# Books and Tutors SFSU

**SW Engineering CSC648/848 Summer2016**

**Milestone 4**

**08/5/16**

**Rev 1.0**

**Group 4**

**Lead:** Abdullah Arikat (aarikat@mail.sfsu.edu)
**Team:** Alex Brown, Izaac Garcilazo, Marc Garde, Juris Puchin, Sang Saephan

## 1)product summary:
## Books and tutors SFSU

Books and Tutors SFSU is a site for SFSU student to sell items such as books, computers, furniture, clothing and more, directly to other SFSU students. In addition to selling items, Books and Tutors SFSU customers can use the site as an easy way to search for and offer tutoring services.

Students of SFSU interested in finding a tutor or advertising their tutoring service will no longer need to rely on a corkboard flier somewhere on campus. Books and Tutors SFSU can be used as a central place to easily search through and find the exact tutor they need, or a place to offer tutoring service themselves.

Our website will initially cater to San Francisco State University students, but as the service grows, we could expand to other schools and offer Books and Tutors UCSF, Books and Tutors UC Berkeley, and so on.

Our service (Books and Tutors SFSU):
We are dedicated to serve the students of SFSU
Nearly 30000 students attend SFSU and are always in search for books as well as other various products at a cheaper price.
We will offer the books as well as other various products listed on our site to the students of SFSU at a very competitive price
We will provide tutoring services to the students of SFSU
If a user is searching for books, we will also have a feature that will recommend tutors at SFSU that match the subject of the books that they are viewing
We believe that our unique feature (tutoring) will have a significant impact on our business plan to help serve the students of SFSU

| **Priority 1:** | |
|---|---|
| *Anonymous User:* | |
| | Use the search bar to look for goods and tutors |
| | Search for items by category |
| | Create an account (if also SFSU student) |
| *User (logged in):* | |

| | |
|---|---|
| | All of the above plus |
| | Access his/her shopping cart |
| | Add tutoring services and items for sale to the cart |
| | Buy items |
| | Sell Items |
| | Order tutoring services |
| | Register as a tutor |
| | Edit items for sale |
| | Edit the shopping cart |
| | Request to become a tutor |
| *Tutor (logged in):* | |
| | All of the above plus |
| | Offer tutoring services |
| | Edit the tutoring services they offer |
| *Administrator:* | |
| | All of the above plus |
| | Manage the database directly |
| **Priority 2:** | |
| *Anonymous User:* | |
| | Sort items by price and date |
| | Get a list of tutors and items by entering a class |
| *User (logged in):* | |
| | All of the above plus |

## Our website:

URL: http://sfsuswe.com/~su16g04/m3/

## 2) **Usability Test Plan**:

**Test Objectives:**
Discoverability of items and tutoring services offered by Books and Tutors SFSU is an important aspect of our website. That's why the following tests focus on finding specific item or tutor a student may need.

**Test Plan:**
*System Setup:*
A desktop computer running a recent version of either Internet Explorer, Edge, Safari, Chrome, or Firefox.
*Starting Point:* Navigate to http://sfsuswe.com/~su16g04/m3/
*Tasks:*

1. From the webpage, search for a Database tutor
2. Search for a Database book
3. Search for a laptop
4. Search for an iClicker

*Intended User:* An SFSU student
*Completion Criteria:* User was able to search for and find an item for each step.

**Questionnaire:**
Please circle the answer that best describes your feelings to the following statements.

***Question 1:*** *It was easy to find a tutor.*

| | | | | |
|---|---|---|---|---|
| Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree |

*Comments:*

***Question 2:*** *It was easy to sell an item.*

| | | | | |
|---|---|---|---|---|
| Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree |

*Comments:*

***Question 3:*** *It was easy to purchase an item*

| | | | | |
|---|---|---|---|---|
| Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree |

*Comments:*

***Question 4:*** *I would recommend Books and Tutors SFSU to other SFSU students*

| | | | | |
|---|---|---|---|---|
| Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree |

*Comments:*

## 3) **QA Test Plan**

**Test Objectives:**
Ensure the usability and functionality of searching products and searching products with a filter.

**HW and SW Setup:**
Website supports web browsers (Chrome, FireFox, IE, etc). Internet connection is required to interact with website.

**Feature To Be Tested:**
Searching products in general and searching products with filter.

**Test Cases:**
To ensure the stability of the functionality and usability of the search feature, requires numerous tests. The tests started searching products in general. By default, the search bar had no filters and as a result, some results were not expected. For example, if the user searched “book”, the user expects the results to be books. However the results also included “Macbook charger”. To over this problem, the categories list was implemented. This makes the search being entered to be more precise when the user types in a more broad word like “book”. The “Books” category list ensures that only books would show up and this goes for the same for the other categories. The next series of tests includes the search of tutors. When the user types in “tutors” all tutors will be retrieved from the database. The “Tutors” category list gives a more precise search result. When a class name is typed in, tutors who are tutoring that class will show up (ie, CSC340). All in all the tests were generally expected. With the appropriate filters, the user can expect a more precise search result.

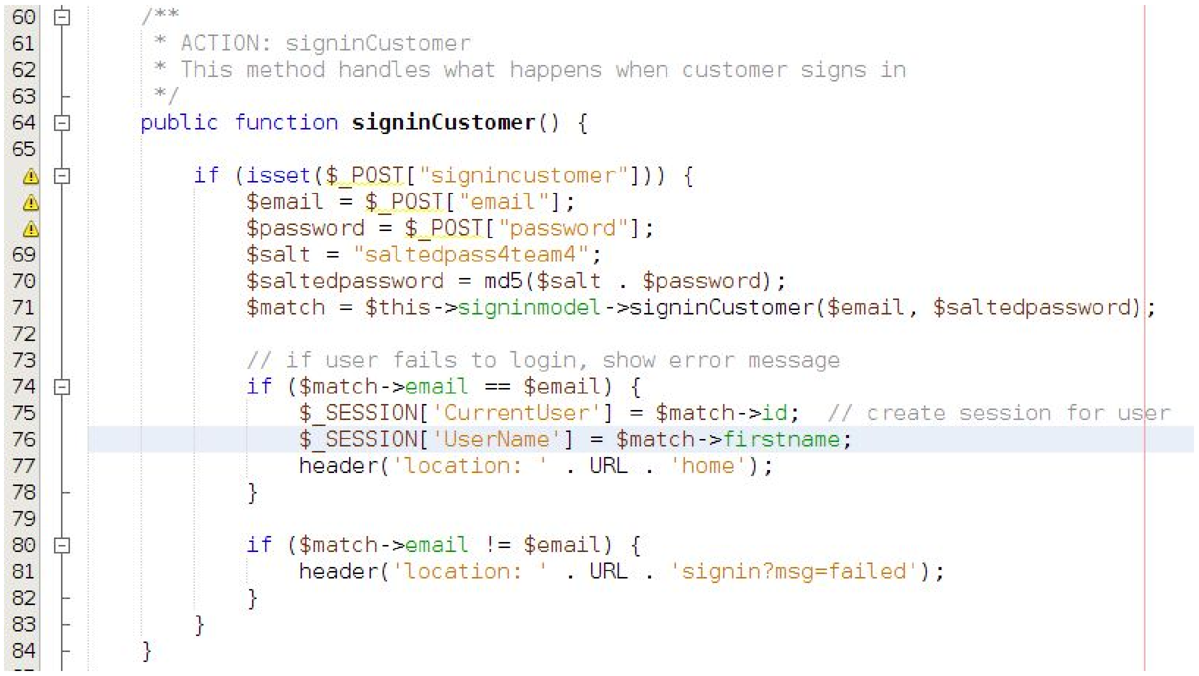
Test Cases

| **Test Number** | **Description** | **Test Input** | **Expected Output** | **PASS/FAIL** |
|---|---|---|---|---|
| 1 | Open the webpage: http://sfsuswe.com/~su16g04/m3/<br><br>Change the category from "All" to "Books" and search with an empty search bar. | **Category:** Books<br><br>**Search:** " "<br><br>**Sort by:** "Best Match" | Products that fall in the "Books" category.<br><br>**6 Results** | PASS |
| 2 | Following Test #1, we change "Sort by" from "Best Match" to "Price: Lowest to Highest". | **Category:** Books<br><br>**Search:** " "<br><br>**Sort by:** Price: Lowest to Highest | Products that fall in the "Books" category by the lowest price to the highest.<br><br>**6 Results** | PASS |
| 3 | Open the webpage: http://sfsuswe.com/~su16g04/m3/<br><br>With the "All" category selected, enter "C++" into the search bar. | **Category:** All<br><br>**Search:** "C++"<br><br>**Sort by:** "Best Match" | Products that contain the word "C++". In addition to products, tutors that offer services related to "C++" .<br><br>**3 Results** | PASS |
| 4 | Following Test #3, select the "c++ introduction book" product. | **Category:** All<br><br>**Search:** "C++"<br><br>**Sort by:** "Best Match" | Tutors that offer services related to "C++" will be recommended to the user.<br><br>**2 Results** | PASS |
| 5 | Open the webpage: http://sfsuswe.com/~su16g04/m3/<br><br>With the "Tutors" category selected, enter "CSC340" to search for tutors related to that class. | **Category:** Tutors<br><br>**Search:** "CSC340"<br><br>**Sort by:** "Best Match" | Tutors that offer services for "CSC340".<br><br>**2 Results** | PASS |

4) **Code Review**:

Every member in our group uses Netbeans IDE for this project. We agreed to use One True Brace style of formatting and change the formatting style within the IDE to match this style. In this way we can use the automated format that comes with Netbeans.
We choose One True Brace Style because we like the opening and closing brackets that this style offers. All if, else, while, and for statements have opening and closing braces, even if they are not necessary. Unlike the K&R style which omits them. The purpose is to make it easy to know where to insert new statements and know precisely how these statements will be grouped. It also makes the code easy to read.

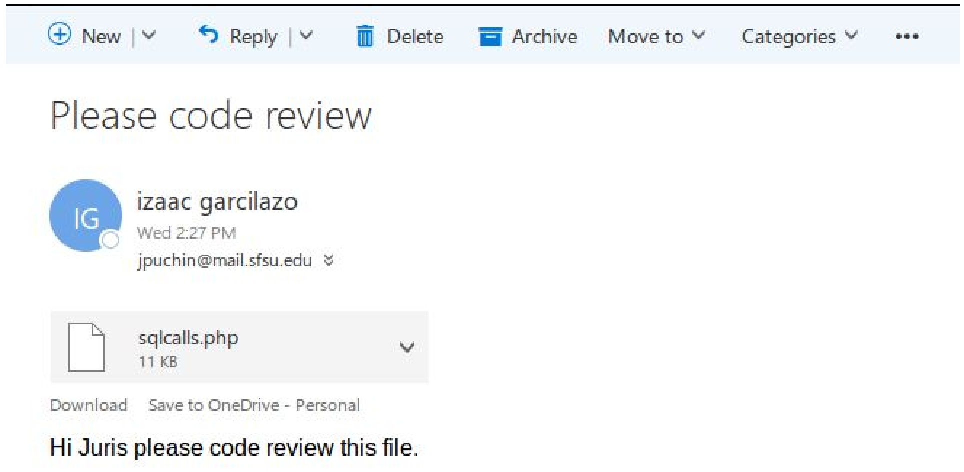
Coding style sample:



```
    /**
     * ACTION: signinCustomer
     * This method handles what happens when customer signs in
     */
    public function signinCustomer() {

        if (isset($_POST["signincustomer"])) {
            $email = $_POST["email"];
            $password = $_POST["password"];
            $salt = "saltedpass4team4";
            $saltedpassword = md5($salt . $password);
            $match = $this->signinmodel->signinCustomer($email, $saltedpassword);

            // if user fails to login, show error message
            if ($match->email == $email) {
                $_SESSION['CurrentUser'] = $match->id;  // create session for user
                $_SESSION['UserName'] = $match->firstname;
                header('location: ' . URL . 'home');
            }

            if ($match->email != $email) {
                header('location: ' . URL . 'signin?msg=failed');
            }
        }
    }
```

Code Review:

Team member Izaac sends the file sqlcalls.php to be review by team member Juris.



