Design Document

CSCE 361- Spring 2017

Asset Management System

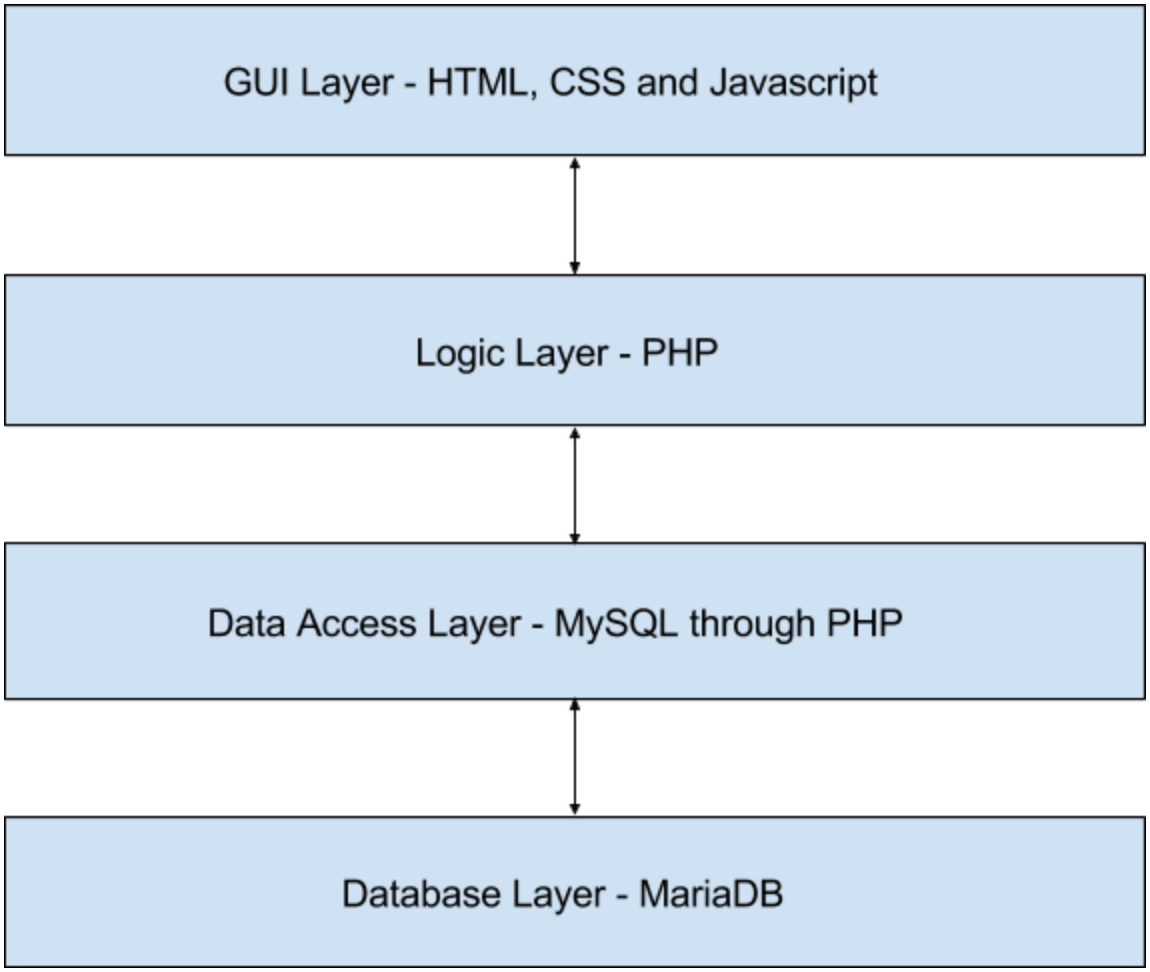
Team 10 - Andy Petrzilka, Johnathan Avery, Marta Kodin, Matthew Maas

1. Introduction

This design document describes the architecture and class relations for the Asset Management System. The document is intended for the coders who will be implementing the AMS. The design document will include a complete, high level description of object classes and database tables.

