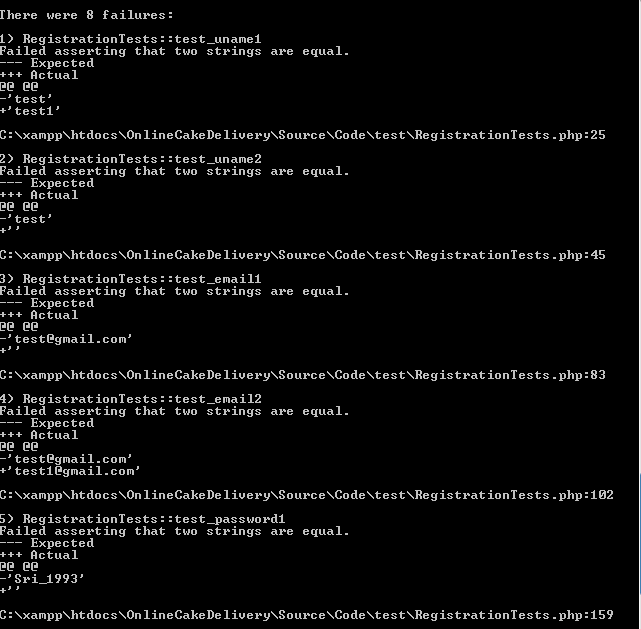
$this->assertEquals("0987654321",$a->getMobileNumber());

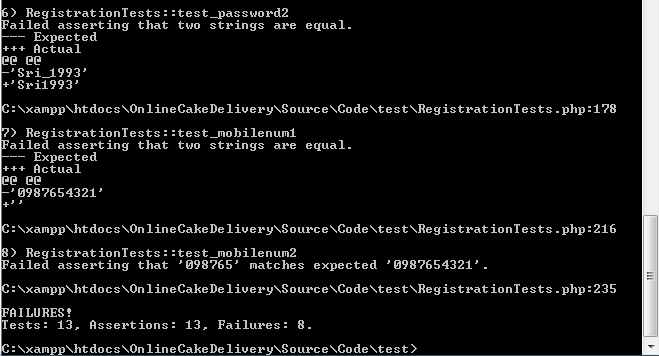
}

}

?>

Screenshot





LoginTest.php:

<?php

require '../src/login.php';

class LoginTest extends PHPUnit\_Framework\_TestCase

{

//checking whether admin is redirecting to correct page or not

public function testusertype1()

{

$obj= new Login();

$usertype="admin";

$this->assertEquals("admin\_features.php",$obj->redirectBrowser($usertype));

}

//checking whether customer is redirecting to correct page or not

public function testusertype2()

{

$obj= new Login();

$usertype="customer";

$this->assertEquals("index\_Customer\_logged.php",$obj->redirectBrowser($usertype));

}

//checking whether deliverer is redirecting to correct page or no

public function testusertype3()

{

$obj= new Login();

$usertype="deliverer";

$this->assertEquals("deliverer\_customerOrders.php",$obj->redirectBrowser($usertype));

}

//checking whether admin login credentials are retreiving from login\_admin table or not

public function test\_admintable()

{

$obj= new Login();

$usertype="admin";

$this->assertEquals("login\_admin",$obj->getTableName($usertype));

}

//checking whether customer login credentials are retreiving from login\_customer table or not

public function testcustomertable()

{

$obj= new Login();

$usertype="customer";

$this->assertEquals("login\_customer",$obj->getTableName($usertype));

}

//checking whether deliverer login credentials are retreiving from login\_deliverertable or not

public function testadmintable()

{

$obj= new Login();

$usertype="deliverer";

$this->assertEquals("login\_deliverer",$obj->getTableName($usertype));

}

}



DelivererTest.php:

<?php

require '../src/deliverer\_customerOrders.php';

class DelivererTest extends PHPUnit\_Framework\_TestCase

{

// checking whether pending orders are retreiving or not

public function testorders()

{

$obj= new Customer\_Orders();

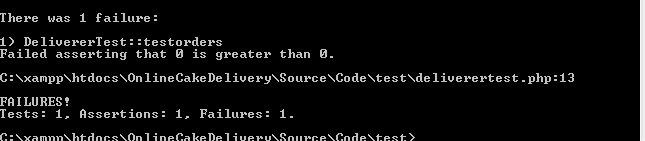
$connection\_object= $obj -> connectToDatabase();

$result= $obj -> pending\_Orders($connection\_object);

$this->assertGreaterThan(0,$result -> num\_rows);

}

}



DelivererOrdersTest:

<?php

require '../src/deliverer\_selectedOrder.php';

class DelivererOrdersTest extends PHPUnit\_Framework\_TestCase

{

// checking whether pending orders are retreiving or not

public function testdelivererorders()

{

$obj= new SelectedOrder();

$conn = $obj->connectToDatabase();

$deliverer\_id=3;

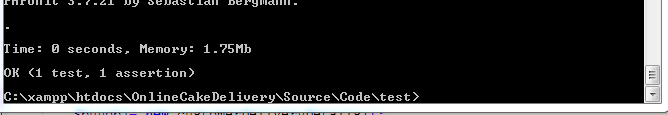
$result= $obj -> confirmedOrderResults($deliverer\_id,$conn);

$this->assertGreaterThan(0,$result -> num\_rows);

}

}

?>



CheckoutTest:

<?php

require '../src/buy.php';

class CheckoutTest extends PHPUnit\_Framework\_TestCase

{

// this test case should fail because email id is invalid

public function testEmailID1()

