**Assignment 2:**

**COMP1341 (Introduction to Web and Database Development)**

**Assignment 2**

**Semester 2 - 2018/19**

**Student number = 18006277**

**WEBSITE (with database) URL:** [**http://garb1-18.wbs.uni.worc.ac.uk/**](http://garb1-18.wbs.uni.worc.ac.uk/)

**Login Details: Register as a new user and sign in with those credentials, you will have admin rights by default. Please do not use a sensitive password due to the lack of a hash function**

□ I do not want my work to be used anonymously to help future students

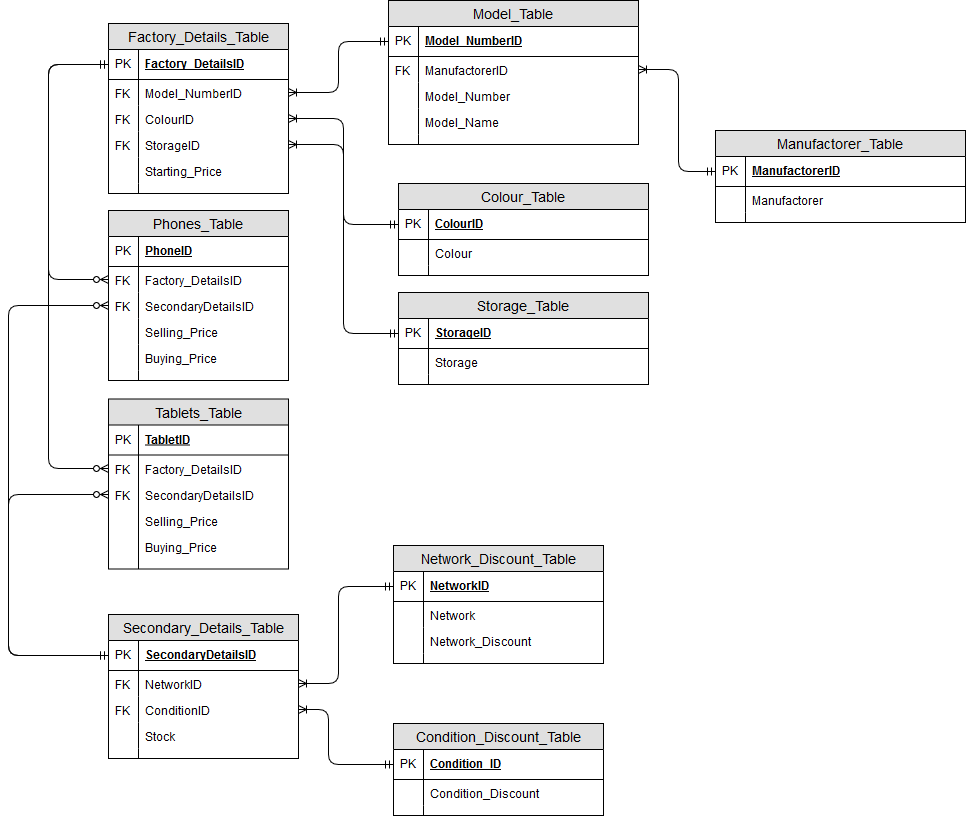
**Designing a relational database**

**Context**

The database I am designing is going to be a device identification database for a second-hand electronic retailer. The database is to provide a record of every potential type of phone that could come into the company to be sold. This would be a combination of both factory properties such as storage and colour, and secondary properties that are not determined by the manufacturer such as the condition of the device and the stock level that may exist inside the company.

The ideal implementation of this database would be for a front-end user such as a sales assistant to use in order to find a specific device and be able to give a price of the device to a customer. For example, if a customer came in with a second-hand phone and wanted to sell the device, then the user could input certain specs that they know and narrow down potential phones. For example, if the user knew that the device was an Apple 5S in Space Grey on the EE network in a B condition, but did not know the storage of the device, the user could still give a rough price based off the price of the device and all its storage possibilities using a wildcard search.