2. Architecture

2.1. Introduction



This architectural model of the system is layered. The top layer, written in Javascript, HTML, and CSS, is the GUI Layer used for user interaction. The next layer, the

Logic Layer will be written in PHP. The next layer, the Data Access Layer, will be written with PHP and use MySQL commands to communicate with the Database Layer, written in MariaDB.

2.2. Modules

2.2.1. GUI Layer

The GUI layer or graphical user interface will allow us to present data to the users in a convenient, organized and neat manner. The GUI will display results from the Logic layer as well as the data access layer and present those results in an easy to use, easy to read fashion. We will display information using html and make change the html based off of CSS styles. The javascript will work to make real time interaction between the user and the html live insteading of requiring a page refresh to change any information. The javascript will make realtime changes possible and change the GUI in real time depending on various information.

2.2.2. Logic Layer

The logical layer will allow us to do the data manipulation as well as create various measurable statistics for reporting. Our logic layer acts as a processor as in it will process data for various reasons and give us useful information to use whether it be for displaying information for the user to see or adding and editing data from the database. Our logic layer will be done using PHP which allows us to displays with PHP or HTML and interact with the database through mySQL and php PDO prepared statements.

2.2.3. Data Access Layer

The data access layer will allow us to prepare data and send it off to the database as well as let us prepare queries so that we can retrieve and manipulate various sets of data. This will allow our update, add and retire feature to work with the click of a button. Our data access layer will use mySQL which can talk with various database engines and provides a fairly standard method of writing requests.

2.2.4. Database Layer

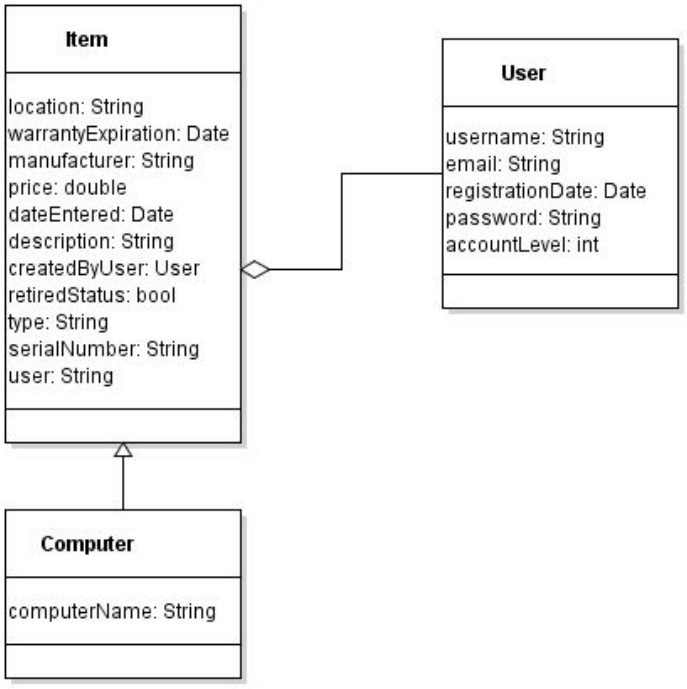
The database layer is responsible for storing all of our data that will need to be accessed at different times and viewed from different locations. MariaDB will interfaced using mySQL which makes it easy to integrate in many applications. We can use various interfaces to manage the data as well as PHP itself to control and access data.

