[IS113] Extra Exercises - Week 11 - Database Interaction

Objectives

- To master the concepts of database interaction in PHP

Instructions

- Questions with no asterisk mark are easy peasy.
- Questions marked with * are slightly challenging.
- Questions marked with ** are challenging.
- Questions marked with *** are very challenging.

Download

- **Resources**: Click here
- **Solutions**: Click here

NOTE: If you spot any mistakes/errors in the questions, please contact your instructors by email and state the issues. We will try to address it as soon as possible.

Database Connection (from inside PHP code)

1) **WAMP Users**
    a) Upon WAMP installation, if you have not changed your MySQL login info will be:
        i) **Username**: root
        ii) **Password**: <left empty>
2) **MAMP Users**
    a) For most students we have assisted, it appears that the default MySQL login info is:
        i) **Username**: root
        ii) **Password**: root
    b) Additionally, your **MySQL port** appears to be **3306** (*please verify this on your own laptop computer and **remember to note it down***).
        i) You will have to specify **port** in **`ConnectionManager.php`**.
        ii) Please remember to configure **`ConnectionManager.php`** on your own in all Extra Exercises as well as in Lab Test 2 questions **on your own** (as we instructional staff do NOT provide a separate **`ConnectionManager.php`** file for non-WAMP users).

Question 1: Person Filter (*)

Given:

- `q1/model`
  - `ConnectionManager.php, Person.php` (**complete**)
  - `PersonDAO.php` ***(partial)***
- `q1/`
  - `common.php` (**complete**)
  - `display.php` ***(partial)***
  - `setup.sql` (**complete**)

Read and use the given `setup.sql` to understand and create the necessary database and tables for this question.

Part A: Complete search method of "PersonDAO.php"

Complete **`search`** method of **`PersonDAO.php`** page to retrieve all persons from **`person`** table that satisfy the search criteria (minimum age, maximum age, and gender). Return all matching persons as an indexed array of **`Person`** objects. If Part A is completed well, the following would be the behavior when **`display.php`** is opened in the web browser:

When the page loads for the first time:

Gender: ○ Male ○ Female ◉ Any
Min Age: 0
Max Age: 200
Filter

Name	Gender	Age
Amy	F	28
Bill	M	18
Charles	M	17
Doraemon	F	32

*When one or more search criteria are specified and **Filter** submit button is clicked:*

(i) search criteria are specified

Gender: ○ Male ◉ Female ○ Any
Min Age: 0
Max Age: 30
Filter

Name	Gender	Age
Amy	F	28
Bill	M	18
Charles	M	17
Doraemon	F	32

(ii) output

Gender: ○ Male ○ Female ◉ Any
Min Age: 0
Max Age: 200
Filter

Name	Gender	Age
Amy	F	28

Part B: Complete `"display.php"`

Complete `display.php` so that the page remembers the values that the user selects before the **Filter** button is clicked. If Part B is completed well, the following would be the behavior when `display.php` is opened on the web browser:

(i) search criteria are specified

Gender: ○ Male ◉ Female ○ Any
Min Age: 0
Max Age: 30
Filter

Name	Gender	Age
Amy	F	28
Bill	M	18
Charles	M	17
Doraemon	F	32

(ii) output

Gender: ○ Male ◉ Female ○ Any
Min Age: 0
Max Age: 30
Filter

Name	Gender	Age
Amy	F	28

Question 2: Warehouse (**)

Given:

`q2/`

- `create.sql`
- `style.css`
- `ConnectionManager.php`, `Product.php`, `Warehouse.php`
- `categoryList.php`
- `searchByCategoriesAndPriceRange.php`
- `searchByCategory.php`
- `searchByPriceRange.php`
- `testGetCategories.php`
- `testSearchByCategory.php`
- `testSearchByPriceRange.php`

[1]Read and use the given **`create.sql`** to understand and create the necessary database and tables for this question.

Part A

Update class `Warehouse` to implement its method `getCategories()` to

1. Retrieve the list of product categories from database table `category`
2. Return the product categories as an indexed array of strings sorted alphabetically (case-sensitive) in ascending order

If done correctly, `testGetCategories.php` should display the following:

Test getCategories()

Alcoholic Drink
Cereals
Drink
Fruit
Meat
Seafood
Sweets and Chocolate

Part B

Create class `Product`

1. Four properties: Product name : String, Category name : String, Quantity : Integer, Price : Float
2. Constructor that takes in 4 parameters to initialize its properties.
3. Getter methods for its properties.

---

[1] This question includes some CSS to make the web page looks nicer and spice things up. CSS is not within the assessment scope of this course. If you wish to learn more about CSS, go to https://www.w3schools.com/css/.

Update class `Warehouse` to implement its method `searchByCategory($category_name)` to

1. Parameter
    a. `$category_name` is the product category to search for
2. Return an indexed array of `Product` objects representing products for the specified category sorted by products' name alphabetically (case-sensitive) in ascending order.

If done correctly, `testSearchByCategory.php` should display the following:

Test searchByCategory()

Seafood

Crab, 35, 20.00
Prawn, 13, 2.15
Promfret, 3, 18.80
Salmon, 23, 5.95

Sweets and Chocolate

Chewy Gums, 26, 3.00
Dark chocolate, 36, 2.25
Lollipop, 16, 1.20
Minty Pops, 62, 2.10
White chocolate, 6, 2.15

No such category

Nothing found.

