|  |
| --- |
| **CS4313** |
| Deliverable 4 |
|  |
| All Database tables |
| Description of all functionality |
| All HTML Pages |
| All PHP scripts |
| List of copied material and sources |
|  |
| **David O Neill 0813001** |
| **James McHugh 0052833** |
| **Keiran Duggan 0866504** |
|  |
| **19/4/2010** |
|  |

Table of Contents

[Database 4](#_Toc259390289)

[Tables 4](#_Toc259390290)

[Auction items 4](#_Toc259390291)

[SQL 5](#_Toc259390292)

[Auction pictures 6](#_Toc259390293)

[SQL 6](#_Toc259390294)

[Auction categories 7](#_Toc259390295)

[SQL 7](#_Toc259390296)

[Auction sub categories 7](#_Toc259390297)

[SQL 7](#_Toc259390298)

[Messages 8](#_Toc259390299)

[SQL 8](#_Toc259390300)

[Auction Bids 9](#_Toc259390301)

[SQL 9](#_Toc259390302)

[Item watching 10](#_Toc259390303)

[SQL 10](#_Toc259390304)

[Auction feedback 11](#_Toc259390305)

[SQL 11](#_Toc259390306)

[Addresses 12](#_Toc259390307)

[SQL 12](#_Toc259390308)

[OU users 13](#_Toc259390309)

[SQL 13](#_Toc259390310)

[OU groups 14](#_Toc259390311)

[SQL 14](#_Toc259390312)

[Countries 14](#_Toc259390313)

[SQL 14](#_Toc259390314)

[Locale 15](#_Toc259390315)

[SQL 15](#_Toc259390316)

[Referential Integrity Constraints 16](#_Toc259390317)

[SQL 16](#_Toc259390318)

[Overview 17](#_Toc259390319)

[Relations Overview 17](#_Toc259390320)

[Design & Implementation 18](#_Toc259390321)

[Requirements 18](#_Toc259390322)

[Major Processes 18](#_Toc259390323)

[User Registration 18](#_Toc259390324)

[Update registration 19](#_Toc259390325)

[Login 19](#_Toc259390326)

[Logout 19](#_Toc259390327)

[Submit Auction 20](#_Toc259390328)

[Browse Auctions 20](#_Toc259390329)

[View auction 21](#_Toc259390330)

[Bidding 21](#_Toc259390331)

[Searching 21](#_Toc259390332)

[Feedback 22](#_Toc259390333)

[Watch item 22](#_Toc259390334)

[Administrator 22](#_Toc259390335)

[Delete User 22](#_Toc259390336)

[Ban User 22](#_Toc259390337)

[Update auction 22](#_Toc259390338)

[Delete auction 22](#_Toc259390339)

[Non Functional Requirements 23](#_Toc259390340)

[Model – View - Controller 23](#_Toc259390341)

[Sample controller 23](#_Toc259390342)

[Sample Template 24](#_Toc259390343)

[View Constructor 24](#_Toc259390344)

[View Includes 24](#_Toc259390345)

[View Processing 25](#_Toc259390346)

[View Process Directive 26](#_Toc259390347)

[View Output 26](#_Toc259390348)

[Localisation 27](#_Toc259390349)

[Security 28](#_Toc259390350)

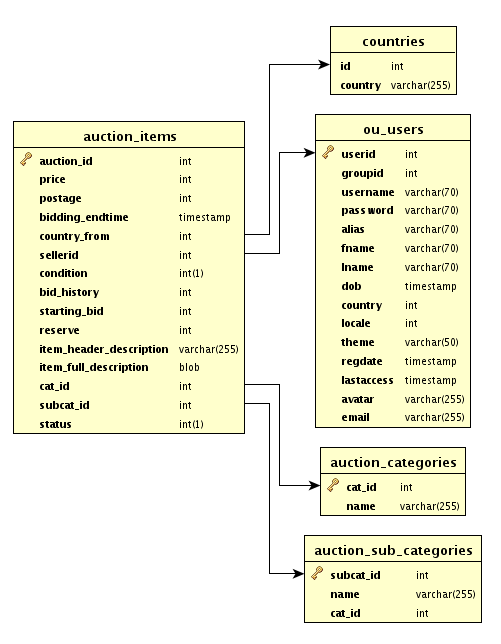
[Prototype UML Class Diagram 29](#_Toc259390351)

[Copied materials 30](#_Toc259390352)