{

$buyobj= new customerDeliveryDetails();

$email\_id="anveshgmail.com";

$this->assertEquals(true,$buyobj->validationOfEmail($email\_id));

}

// this test case should pass because email id is valid

public function testEmailID2()

{

$buyobj= new customerDeliveryDetails();

$email\_id="anvesh@gmail.com";

$this->assertEquals(true,$buyobj->validationOfEmail($email\_id));

}

// this test case should fail because phone number is invalid

public function testPhoneNumber1()

{

$buyobj= new customerDeliveryDetails();

$phone\_number="940220";

$this->assertEquals(true,$buyobj->validation\_phonenumber($phone\_number));

}

// this test case should pass because phone number is valid

public function testPhoneNumber2()

{

$buyobj= new customerDeliveryDetails();

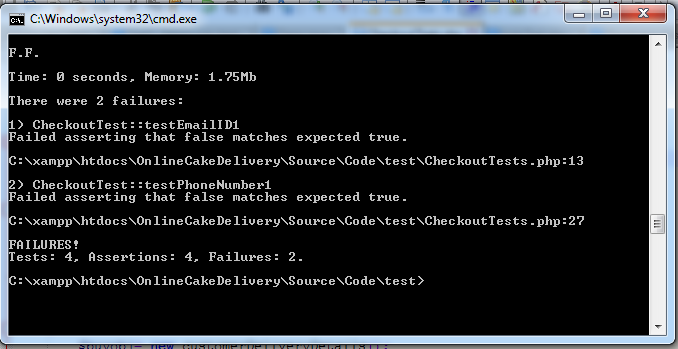
$phone\_number='9402200017';

$this->assertEquals(true,$buyobj->validation\_phonenumber($phone\_number));

}

}

?>



**4.2Problems encountered during Testing:**

* As we are not familiar with PHPUnit it took considerable amount of time in learning and implementing test cases.
* Since it is web based application, our controller (PHP code) is linked with User Interface (html) hence we had to make lot of assumptions in writing test cases in eliminating User Interface part.
* Unit testing is not applicable for some of the functions for eg: it is not possible to test the insert\_cake functionality as it contains browse option to upload a cake requiring inbuilt php features to be available during the test scenario making the setup hard to implement.
* As the complete testing of web application cannot be done through Unit testing we have also done functional testing using Selenium//

**5.Features Implemented and Limitations:**

This section has the features successfully implemented and limitations of the application.

**5.1 Features Successfully Implemented:**

* Ordering a cake through online by the customer.
* Feedback by the customer for the website.
* Uploading new cakes by the admin
* Checking feedback by admin submitted by customer
* Checking available orders and already delivered orders by the deliverer

**5.2 Limitations:**

* Payment through online is not available, Payments are through cash on delivery only.
* Assurance of delivery is not given to the customer by email.
* Application is browser specific, it doesn’t show some CSS styles in other than Google chrome.
* Customer cannot buy multiple cakes at a single instance as shopping cart is not included in the current version.

1. **Meeting Minutes:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DATE | DISCUSSION HIGHLIGHTS | TIME | MEMEMBERS ATTENDED | ATTENDEES |
| 30th AUG | Project idea brain storming | 1hr 30min | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 1st SEP | Defining Functionalities precisely | 2 hours | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 4th SEP | Research on platforms and languages | 1hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 6th SEP | Identifying the risks in the project | 1hr. 20 min | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 8th SEP | Gathering the requirements and installations | 1hr. | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 9th SEP | Work Sharing of Delivarble -1 | 3hrs | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 10th SEP | Presentation of Delivearble -1 | 1hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 11th SEP | Gantt Chart making | 1hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 13th SEP | Study and quick review of concepts | 50 mins | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 15th SEP | Presentation for the deliverable 1 | 30 min | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 20th SEP | Conducted workshop, talked with a group of clients and decided to implement feedback system. | 40 min | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 22nd SEP | Feedback of the deliverable 1 obtained and analyzed. | 20 min | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 27th SEP | Reanalyzed and modified the deliverable 1 document on discussing. | 1hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 29th SEP | Use case diagram for the project discussed. | 1hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 4th OCT | Functional and non-functional requirements discussed. | 30 min | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 6th OCT | System architecture discussed. | 45 min | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 11th OCT | Deliverable 2 documents and related matter discussed. | 2 hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 13th Oct | Review discussion on Deliverable 2 | 1 hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 18th Oct | Rectified the misconceptions over deliverable 2 | 2 Hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 20th Oct | Discussed about coding standards | 1hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 1st Nov | Reusable code written using mock data and discussed | 3hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 3rd Nov | Class based php coding discussed | 1hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 8th Nov | Individual Modules divided and started working. | 2 Hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 10th Nov | Discussed on database tables and their dependencies. | 2 hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 15th Nov | Testing using PHP unit discussed | 4hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |
| 17th Nov | Drawbacks and reviews about code inspection considered. | 3hr | 4 | Ashik, Anvesh, Srikanth, Sudhira |

1. **Member Contribution:**

|  |  |  |  |
| --- | --- | --- | --- |
| Member name | Contribution description | Overall Contribution (%) | Note  (if applicable) |
| Anvesh Athmakuri | Deliverer Module  Test cases | 25 |  |
| Ashik Shaik | Registration  Login  Feed back  Documentation | 25 |  |
| Srikanth Pusapati | Admin Module  Test cases | 25 |  |
| Sudhira Badugu | Checkout  Test cases  Documentation | 25 |  |