**CSCE 5430 DELIVARABLES 4**

**Online Pharmacy Portal (Group I)**

**Phase II for Development**

**For**

**Online Pharmacy Portal (web page)**

**Version 1.0 (approved)**

**Prepared by: Akella Soujanya**

**Akella Gayatri Sravani**

**Chahnitha Chandu**

**Dodla Nikithanjali**

**University of North Texas, Denton**

**Date: 4th November, 2019**

**Table of Contents**

1. List of Requirements ………………………………………… 3
2. UML Design ………………………………………………… 4
3. Test cases ……………………………………………………. 6
4. Unit testing ………………………………………………….. 7
5. Workload division …………………………………………… 9
6. User manual & Compilation instructions ……………………. 10
7. Peer review feedback ………………………………………… 13
8. **Requirements for Phase II**

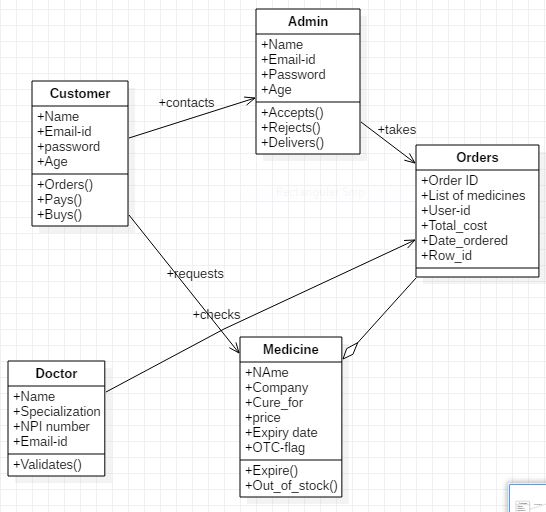
In the first phase we have implemented the login pages for both user and admin. Also, admin can add or edit products in his page and the changes can be viewed in the user page. In the second installment of the webpage, we have implemented few more functionalities in the page.

The requirements implemented in phase II below:

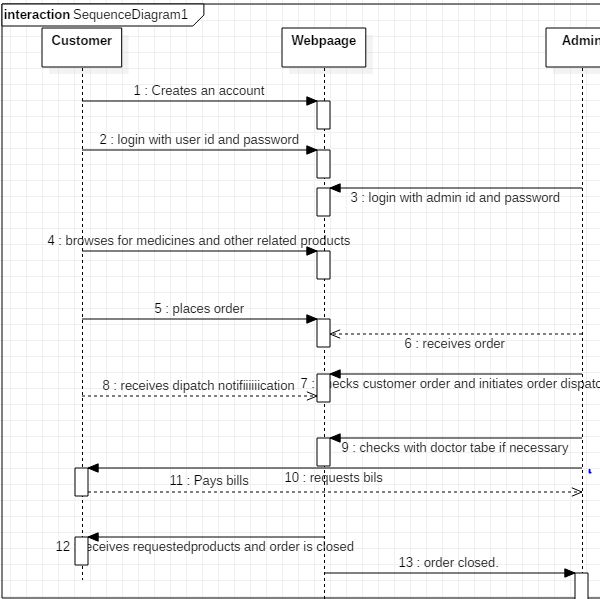
(Requirements were not properly specified in deliverables 2)

* Users should be able to add favorites in their account, to be able to purchase them at a later point (cart functionality)
* User and admin should be able to search for specific medicines using the search bar on top of their respective home pages
* User should be able to order medicines and this should be reflected on the admin’s page
* User should get a prompt to fill up additional details in case of OTC medicines

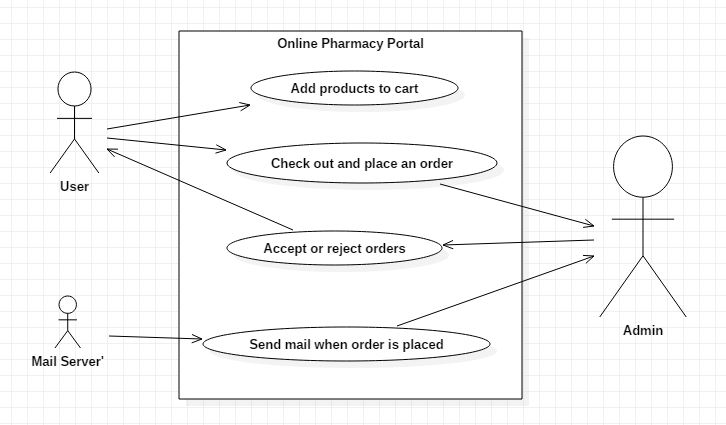
1. **UML Design**
   1. **Class diagram**