# Database

## Tables

### Auction items



#### SQL

CREATE TABLE auction\_items (

auction\_id int(11) NOT NULL AUTO\_INCREMENT,

price int(11) NOT NULL,

postage int(11) NOT NULL,

bidding\_endtime timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',

country\_from int(11) NOT NULL,

sellerid int(11) NOT NULL,

`condition` int(1) NOT NULL,

bid\_history int(11) NOT NULL,

starting\_bid int(11) NOT NULL,

reserve int(11) NOT NULL,

item\_header\_description varchar(255) NOT NULL,

item\_full\_description blob NOT NULL,

cat\_id int(11) NOT NULL,

subcat\_id int(11) NOT NULL,

`status` int(1) NOT NULL,

PRIMARY KEY (auction\_id),

KEY country\_from (country\_from),

KEY sellerid (sellerid),

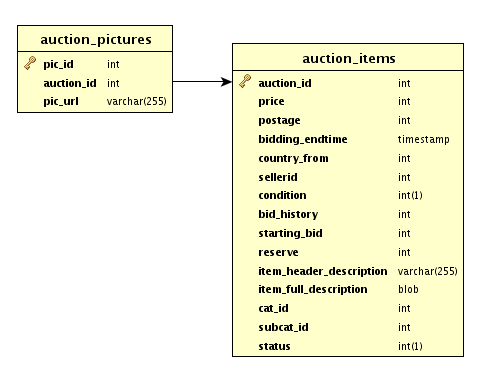
KEY cat\_id (cat\_id),

KEY subcat\_id (subcat\_id)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

### 

### Auction pictures



#### SQL

CREATE TABLE IF NOT EXISTS `auction\_pictures` (

`pic\_id` int(11) NOT NULL AUTO\_INCREMENT,

`auction\_id` int(11) NOT NULL,

`pic\_url` varchar(255) NOT NULL,

PRIMARY KEY (`pic\_id`),

KEY `auction\_id` (`auction\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=9 ;

### Auction categories



#### SQL

CREATE TABLE auction\_categories (

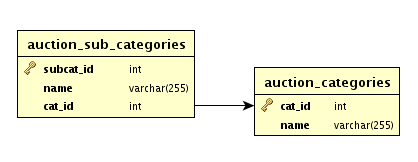
cat\_id int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(255) NOT NULL,

PRIMARY KEY (cat\_id)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

### Auction sub categories



#### SQL

CREATE TABLE auction\_sub\_categories (

subcat\_id int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(255) NOT NULL,