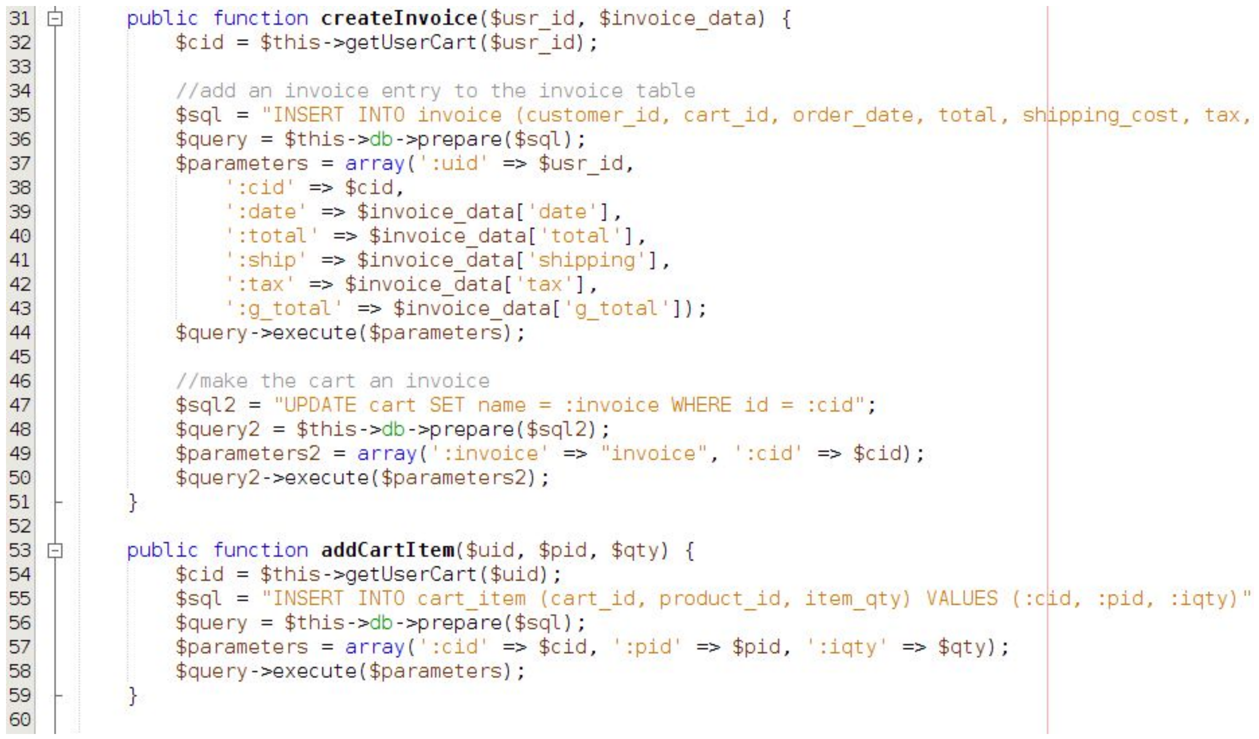
Please code review

izaac garcilazo
Wed 2:27 PM
jpuchin@mail.sfsu.edu

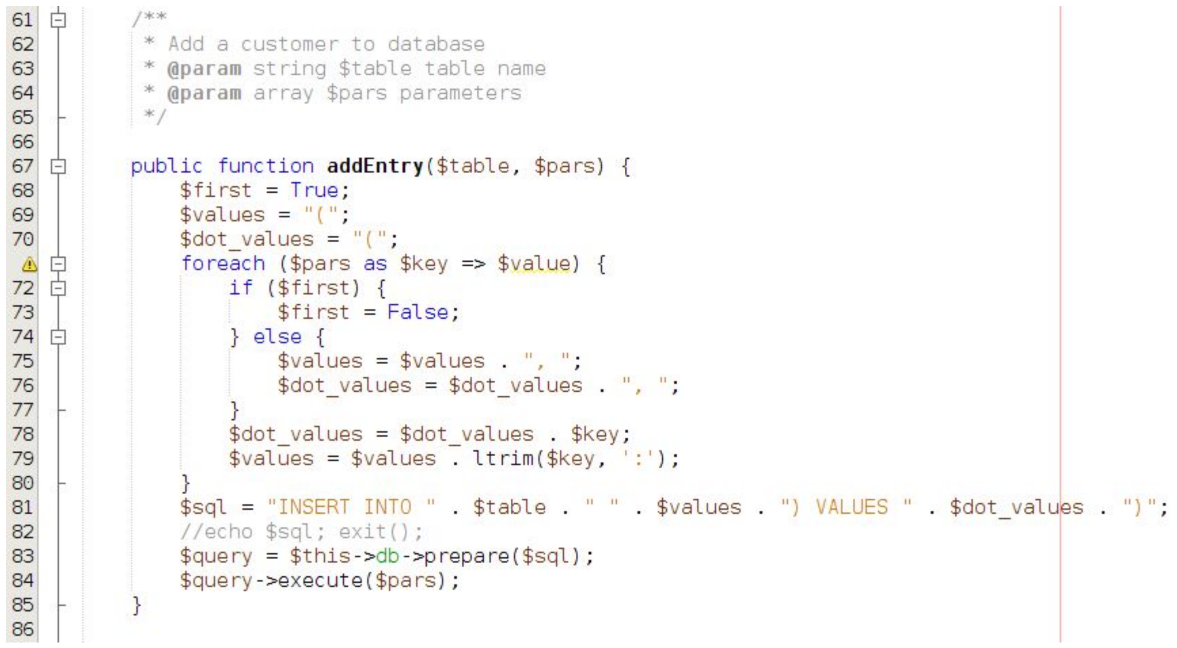
sqlcalls.php
11 KB

Hi Juris please code review this file.

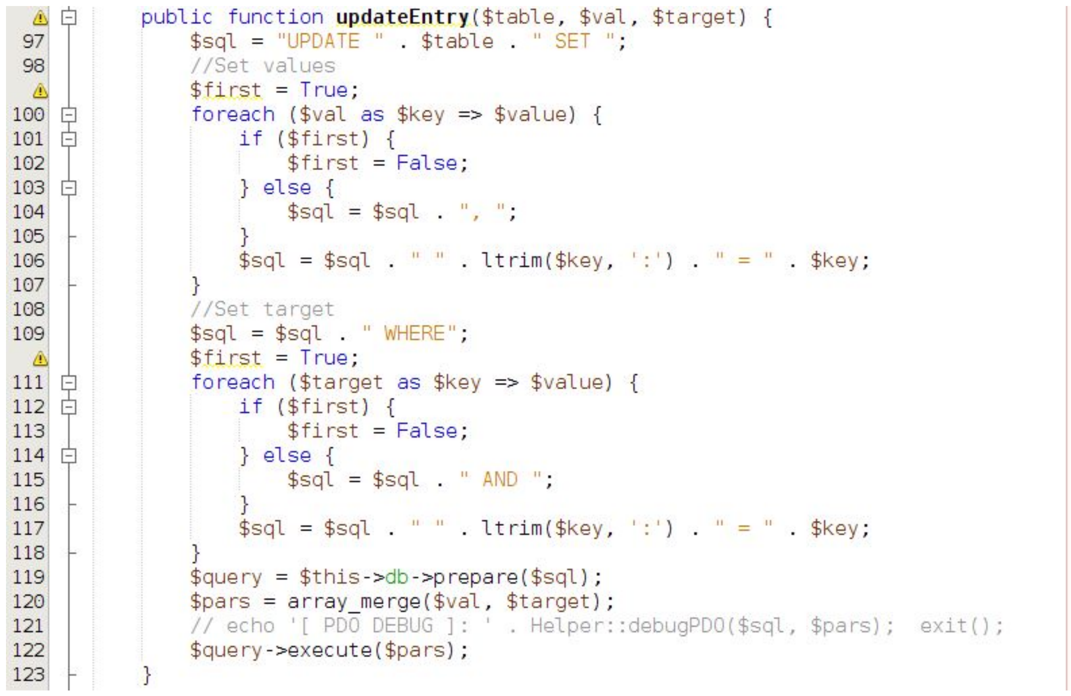
Sqlcalls.php



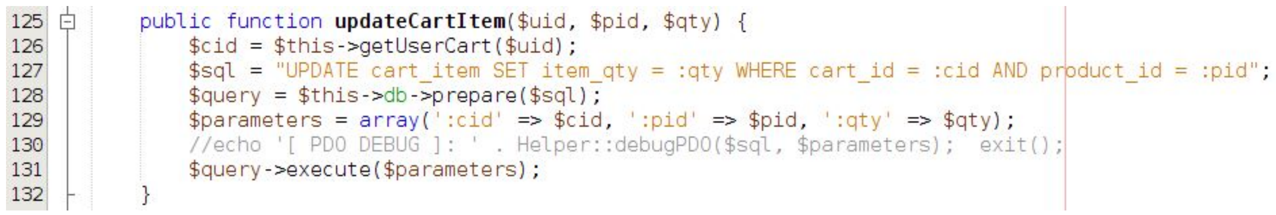
```
    public function createInvoice($usr_id, $invoice_data) {
        $cid = $this->getUserCart($usr_id);

        //add an invoice entry to the invoice table
        $sql = "INSERT INTO invoice (customer_id, cart_id, order_date, total, shipping_cost, tax,
        $query = $this->db->prepare($sql);
        $parameters = array(':uid' => $usr_id,
            ':cid' => $cid,
            ':date' => $invoice_data['date'],
            ':total' => $invoice_data['total'],
            ':ship' => $invoice_data['shipping'],
            ':tax' => $invoice_data['tax'],
            ':g_total' => $invoice_data['g_total']);
        $query->execute($parameters);

        //make the cart an invoice
        $sql2 = "UPDATE cart SET name = :invoice WHERE id = :cid";
        $query2 = $this->db->prepare($sql2);
        $parameters2 = array(':invoice' => "invoice", ':cid' => $cid);
        $query2->execute($parameters2);
    }

    public function addCartItem($uid, $pid, $qty) {
        $cid = $this->getUserCart($uid);
        $sql = "INSERT INTO cart_item (cart_id, product_id, item_qty) VALUES (:cid, :pid, :iqty)"
        $query = $this->db->prepare($sql);
        $parameters = array(':cid' => $cid, ':pid' => $pid, ':iqty' => $qty);
        $query->execute($parameters);
    }