****

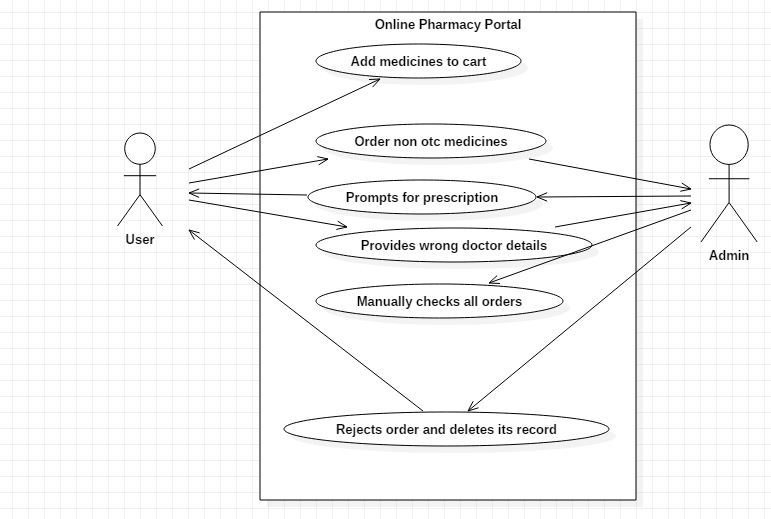
* 1. **Sequence diagram**

****

* 1. **Use case diagrams**
     1. **Normal case**

****

* + 1. **Error case**

****

1. **Test Cases**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No.** | **Requirement** | **Test Priority** | **Test steps, pre-conditions** | **Expected Result** | **Obtained result** | **Success/ Failure** | **Name of Tester** |
| 1 | User clicks on the ‘add to cart’ button for any product | Medium | User should already have an account and must be logged in | The product should be added and visible in the user’s cart | The product is now visible in the user’s cart | Success | Gayatri |
| 2 | User types the word ‘Candy’ in the search bar on his home page | Medium | User should have an account and be logged in | The page must display all products related to candy as the result | The page shows a list of candies and candy bars sold in it | Success | Soujanya |
| 3 | Admin types name of any product in the search bar on his home page | Low | The admin should be logged in | The page must display the product | The product searched for was displayed | Success | Nikitha |
| 4 | User places an order | High | User should have products in their cart. Click on the ‘Check out’ button | The user should be redirected to a page where he will be asked for an address | User is asked for shipping address | Success | Chahnitha |
| 5 | When user enters an appropriate address | High | User must type in the address and click on ‘save address’ and then click on ‘continue’ | User should see that his address is saved and that order is placed | First user can see a message ‘address saved’ and order is successfully placed | Success | Gayatri |
| 6 | When user places an order, on admin page... | High | Admin should be logged in, click on ‘orders’ button | Admin should be able to see orders placed | Admin can see 1 order from the customer | Success | Soujanya |
| 7 | When the admin opens the order and responds to it as reject | High | Admin should click on ‘orders’, open any order and click on either ‘reject’ | Order should be deleted | Order is no longer visible in the list of orders | Success | Nikitha |
| 8 | When the admin responds to any order, the respective user… | Medium | User should login to their account and click on ‘orders’ | User should be able to see the change in their account | User can see the order only if they are either accepted or not yet seen. If they rejected, the order is automatically deleted | Success | Soujanya |
| 9 | User orders product which is not over the counter | High | User should have a product in his cart which is not over the counter | User should be prompted to fill up extra details | User is asked for prescription, doctor name, email-id and NPI number | Success | Gayatri |

1. **Unit Testing**

**U**nit testing is done using PHPunit in our project.

I’m mentioning below, some of the many test cases that we have run as part of our unit testing along with screenshots showing that they execute well.

<?php

require\_once('checkDocDetails.php');

use PHPUnit\Framework\TestCase;

class TestDocDetails extends TestCase

{

protected function setUp()

{

parent::setUp();

$\_POST = array();

}

public function tearDown(){ }

public function testDoctorDetails()

{

$doctor = new CheckDocDetails();

$\_POST = array(

'dName' => 'Derek Shepherd',

'dNPINum' => '7654567896',

'prescription' => 'pr1.png',

'dEmailId' => 'derekshepherd@gmail.com',

'UserId' => 'alicia@gmail.com',

'submit' => 1

);

$\_SESSION['id'] = 'mpalkar@gmail.com';