**ERD**

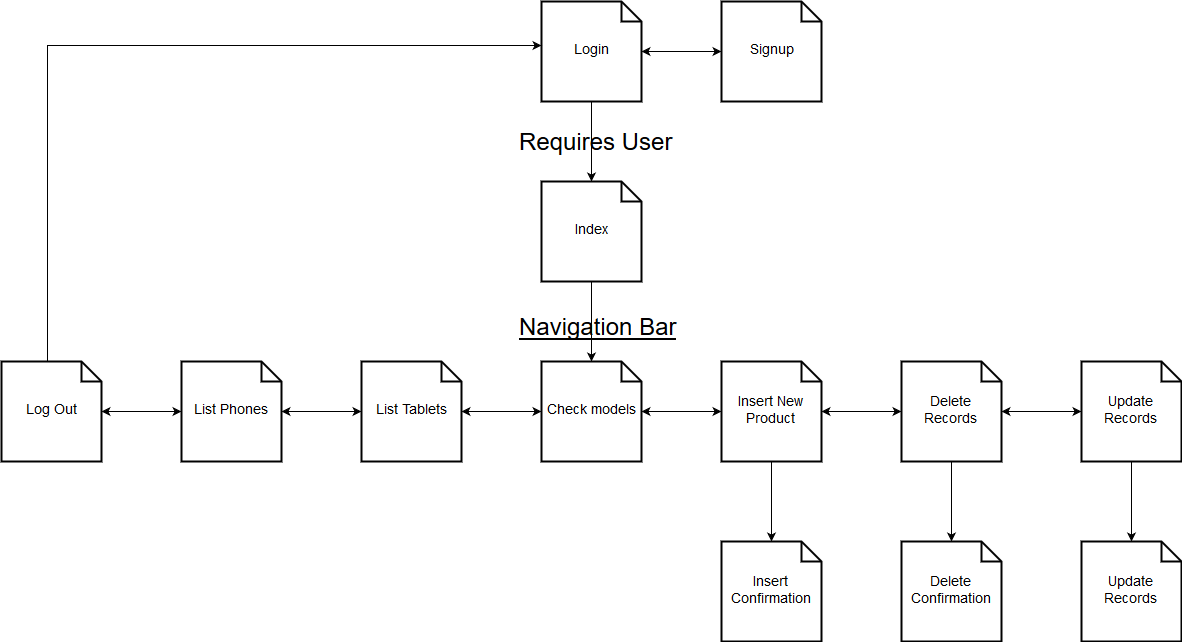
****

**ERD Explanation**

The ERD focuses around two products in which the database centers on. The Phones\_Table and the Tablets\_Tablet. The Factory\_Details\_Table feeds into both tables, but only one at a time, making the relationship one mandatory to one optional. The factory details encompass all properties of the phone that are determined during the manufacturing of the device. This includes the model, colour, storage amount, and starting price (RRP brand new). The model number has a one to one mandatory relationship with the manufacturer. The model number and model name is determined by the manufacturer resulting in a one to one mandatory relationship between model and manufacturer. The colour and storage both branch off into their own respective tables with CoelourID and StorageID working as the primary key, and each colour and storage has their own unique primary key.

The Secondary\_Details\_Table is again a one mandatory to one optional between both phones and tablets. These are properties that the manufacturer did not control, and instead determined by the previous owner of the device. This includes the network in which the device is locked to (if any), the physical condition of the device, and the level of stock of that device, the company holds.

Both Network, and Condition have separate tables which is a discount determination table. This is a one to one mandatory relationship between the NetworkID and Network\_Discount\_Table. This table provides a percentage discount to the Retail Price based on Network. The same applies to the ConditionID having a one to one mandatory relationship with Conditon\_Discount\_Table. This applies a percentage discount to the Retail price based off the condition of the device.

**Site Map**

Site navigation works through the initial login screen. The site requires that all users of this site have a username, password, and have logged in to the site. If they have no credentials, there is a signup form which a user can create credentials. Upon successful login to the site, the user is taken to an index page which serves no purpose other than to navigate to other pages on the site. To the left of the Index is a vertical navbar which takes the user to all other pages within the site, as well as gives an option of logging out. If the user logs out, they are taken back to the login screen. All pages within the site have access to the vertical navigation bar, and thus all pages can be reached from one another. The only exception is the signup page, which is loaded upon a successful register to the site. This page the automatically redirects the user to the login page after three seconds have passed.

**Assumptions:**

* All phones are European models
* All Network providers are UK based
* All Network providers are broken down into the “big four”, so all phones networks will operate either directly as the named operator or piggyback off one of the big four.
  + Example: GiffGaff piggyback off O2, so any phone operating off GiffGaff will be added to the database as O2 due to working off the O2 network.
  + The “WiFi” network is for use on devices that only have an exclusive WiFi card, with no potential to connect to a roaming network via sim card. This would be used exclusively on Tablets due to not all tablets possessing a sim card tray
  + All phones within the database will have a network other than WiFi.
* All phones are assumed to be fully working, with condition referring to cosmetic condition, rather than if the phone is functioning
* All prices are presented as sterling
* Colour will be the manufacturers specified colour, not a generic colour description of the product
  + Example – Apple call their white iPhone 5S “Silver”, and Samsung refer to their black coloured S7 as “Midnight Black”
* Whilst IMEI is a unique identifier of each phone sold globally, it will not be stored in this database due to it being a unique identifier to a single phone. So, one PhoneID could have many IMEIs.
* Selling price refers to the price of which the phone would be sold to the public
  + Discount is a flat % for each phone based upon fields
    - Example: An unlocked phone would have a 0% deducted, a phone locked on the Vodaphone network would have a 10% discount
    - If multiple discounts are applied, for example a phone being locked to O2 and the phone being of a condition grade C, the two discounts would be added together, to form one discount. Rather than having one discount applied first, then the second discount being applied on the lower price of the item.
* Buying price refers to the price of which the phone would be bought off a hypothetical customer after all relevant discounts have been applied.
* Starting price is the price of the device with no discounts applied to it and is based on current market value of the product being sold as a new product, or in rare cases, the antique value of certain old devices that have become collectible. An example could be that if the company were to expand into selling laptops, then the old IBM ThinkPads would be worth more than some modern laptops due to their age, and some being deemed collectable.
* All users will have the basic understanding of how the system works.
* All users will follow basic DDL etiquette then using the site.