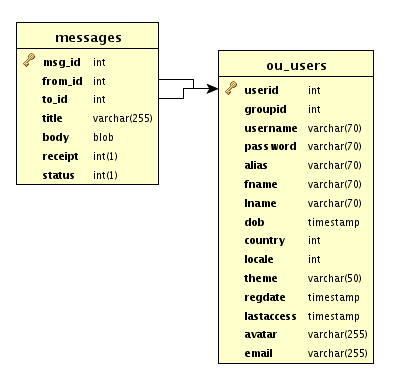
cat\_id int(11) NOT NULL,

PRIMARY KEY (subcat\_id),

KEY cat\_id (cat\_id)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

### Messages



#### SQL

CREATE TABLE messages (

msg\_id int(11) NOT NULL AUTO\_INCREMENT,

from\_id int(11) NOT NULL,

to\_id int(11) NOT NULL,

title varchar(255) NOT NULL,

body blob NOT NULL,

receipt int(1) NOT NULL,

`status` int(1) NOT NULL,

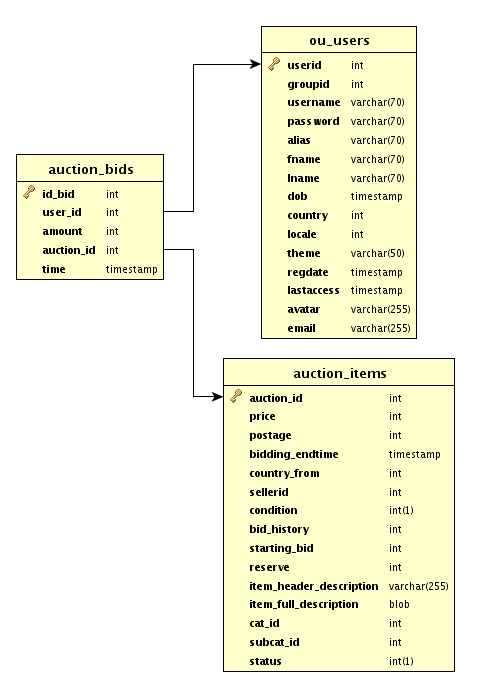
PRIMARY KEY (msg\_id),

KEY from\_id (from\_id),

KEY to\_id (to\_id)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

### Auction Bids



#### SQL

CREATE TABLE auction\_bids (

id\_bid int(11) NOT NULL AUTO\_INCREMENT,

user\_id int(11) NOT NULL,

amount int(11) NOT NULL,

auction\_id int(11) NOT NULL,

`time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,

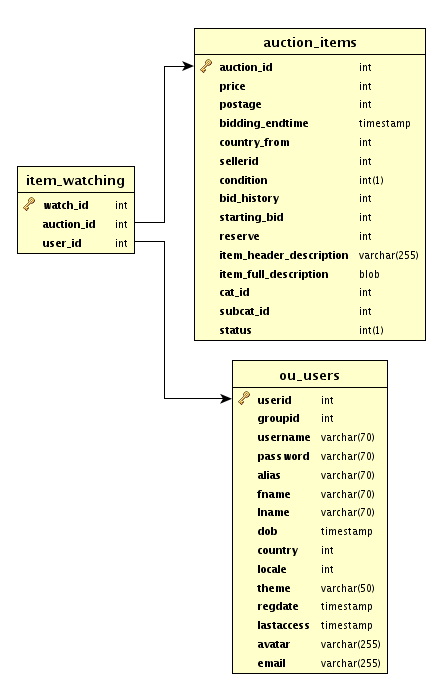
PRIMARY KEY (id\_bid),

KEY auction\_id (auction\_id),

KEY user\_id (user\_id)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

### Item watching



#### SQL

CREATE TABLE item\_watching (

watch\_id int(11) NOT NULL AUTO\_INCREMENT,

auction\_id int(11) NOT NULL,

user\_id int(11) NOT NULL,

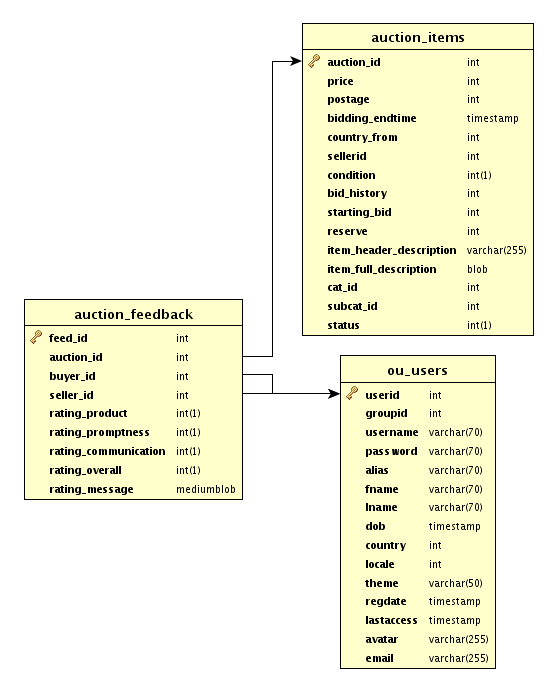
PRIMARY KEY (watch\_id),

KEY auction\_id (auction\_id),

KEY user\_id (user\_id)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

### Auction feedback



#### SQL

CREATE TABLE auction\_feedback (

feed\_id int(11) NOT NULL AUTO\_INCREMENT,

auction\_id int(11) NOT NULL,

buyer\_id int(11) NOT NULL,

seller\_id int(11) NOT NULL,

rating\_product int(1) NOT NULL,

rating\_promptness int(1) NOT NULL,

rating\_communication int(1) NOT NULL,

rating\_overall int(1) NOT NULL,

rating\_message mediumblob NOT NULL,

PRIMARY KEY (feed\_id),

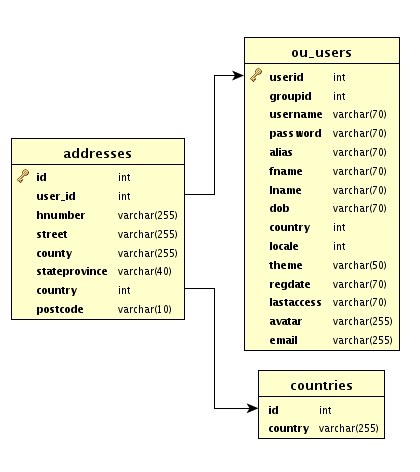
KEY auction\_id (auction\_id),

KEY buyer\_id (buyer\_id),

KEY seller\_id (seller\_id)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

### Addresses



#### SQL

CREATE TABLE addresses (

id int(11) NOT NULL AUTO\_INCREMENT,

user\_id int(11) NOT NULL,

hnumber varchar(255) NOT NULL,

street varchar(255) NOT NULL,

county varchar(255) NOT NULL,

stateprovince varchar(40) NOT NULL,

country int(11) NOT NULL,

postcode varchar(10) NOT NULL,

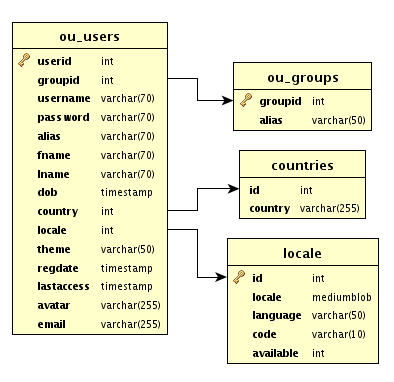
PRIMARY KEY (id),

KEY user\_id (user\_id),

KEY country (country)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

### OU users



#### SQL

CREATE TABLE `ou\_users` (

`userid` int(11) NOT NULL AUTO\_INCREMENT,

`groupid` int(11) NOT NULL,

`username` varchar(70) NOT NULL,

`password` varchar(70) NOT NULL,

`alias` varchar(70) NOT NULL,

`fname` varchar(70) NOT NULL,

`lname` varchar(70) NOT NULL,

`dob` varchar(70) NOT NULL,

`country` int(11) NOT NULL,

`locale` int(11) NOT NULL,

`theme` varchar(50) NOT NULL,

`regdate` varchar(70) NOT NULL,

`lastaccess` varchar(70) NOT NULL,

`avatar` varchar(255) NOT NULL,

`email` varchar(255) NOT NULL,

PRIMARY KEY (`userid`),

KEY `groupid` (`groupid`),

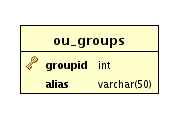
KEY `country` (`country`),

KEY `locale` (`locale`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=3 ;

### 

### OU groups



#### SQL

CREATE TABLE `ou\_groups` (

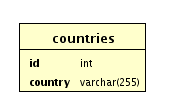
`groupid` int(11) NOT NULL AUTO\_INCREMENT,