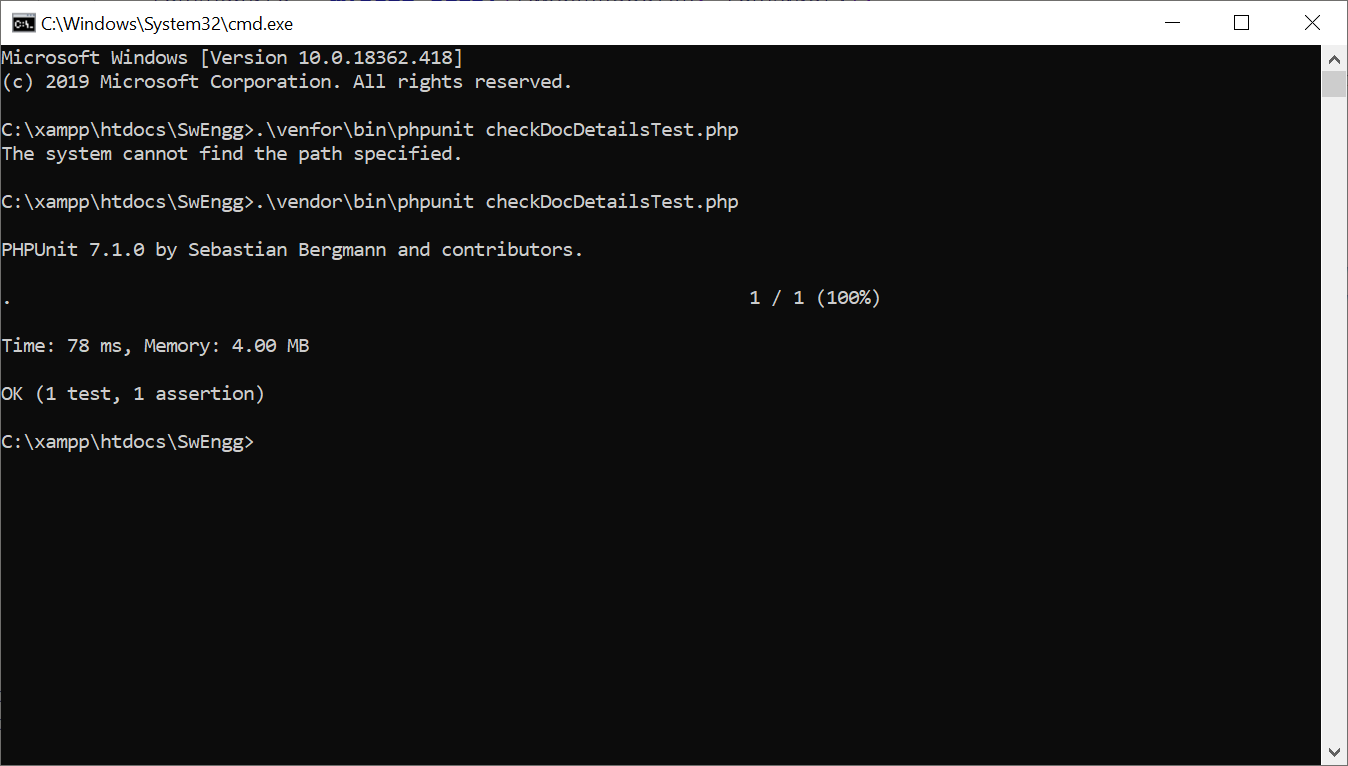
$\_REQUEST = 'POST';

echo $this->assertTrue($doctor->CheckDoc($\_POST, $\_REQUEST) == 3);

}

}

?>



<?php

require\_once('addToCart.php');

use PHPUnit\Framework\TestCase;

class TestAddToCart extends TestCase

{

protected function setUp()

{

parent::setUp();

$\_POST = array();

}

public function tearDown(){ }

public function testItemsToCart()

{

$cart = new AddToCart;

$\_POST = array(

'ProductId' => 28,

'cartQty' => 2,

'submit' => 1

);

$\_SESSION['id'] = 'mpalkar@gmail.com';

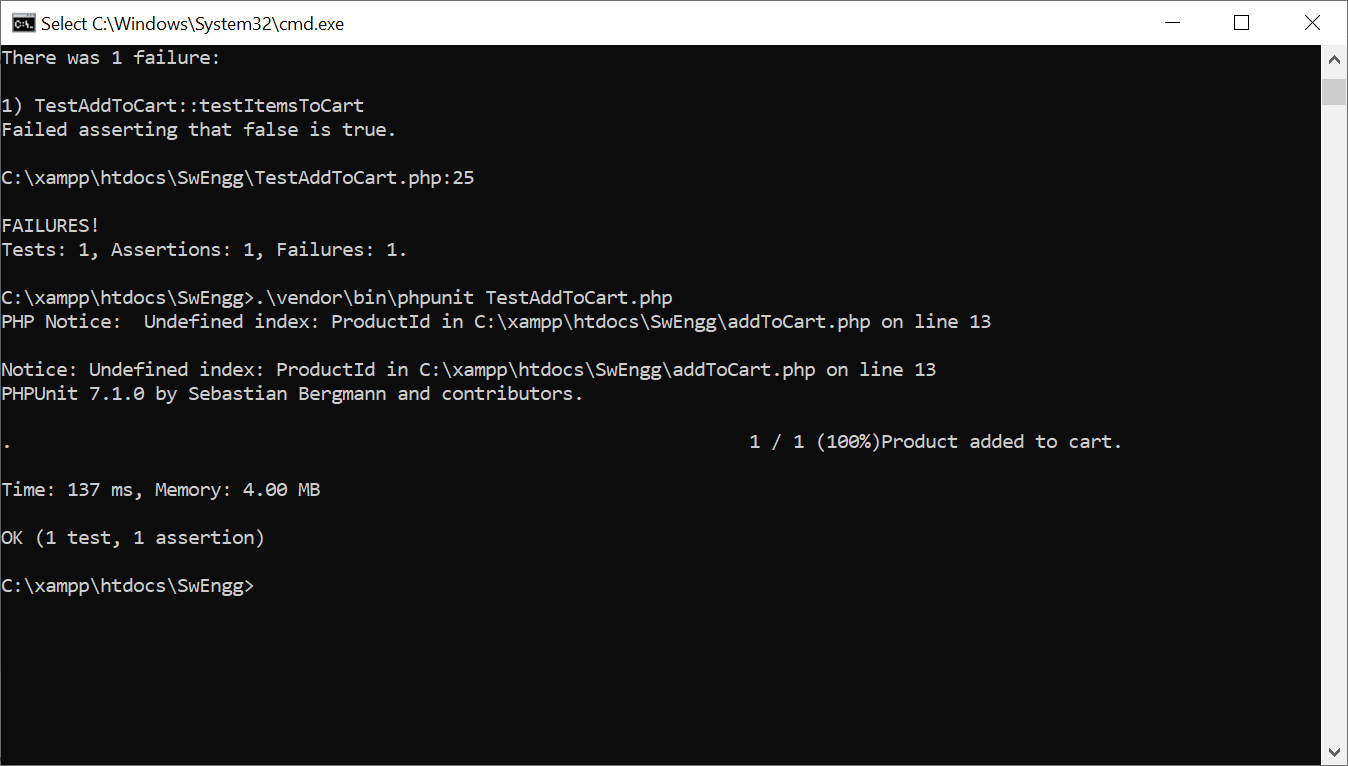
$\_REQUEST = 'POST';