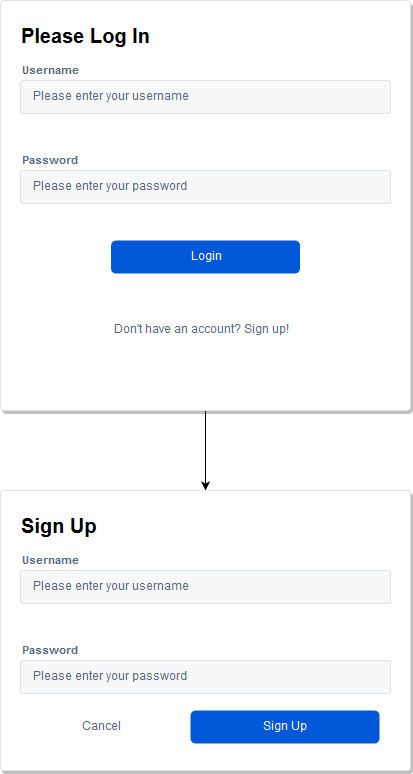
**User Created Data Dictionary:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Manufacturer\_Table | Type | Description | Example | Null (Y/N) |
| ManufacturerID | Char(5) | Unique identifier for Manufacturer | M1001 | N |
| Manufacturer | Varchar(7) | Name of Manufacturer | Apple | N |
|  |  |  |  |  |
| Model\_Table | Type | Description | Example | Null (Y/N) |
| Model\_NumberID | Char(6) | Unique identifier for Model Number and Name | MN1001 | N |
| ManufacturerID\* | Char(5) | Unique identifier for Manufacturer | M1001 | N |
| Model\_Number | Varchar(8) | Model number of the product | A1357 | N |
| Model\_Name | Varchar(15) | Model name of the product | 5S | N |
|  |  |  |  |  |
| Network\_Discount\_Table | Type | Description | Example | Null(Y/N) |
| NetworkID | Char(5) | Unique identifier for the network | N1001 | N |
| Network | Varchar(8) | Network attached to the device | Vodafone | N |
| Network\_Discount | Tinyint(2) | Discount percentage for a devices network | 10 | N |
|  |  |  |  |  |
| Condition\_Discount\_Table | Type | Description | Example | Null(Y/N) |
| ConditionID | Char(6) | Unique identifier for the condition | CN1001 | N |
| Phone\_Condition | Char(1) | Condition of the device | A | N |
| Condition\_Discount | Tinyint(2) | Discount percentage for a device’s condition | 5 | N |
|  |  |  |  |  |
| Factory\_Details\_Table | Type | Description | Example | Null(Y/N) |
| Factory\_DetailsID | Int(255) | Unique identifier for the details determined by the factory | 2 | N |
| Model\_NumberID\* | Char(6) | Unique identifier for the Model Number and Name | MN1001 | N |
| ColourID\* | Varchar(6) | Unique identifier for the colour | C1001 | N |
| StorageID\* | Varchar(6) | Unique identifier for the storage | ST1001 | N |
| Starting\_Price | Decimal(6,2) | Starting price before any discounts are applied | 299.01 | N |
|  |  |  |  |  |
| Colour\_Table | Type | Description | Example | Null(Y/N) |
| ColourID | Char(6) | Unique identifier for the details determined by the factory | C1001 | N |
| Colour | Varchar(14) | Colour of the device | Example | N |
|  |  |  |  |  |
| Storage\_Table | Type | Description | Example | Null(Y/N) |
| StorageID | Char(6) | Unique identifier for the storage | ST1001 | N |
| Storage | Varchar(5) | Storage of the device | 32GB | N |
|  |  |  |  |  |
| Secondary\_Details\_Table | Type | Description | Example | Null(Y/N) |
| Secondary\_DetailsID | Char(6) | Unique identifier for the details determined by the previous owner | S1001 | N |
| NetworkID\* | Char(5) | Unique identifier for the network | N1001 | N |
| ConditionID\* | Char(6) | Unique identifier for the condition | CN1001 | N |
|  |  |  |  |  |
| Phones\_Table | Type | Description | Example | Null(Y/N) |
| PhoneID | Int(255) | Unique identifier for a unique phone | 2007 | N |
| Factory\_DetailsID\* | Char(5) | Unique identifier for the details determined by the factory | F1001 | N |
| Secondary\_DetailsID\* | Char(5) | Unique identifier for the details determined by the previous owner | S1001 | N |
| Selling\_Price | Decimal(6,2) | Price the device will be sold at in the company after discounts applied | 249.99 | N |
| Buying\_Price | Decimal(6,2) | Price the company will pay the customer for the phone | 249.99 | N |
| Stock | Smallint(4) | Levels of stock within the company for the exact device | 47 | N |
|  |  |  |  |  |
| Tablet\_Table | Type | Description | Example | Null(Y/N) |
| TabletID | Int(255) | Unique identifier for a unique tablet | 3002 | N |
| Factory\_DetailsID\* | Char(5) | Unique identifier for the details determined by the factory | F1001 | N |
| Secondary\_DetailsID\* | Char(5) | Unique identifier for the details determined by the previous owner | S1001 | N |
| Selling\_Price | Decimal(6,2) | Price the device will be sold at in the company after discounts applied | 249.99 | N |
| Buying\_Price | Decimal(6,2) | Price the company will pay the customer for the tablet | 249 | N |
| Stock | Smallint(4) | Levels of stock within the company for the exact device | 47 | N |
|  |  |  |  |  |
| User\_Table | Type | Description | Example | Null(Y/N) |
| UserID | Int(255) | Unique user identifier | 1 | N |
| Username | Varchar(255) | Unique user username | PCLEWS | N |
| Password | Varchar(255) | Unique user password | £%$^"& | N |