`alias` varchar(50) NOT NULL,

PRIMARY KEY (`groupid`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=7 ;

### Countries



#### SQL

CREATE TABLE `countries` (

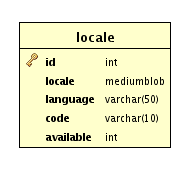
`id` int(11) NOT NULL,

`country` varchar(255) NOT NULL,

KEY `id` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

### Locale



#### SQL

CREATE TABLE `locale` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`locale` varchar(50) NOT NULL,

`language` varchar(50) NOT NULL,

`code` varchar(10) NOT NULL,

`available` int(11) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=93 ;

## 

## Referential Integrity Constraints

### SQL

## 

ALTER TABLE `addresses`

ADD CONSTRAINT `addresses\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `ou\_users` (`userid`),

ADD CONSTRAINT `addresses\_ibfk\_2` FOREIGN KEY (`country`) REFERENCES `countries` (`id`);

ALTER TABLE `auction\_bids`

ADD CONSTRAINT `auction\_bids\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `ou\_users` (`userid`),

ADD CONSTRAINT `auction\_bids\_ibfk\_2` FOREIGN KEY (`auction\_id`) REFERENCES `auction\_items` (`auction\_id`);

ALTER TABLE `auction\_feedback`

ADD CONSTRAINT `auction\_feedback\_ibfk\_1` FOREIGN KEY (`auction\_id`) REFERENCES `auction\_items` (`auction\_id`),

ADD CONSTRAINT `auction\_feedback\_ibfk\_2` FOREIGN KEY (`buyer\_id`) REFERENCES `ou\_users` (`userid`),

ADD CONSTRAINT `auction\_feedback\_ibfk\_3` FOREIGN KEY (`seller\_id`) REFERENCES `ou\_users` (`userid`);

ALTER TABLE `auction\_items`

ADD CONSTRAINT `auction\_items\_ibfk\_1` FOREIGN KEY (`country\_from`) REFERENCES `countries` (`id`),

ADD CONSTRAINT `auction\_items\_ibfk\_2` FOREIGN KEY (`sellerid`) REFERENCES `ou\_users` (`userid`),

ADD CONSTRAINT `auction\_items\_ibfk\_3` FOREIGN KEY (`cat\_id`) REFERENCES `auction\_categories` (`cat\_id`),

ADD CONSTRAINT `auction\_items\_ibfk\_4` FOREIGN KEY (`subcat\_id`) REFERENCES `auction\_sub\_categories` (`subcat\_id`);

ALTER TABLE `auction\_pictures`

ADD CONSTRAINT `auction\_pictures\_ibfk\_1` FOREIGN KEY (`auction\_id`) REFERENCES `auction\_items` (`auction\_id`);

ALTER TABLE `auction\_sub\_categories`

ADD CONSTRAINT `auction\_sub\_categories\_ibfk\_1` FOREIGN KEY (`cat\_id`) REFERENCES `auction\_categories` (`cat\_id`);

ALTER TABLE `item\_watching`

ADD CONSTRAINT `item\_watching\_ibfk\_1` FOREIGN KEY (`auction\_id`) REFERENCES `auction\_items` (`auction\_id`),

ADD CONSTRAINT `item\_watching\_ibfk\_2` FOREIGN KEY (`user\_id`) REFERENCES `ou\_users` (`userid`);

ALTER TABLE `messages`

ADD CONSTRAINT `messages\_ibfk\_1` FOREIGN KEY (`from\_id`) REFERENCES `ou\_users` (`userid`),

ADD CONSTRAINT `messages\_ibfk\_2` FOREIGN KEY (`to\_id`) REFERENCES `ou\_users` (`userid`);

ALTER TABLE `ou\_users`

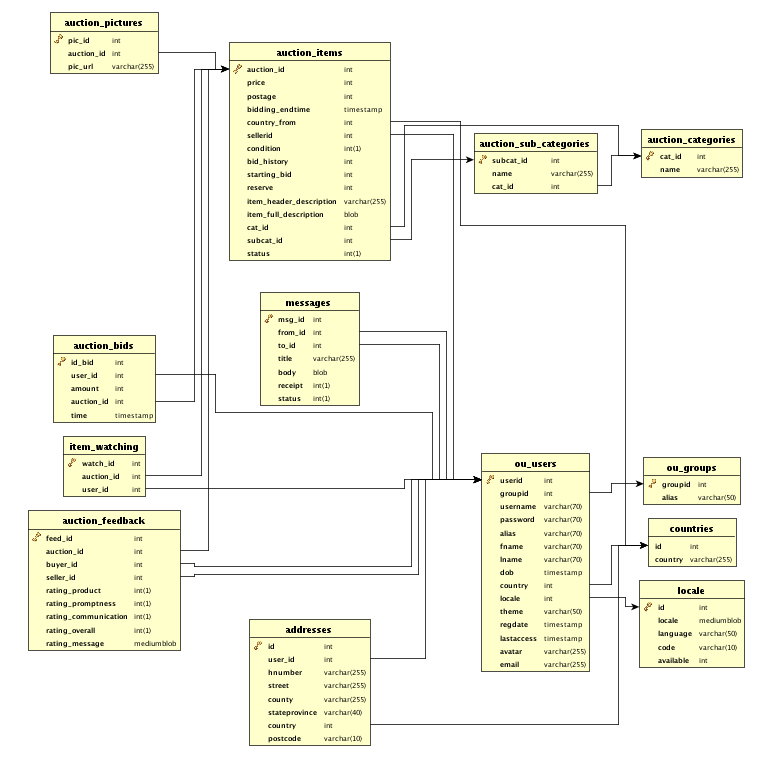
ADD CONSTRAINT `ou\_users\_ibfk\_1` FOREIGN KEY (`groupid`) REFERENCES `ou\_groups` (`groupid`),