$this->assertTrue($cart->itemsToCart($\_POST,$\_SESSION, $\_REQUEST) == 1);

}

}

?>



1. **Workload division**

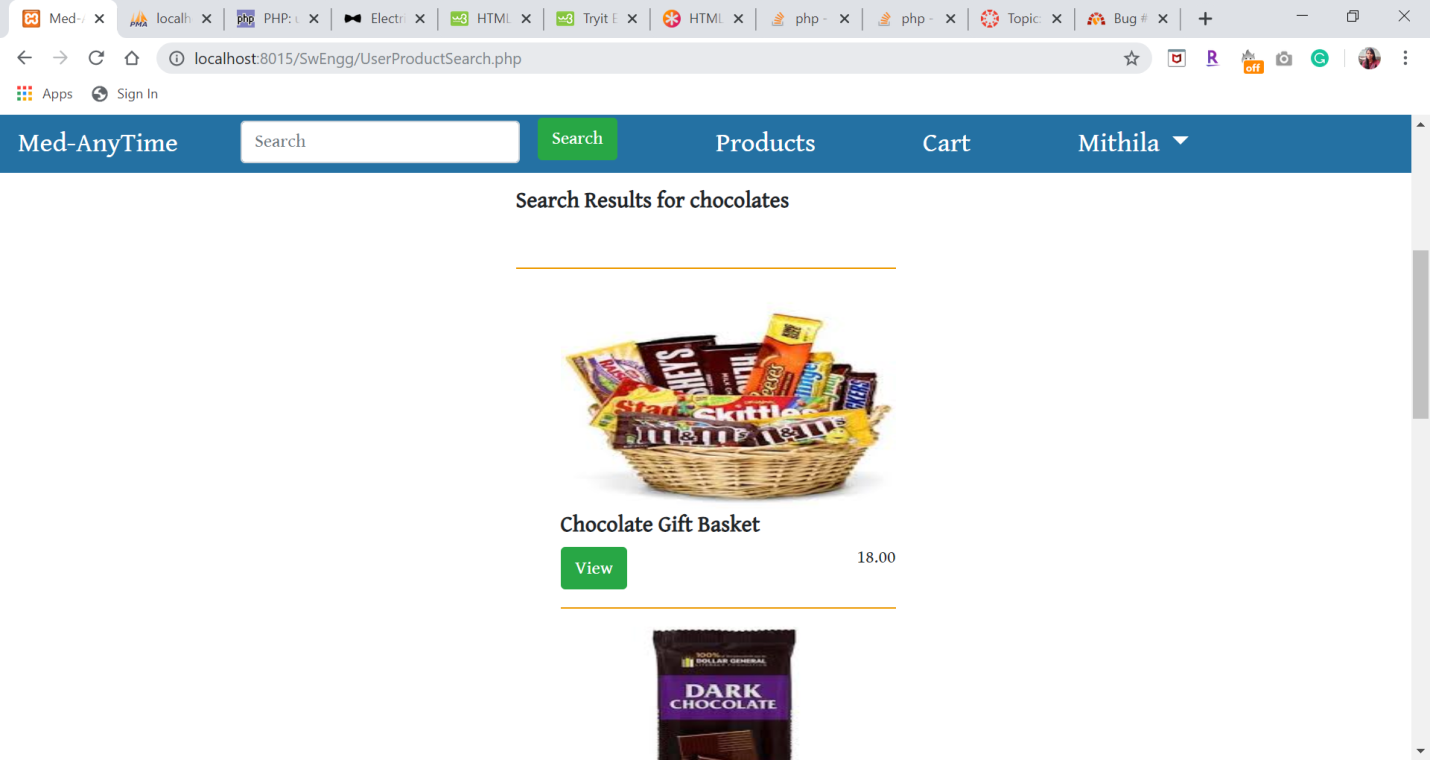
The entire work load was equally divided among all team members. Each of us had a piece each of the coding, testing, documentation, unit testing and presentation to do. Below, I’m mentioning a few of the tasks that were accomplished between phase I and phase II of development along with the name of the person who has accomplished the tasks.