**PHP MyAdmin generated Data Dictionary:**

See “phpMyAdmin\_Data\_Dictionary.pdf” in files

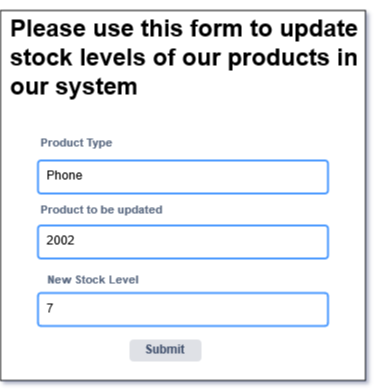
**User login form:**



**User login form pseudocode:**

Uses user table within database which validates through a select statement. Username and password are case sensitive. Would generate session token which allows user to access all other pages on the site. If a user does not log in, then the rest of the site is inaccessible. If the user knows the names of pages such as index.php and tries to go directly to that page, the session token will be void and the user is immediately redirected to the login page. Sign up uses standard insert into user table with a username and password, which is then immediately available to log in with. All users by default are admins due to limitations of plex usage. Ideally, I would have all users have base permissions rather than full admin access to the site, then have SQL table properties which determine what permissions a user has.

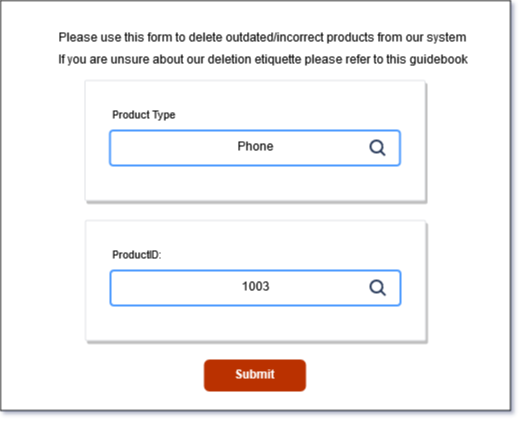
**Update data form:**

****

**Update data pseudocode:**

Data updating would work through an update statement for a desired table. In this instance, the table selected (phone or tablet) would work off the first form selection being a dropdown form. This specifies the table in the SQL. The product to be updated will be the exact product id that will be inputted. The new stock level will just be an int update which will overwrite the existing stock. So, if product 2002 has a stock level of 2, and the user updates it to 6, the new stock level will be 6, 6 will not be added on to the current stock.

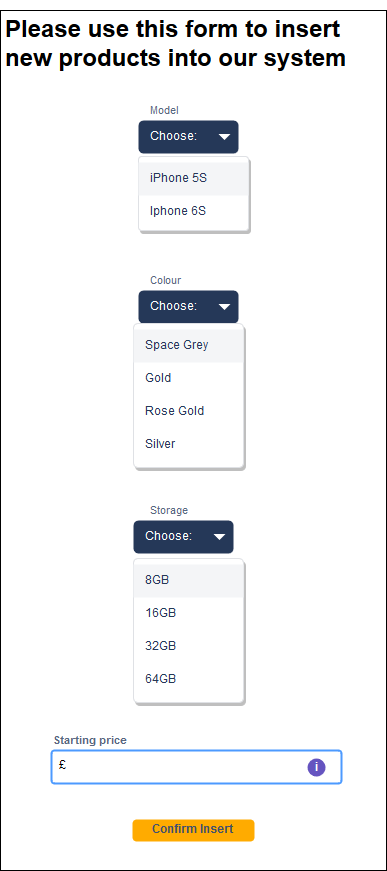
**Data Deletion form:**



**Data deletion form pseudocode:**

Admin access only form that would allow simple data deletion. Would assume user would know product code, or at least most of product code of the data entry they wish to remove. Would likely be used exclusively for deletion of data that was entered by mistake. Any data of products that would no longer be sold would be added to an archive or excluded from search results but kept within the database. Simple form design with clear warning about data deletion being permanent. Would work as a simple delete statement, with the first dropdown acting as a table to delete from, and then a product code for e.g. ‘delete from “x” where “Y” = “Z”’. with x being table, y being product type, and z being product.