ADD CONSTRAINT `ou\_users\_ibfk\_2` FOREIGN KEY (`country`) REFERENCES `countries` (`id`),

ADD CONSTRAINT `ou\_users\_ibfk\_3` FOREIGN KEY (`locale`) REFERENCES `locale` (`id`);

## Overview

### Relations Overview



# Design & Implementation

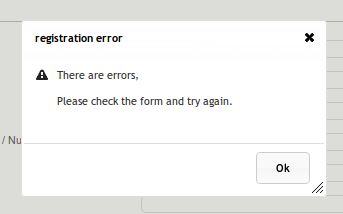
## Requirements

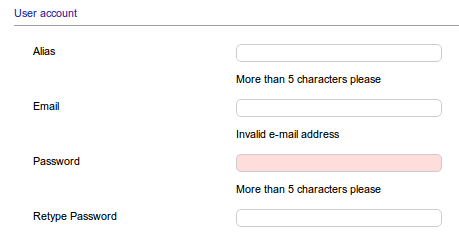
### Major Processes

#### User Registration

The PHP logic implements a series of validation checks to prevent the user from entering unwanted data. Email, username, minimum password length, addresses, DOB etc. are further checked with regular expressions to ensure coherent data. Any errors will be shown to the user by indicating which fields had incorrect data. Also the PHP logic has built in methods for taking AJAX requests, to check availability of username and email already is use prohibitions. Activation, although can be completed by email; is disabled as the server not have ‘sendmail ‘enabled.

ALL html, CSS and JavaScript completed. JQuery and plugins are used for AJAX requests to the server on form elements to verify email, username are not already in use. Using JQuery, problems fields are identified immediately indicating to the user which fields have problem data. Dates are handled by JQuery plugin DATEPICKER to ensure valid dates. JQueryUI Modal windows give further indication to the user when problems occur to increase the visibility of system status and error prevention. (Nielsen's heuristics) Minor improvements to be made to CSS and avatar upload.





#### Update registration

The PHP logic implements a series of validation checks to prevent the user from entering unwanted data. Any errors will be shown to the user by indicating which fields had incorrect data. Ajax and supported PHP logic upload allows the user to change the avatar, address, password and other details without refreshing the page.

ALL html, CSS and JavaScript completed. JQuery and plugins are used to show errors and prevent errors in user entered data. All updates are done using AJAX. Minor improvements to be made, (change email address) but cannot be implemented till sendmail is enabled.



#### Login

The PHP logic implements a series of validation checks to prevent the user from entering unwanted data. Any errors will be shown to the user by indicating which fields had incorrect data. On login complete the users profile will decide which locale will be used on the following pages. If a user is banned they will be notified and prevented from logging in.

ALL html, CSS and JavaScript completed. JQuery and plugins are used to show error and prevent error in user entered data.

#### Logout

The user’s session is invalidated and the script returns the user to the main page.

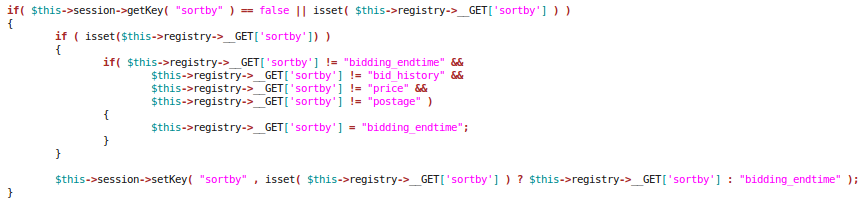
#### Submit Auction

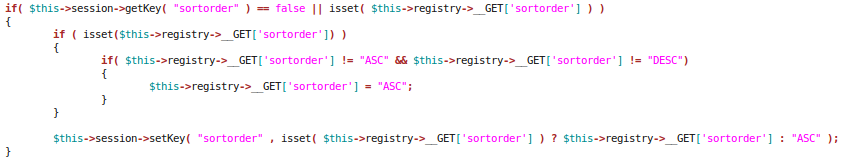
The PHP logic implements a series of validation checks to prevent the user from entering unwanted data. Any errors will be shown to the user by indicating which fields had incorrect data. Prices, times etc are checked using regular expressions to check for coherent data. If there are any errors the page will be represented with notice errors next to the offending fields. PHP implemented to handle temporary storage of upload images until the process is complete.

ALL html, CSS and JavaScript completed. JQuery and plugins are used to show error and prevent error in user entered data. Ajax is used to upload images and is immediately presented on the auction item page. The user has the ability to remove a picture immediately if they wish to do so. Times and prices fields are validate by JQuery Date picker, time picker, price picker to prevent invalid data before submitting to the server.