* Search functionality for both the admin and user pages. (By Chahnitha)
* User should be able to add products in their cart (By Gayatri)
* At the cart, if user wishes to check out, prompt for address and doctor details in case of non-OTC meds (By Soujanya)
* Generation of tracking number when order placed (By Soujanya)
* Admin page being notified when Order is placed by user (By Nikitha)
* User being able to view their own orders, and their status when available on the orders page (By Nikitha)
* Unit testing using PHPUNIT (By Gayatri & Chahnitha)
* Documentation and Repository Management (By Soujanya & Chahnitha)

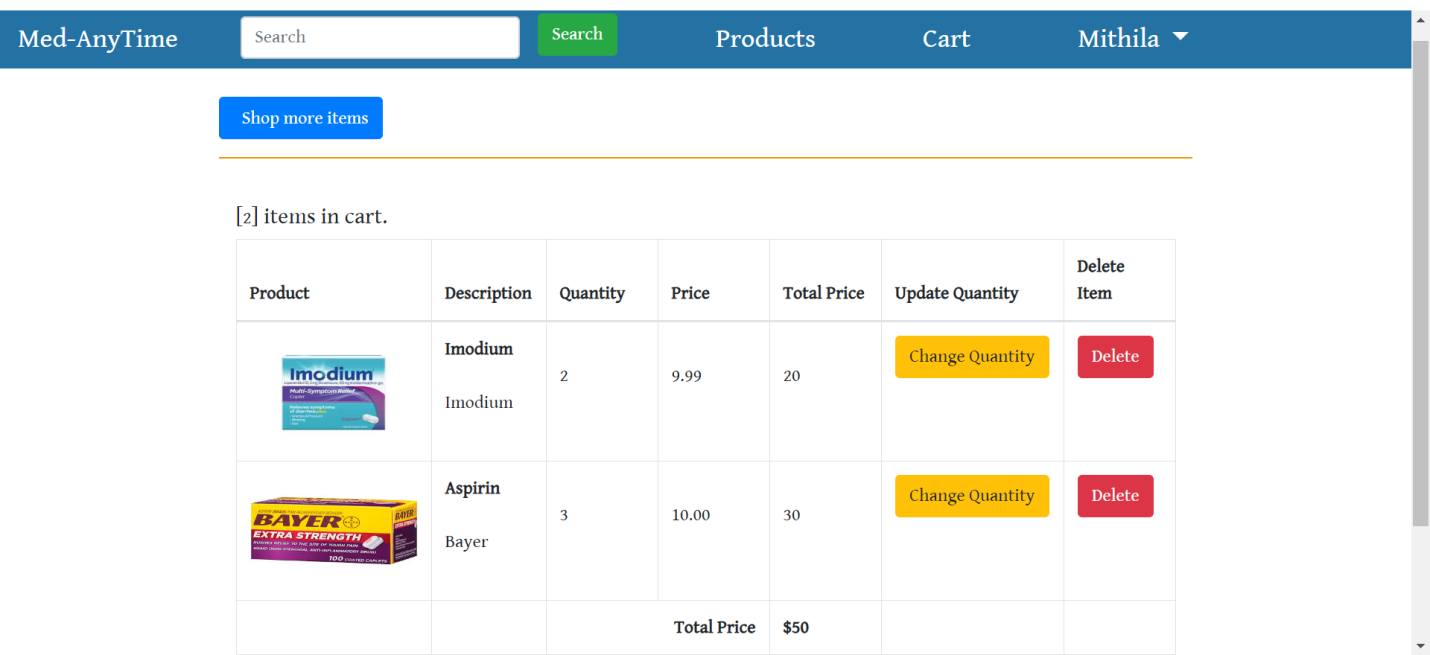
1. **User Manual & Compilation Instructions**

The user manual here is a mere continuation of the previous one (in deliverables 3.docx). The user manual covers instructions on how to use all the functionalities implemented in this phase of development.