```



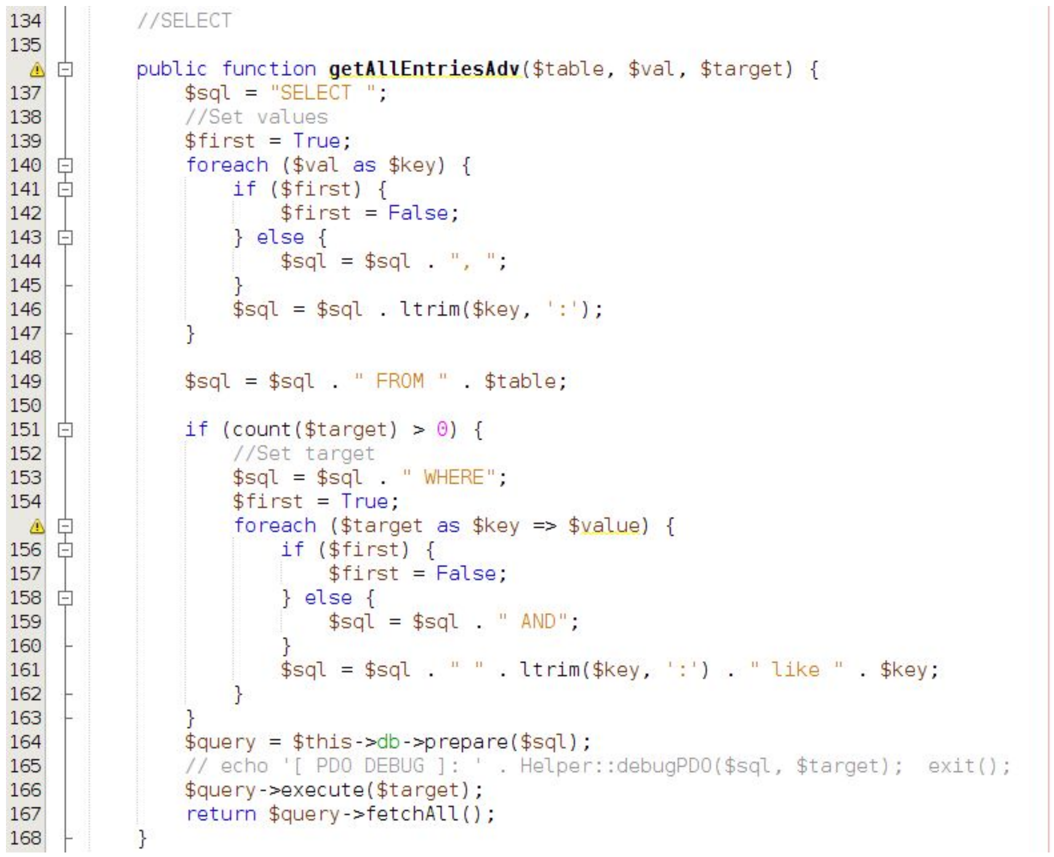
```php
    /**
     * Add a customer to database
     * @param string $table table name
     * @param array $pars parameters
     */

    public function addEntry($table, $pars) {
        $first = True;
        $values = "(";
        $dot_values = "(";
        foreach ($pars as $key => $value) {
            if ($first) {
                $first = False;
            } else {
                $values = $values . ", ";
                $dot_values = $dot_values . ", ";
            }
            $dot_values = $dot_values . $key;
            $values = $values . ltrim($key, ':');
        }
        $sql = "INSERT INTO " . $table . " " . $values . ") VALUES " . $dot_values . ")";
        //echo $sql; exit();
        $query = $this->db->prepare($sql);
        $query->execute($pars);
    }

```



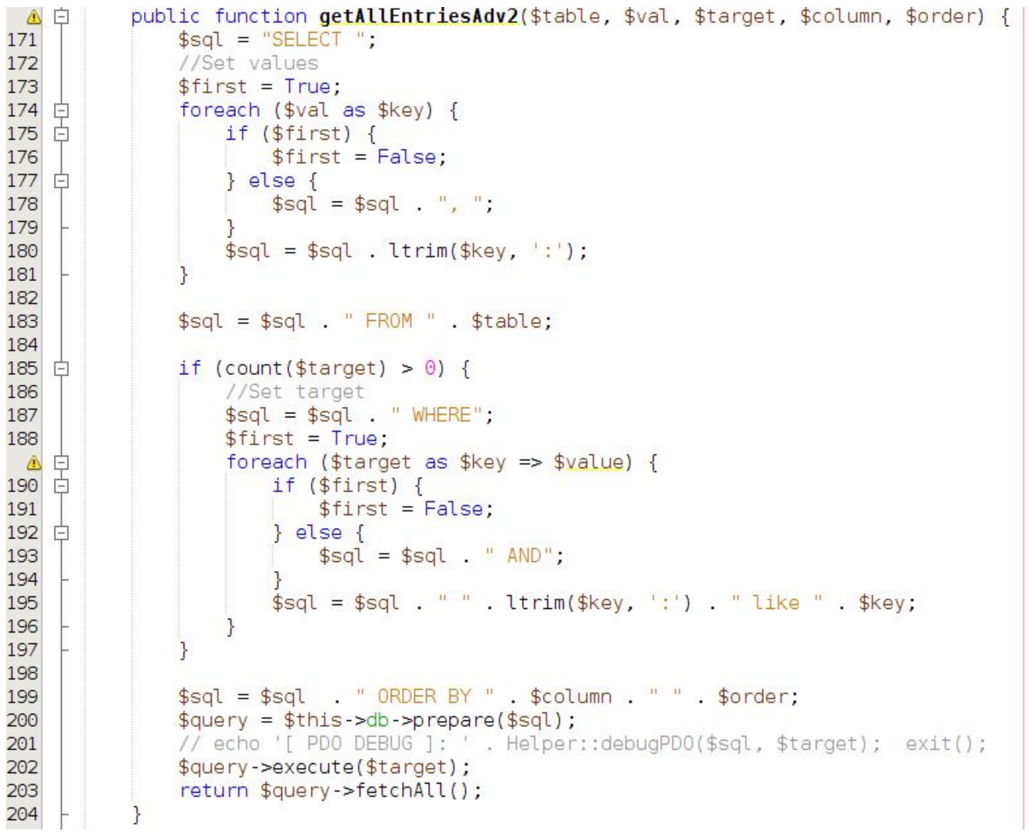
```php
    public function updateEntry($table, $val, $target) {
        $sql = "UPDATE " . $table . " SET ";
        //Set values
        $first = True;
        foreach ($val as $key => $value) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . ", ";
            }
            $sql = $sql . " " . ltrim($key, ':') . " = " . $key;
        }
        //Set target
        $sql = $sql . " WHERE";
        $first = True;
        foreach ($target as $key => $value) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . " AND ";
            }
            $sql = $sql . " " . ltrim($key, ':') . " = " . $key;
        }
        $query = $this->db->prepare($sql);
        $pars = array_merge($val, $target);
        // echo '[ PDO DEBUG ]: ' . Helper::debugPDO($sql, $pars);  exit();
        $query->execute($pars);
    }
```



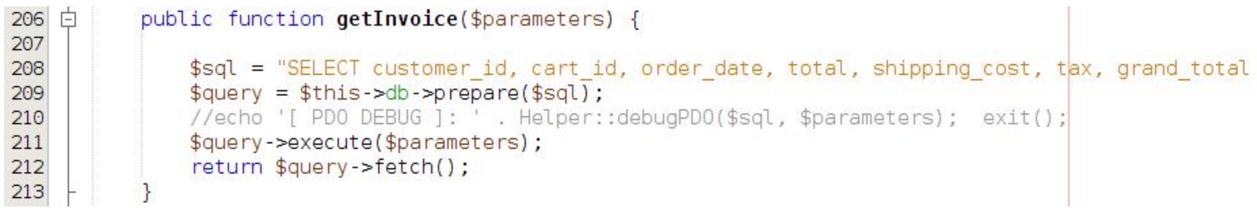
```php
    public function updateCartItem($uid, $pid, $qty) {
        $cid = $this->getUserCart($uid);
        $sql = "UPDATE cart_item SET item_qty = :qty WHERE cart_id = :cid AND product_id = :pid";
        $query = $this->db->prepare($sql);
        $parameters = array(':cid' => $cid, ':pid' => $pid, ':qty' => $qty);
        //echo '[ PDO DEBUG ]: ' . Helper::debugPDO($sql, $parameters);  exit();
        $query->execute($parameters);
    }
```



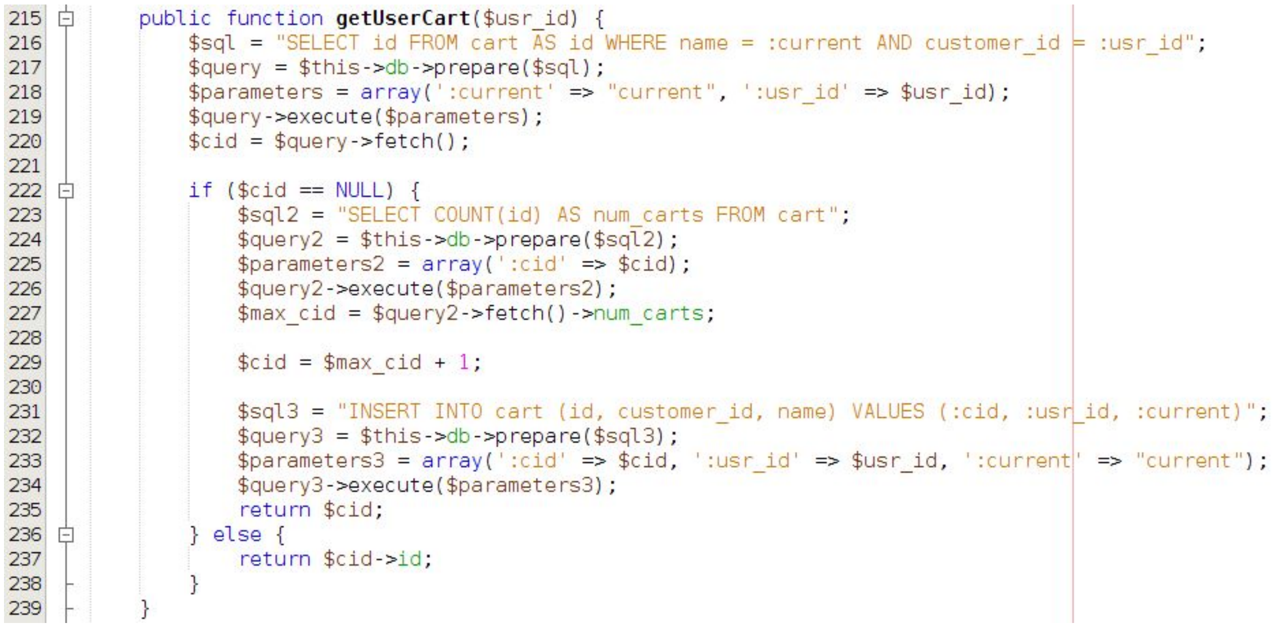
```php
    //SELECT

    public function getAllEntriesAdv($table, $val, $target) {
        $sql = "SELECT ";
        //Set values
        $first = True;
        foreach ($val as $key) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . ", ";
            }
            $sql = $sql . ltrim($key, ':');
        }

        $sql = $sql . " FROM " . $table;

        if (count($target) > 0) {
            //Set target
            $sql = $sql . " WHERE";
            $first = True;
            foreach ($target as $key => $value) {
                if ($first) {
                    $first = False;
                } else {
                    $sql = $sql . " AND";
                }
                $sql = $sql . " " . ltrim($key, ':') . " like " . $key;
            }
        }
        $query = $this->db->prepare($sql);
        // echo '[ PDO DEBUG ]: ' . Helper::debugPDO($sql, $target);  exit();
        $query->execute($target);
        return $query->fetchAll();
    }
```