**Data insertion form**

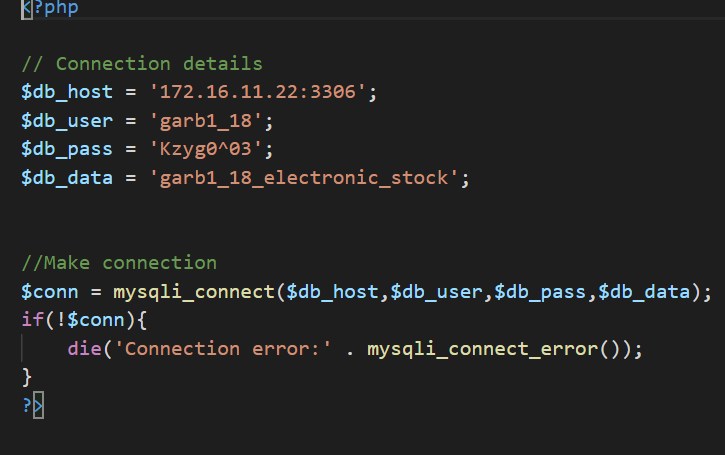


**Data insertion from pseudocode:**

Simple insertion form that uses an insert statement, with each part of the statement having its values stored within variables that get posted from this form. Would also have a SQL statement which checks for duplicate values and prevents the insert if a value with the exact specifications already exists. This would be a simple select statement with values equal to the posted variables. If the insertion is unique, then it carries on and works as an insert V X Y & Z into the corresponding table, which is determined by the dropdown options. Form would work entirely as a series of dropdowns, with values being pulled from the database. This ensures the form is always up to date with the correct information regarding the database.

**Database Connection Code:**

**Config.php – Makes database connection (1/2)**



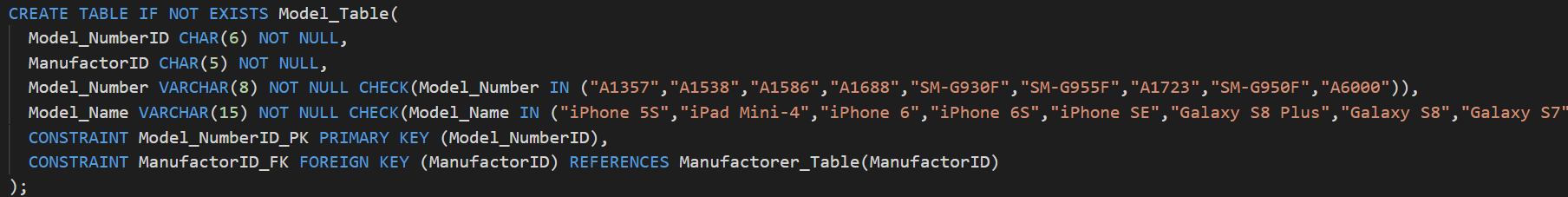
Connection details are what every user that logs into the site uses to manipulate the database. Login details are generated in Plesk. Simple PHP that checks connection details are correct, and if so allows connection. If details are incorrect, then kills the connection to the database and throws an error. This should never happen though.

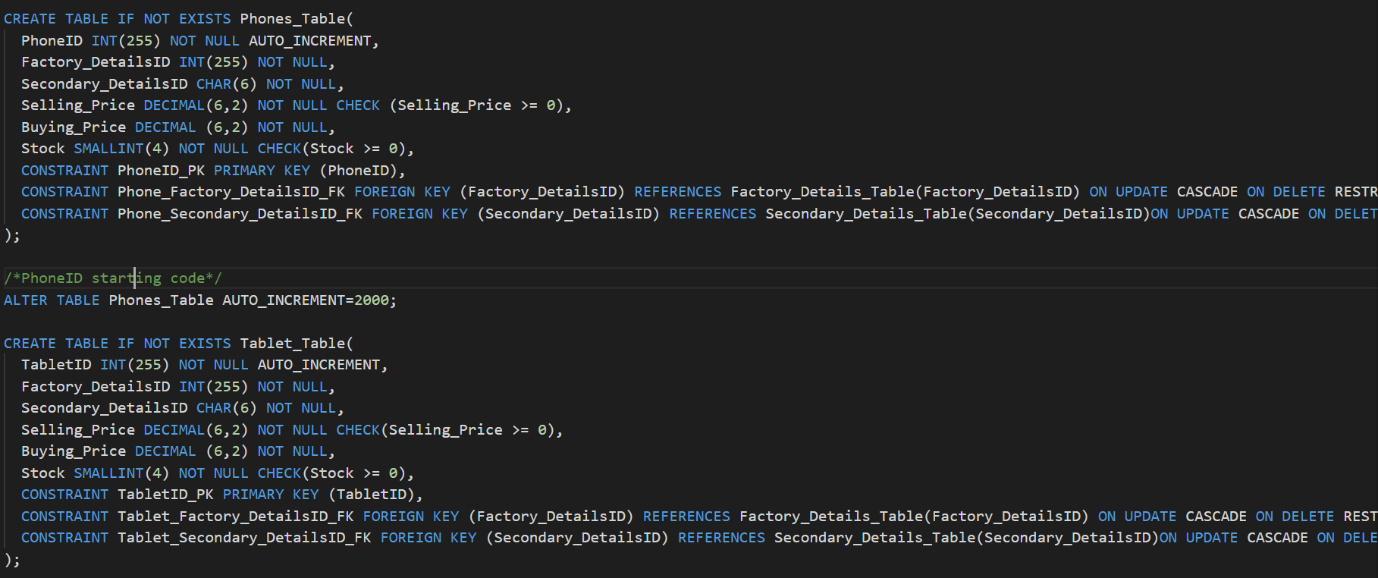
**Login.php – Logs user in and checks if they exist in the database. If user exists, session starts, and user can access rest of site(2/2)**