Part C

Edit `searchByCategory.php` such that it has a form with

1. A drop down list of product categories retrieved using class `Warehouse`'s method `getCategories()`.
2. Button 'Search' that submits the form back to itself (same page) via HTTP GET.

Product Search by Category

Category Alcoholic Drink ▾ Search

Upon form submission, the page does the following:

1. The drop down list should show the category that user has selected.
2. Retrieves all products of the specified category using class `Warehouse`'s method `searchByCategory($category_name)`.

3. Display details of the products as shown in the table below sorted by products' name alphabetically (case-sensitive) in ascending order.
4. For quantity,
    a. If quantity is less than 10, display quantity in red.
    b. If quantity is less than 20, display quantity in orange.
    c. Otherwise, black.
    d. Look at the given `searchByCategory.php` for the CSS code for text color.

If done correctly, when user searches for category 'Sweets and Chocolate', the page should look like this:

Product Search by Category

Category Sweets and Chocolate ▾ Search

S/N	Product	Quantity	Price
1	Chewy Gums	26	$3.00
2	Dark chocolate	36	$2.25
3	Lollipop	16	$1.20
4	Minty Pops	62	$2.10
5	White chocolate	6	$2.15

Part D

Update class `Warehouse` to implement its method `searchByPriceRange($min_price, $max_price)` to

1. Parameters
    a. `$min_price` (float) is the minimum price to search for
    b. `$max_price` (float) indicate the price range to search for.
2. You may assume that `$min_price` is less than or equal to `$max_price`.
3. Return an indexed array of Product objects whose price is between `$min_price` and `$max_price` inclusive. The products are sorted by **price, then product's name** alphabetically (case-sensitive) in ascending order.

If done correctly, `testSearchByPriceRange.php` should display the following:

Test searchByPriceRange()

Between $1.2 and $1.6

Barley, Drink, 4, $1.20
Lemon Tea, Drink, 42, $1.20
Lollipop, Sweets and Chocolate, 16, $1.20
Coke, Drink, 24, $1.40
Pepsi, Drink, 34, $1.40
100 Plus, Drink, 41, $1.60

Between $10 and $18

Soju, Alcoholic Drink, 0, $10.00
Champion Champagne, Alcoholic Drink, 0, $12.45

Between $100 and $100

Nothing found.

Part E

Edit `searchByPriceRange.php` such that it has a form with

1. Text field 'Min price' with default value 0.
2. Text field 'Max price' with default value 100.
3. Button 'Search' that submits the form back to itself (same page) via HTTP GET.

Product Search by Price Range

Min price 0
Max price 100
Search

You may assume that user will always enter valid floating numbers for min and max prices, and min price is less than or equal to max price.

Upon form submission, the page does the following:

1. The two text fields should show the values that user has entered.
2. Retrieves all products whose price is between min and max prices inclusive.
3. Display details of the products as shown in the table below sorted by **price, then product's name** alphabetically (case-sensitive) in ascending order.
5. Do the same color coding for quantity as part C.

If done correctly, when user searches for prices between 1.2 and 1.6 inclusive, the page should look like this:

Product Search by Price Range

Min price 1.2
Max price 1.6
Search

S/N	Product	Category	Quantity	Price
1	Barley	Drink	4	$1.20
2	Lemon Tea	Drink	42	$1.20
3	Lollipop	Sweets and Chocolate	16	$1.20
4	Coke	Drink	24	$1.40
5	Pepsi	Drink	34	$1.40
6	100 Plus	Drink	41	$1.60

When user searches for prices between 1.2 and 1.6 inclusive, the page should look like this:

Product Search by Price Range

Min price 100
Max price 100
Search

Nothing found.

Part F

Update `categoryList.php` to display an ordered list of the product categories. Each category name is linked to `searchByCategory.php`. Upon clicking the category-name-hyperlink, `searchByCategory.php` should display the products for that category.

If done correctly, the page looks like this:

Product Category List

1. Alcoholic Drink
2. Cereals
3. Drink
4. Fruit
5. Meat
6. Seafood
7. Sweets and Chocolate

If user clicks on ‘Seafood’, the browser goes to `searchByCategory.php` (screenshot below) and displays ‘Seafood’ (as though user had selected ‘Seafood’ from the drop down list).

Product Search by Category

Category Seafood ▾ Search

S/N	Product	Quantity	Price
1	Crab	35	$20.00
2	Prawn	13	$2.15
3	Promfret	3	$18.80
4	Salmon	23	$5.95

Part G

Update class `Warehouse` to implement its method `searchByCategoriesAndPriceRange($category_name, $min_price, $max_price)` to

1. Parameters
    a. `$category_names` is an indexed array of strings representing the categories to search for
    b. `$min_price` (float) is the minimum price to search for
    c. `$max_price` (float) indicate the price range to search for.
2. You may assume that `$min_price` is less than or equal to `$max_price`.
3. Return an **associative array.**
    a. Key is product category name
    b. Value is an indexed array of Product objects for the specified category and whose price is between $min_price and $max_price inclusive. The products are sorted by **category name then product's name** alphabetically (case-sensitive) in ascending order.