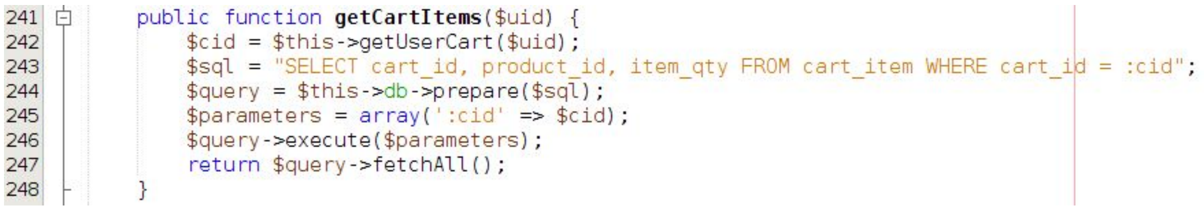
```php
    public function getAllEntriesAdv2($table, $val, $target, $column, $order) {
        $sql = "SELECT ";
        //Set values
        $first = True;
        foreach ($val as $key) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . ", ";
            }
            $sql = $sql . ltrim($key, ':');
        }

        $sql = $sql . " FROM " . $table;

        if (count($target) > 0) {
            //Set target
            $sql = $sql . " WHERE";
            $first = True;
            foreach ($target as $key => $value) {
                if ($first) {
                    $first = False;
                } else {
                    $sql = $sql . " AND";
                }
                $sql = $sql . " " . ltrim($key, ':') . " like " . $key;
            }
        }

        $sql = $sql  . " ORDER BY " . $column . " " . $order;
        $query = $this->db->prepare($sql);
        // echo '[ PDO DEBUG ]: ' . Helper::debugPDO($sql, $target);  exit();
        $query->execute($target);
        return $query->fetchAll();
    }
```



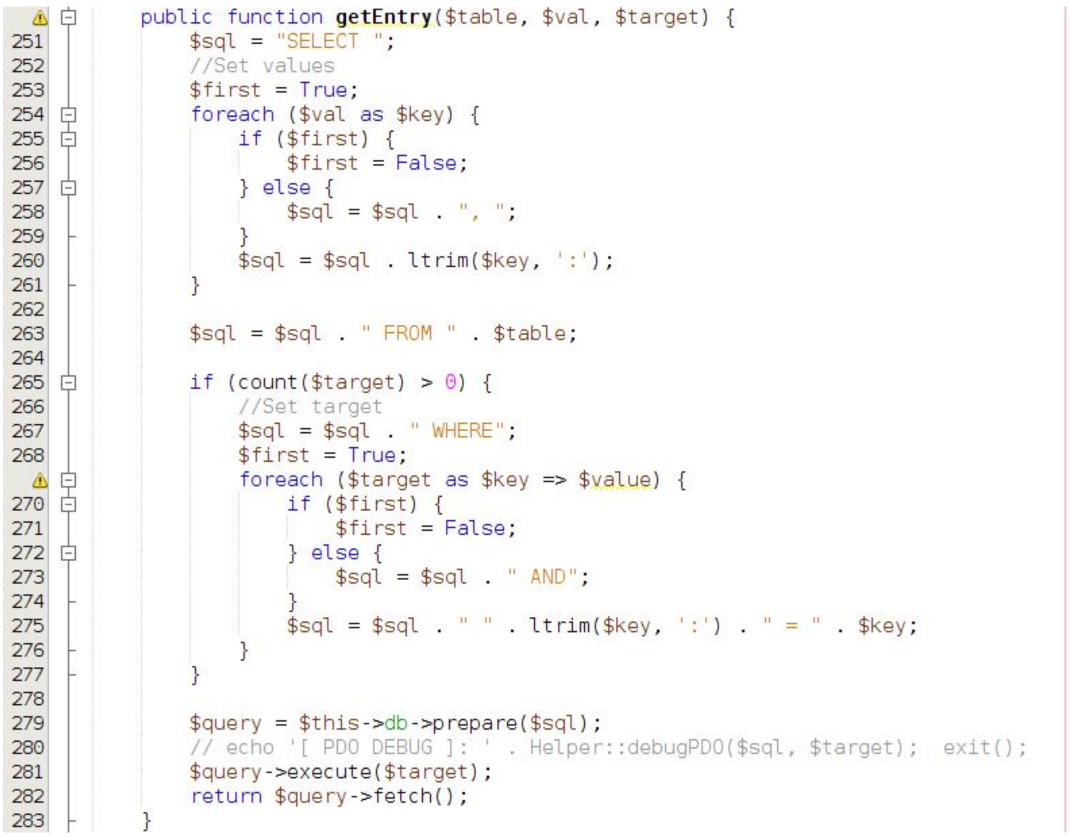
```php
    public function getInvoice($parameters) {

        $sql = "SELECT customer_id, cart_id, order_date, total, shipping_cost, tax, grand_total
        $query = $this->db->prepare($sql);
        //echo '[ PDO DEBUG ]: ' . Helper::debugPDO($sql, $parameters);  exit();
        $query->execute($parameters);
        return $query->fetch();
    }
```



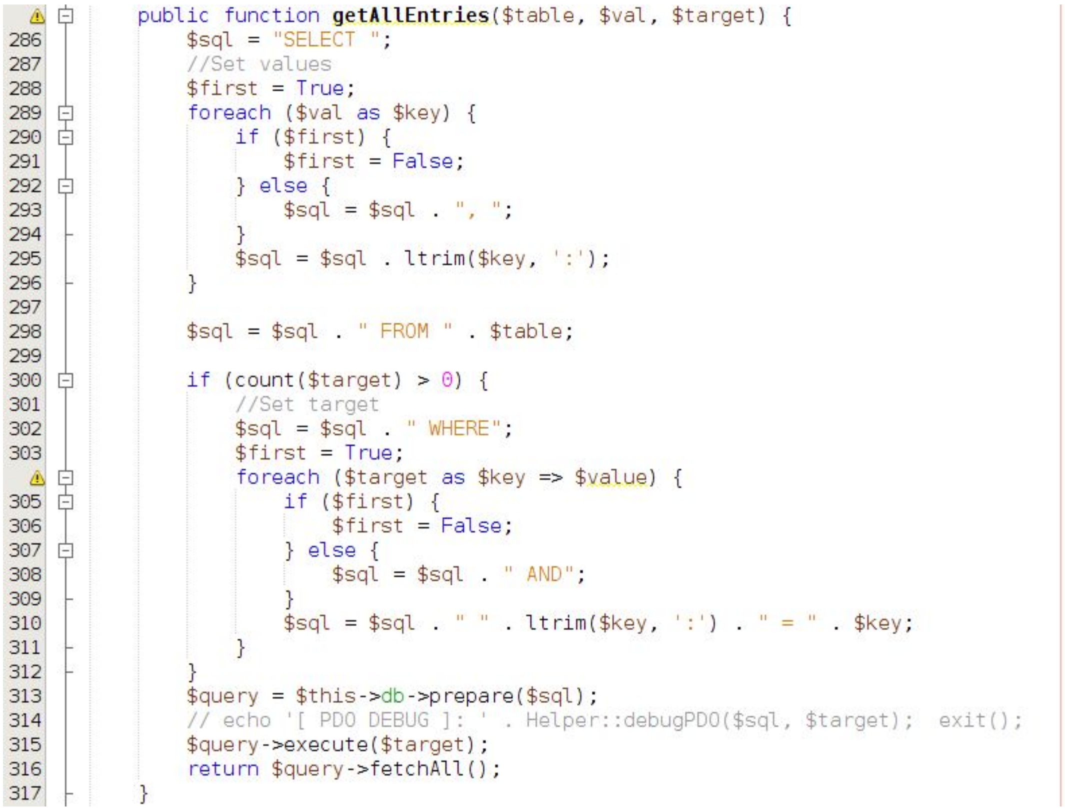
```php
    public function getUserCart($usr_id) {
        $sql = "SELECT id FROM cart AS id WHERE name = :current AND customer_id = :usr_id";
        $query = $this->db->prepare($sql);
        $parameters = array(':current' => "current", ':usr_id' => $usr_id);
        $query->execute($parameters);
        $cid = $query->fetch();

        if ($cid == NULL) {
            $sql2 = "SELECT COUNT(id) AS num_carts FROM cart";
            $query2 = $this->db->prepare($sql2);
            $parameters2 = array(':cid' => $cid);
            $query2->execute($parameters2);
            $max_cid = $query2->fetch()->num_carts;

            $cid = $max_cid + 1;

            $sql3 = "INSERT INTO cart (id, customer_id, name) VALUES (:cid, :usr_id, :current)";
            $query3 = $this->db->prepare($sql3);
            $parameters3 = array(':cid' => $cid, ':usr_id' => $usr_id, ':current' => "current");
            $query3->execute($parameters3);
            return $cid;
        } else {
            return $cid->id;
        }
    }
```



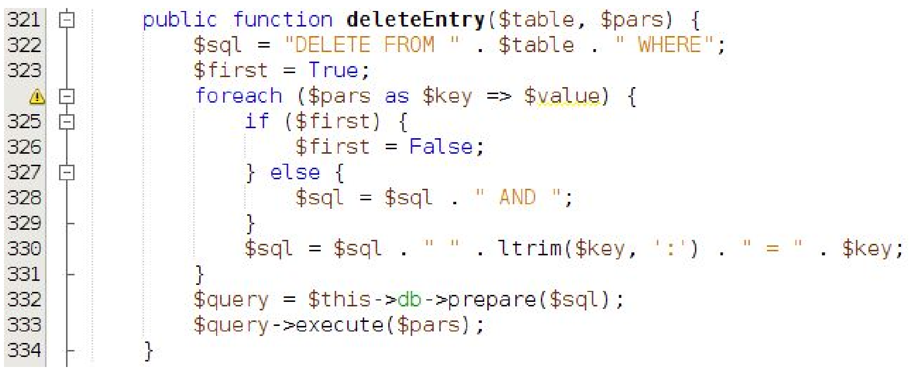
```php
    public function getCartItems($uid) {
        $cid = $this->getUserCart($uid);
        $sql = "SELECT cart_id, product_id, item_qty FROM cart_item WHERE cart_id = :cid";
        $query = $this->db->prepare($sql);
        $parameters = array(':cid' => $cid);
        $query->execute($parameters);
        return $query->fetchAll();
    }
```



```php
public function getEntry($table, $val, $target) {
    $sql = "SELECT ";
    //Set values
    $first = True;
    foreach ($val as $key) {
        if ($first) {
            $first = False;
        } else {
            $sql = $sql . ", ";
        }
        $sql = $sql . ltrim($key, ':');
    }

    $sql = $sql . " FROM " . $table;

    if (count($target) > 0) {
        //Set target
        $sql = $sql . " WHERE";
        $first = True;
        foreach ($target as $key => $value) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . " AND";
            }
            $sql = $sql . " " . ltrim($key, ':') . " = " . $key;
        }
    }

    $query = $this->db->prepare($sql);
    // echo '[ PDO DEBUG ]: ' . Helper::debugPDO($sql, $target);  exit();
    $query->execute($target);
    return $query->fetch();
}
```