Login form php which reads users login details. The username and password variables are posted from the login form on the same page. The $SQL variable then is a select statement which checks to see if the user exists and marks the session as active. If there is no user with that username or password, it throws an error stating that user or password may be incorrect. Should the user exist, then sets the session variable to the username, and redirects the user to the home page.

**Create a table – Phone\_Database\_Assignment.sql**

****

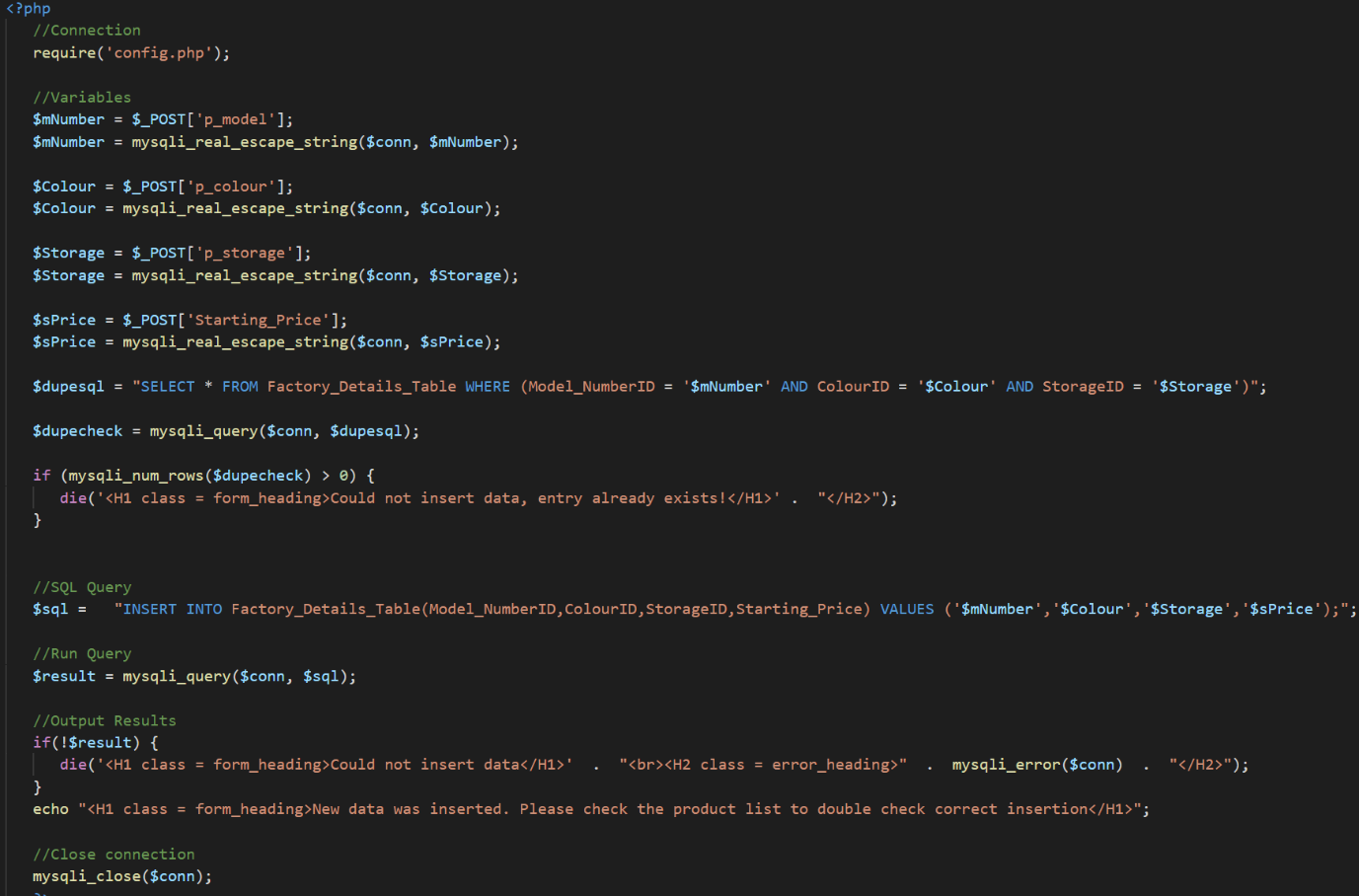
****

Uses standard create table DDL to create every table within the database. If not exist clause is added to prevent conflict through multiple reuploads and if the database isn’t dropped upon reupload for any reason. All tables follow the same format of the first entity being the primary key, then followed by all relevant secondary keys, then finally the data entities. All data entries have the relevant information such as type, length and relevant constraints. Some tables such as phone tables are missing constraints, but this is due to them being declared within triggers. The constraints set within table creation only work upon hard coded data insertion, such as dummy data. The constraints declared at the end of the table creation are the table links through primary and foreign keys. These are declared as “Constraint X PK from Y Table”, or “Constraints X FK from Y entity References Z table”. All foreign keys also cascade on update and restrict on delete.