Edit `searchByCategoriesAndPriceRange.php` such that it has a form with

1. A list of checkboxes for the product categories.
2. Text field ‘Min price’ with default value 0.
3. Text field ‘Max price’ with default value 100.
4. Button ‘Search’ that submits the form back to itself (same page) via HTTP GET.

Product Search by Category

Categories:	☐ Alcoholic Drink ☐ Cereals ☐ Drink ☐ Fruit ☐ Meat ☐ Seafood ☐ Sweets and Chocolate
Min price	0
Max price	100

Search

You may assume that user will always enter valid floating numbers for min and max prices, and min price is less than or equal to max price.

Upon form submission, the page does the following:

1. The form should show the values that user has selected or entered.
2. Retrieves all products for the specified category and whose price is between min and max prices inclusive.
3. Display details of the products as shown in the table below sorted by category name then product's name alphabetically (case-sensitive) in ascending order.
4. Do the same color coding for quantity as before.

If done correctly, sample screenshots of the page:

Product Search by Category

Categories:	☑ Alcoholic Drink ☑ Cereals ☐ Drink ☑ Fruit ☑ Meat ☐ Seafood ☐ Sweets and Chocolate
Min price	5.5
Max price	9.9

Search

S/N	Category	Product	Quantity	Price
1	Alcoholic Drink	Hadoken Beer	20	$6.55
2		Liger Beer	10	$6.55
3		Shoruken Beer	0	$5.55
4	Fruit	Papaya	22	$7.15
5	Meat	Half chicken	1	$5.50
6		Whole chicken	11	$9.90

Product Search by Category

Categories:	☐ Alcoholic Drink ☑ Cereals ☐ Drink ☐ Fruit ☐ Meat ☐ Seafood ☐ Sweets and Chocolate
Min price	2
Max price	4.5

Search

S/N	Category	Product	Quantity	Price
1	Cereals	Aunty Toby	51	$3.45
2		Cocoa Pebbles	5	$4.50
3		Crunchy Peanut	15	$4.50
4		Honey Star	35	$3.45
5		Oats and Oats	25	$2.00
6		Rasin Bran	5	$2.15

Product Search by Category

Categories:	☐ Alcoholic Drink ☐ Cereals ☐ Drink ☐ Fruit ☐ Meat ☐ Seafood ☐ Sweets and Chocolate
Min price	0
Max price	100

Search

Nothing found.

Product Search by Category

Categories:	☑ Alcoholic Drink ☐ Cereals ☑ Drink ☐ Fruit ☐ Meat ☐ Seafood ☐ Sweets and Chocolate
Min price	20
Max price	30

Search

Nothing found.

Question 3: KPop Stars (**)

Given:

```
q3/
```

- `ConnectionManager.php, common.php, create.sql`
- `Star.php, StarDAO.php`
- `display.php, edit.php, update.php`
- `images/* (there are FOUR (4) JPG image files)`

Read and use the given **`create.sql`** to understand and create the necessary database and tables for this question.

- Open `create.sql`. Take the SQL statements in this file and execute it (you may use **WorkBench** or **PHPMyAdmin**, whichever one you are comfortable with).
- It creates a schema **`kpop`**. Inside **`kpop`**, it creates a table **`star`**.

Part A (Difficulty: **)

Edit `display.php` such that it:

- Uses **`StarDAO` object** to query the database table **`star`** via public method **`getAll()`**, which returns an Indexed Array of **`Star` objects**.
- Receives an Indexed Array of **`Star` objects** and displays the stars' information in an HTML table.

`display.php`

Photo	Name	Gender	Headline	Edit Link
	Jennie	F	She is NOT related to Kim Jong Un	Edit
	Seolhyun	F	Her nose is natural	Edit
	Seungri	M	His name is all over the news now!	Edit
	Taeyang	M	He is still married	Edit

The last table column **"Edit Link"** must display a HyperLink to page **`edit.php`**.

- The HyperLink URL will look like this: **`edit.php?id=2`**
- Clicking on this link will make a new HTTP GET request to **`edit.php`** with one parameter with the name **`id`**. The value (e.g. `2` in the above example) is a particular star's **id** (as retrieved from the database). Your code can obtain this **id** from each **`Star` object** via public Getter method **`getID()`**.

Part B (Difficulty: **)

Suppose that the user clicks on **Seolhyun**'s **Edit** HyperLink. The user will be taken to **`edit.php`** with a particular **id**, e.g. **2** (this is the **ID** of **Seoulhyun** in my local MySQL database table **`star`**).

- Link: **`edit.php?id=2`**

Edit `edit.php` such that it:

- Retrieves the value of the parameter **id** from HTTP GET request.
- Takes this **id** value and calls **`StarDAO`** object's public method **`getStarByID($id)`**. This method is defined in **`StarDAO.php`**. Please go and have a look at the method. ***What does it do?***
  - It retrieves a row from the database table **`star`** where the **id** column value matches that of the method parameter **`$id`**.
  - If a matching row is found in table **`star`**, this method retrieves all column data and create a new **`Star` object**. This **`Star` object** is then returned to **`edit.php`**.
- Takes the **`Star` object** and displays the star's information as shown below:

`edit.php?id=2`
Name: Seolhyun Gender: F Headline: Her nose is natural Update Info

- Only ONE (1) property (or attribute) is **editable**.
  - **Headline** text can be updated by the user.
- **Name** and **Gender** cannot be updated via this webpage. Hence, we display them as text (without editable input fields).
- Upon keying in new data for **Headline** input field (text), the user clicks on the SUBMIT button "Update Info". It will then submit to **`update.php`** via HTTP POST method.
- Please see additional guiding comments inside **`edit.php`** for further instructions.