3. Class Diagrams

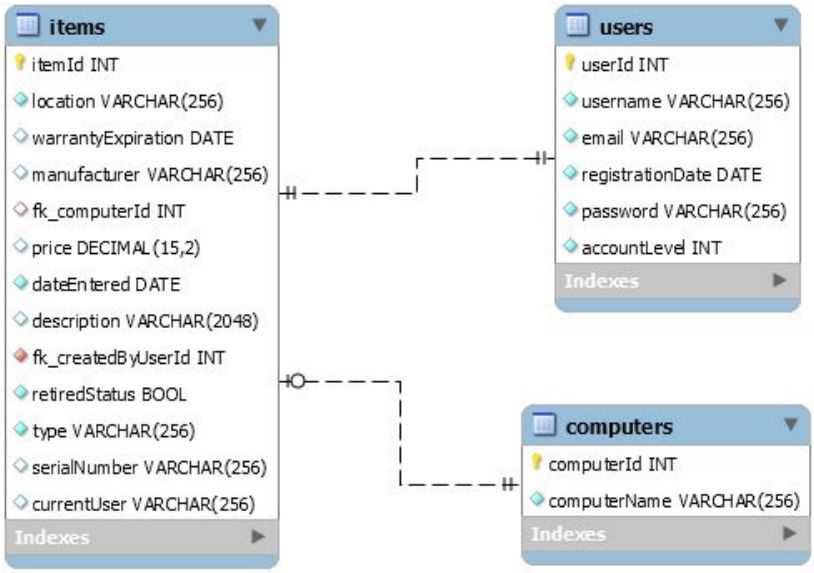
3.1. Data Table Classes

3.1.1. Schema

3.1.1.1 PHP Schema



3.1.1.2. MySQL Schema



3.2. Class Information

The classes will be implemented within PHP and will be constructed using information from the database. Classes will follow identical structures as the database and will contact various functions to set, retrieve and manipulate data. There will be few parent/child classes and a couple normal classes without any inheritance. The classes will interact directly with the data access layer and be used during the logic layer stage.

3.3. GUI Layer

The interface level of the AMS is to be straightforward. Users will log in through a login screen. After user identity has been verified, users are taken to a home screen. All of the functionality will be immediately accessible through a dashboard along the top. The navigation will provide access to pages for updating, searching, adding and viewing records. Each function will be given its own page to proceed with those functions and the pages will interact with the classes which follow the same design as the database. The classes will interact with various functions to pull data from the class as well as update data. Not all information stored in a database will be displayed at each step as some of it will be auto generated and information will be displayed on a need to know basis for each report or function.

4. Database structure

4.1. Users

4.1.1. userID (int)

The userID will be a personal, auto-incremented key for each user.

4.1.2. username varchar(256)

The username will be unique to each other and allowed to be up to 256 characters.

4.1.3. email varchar(256)

The user's email will allow up to 256 characters.

4.1.4. password varchar(256)

The user password will be allowed to be up to 256 characters.

4.1.5. registrationDate(date)

There will be a field for the date that the user registered set up by the database.

4.1.6. accountLevel (int)

This field will specify the account level of the user, and therefore the privileges the user may perform through his/her account.

4.2 Items

4.2.1. itemID (int)

The itemID will be a personal, auto-incremented, non-null key for each item.

4.2.2. location varchar(256)

This field is to specify the location of the item. It shall not be null.

4.2.3. warrantyExpiration (Date)

This field is for the expiration date of any warranty purchased or that came with an item. It may be null if the item was obtained without a warranty.

4.2.4. manufacturer varchar(256)

This field is for the manufacturer of the item, allowing up to 256 characters.

4.2.5. price (double)

This field specifies the price for which an item was bought for insurance purposes.

4.2.6. dateEntered (date)

This field specifies the date an item was entered into the database. This field shall not be null and be determined by the back end through accessing the timestamp.

4.2.7. description varchar(2048)

This field shall specify a physical description of the item, allowing 2048 characters which shall be sufficient for a description.

4.2.8. createdByUserID (int)

This specified which user entered this item into the database through their personal key. It is therefore a foreign key connecting to the users table. This field shall not be null.

4.2.9. retiredStatus (bool)

This field is a boolean that is true when an item has been retired and false if it has not yet been retired. This field shall not be null.

4.2.10. itemType varchar(256)

Unlike the description, this is just the type of item. e.g. computer, desk, table, chair, etc. This field shall not be null.

4.2.11. serialNumber varchar(256)

This is the field for the serial number of an item if it was provided at the time of purchase or acquisition. This field may be null if the item did not come with a serial number.

4.2.12. currentUser varchar(256)

This field specifies which user is currently using the item. The user may not be registered as a user in the system and as such the field allows up to 256 characters. The current user of the item may be different to the user who entered the item into the database. This field may be null if the item is retired, in storage, or in a location where no users are actively operating it like a conference room.

4.3 Computers

4.3.1. computerID (int)

The computerID will be a personal, auto-incremented, non-null key for each computer.

4.3.2. computerName varchar(256)

This field specifies the name of the computer, allowing up to 256 characters. This field shall not be null.