#### Browse Auctions

The sections are divided into categories and sub categories. PHP implements pagination and items per page. Auctions are shown only if the are newer than the current date. Listing can be viewed by category or by category and sub category. Sorting is also implemented, currently is allows for sorting by date, price, bids, postage and also by name. A series of validation checks prevents the user from sending mal formed GET queries which control the flow of these features. If malformed data is sent the script which go back to standard view.



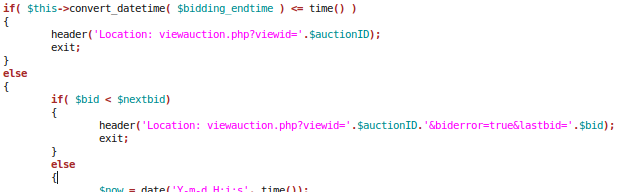
****

#### View auction

The view auction page presents all auctions present and past. The output differs if the auction is complete and presents the winner on the page. The place bid option is removed. An auction is selected by providing the auctioned in a GET request. Validation checks take place to prevent malformed data from being sent in the GET request.

#### Bidding

The bidding process implements a series of checks to prevent a user from bidding under the current bid. Firstly the user must be logged in, on the auction page the next bidding multiple will be presented, the use can bid higher be must be at least the bid presented in order to successfully bid. Once the bid is added the user can view his current bids on the ‘my bids’ section of the web page. He can see his / hers current bid and the current high bid.



#### Searching

The simple search is an auto suggests form on the main page. It allows the user to search by a name. The advanced search however allows the user to submit a series of predicates. The request and its predicates are sent to the server via AJAX. The PHP implements a query builder to query the SQL database and find the results the user is looking for. Finally the information is sent back to the browser and JavaScript is used to create a table which is then sort able using JavaScript.

#### Feedback

After an auction is completed, the winning party has the opportunity to leave feedback on the item won. They access this through ‘my bids’ where a feedback button is available on the auctions they have won and where feedback has not already been left. The feedback page allows the user to leave feedback and levels of acceptance of promptness, communication, etc. Once feedback is left, the seller is able to view the feedback on the feedback page.

#### Watch item

On the auction page a watch item button is available to allow users to add the item to their watching list. On the watching page the user can quickly view items there are watching and have the ability to remove the item form their watch list.

### Administrator

#### Delete User

The administrator has the ability to remove a user. When removing the users the auctions for that user, pictures, feedback, messages, address is removed also because of the foreign key constraints existing in the database.

#### Ban User

Banning users changes there group to group6. When the banned user attempts to login they are presented with error. Immediately there session is invalidated so as they cannot do anything with the account.

#### Update auction

The administrator has the ability to change the Time and date of the auction end time and the ability to completely remove the auction if they wish to do so.

#### Delete auction

The administrator first selects the user which is auctioning the item. Then a list of his or hers items is shown and the administrator selects the auction to remove. On removing the pictures and feedback are also removed.

### Non Functional Requirements

#### Model – View - Controller

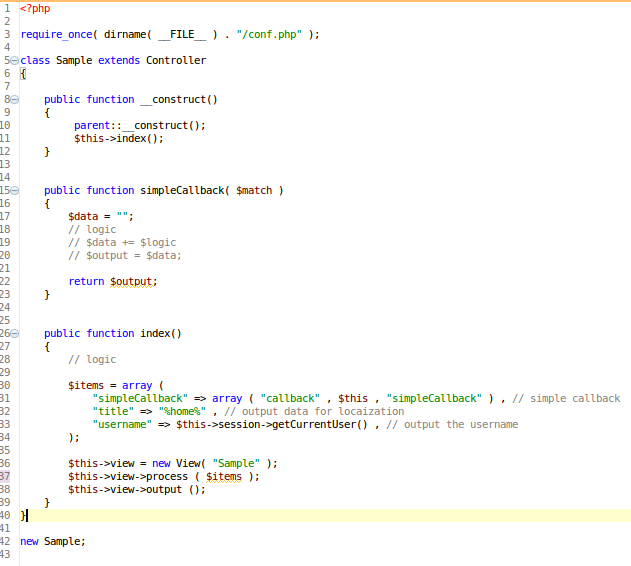
##### Sample controller

The business logic is done in the controller instances before handing off to the view for output. Once to business logic is completed an array of Keys (replacements) and there values are constructed to tell the view which items are to be replaced within the output template html.

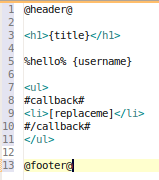
Three distinct types of values are present in the example below.

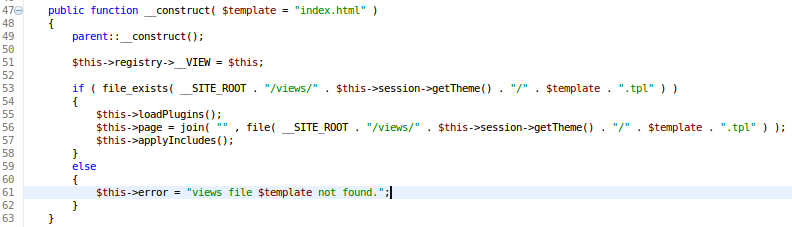
* A call back to the controller instance for a matched block that requires more processing.
* A replacement that requires a language translation
* A standard replacement.