Part C (Difficulty: **)

*(Continuing with **Part B** example)*

Suppose that the same user clicks on "Update Info" SUBMIT button. It submits to **`update.php`**.

Edit `update.php` such that it:

- Retrieves the value of the parameter **id** AND the parameter **headline** from HTTP POST request.
- Calls **`StarDAO`** object's public method **`updateHeadline($id, $headline)`**. This method is defined in **`StarDAO.php`**. Please go and have a look at the method. ***What does it do?***
  - It updates the table (**`star`**) **row** where the row's **id** column value matches that of the method parameter **`$id`**. Specifically, it updates the value of the column **headline** in the matching row.
  - If the query executes successfully, then the method will return **Boolean True**.
  - Else, it will return **Boolean False**.
  - How do you check if **query** ran **successfully**?
    - See what **`$stmt->execute()`** returns. Does it return a Boolean value?

BEFORE editing the "Headline" text

`edit.php?id=2`

Name: Seolhyun
Gender: F
Headline: Her nose is natural

Update Info

AFTER editing the "Headline" text

`edit.php?id=2`

Name: Seolhyun
Gender: F
Headline: Her nose is really natural meh???

Update Info

AFTER clicking on the "Update Info" SUBMIT button in `edit.php`

`update.php`

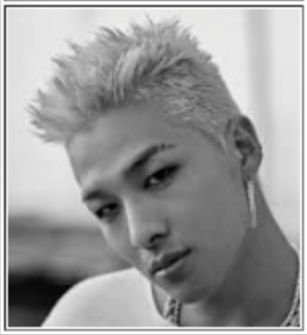
Update was successful!

Click here to return to Main Page

AFTER clicking on the HyperLink here

`display.php` shows the updated “Headline” text for Seolhyun (3rd row of the HTML table)

`display.php`

Photo	Name	Gender	Headline	Edit Link
	Jennie	F	She is NOT related to Kim Jong Un	Edit
	Seolhyun	F	Her nose is really natural meh???	Edit
	Seungri	M	His name is all over the news now!	Edit
	Taeyang	M	He is still married	Edit

Question 4: Location and Store Filter (*)

Given:

- `q4/model`
  - `ConnectionManager.php, Product.php Shop.php` (**complete**)
  - `ProductDAO.php` ***(partial)***
  - `ShopDAO.php` ***(partial)***
  -
- `q4/`
  - `common.php` (**complete**)
  - `display.php` ***(partial)***
  - `setup.sql` ***(run this before you start)***

The page allows the user to select a location and a shop name. These values are retrieved from the database and should not be hardcoded. The list of location and shop names are distinct, i.e. there should not be any duplicates. With these inputs, the application will proceed to check if the selected shop name exists at the selected location. If it does, it will return a list of products that are available by the shop. You can assume that a shop can exist in multiple locations, offering similar products. There are exceptional cases where the shop exists at the selected location but does not offer any products at this point of time.

NOTE: The suggested solution does not uses join tables in the SQL statements.

Part A: Complete "ProductDAO.php and ShopDAO.php"

Complete functions to retrieve the distinct list of locations, distinct list of shop names and list of products available at a shop. Return all matching data from `ProductDAO.php` as an indexed array of `Product` objects. The return of data from `ShopDAO.php` can be in the form of either an indexed array of `Shop` objects or strings.

Part B: Complete "display.php"

Complete `display.php` so that the page remembers the values that the user selects before the **Submit** button is clicked. If Part B is completed well, the following would be the behavior when `display.php` is opened on the web browser:

The following would be the behavior when `display.php` is opened in the web browser:

- *When the page loads for the first time, the distinct list of locations and shop names are provided in the drop down list. If nothing is selected, the default will be the first in the list:*

Products Available for the location and store :

Enter the location : Jurong ▾
Enter choose the shop name : MyShop ▾

Submit

The list of Products available at **MyShop** in **Jurong**

Item	Category	Price
apple	fruit	$0.65
orange	fruit	$0.80
celery	vegetable	$2.70
cabbage	vegetable	$3.00
broccoli	vegetable	$1.85

- *When the user selects a location and a shop name, the application will do a check. If the shop exists in the location but does not have any available products, the page will display the following:*

Products Available for the location and store :

Enter the location : Jurong ▾
Enter choose the shop name : BuyFromMe ▾

Submit

The store **BuyFromMe** in **Jurong** currently does not have any products for sale.

- *When the user selects a location and a shop name, the application will do a check. If the shop does not exist in the selected location, the page will display the following:*

Products Available for the location and store :

Enter the location : Jurong ▾
Enter choose the shop name : LowestPrice ▾

Submit

The location **Jurong** does not have shop **LowestPrice**.

Question 4b: Location and Store Filter Using Indexed Array within the Shop Class (**)

Copy the following files from Question 4 into the following drive:

- `q4/model`
  - `ConnectionManager.php` (no changes required)
  - `Product.php` (no changes required)
  - `ProductDAO.php` (no changes required)
  - `Shop.php` (***changes required as below***)
  - `ShopDAO.php` *(may need modifications)*
- `q4/`
  - `common.php` (no changes required)
  - `display.php` ***(changes required)***
  - `setup.sql` *(run this if you need to refresh your database)*

In this version, the class `Shop` will have three properties – the shop name, the shop location and an indexed array of products sold by the shop. The indexed array will consist of `Product` objects which can be retrieved from the database by calling the `ProductDAO` class at the constructor. The modifications will be in the constructor.