**Insert into the database (1/2) – insert.php**

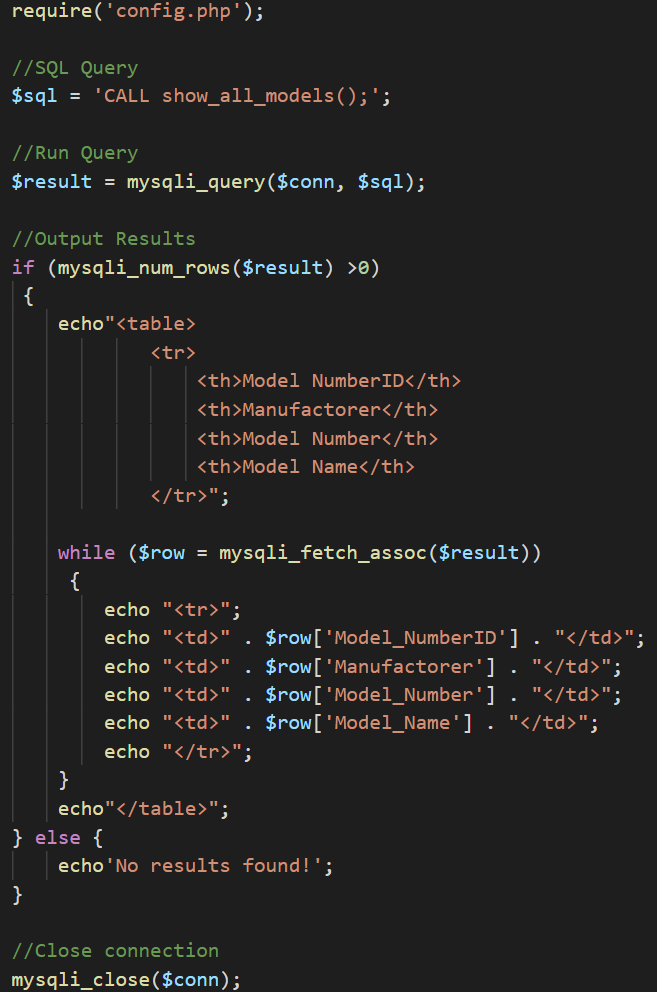
****

**Insert into the database (2/2) – database\_insert.php**



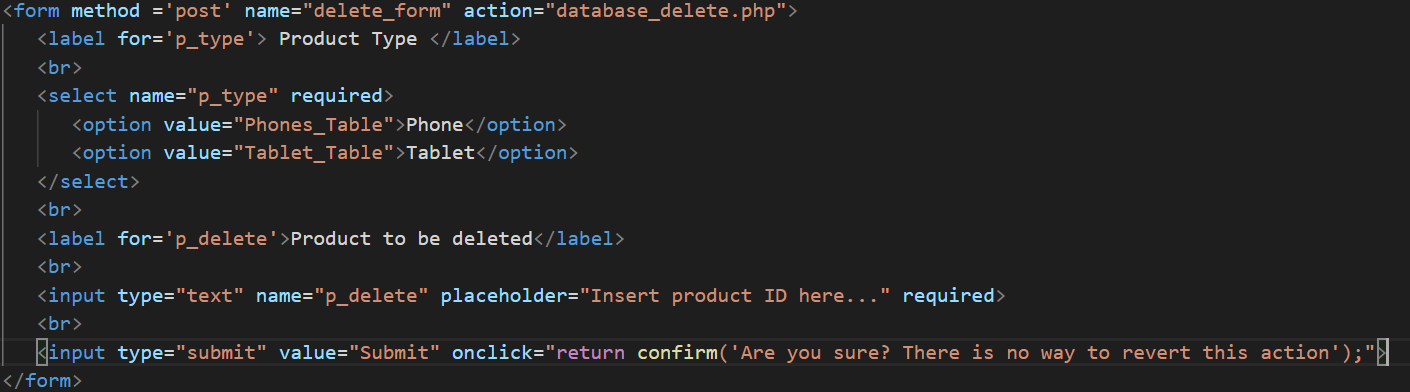
Database insert works using an insert query with the values being assigned through posted variables that are assigned through a form. The form is populated through a select \* from x table query which automatically updates the dropdown form whenever a new entry is inserted/deleted. The value is the same value as what is required within sql queries, but the form shows a value more readable to users, so instead of reading a cryptic model code, they would instead see the name of a model. The insertion itself first assigns all the values from the form to variables. These variables are then passed through a “select \* where” query which checks to see if whatever is trying to be inserted already exists. If the value already exists within the database, it kills the connection and throws and error code at the user. If the entry does not already exist, it then runs the insert query into the appropriate table. If for some reason the entry fails, it kills the connection and throws an error. This should never happen though due to all options being from a drop-down list, aside from price, but that requires a number input. Upon successful insertion, it notifies the user that the insertion was successful.