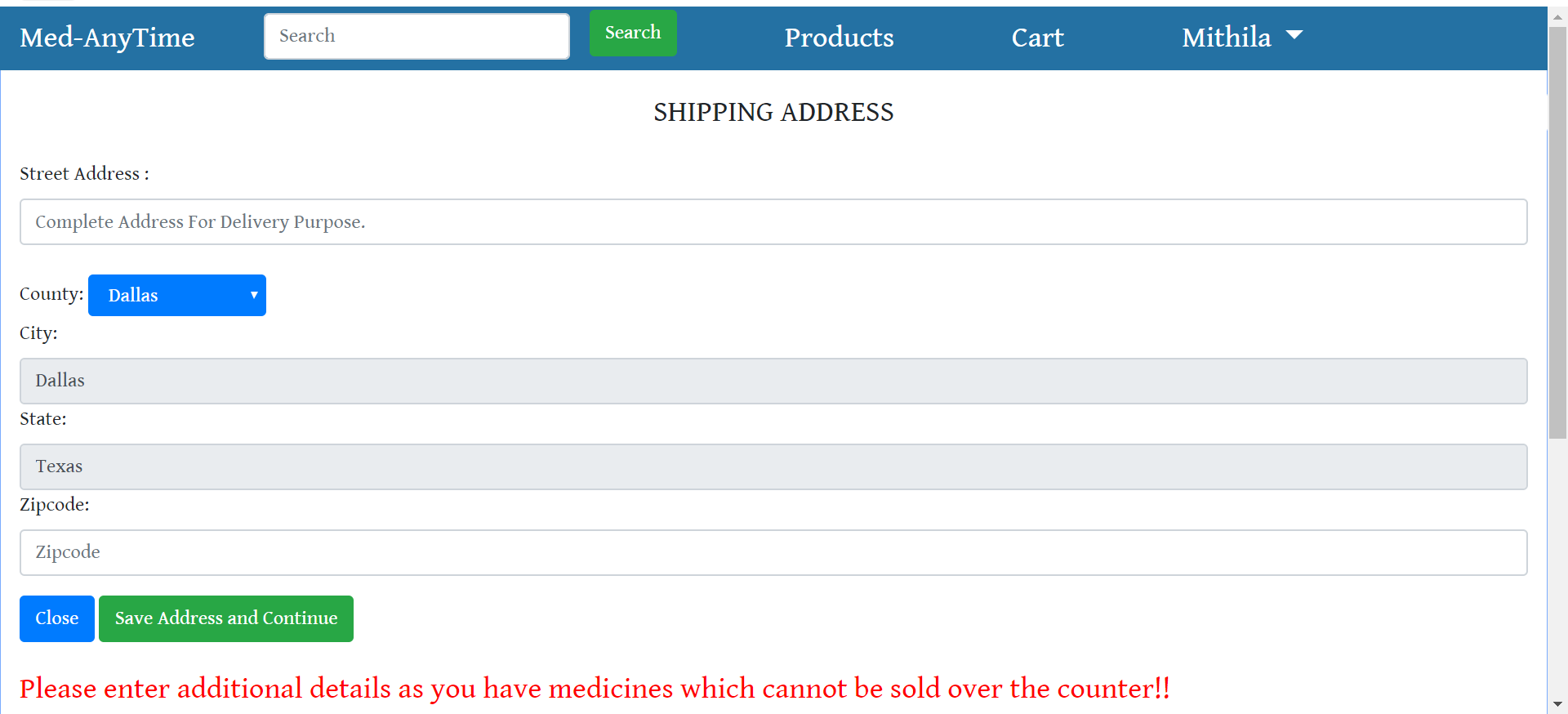
So far, we have covered how the user can login and view product details if the products are featured on the home page. This installment release of the webpage allows the user to search for specific medicines by typing in the title of the medicine or the problem for which they wish to get a cure. I wish to make note of an important point here. As we are talking of the webpage as an online pharmacy portal, we might frequently use the term medicines, but it really means products. The webpage also sells things like moisturizers, certain accessories, candy and mouth fresheners, etc. Therefore, the user can say just type in the search bar ‘chocolate’ and he/she will see all the range of candy sold on MedAnytime. Similarly, on the admin page as well, the admin can search for products instead of having to go through the whole list of products every time.