The class `Shop` will looks like this:

```
class Shop{
     private $name;
     private $location;
     private $items;
// this is an indexed array of Product objects.

     public function __construct ($name, $location) {
          $this->name = $name;
          $this->location = $location;

// Use of ProductDAO to retrieve the list of products
// available at the store.
          $dao = new ProductDAO();

          ... /* enter your codes here */
     }

}
```

Make the appropriate modifications of the codes in `Shop.php`, `ShopDAO.php` and `display.php` such that it is able to make use of the updated class `Shop`. The codes in `display.php` will only need to call `ShopDAO` objects. The behavior of `display.php` is the same as Question 4.

Question 5: Employment Statistics (**)

Given:

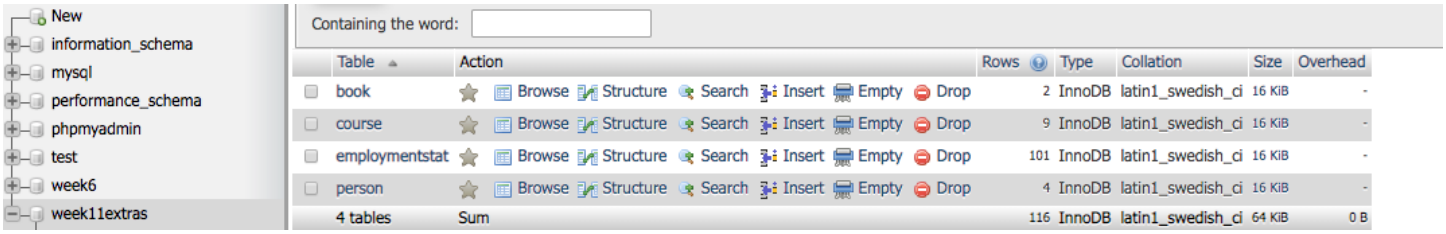
`q5/`

- `model/setup.sql` **(complete)**
- `model/populateDatabase.php` **(complete)**
- `model/ConnectionManager.php` **(complete)**
- `model/EmploymentStat.php` ***(partial)***
- `model/EmploymentStatDAO.php` ***(partial)***
- `common.php` **(complete)**
- `viewEmployment.php` ***(partial)***
- `updateEmployment.php` ***(partial)***
- `viewEmploymentB.php` ***(partial)***

Part A: Run `populateDatabase.php`

Create `week11extras` database using `setup.sql` given.
Run `populateDatabase.php` from your localhost to create `employmentstat` table in `week11extras` database.
(note: `populateDatabase.php` accesses the statistics provided by data.gov.sg via an API and loads them into `employmentstat` table)

If done correctly, you should see some records in `employmentstat` table as follows:



1 > >> | Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

←T→	id	year	university	school	degree	employment_rate	salary
Edit Copy Delete	1	2017	National University of Singapore	Faculty of Science	Bachelor of Science (Pharmacy) ##	99.1	3473
Edit Copy Delete	2	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration	94.9	3770
Edit Copy Delete	3	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration (Hons)	98.2	4272
Edit Copy Delete	4	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration (Accountancy)	97.2	3396
Edit Copy Delete	5	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration (Accountancy) ...	100	3689
Edit Copy Delete	6	2017	National University of Singapore	School of Computing	Bachelor of Computing (Communications And Media) *...	na	na
Edit Copy Delete	7	2017	National University of Singapore	School of Computing	Bachelor of Computing (Computational Biology) **	na	na
Edit Copy Delete	8	2017	National University of Singapore	School of Computing	Bachelor of Computing (Computer Science)	93.5	4510
Edit Copy Delete	9	2017	National University of Singapore	School of Computing	Bachelor of Computing (Electronic Commerce) **	na	na
Edit Copy Delete	10	2017	National University of Singapore	School of Computing	Bachelor of Computing (Information Security) **	na	na
Edit Copy Delete	11	2017	National University of Singapore	School of Computing	Bachelor of Computing (Information Systems)	94.5	4061
Edit Copy Delete	12	2017	National University of Singapore	School of Computing	Bachelor of Science (Business Analytics)	97.6	4114
Edit Copy Delete	13	2017	National University of Singapore	School of Design and Environment	Bachelor of Arts (Architecture) ##	91.3	4037
Edit Copy Delete	14	2017	National University of Singapore	School of Design and Environment	Bachelor of Arts (Industrial Design)	93.3	3034
Edit Copy Delete	15	2017	National University of Singapore	School of Design and Environment	Bachelor of Science (Project and Facilities Manage...	90.8	3105
Edit Copy Delete	16	2017	National University of Singapore	School of Design and Environment	Bachelor of Science (Real Estate)	93.8	3090
Edit Copy Delete	17	2017	National University of Singapore	Yong Loo Lin School (Medicine)	Bachelor of Medicine And Bachelor Of Surgery ##	100	4367
Edit Copy Delete	18	2017	National University of Singapore	Yong Loo Lin School (Medicine)	Bachelor of Science (Nursing)	97.4	3165

Part B: Create class `EmploymentStat`

1. Seven properties: id : Integer, year : Integer, university: String, school : String, degree : String, employment_rate : Float, avgSalary: Integer
2. Constructor that takes in 7 parameters to initialize its properties.
3. Getter methods for its properties.