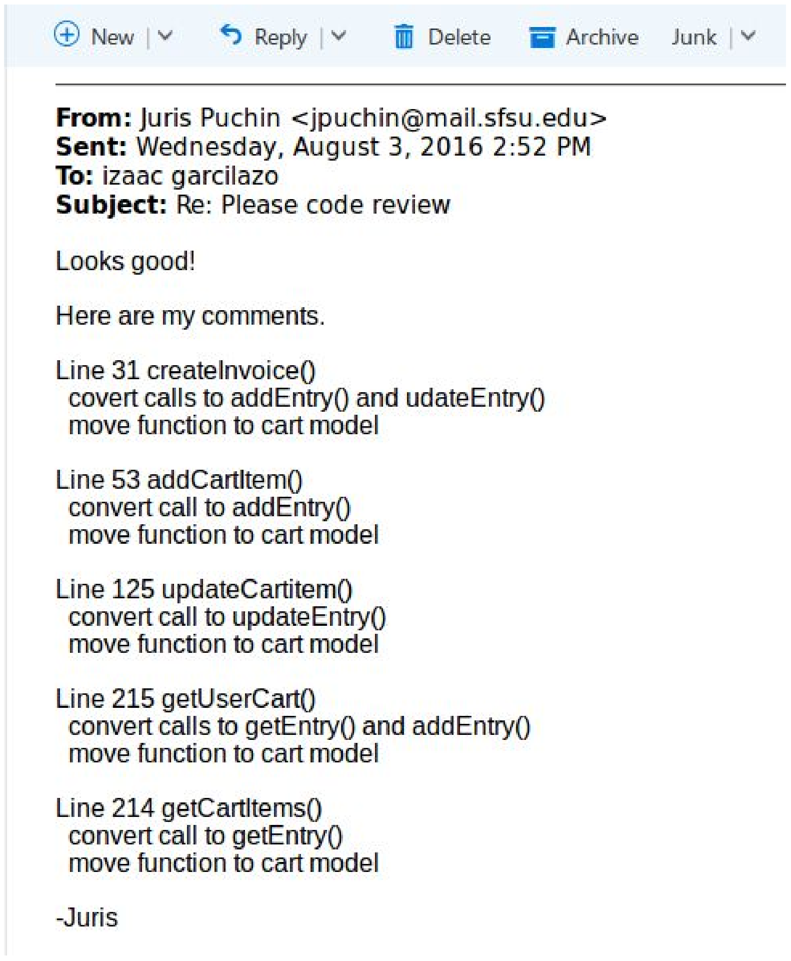


```php
public function getAllEntries($table, $val, $target) {
    $sql = "SELECT ";
    //Set values
    $first = True;
    foreach ($val as $key) {
        if ($first) {
            $first = False;
        } else {
            $sql = $sql . ", ";
        }
        $sql = $sql . ltrim($key, ':');
    }

    $sql = $sql . " FROM " . $table;

    if (count($target) > 0) {
        //Set target
        $sql = $sql . " WHERE";
        $first = True;
        foreach ($target as $key => $value) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . " AND";
            }
            $sql = $sql . " " . ltrim($key, ':') . " = " . $key;
        }
    }
    $query = $this->db->prepare($sql);
    // echo '[ PDO DEBUG ]: ' . Helper::debugPDO($sql, $target);  exit();
    $query->execute($target);
    return $query->fetchAll();
}
```



```
    public function deleteEntry($table, $pars) {
        $sql = "DELETE FROM " . $table . " WHERE";
        $first = True;
        foreach ($pars as $key => $value) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . " AND ";
            }
            $sql = $sql . " " . ltrim($key, ':') . " = " . $key;
        }
        $query = $this->db->prepare($sql);
        $query->execute($pars);
    }
```

Juris reads the code and sends an email to Izaac containing comments about the code.



**From:** Juris Puchin <jpuchin@mail.sfsu.edu>
**Sent:** Wednesday, August 3, 2016 2:52 PM
**To:** izaac garcilazo
**Subject:** Re: Please code review

Looks good!

Here are my comments.

Line 31 createInvoice()
  covert calls to addEntry() and udateEntry()
  move function to cart model

Line 53 addCartItem()
  convert call to addEntry()
  move function to cart model

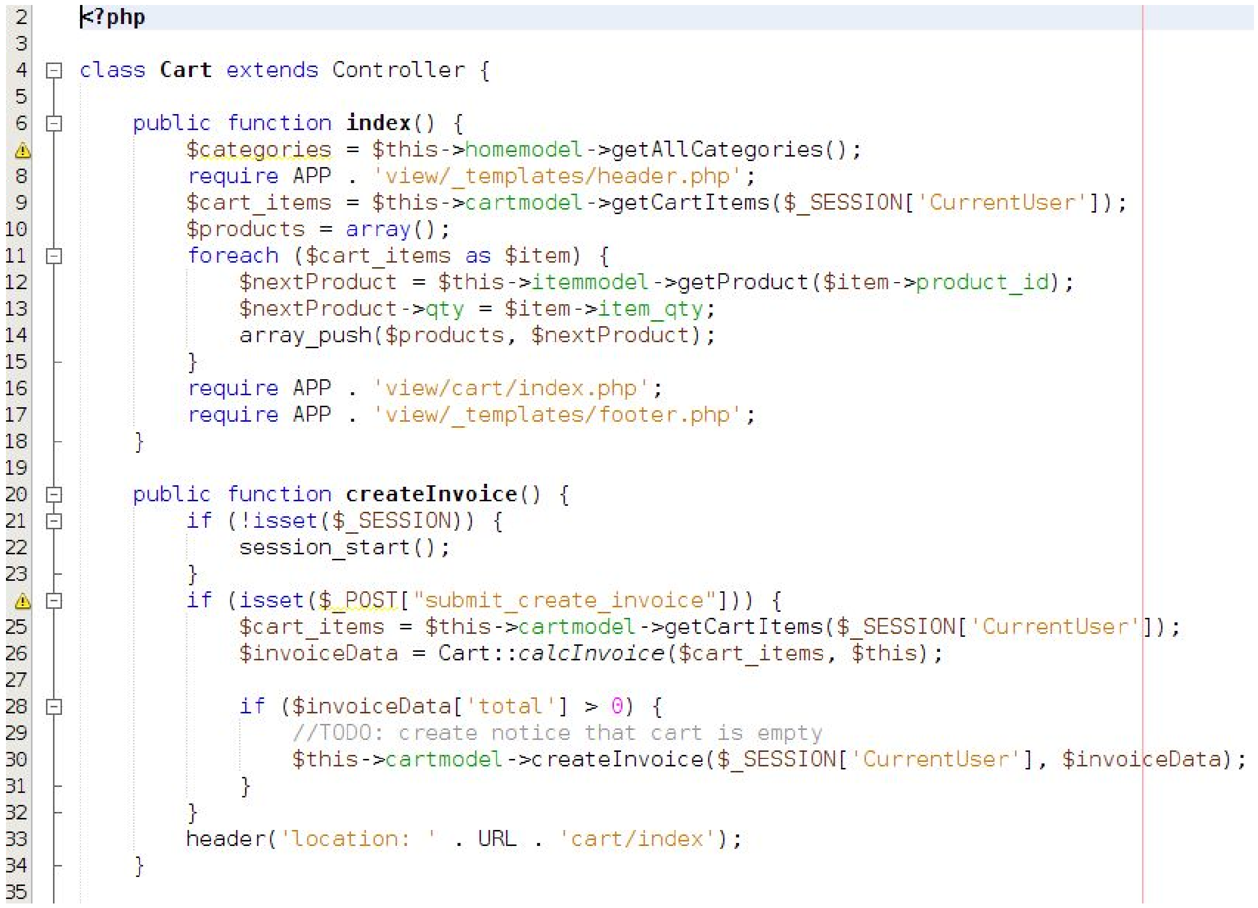
Line 125 updateCartitem()
  convert call to updateEntry()
  move function to cart model

Line 215 getUserCart()
  convert calls to getEntry() and addEntry()
  move function to cart model

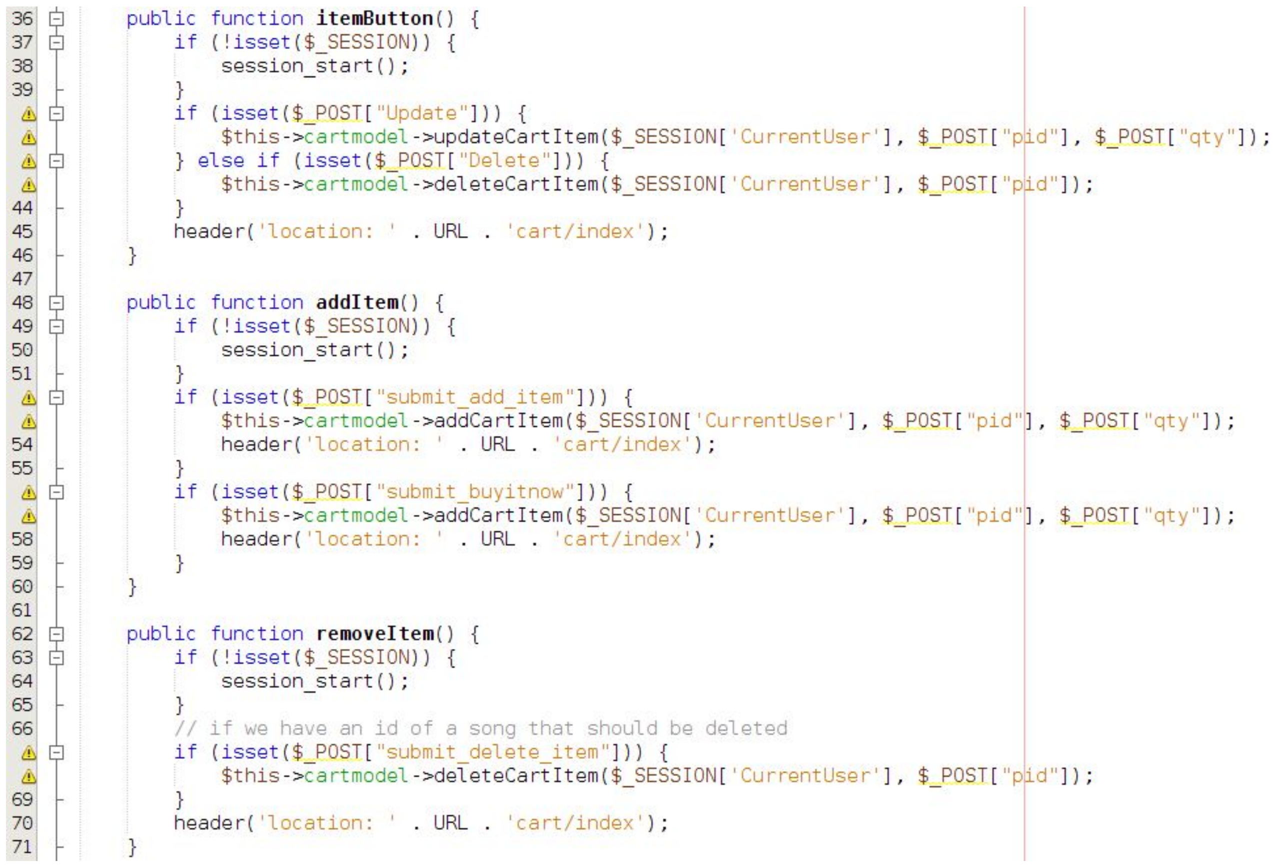
Line 214 getCartItems()
  convert call to getEntry()
  move function to cart model