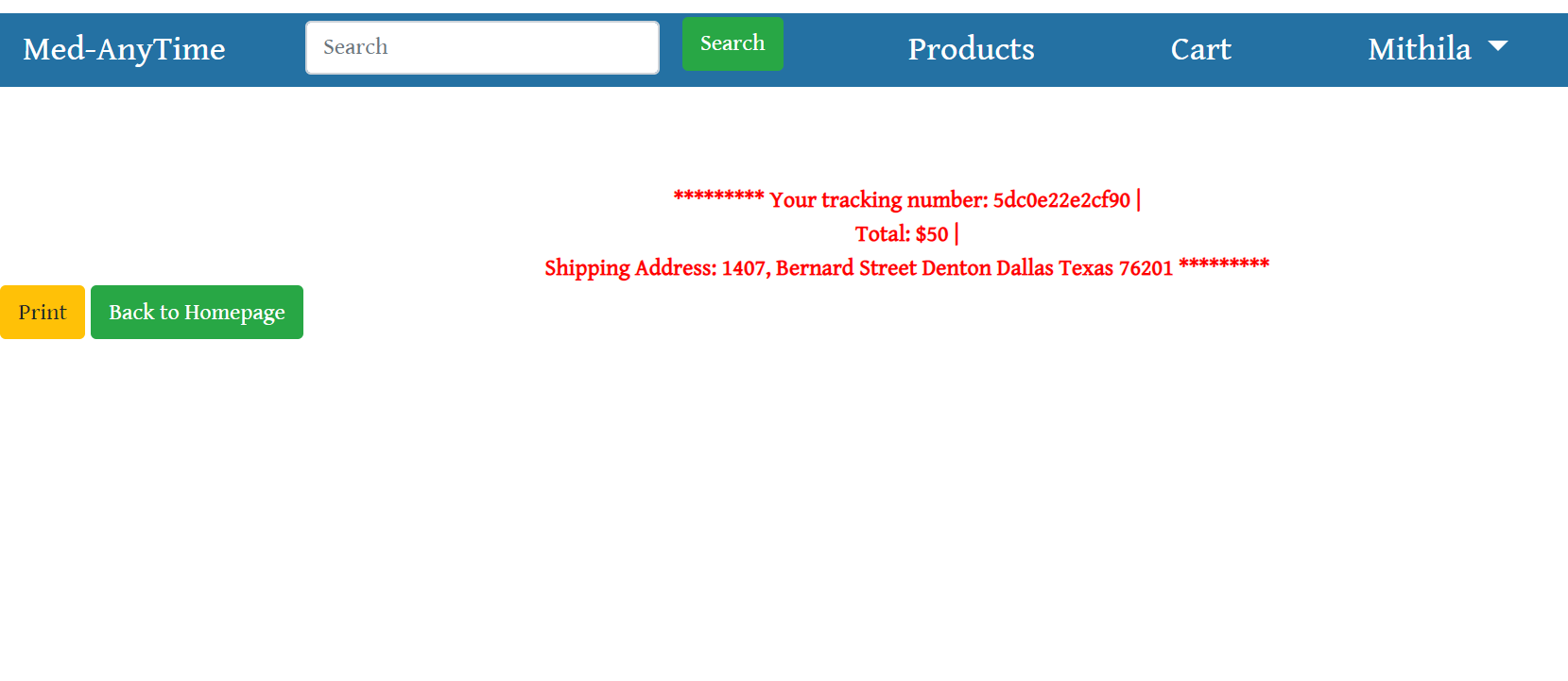


Next, the user can add products to his/her cart. To do so, user must click on the specific product to open its details. The page should show a button ‘Add to cart’ on it. The user should click on the button and the product will be added to the cart in their account. The below screenshot shows that a user has added a product to their cart

****

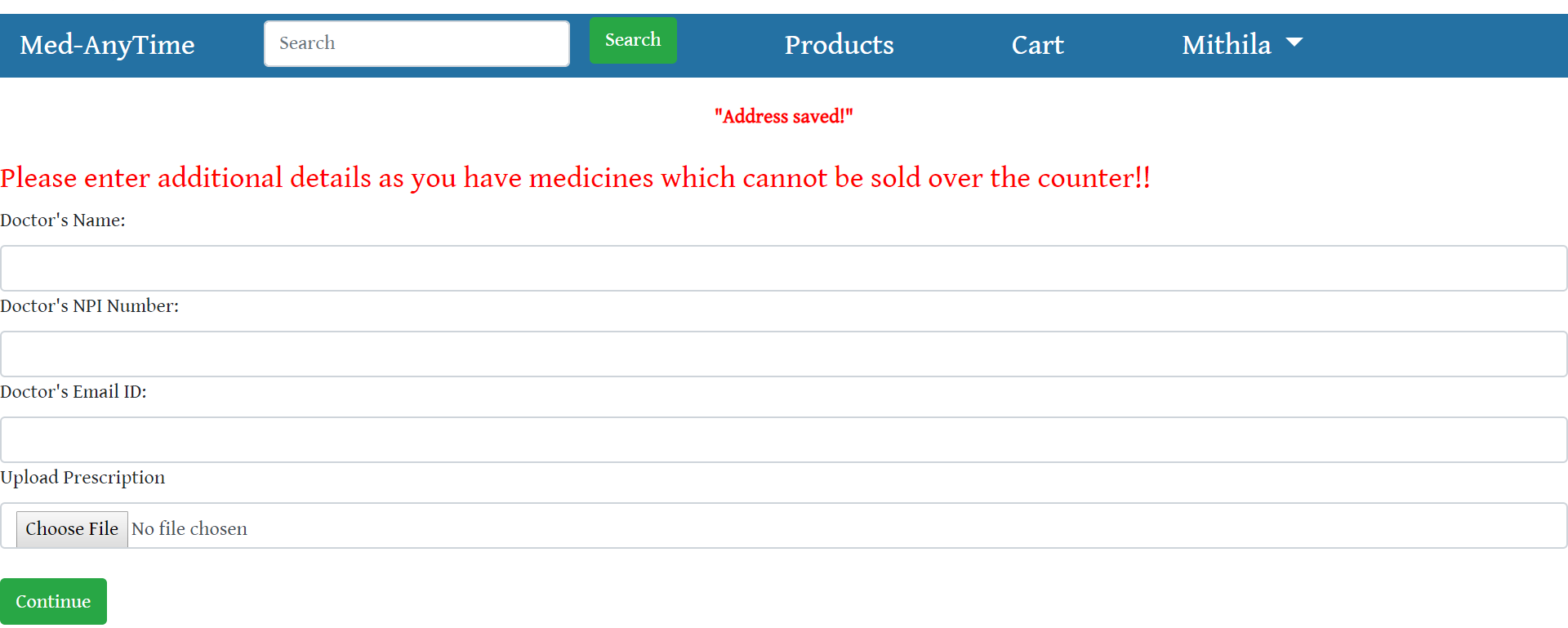
Next, to place an order, the user needs to click on the cart symbol on their page, where they will be able to view all the products in it. It also shows a button ‘check out’ in the cart page. If the user clicks on it, they will be asked for an address to ship the order to. The user should now type in the address, as street address, county, city, state and Zip code. After typing the details, click on ‘save address’ and then click on ‘continue’.