Part C: Complete retrieveAll method of `EmploymentStatDAO.php`

Complete **retrieveAll** method of **`EmploymentStatDAO.php`** to retrieve all employment statistics from **`employmentstat`** table. Return the employment statistics data as an indexed array of **`EmploymentStat`** objects.

If done correctly, the following would be the behavior when **`viewEmployment.php`** is opened in the web browser:

Graduate Employment Survey Results - NTU, NUS, SIT, SMU & SUTD

Search employment statistics by University

University:

Filter Reset

Record ID	Year	University	School	Degree	Employment Rate	Average Salary (S$)
1	2017	National University of Singapore	Faculty of Science	Bachelor of Science (Pharmacy) ##	99.1	3473
2	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration	94.9	3770
3	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration (Hons)	98.2	4272
4	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration (Accountancy)	97.2	3396
5	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration (Accountancy) (Hons)	100	3689
6	2017	National University of Singapore	School of Computing	Bachelor of Computing (Communications And Media) **	na	na
7	2017	National University of Singapore	School of Computing	Bachelor of Computing (Computational Biology) **	na	na
8	2017	National University of Singapore	School of Computing	Bachelor of Computing (Computer Science)	93.5	4510
9	2017	National University of Singapore	School of Computing	Bachelor of Computing (Electronic Commerce) **	na	na
10	2017	National University of Singapore	School of Computing	Bachelor of Computing (Information Security) **	na	na
11	2017	National University of Singapore	School of Computing	Bachelor of Computing (Information Systems)	94.5	4061
12	2017	National University of Singapore	School of Computing	Bachelor of Science (Business Analytics)	97.6	4114
13	2017	National University of Singapore	School of Design and Environment	Bachelor of Arts (Architecture) ##	91.3	4037
14	2017	National University of Singapore	School of Design and Environment	Bachelor of Arts (Industrial Design)	93.3	3034
15	2017	National University of Singapore	School of Design and Environment	Bachelor of Science (Project and Facilities Management)	90.8	3105
16	2017	National University of Singapore	School of Design and Environment	Bachelor of Science (Real Estate)	93.8	3090
17	2017	National University of Singapore	Yong Loo Lin School (Medicine)	Bachelor of Medicine And Bachelor Of Surgery ##	100	4367
18	2017	National University of Singapore	Yong Loo Lin School (Medicine)	Bachelor of Science (Nursing)	97.4	3165
19	2013	National University of Singapore	Faculty of Engineering	Bachelor of Engineering (Civil Engineering)	96.1	3140
20	2017	National University of Singapore	Yong Loo Lin School (Medicine)	Bachelor of Science (Nursing) (Hons)	91.8	3280

Part D: Implement SearchByUniversity functionality

1. Complete **`searchByUniversity`** method of **`EmploymentStatDAO.php`** to retrieve the employment statistics of **a given university** from **`employmentstat`** table. Return the employment statistics data as an indexed array of **`EmploymentStat`** objects.
2. Complete **`viewEmployment.php`** by adding code to read the **university** input from the user and retrieve the employment statistics of that university, by using the **`searchByUniversity`** method implemented above.
3. Upon clicking the **Filter** button in **`viewEmployment.php`**, it should display the employment statistics of a given university. Upon clicking the **Reset** button, it should display back all the employment statistics of all the universities.

Your code should handle possible exceptions and invalid scenarios, such as errors in accessing the database, clicking the Filter button without entering the university value, entering an invalid university value, etc.

If done correctly, the following would be the behavior when **`viewEmployment.php`** is run in the web browser:

Entering the university input as "Singapore Management University":

Upon clicking Filter button:

Graduate Employment Survey Results - NTU, NUS, SIT, SMU & SUTD

Search employment statistics by University

University:

Filter Reset

Record ID	Year	University	School	Degree	Employment Rate	Average Salary (S$)
26	2017	Singapore Management University	School of Accountancy (4-years programme) *	Accountancy	98	3569
27	2017	Singapore Management University	School of Accountancy (4-years programme) *	Accountancy Cum Laude and above	97.8	4037
28	2017	Singapore Management University	School of Business (4-years programme) *	Business Management	93.3	3862
29	2017	Singapore Management University	School of Business (4-years programme) *	Business Management Cum Laude and above	95.8	4364
30	2017	Singapore Management University	School of Economics (4-years programme) *	Economics	91.3	4013
31	2017	Singapore Management University	School of Economics (4-years programme) *	Economics Cum Laude and above	93.2	4591
32	2017	Singapore Management University	School of Information Systems (4-years programme) *	Information Systems Management	95.1	3922
33	2017	Singapore Management University	School of Information Systems (4-years programme) *	Information Systems Management Cum Laude and above	96.1	4211
34	2017	Singapore Management University	School of Social Sciences (4-years programme) *	Social Sciences	91.3	3344
35	2017	Singapore Management University	School of Social Sciences (4-years programme) *	Social Sciences Cum Laude and above	87.9	3810
36	2017	Singapore Management University	School of Law (4-years programme) *	Law ++	97.6	4778
37	2017	Singapore Management University	School of Law (4-years programme) *	Law Cum Laude and above	97.9	5163
103	2018	Singapore Management University	School of Information Systems (4-years programme) *	Information System	98.4	4300