-Juris

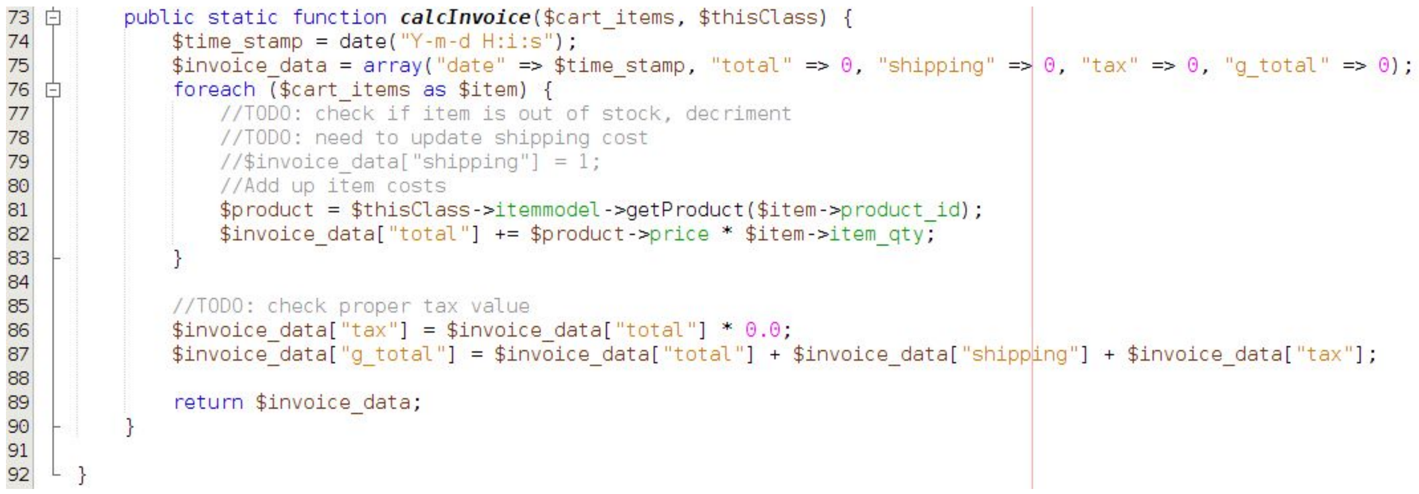
Izaac gets the code review and fixes the problems.
cart.php



```php
<?php

class Cart extends Controller {

    public function index() {
        $categories = $this->homemodel->getAllCategories();
        require APP . 'view/_templates/header.php';
        $cart_items = $this->cartmodel->getCartItems($_SESSION['CurrentUser']);
        $products = array();
        foreach ($cart_items as $item) {
            $nextProduct = $this->itemmodel->getProduct($item->product_id);
            $nextProduct->qty = $item->item_qty;
            array_push($products, $nextProduct);
        }
        require APP . 'view/cart/index.php';
        require APP . 'view/_templates/footer.php';
    }

    public function createInvoice() {
        if (!isset($_SESSION)) {
            session_start();
        }
        if (isset($_POST["submit_create_invoice"])) {
            $cart_items = $this->cartmodel->getCartItems($_SESSION['CurrentUser']);
            $invoiceData = Cart::calcInvoice($cart_items, $this);

            if ($invoiceData['total'] > 0) {
                //TODO: create notice that cart is empty
                $this->cartmodel->createInvoice($_SESSION['CurrentUser'], $invoiceData);
            }
        }
        header('location: ' . URL . 'cart/index');
    }

```

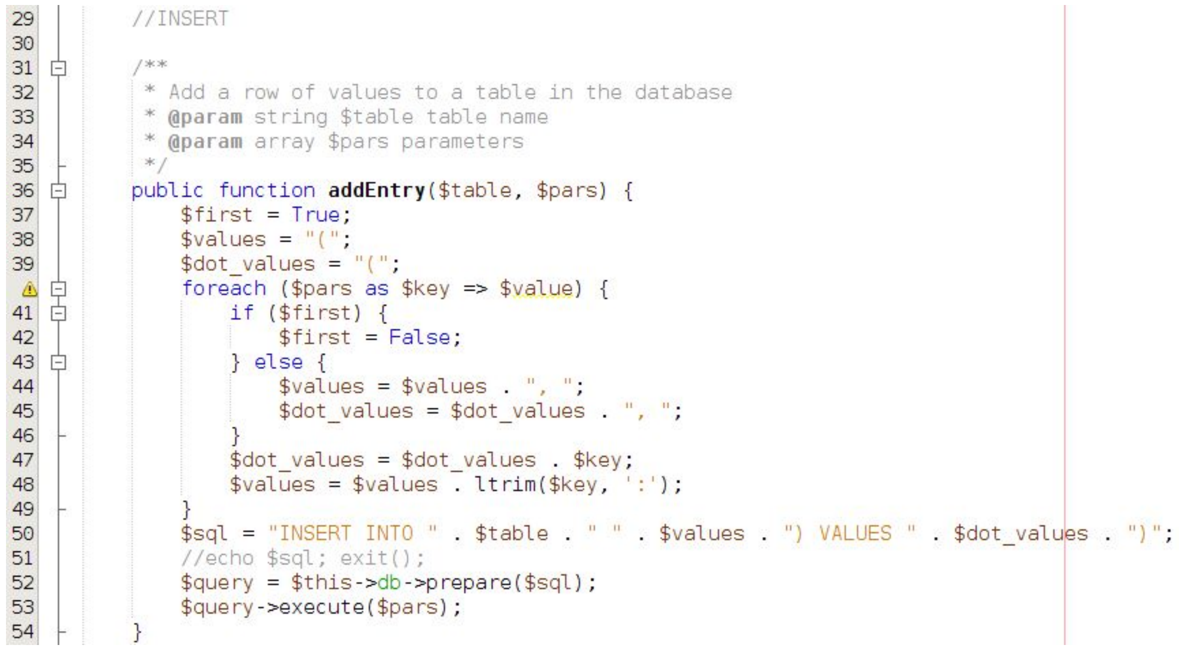


```php
    public function itemButton() {
        if (!isset($_SESSION)) {
            session_start();
        }
        if (isset($_POST["Update"])) {
            $this->cartmodel->updateCartItem($_SESSION['CurrentUser'], $_POST["pid"], $_POST["qty"]);
        } else if (isset($_POST["Delete"])) {
            $this->cartmodel->deleteCartItem($_SESSION['CurrentUser'], $_POST["pid"]);
        }
        header('location: ' . URL . 'cart/index');
    }

    public function addItem() {
        if (!isset($_SESSION)) {
            session_start();
        }
        if (isset($_POST["submit_add_item"])) {
            $this->cartmodel->addCartItem($_SESSION['CurrentUser'], $_POST["pid"], $_POST["qty"]);
            header('location: ' . URL . 'cart/index');
        }
        if (isset($_POST["submit_buyitnow"])) {
            $this->cartmodel->addCartItem($_SESSION['CurrentUser'], $_POST["pid"], $_POST["qty"]);
            header('location: ' . URL . 'cart/index');
        }
    }

    public function removeItem() {
        if (!isset($_SESSION)) {
            session_start();
        }
        // if we have an id of a song that should be deleted
        if (isset($_POST["submit_delete_item"])) {
            $this->cartmodel->deleteCartItem($_SESSION['CurrentUser'], $_POST["pid"]);
        }
        header('location: ' . URL . 'cart/index');
    }
```

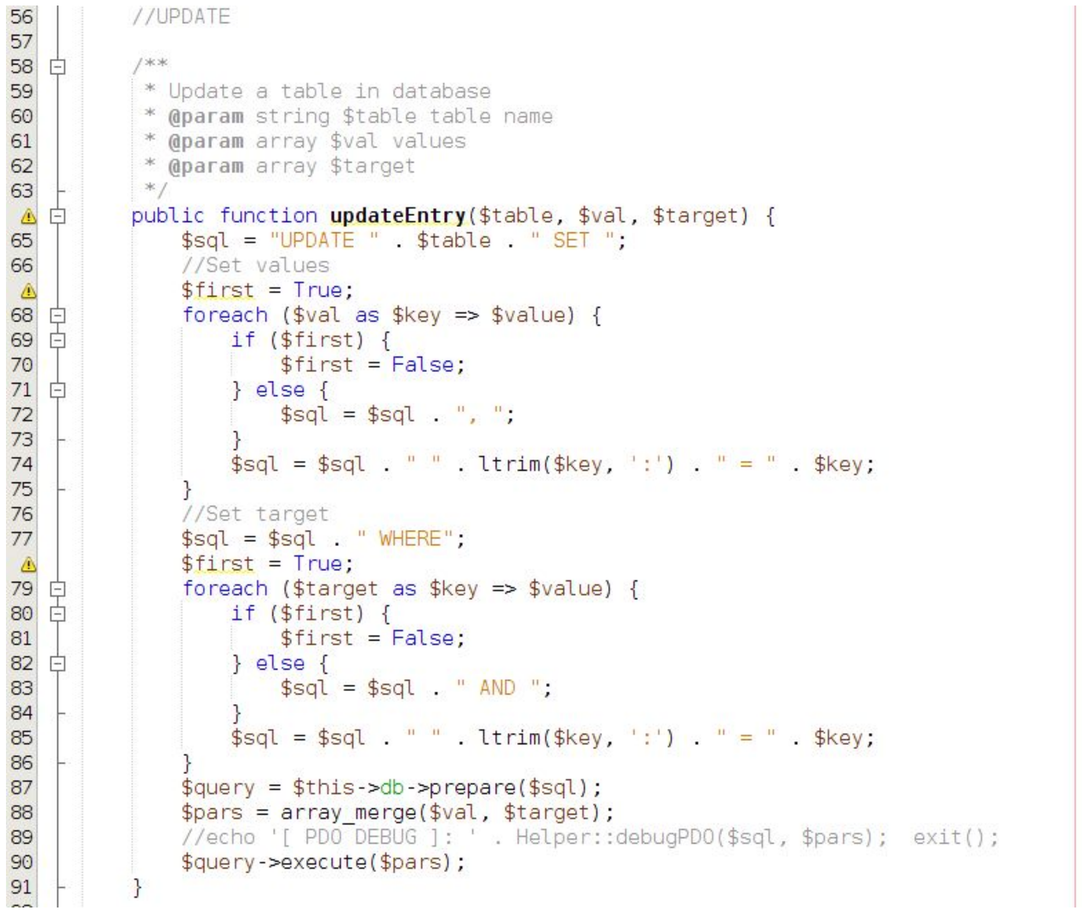


```php
    public static function calcInvoice($cart_items, $thisClass) {
        $time_stamp = date("Y-m-d H:i:s");
        $invoice_data = array("date" => $time_stamp, "total" => 0, "shipping" => 0, "tax" => 0, "g_total" => 0);
        foreach ($cart_items as $item) {
            //TODO: check if item is out of stock, decriment
            //TODO: need to update shipping cost
            //$invoice_data["shipping"] = 1;
            //Add up item costs
            $product = $thisClass->itemmodel->getProduct($item->product_id);
            $invoice_data["total"] += $product->price * $item->item_qty;
        }

        //TODO: check proper tax value
        $invoice_data["tax"] = $invoice_data["total"] * 0.0;
        $invoice_data["g_total"] = $invoice_data["total"] + $invoice_data["shipping"] + $invoice_data["tax"];

        return $invoice_data;
    }

}
```