This array is processed by the view, providing the name of the template to use in the View constructor.

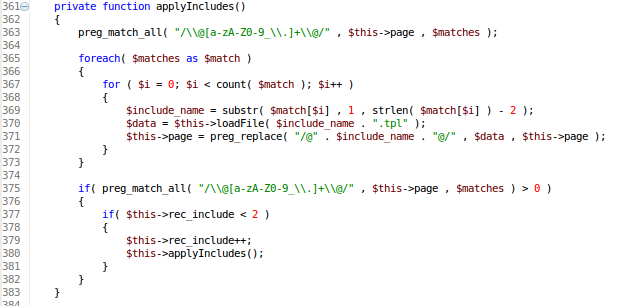


##### Sample Template



View Constructor

##### View Includes



Includes are handled within the templates by matching the @INCLUDE\_THIS@ syntax. The file contents of file are then included in the final output. Levels of recursion are allowed, if the file included has further includes.

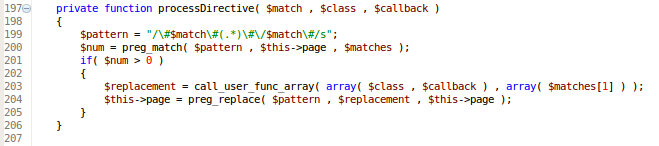
##### View Processing

##### C:\Documents and Settings\Administrator\Desktop\viewprocessing.png

The first element in the replacement tags is an array which is identified as the processing directive. Line 132 identifies that a call back is required. The process directive method is called in the view and the values are handed off to this method. When the method is done processing the output is return and inputted in the html.

The second element is a replacement; line 164 handles this replacement by matching {title} in the view and replacing it with %home%. The third element is handled the same way.

##### View Process Directive



Using the call user function array method in PHP we call the controller instance method specified and pass the html matched (Sample view lines 8 -10) to this method for further processing.

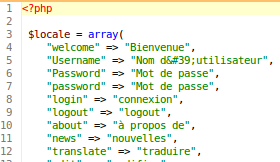
##### View Output



Once the complete page is built the page is then printed to the output stream.

#### Localisation

Localisation support was implemented using the Google translate service. Language packs where created using associative arrays. The key represented the text to replace and the value was the replacement text.



The last step in the view before output the html page is replacing all matching %welcome% using a regular expression replacement. The locality is determined by the users $\_SESSION[‘locale’] and the correct language pack is included. Lines 320 - 327

In the case of %welcome% it would be replaced by “Bienvenue”. Lines 338 – 341

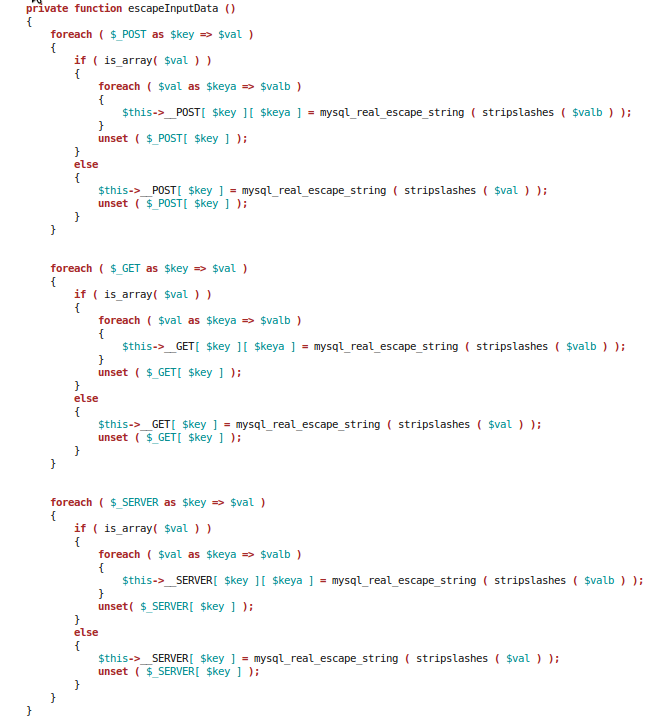


Depending on the locality is use the DOCTYPE for the html output was also replaced to confirm to the internet standards and ensure correct layout and reading orientation in the case of right to left languages.