Upon clicking Reset button:

Graduate Employment Survey Results - NTU, NUS, SIT, SMU & SUTD

Search employment statistics by University

University:

Filter Reset

Record ID	Year	University	School	Degree	Employment Rate	Average Salary (S$)
1	2017	National University of Singapore	Faculty of Science	Bachelor of Science (Pharmacy) ##	99.1	3473
2	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration	94.9	3770
3	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration (Hons)	98.2	4272
4	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration (Accountancy)	97.2	3396
5	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration (Accountancy) (Hons)	100	3689
6	2017	National University of Singapore	School of Computing	Bachelor of Computing (Communications And Media) **	na	na
7	2017	National University of Singapore	School of Computing	Bachelor of Computing (Computational Biology) **	na	na
8	2017	National University of Singapore	School of Computing	Bachelor of Computing (Computer Science)	93.5	4510
9	2017	National University of Singapore	School of Computing	Bachelor of Computing (Electronic Commerce) **	na	na
10	2017	National University of Singapore	School of Computing	Bachelor of Computing (Information Security) **	na	na
11	2017	National University of Singapore	School of Computing	Bachelor of Computing (Information Systems)	94.5	4061
12	2017	National University of Singapore	School of Computing	Bachelor of Science (Business Analytics)	97.6	4114

Part E: Implement Create, Update, and Delete Employment Statistics functionality

1. Complete **`add`** method of **`EmploymentStatDAO.php`** to insert a new employment statistics into the **`employmentstat`** table. Return the Boolean value "TRUE" if insert operation is successful.

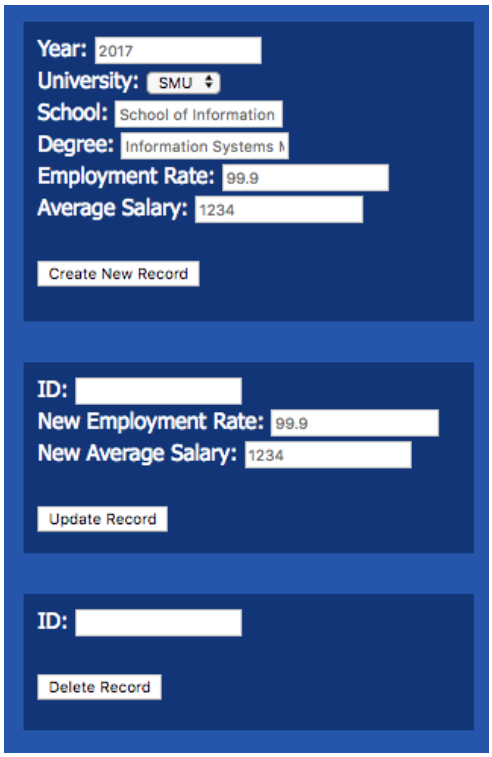
2. Complete **`update`** method of **`EmploymentStatDAO.php`** to update the **employment_rate** and **salary** of an existing employment statistics record in the **`employmentstat`** table, given its **id**. Return the Boolean value "TRUE" if update operation is successful.

3. Complete **`delete`** method of **`EmploymentStatDAO.php`** to delete an existing employment statistics record in the **`employmentstat`** table, given its **id**. Return the Boolean value "TRUE" if update operation is successful.

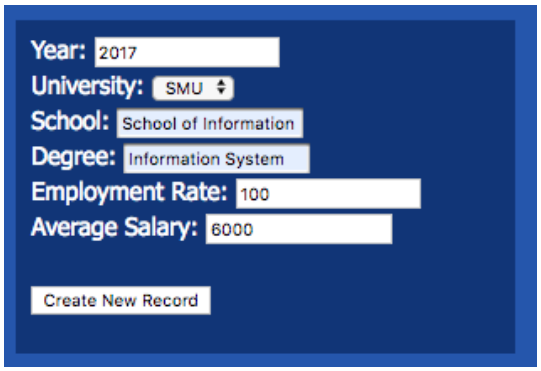
Your code should handle possible exceptions and invalid scenarios, such as errors in accessing the database, entering invalid values, etc.

If done correctly, the following would be the behavior when **`updateEmployment.php`** is run in the web browser:

Loading **`updateEmployment.php`** *for the first time:*

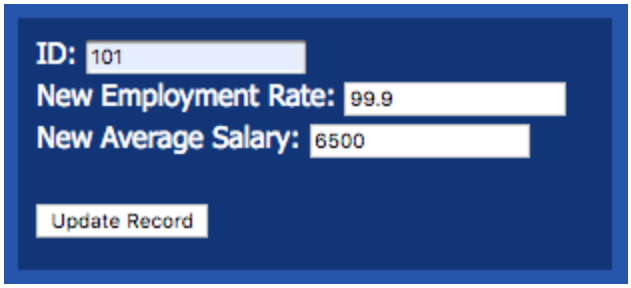


Creating a new employment statistics:



Upon clicking "Create New Record":

Updating an employment statistics:

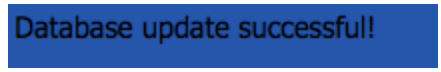


Upon clicking "Update Record":

If create and update operations are successful, some changes should be observed in the employmentstat table in the database:

Deleting an existing employment statistics:

Upon clicking "Delete Record":



In the employmentstat table in the database, record with id "101" should be deleted.