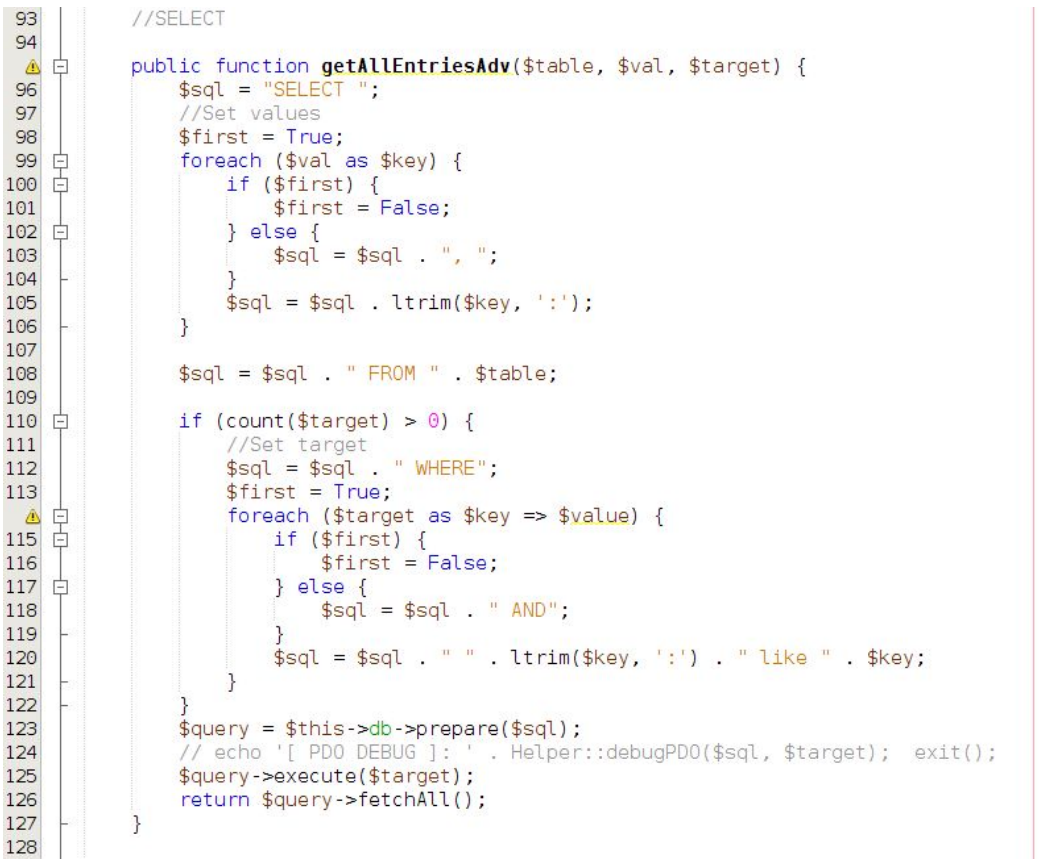
Sqlcalls.php



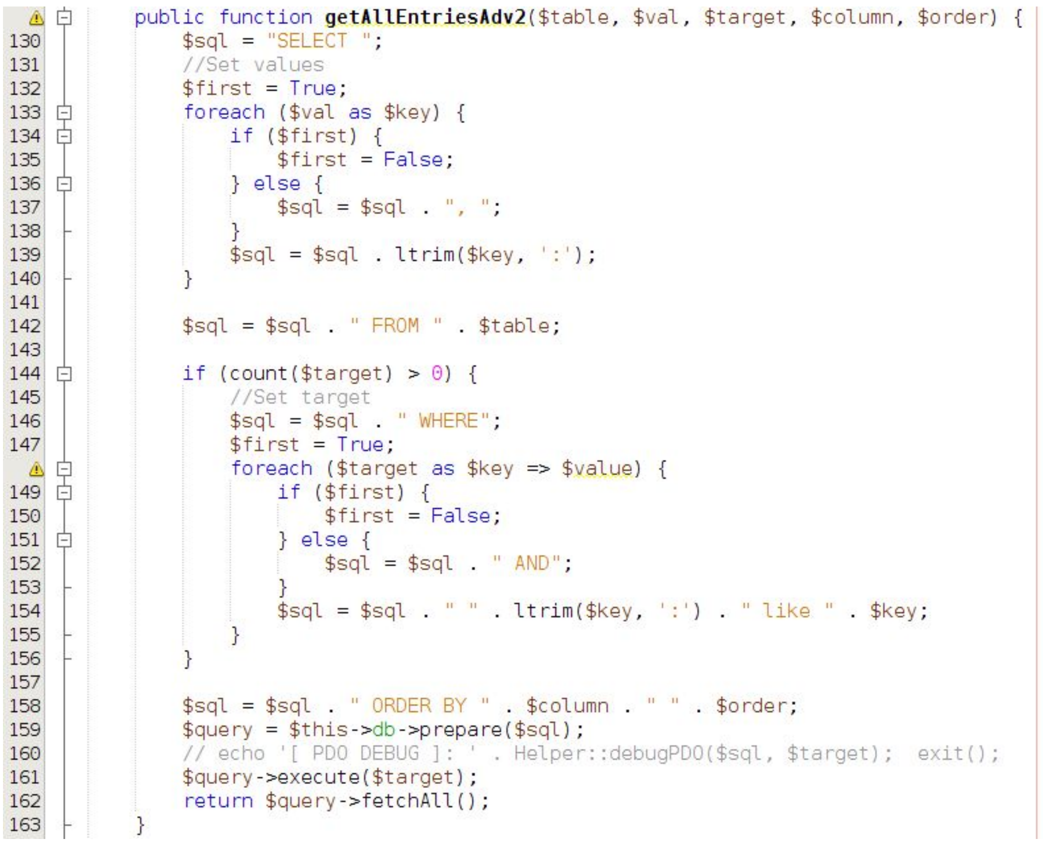
```php
    //INSERT

    /**
     * Add a row of values to a table in the database
     * @param string $table table name
     * @param array $pars parameters
     */
    public function addEntry($table, $pars) {
        $first = True;
        $values = "(";
        $dot_values = "(";
        foreach ($pars as $key => $value) {
            if ($first) {
                $first = False;
            } else {
                $values = $values . ", ";
                $dot_values = $dot_values . ", ";
            }
            $dot_values = $dot_values . $key;
            $values = $values . ltrim($key, ':');
        }
        $sql = "INSERT INTO " . $table . " " . $values . ") VALUES " . $dot_values . ")";
        //echo $sql; exit();
        $query = $this->db->prepare($sql);
        $query->execute($pars);
    }
```



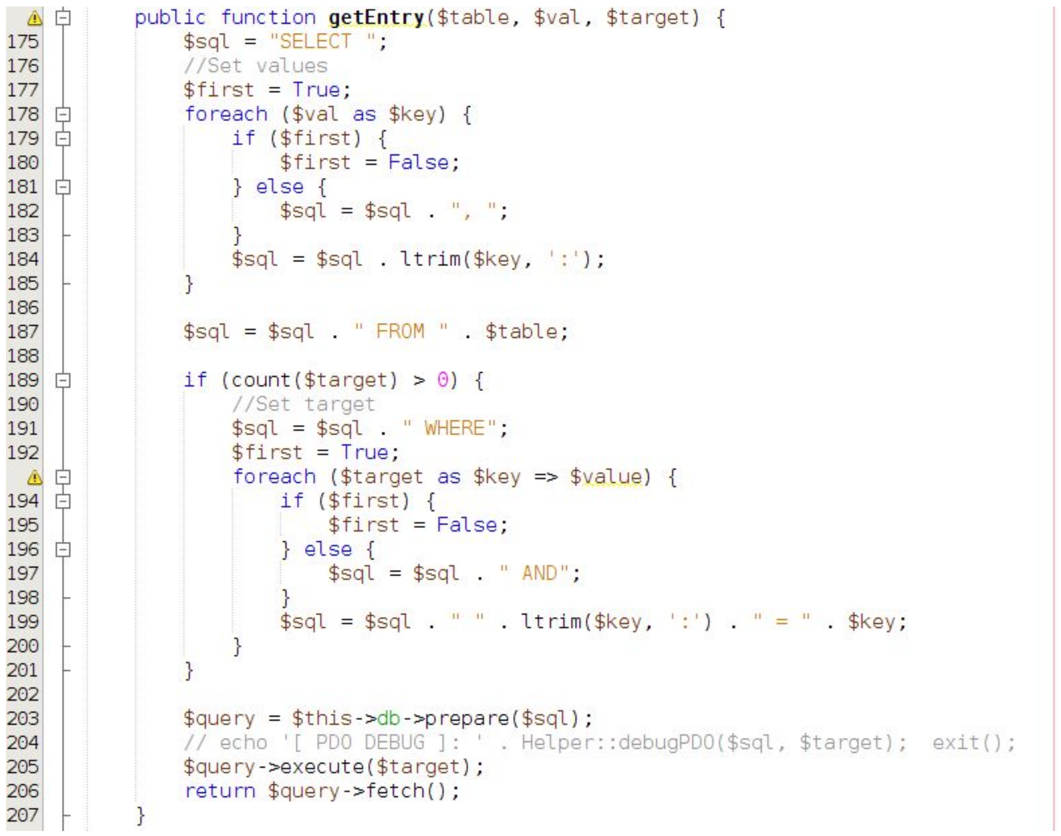
```php
    //UPDATE

    /**
     * Update a table in database
     * @param string $table table name
     * @param array $val values
     * @param array $target
     */
    public function updateEntry($table, $val, $target) {
        $sql = "UPDATE " . $table . " SET ";
        //Set values
        $first = True;
        foreach ($val as $key => $value) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . ", ";
            }
            $sql = $sql . " " . ltrim($key, ':') . " = " . $key;
        }
        //Set target
        $sql = $sql . " WHERE";
        $first = True;
        foreach ($target as $key => $value) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . " AND ";
            }
            $sql = $sql . " " . ltrim($key, ':') . " = " . $key;
        }
        $query = $this->db->prepare($sql);
        $pars = array_merge($val, $target);
        //echo '[ PDO DEBUG ]: ' . Helper::debugPDO($sql, $pars);  exit();
        $query->execute($pars);
    }
```