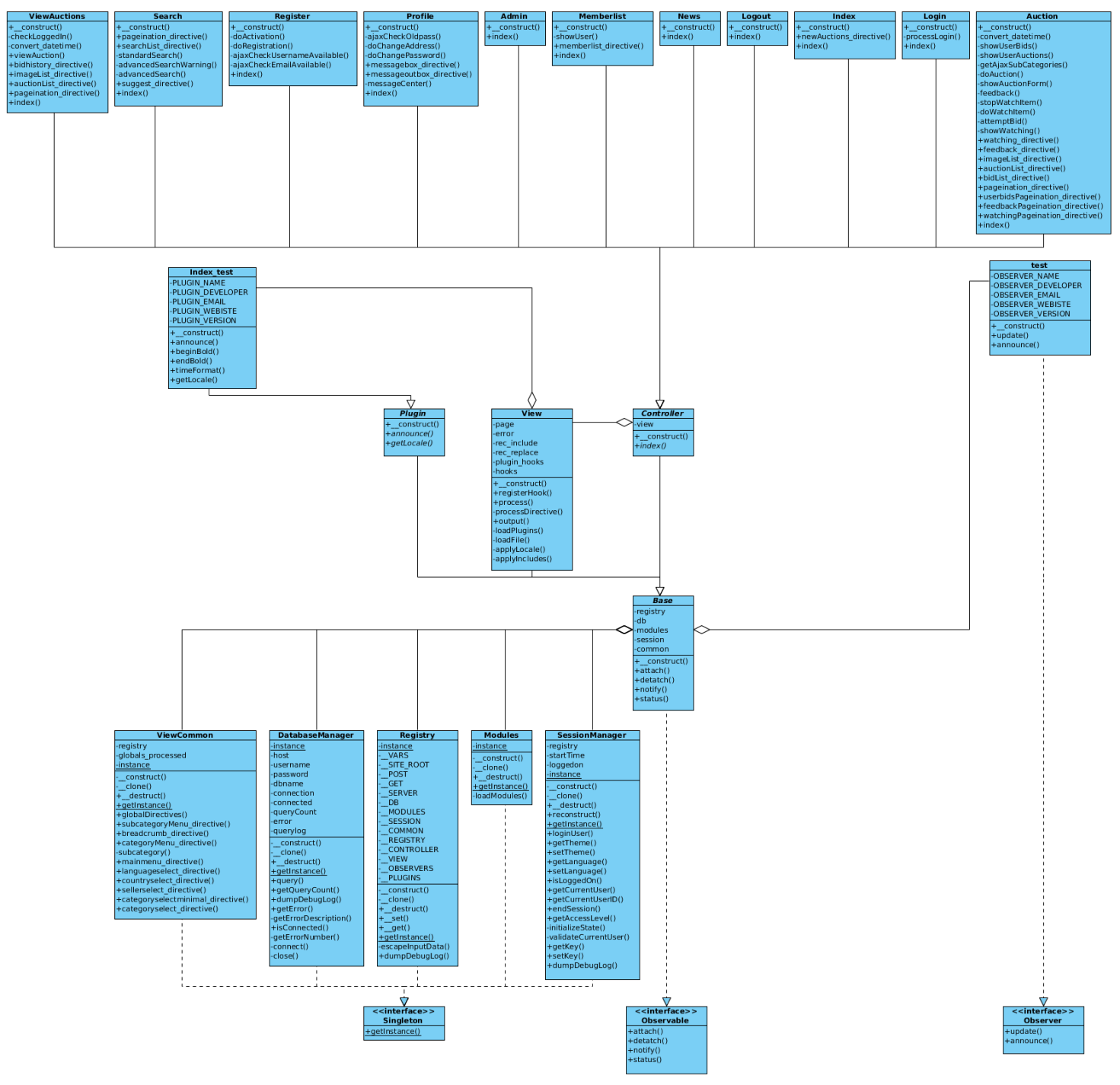
#### Security

First level of security is the un-setting the $\_POST, $\_GET and $\_SERVER. These session datasets are stored in the registry class for later retrieval for use in the controllers. Each key pair in the datasets is stripped of slashes and then using ‘mysql\_real\_escape\_string’ is prepared for database usage.

As the inheritance graph indicates in the UML class diagram all of this processing is done before any of the controller logic, forcing the programmer to use the inbuilt protection overrides. Further checking is done in the Controller logic to ensure safe data.



## Prototype UML Class Diagram



## Copied materials

* **JQuery** 
  + **JavaScript Framework**
    - http://jquery.com/
* **JQueryUI** 
  + **Tabs, Date picker, Dialog, effects**
    - http://jqueryui.com/
* **JQuery Plugins**
  + **Ajax Upload**
    - http://plugins.jquery.com/project/uploadify
  + **Cross Slide (images)**
    - http://plugins.jquery.com/project/easyslideshow
  + **Format currency**
    - http://plugins.jquery.com/project/currency
  + **Table Sorter**
    - http://plugins.jquery.com/project/tableSort
  + **Sizeable text area**
    - http://plugins.jquery.com/project/TextAreaResizer
  + **Time Picker**
    - http://plugins.jquery.com/project/timeEntry
* **Fancy Box** 
  + **Image Preview**
    - http://fancybox.net/
* **GTranslate**
  + **Google Translate API Class** 
    - http://code.google.com/p/gtranslate-api-php/
* **Regular Expressions**
  + **Email**
    - http://php.net/manual/en/function.preg-match.php
  + **Date**
    - http://dotnetslackers.com/Regex/re-16542\_Regex\_matches\_a\_date\_in\_dd\_mm\_yyyy\_format.aspx
* **Reports Diagrams**
  + **Db Visualizer** (Entity relationships)
    - http://www.dbvis.com/
  + **yEd Graph Editor** (Flow Charts)
    - http://www.yworks.com/
  + **Visual Paradigm For UML**
    - <http://www.visual-paradigm.com/>
* **Theme**
  + **Simple auction template**
    - http://www.freewebtemplates.com