**Display data from the database – Model\_check.php**



The table within model\_check.php which outputs all models currently stored within the database. Uses a stored procedure called “show\_all\_models” which is a select \* from x table query. This is the outputted into a hard-coded table with predefined headings. The content of the table however is dynamic in the sense that it updates whenever the database is updated. So, if a new model is inserted into the database, the table does not need to be edited, it simply updates itself. It uses the select \* query to populate the table from the database.

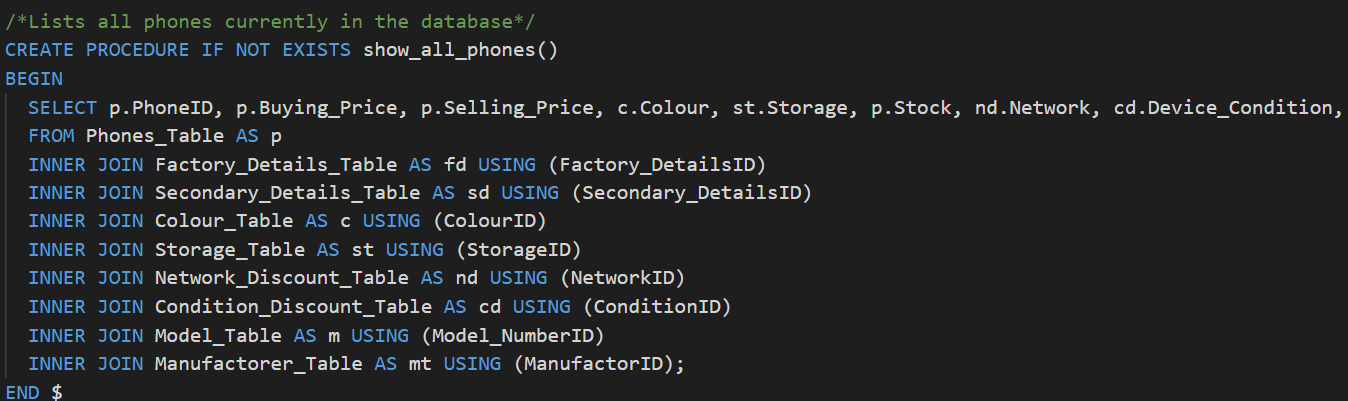
**Delete data from database – database\_delete.php**

****



Uses a standard delete statement, with values and tables being allocated through variable use which are posted values from the delete form. A switch statement is used to determine the productid type based on which table was selected. So, if the user wants to delete a tablet, the drop down selects the tablet table, and the switch statement changes the WHERE clause to be WHERE ‘x’ id, in this case, TabletID. The product to be deleted is a manual entry of the exact product id e.g. 3001 for a tablet. Should the entry not exist, the connection is killed, and an error message is thrown. Upon a successful deletion, the user gets a confirmation that the data was successfully deleted.

**Show at least one join – Phone\_database\_assignment.sql**

****

An inner join which joins all relevant tables to output all necessary information about phones within the database. This inner join is presented inside of a stored procedure and is called within show\_all\_phones.php to output the results to a table. Tables are declared as single letters within the inner join to reduce bloat of the length select query.

**References:**

* W3Schools “CSS Navigation Bar” [Online] Available at: <https://www.w3schools.com/css/css_navbar.asp> [Accessed 20th March, 2019]
* W3Schools “How to create a sign up form” [Online] Available at: <https://www.w3schools.com/howto/howto_css_signup_form.asp> [Accessed 20th March, 2019]
* Thatstevensguy “Disable arrows on number inputs [Online] Available at: <https://www.thatstevensguy.com/programming/disable-arrows-on-number-inputs/> [Accessed 26th March, 2019]
* TutorialsPoint “PHP Login Example” [Online] Available at: <https://www.tutorialspoint.com/php/php_login_example.htm> [Accessed 26th March, 2019