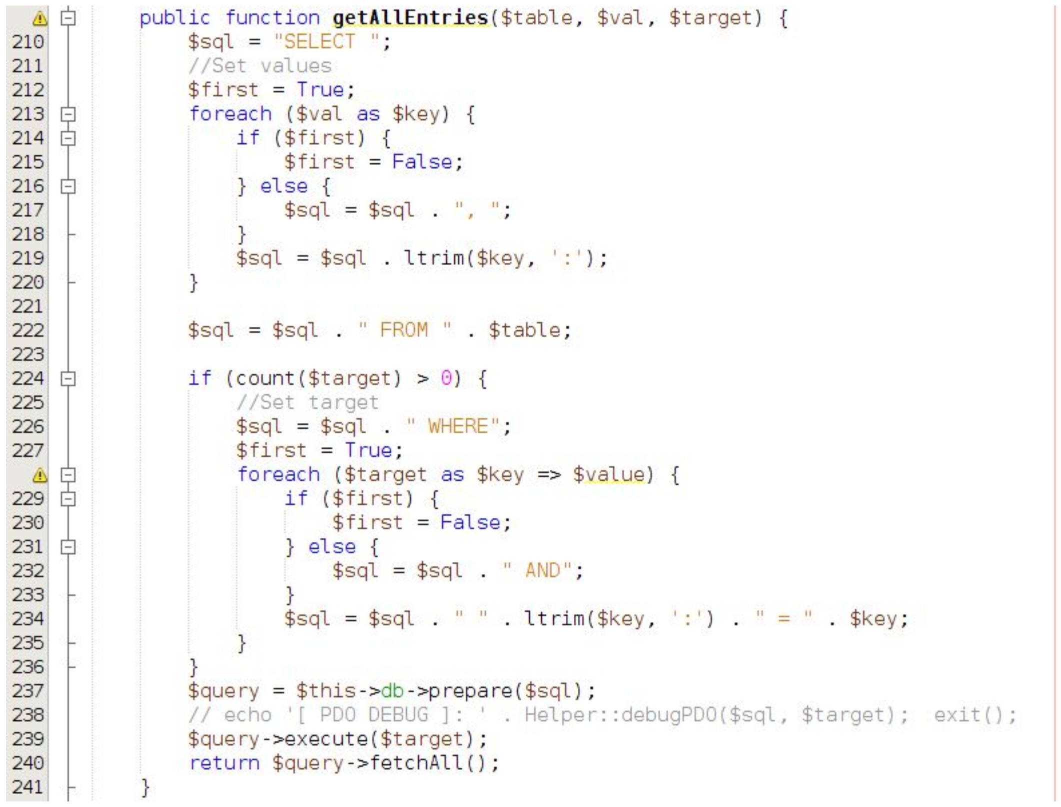
```php
    //SELECT

    public function getAllEntriesAdv($table, $val, $target) {
        $sql = "SELECT ";
        //Set values
        $first = True;
        foreach ($val as $key) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . ", ";
            }
            $sql = $sql . ltrim($key, ':');
        }

        $sql = $sql . " FROM " . $table;

        if (count($target) > 0) {
            //Set target
            $sql = $sql . " WHERE";
            $first = True;
            foreach ($target as $key => $value) {
                if ($first) {
                    $first = False;
                } else {
                    $sql = $sql . " AND";
                }
                $sql = $sql . " " . ltrim($key, ':') . " like " . $key;
            }
        }
        $query = $this->db->prepare($sql);
        // echo '[ PDO DEBUG ]: ' . Helper::debugPDO($sql, $target);  exit();
        $query->execute($target);
        return $query->fetchAll();
    }

```



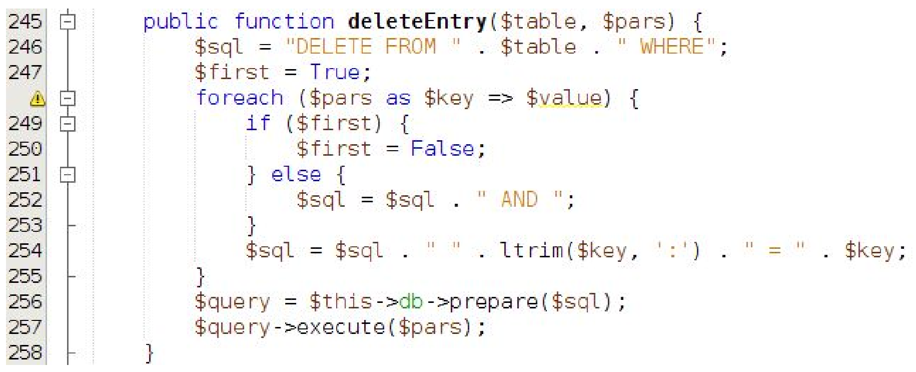
```php
    public function getAllEntriesAdv2($table, $val, $target, $column, $order) {
        $sql = "SELECT ";
        //Set values
        $first = True;
        foreach ($val as $key) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . ", ";
            }
            $sql = $sql . ltrim($key, ':');
        }

        $sql = $sql . " FROM " . $table;

        if (count($target) > 0) {
            //Set target
            $sql = $sql . " WHERE";
            $first = True;
            foreach ($target as $key => $value) {
                if ($first) {
                    $first = False;
                } else {
                    $sql = $sql . " AND";
                }
                $sql = $sql . " " . ltrim($key, ':') . " like " . $key;
            }
        }

        $sql = $sql . " ORDER BY " . $column . " " . $order;
        $query = $this->db->prepare($sql);
        // echo '[ PDO DEBUG ]: ' . Helper::debugPDO($sql, $target);  exit();
        $query->execute($target);
        return $query->fetchAll();
    }
```



```php
public function getEntry($table, $val, $target) {
    $sql = "SELECT ";
    //Set values
    $first = True;
    foreach ($val as $key) {
        if ($first) {
            $first = False;
        } else {
            $sql = $sql . ", ";
        }
        $sql = $sql . ltrim($key, ':');
    }

    $sql = $sql . " FROM " . $table;

    if (count($target) > 0) {
        //Set target
        $sql = $sql . " WHERE";
        $first = True;
        foreach ($target as $key => $value) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . " AND";
            }
            $sql = $sql . " " . ltrim($key, ':') . " = " . $key;
        }
    }

    $query = $this->db->prepare($sql);
    // echo '[ PDO DEBUG ]: ' . Helper::debugPDO($sql, $target);  exit();
    $query->execute($target);
    return $query->fetch();
}
```

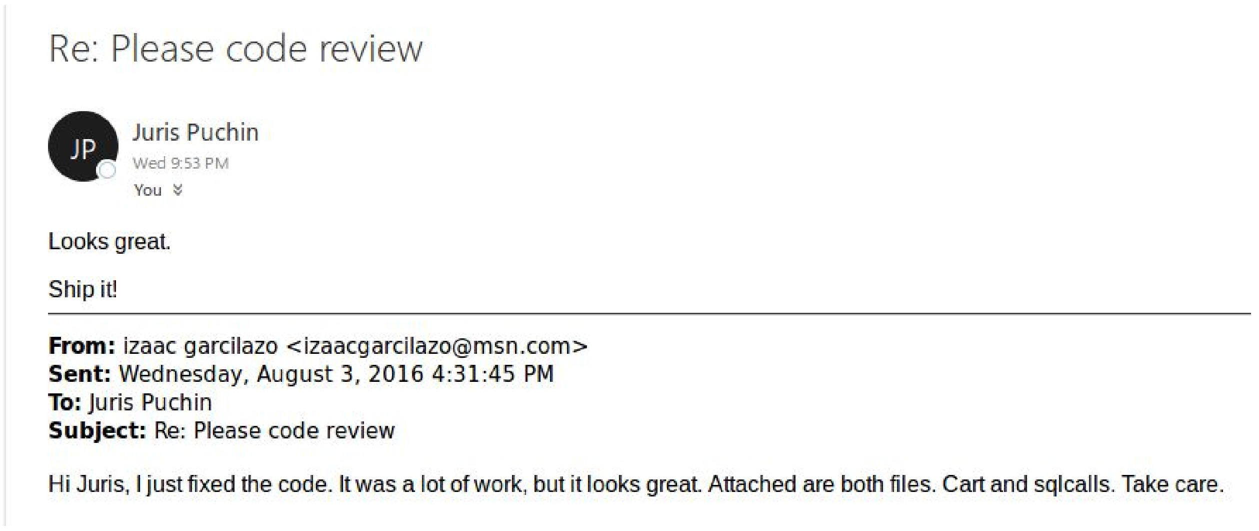


```php
public function getAllEntries($table, $val, $target) {
    $sql = "SELECT ";
    //Set values
    $first = True;
    foreach ($val as $key) {
        if ($first) {
            $first = False;
        } else {
            $sql = $sql . ", ";
        }
        $sql = $sql . ltrim($key, ':');
    }

    $sql = $sql . " FROM " . $table;

    if (count($target) > 0) {
        //Set target
        $sql = $sql . " WHERE";
        $first = True;
        foreach ($target as $key => $value) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . " AND";
            }
            $sql = $sql . " " . ltrim($key, ':') . " = " . $key;
        }
    }
    $query = $this->db->prepare($sql);
    // echo '[ PDO DEBUG ]: ' . Helper::debugPDO($sql, $target);  exit();
    $query->execute($target);
    return $query->fetchAll();
}
```



```php
    public function deleteEntry($table, $pars) {
        $sql = "DELETE FROM " . $table . " WHERE";
        $first = True;
        foreach ($pars as $key => $value) {
            if ($first) {
                $first = False;
            } else {
                $sql = $sql . " AND ";
            }
            $sql = $sql . " " . ltrim($key, ':') . " = " . $key;
        }
        $query = $this->db->prepare($sql);
        $query->execute($pars);
    }
```

Izaac sends the new code back to Juris for a second review. Juris approves the new code.



Re: Please code review

JP Juris Puchin
Wed 9:53 PM
You

Looks great.

Ship it!

---

**From:** izaac garcilazo <izaacgarcilazo@msn.com>
**Sent:** Wednesday, August 3, 2016 4:31:45 PM
**To:** Juris Puchin
**Subject:** Re: Please code review

Hi Juris, I just fixed the code. It was a lot of work, but it looks great. Attached are both files. Cart and sqlcalls. Take care.

## 5).Adherence to original non-functional spec:

1. Application shall be developed using class provided LAMP stack(**DONE**)
2. Application shall be developed using pre-approved set of SW development and collaborative tools provided in the class. Any other tools or frameworks have to be explicitly approved by Marc Sosnick on a case by case basis.(**DONE**)
3. Application shall be hosted and deployed on Amazon Web Services as specified in the class(**DONE**)
4. Application shall be optimized for standard desktop/laptop browsers, and shall render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome and IE. It shall degrade nicely for different sized windows using class approved programming technology and frameworks(**DONE**)
5. Data shall be stored in the database on the class server in the team's account(**DONE**)
6. Application shall be served from the team's account (**DONE**)
7. No more than 50 concurrent users shall be accessing the application at any time
8. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. (**DONE**)
9. The language used shall be English.(**DONE**)
10. Application shall be very easy to use and intuitive. No prior training shall be required to use the website.(**DONE**)
11. Google analytics shall be added for major site functions.
12. Messaging between users shall be done only by class approved methods to avoid issues of security with e-mail services.(**DONE**)
13. Site security: basic best practices to be applied (as covered in the class)(done)
14. Modern SE processes and practices must be used as specified in the class, including collaborative and continuous SW development, using the tools approved by instructors(**DONE**)
15. The website shall prominently display the following text on all pages "SFSU/FAU/Fulda Software Engineering Project, Summer 2016. For Demonstration Only". (Important so as to not confuse this with a real application).(**DONE**)