****

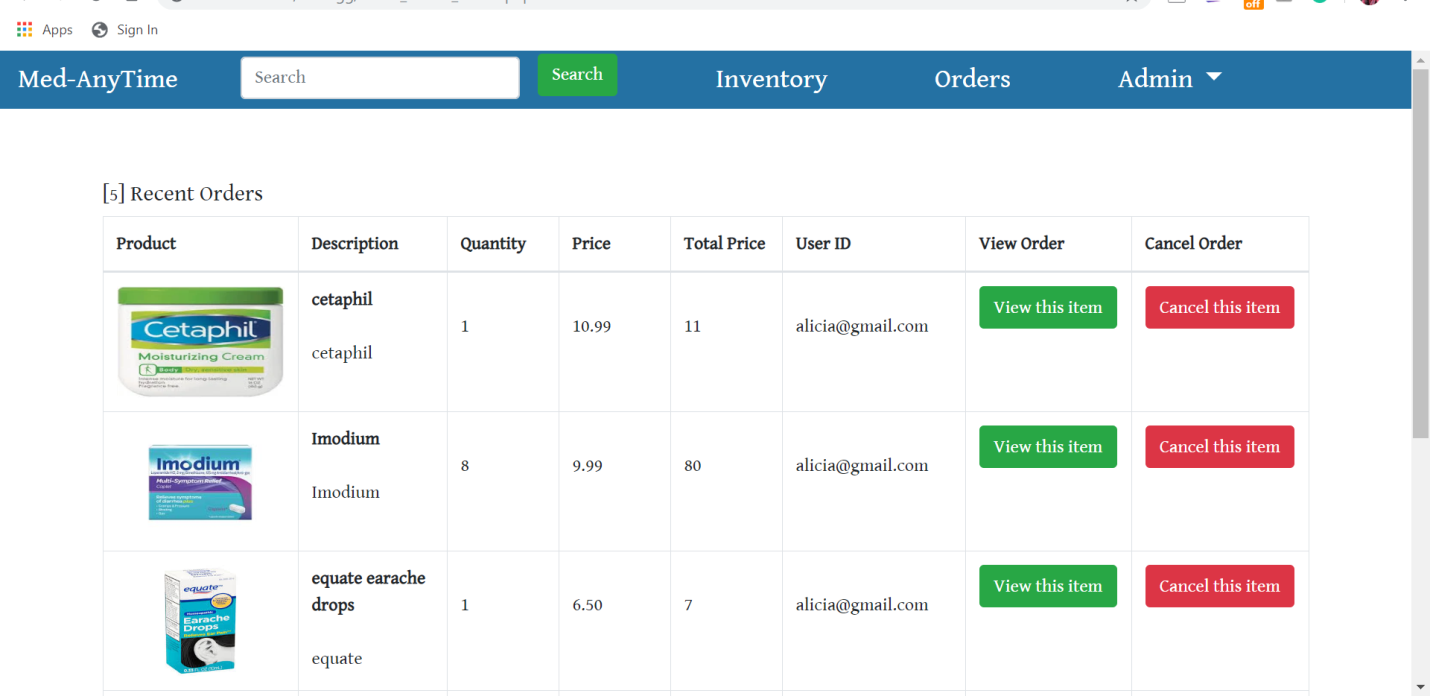
****

Now the user should see a message that order is placed. The admin can also view all orders in his page by clicking on the ‘orders’ button. Here, he can either accept or reject orders. Similarly, the user can also view his orders, by clicking on the ‘orders’ button on his home page.

Lastly, the user can also order for medicines with prescriptions. These medicines are specifically flagged. Therefore, if they are present in the cart, the user will be prompted for a prescription and doctor details (doctor name, email id and an NPI number). These details will be physically checked by the admin.



As mentioned above, the admin can view orders when they are placed by the users. The admin can view the order details by clicking on the ‘view this item’ button. The admin also has the authority to reject orders. This will result in deletion of the order on both admin side and the user side. To do that, admin has to click on the ‘cancel this item’ button. This is what the orders page looks like.



The compilation and installation instructions remain the same as the instructions in the previous document (deliverable 3.docx).

1. **Peer review feedback**

The peer review and code inspection for the code in deliverable 3 (phase I of development) was conducted during class hours on the 24th of October, 2019. On this day, our peers inspected our code, and discussed their questions and concerns with us.

During the review session, the suggestions and feedback we received are mentioned below:

* Functionalities were well mentioned with proper details as to why each of them exist
* However there was no mention of which team member wrote each functionality in the code document. We told them that this was mentioned in the documentation for this code.
* There was no dead code in the code delivered. All methods and functions have appropriate parameters and return types that are tested.
* All files are properly handled, opened and closed as and when necessary.
* **Suggestion:** Include exception handling in all future deliverables, as it is essential that the code has some contingency plan in case of error or exception.

This proved to be a really useful session for our project development as it was essential for us to know where our ideas were not on par with the customer expectations and requirements. We also understood what other important points we must bear in mind while developing the product further.