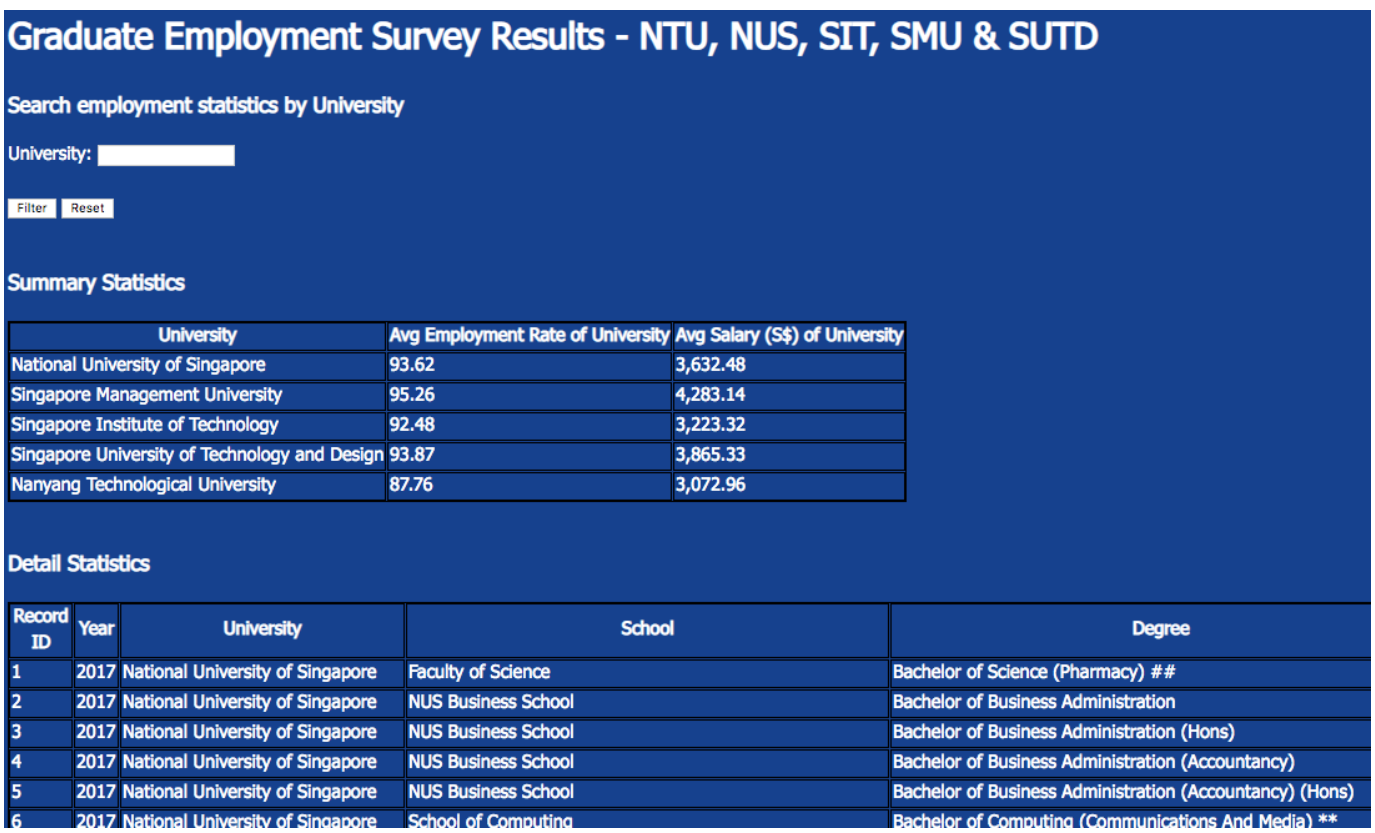
Part F: Implement computing average employment rate and salary of each university and each school

1. Complete **`viewEmploymentB.php`** such that by default (when the page loads the first time or when the user clicks the **Reset** button), it displays the average salary and employment rate of each university.

2. Complete **`viewEmploymentB.php`** such that when the user provides a university input, it should display the average salary and employment rate of each school in that university.

If done correctly, the following would be the behavior when **`viewEmploymentB.php`** is run in the web browser:

Loading **`viewEmploymentB.php`** *for the first time:*



Entering the university input as "Singapore Management University":

Graduate Employment Survey Results

Search employment statistics by University

University: Singapore Managemei

Filter Reset

Upon clicking the Filter button:

Graduate Employment Survey Results - NTU, NUS, SIT, SM

Search employment statistics by University

University:

Filter Reset

Summary Statistics

School	Avg Employment Rate of School	Avg Salary (S$) of School
School of Accountancy (4-years programme) *	97.90	3,803.00
School of Business (4-years programme) *	94.55	4,113.00
School of Economics (4-years programme) *	92.25	4,302.00
School of Information Systems (4-years programme) *	97.40	4,608.25
School of Social Sciences (4-years programme) *	89.60	3,577.00
School of Law (4-years programme) *	97.75	4,970.50

Upon clicking the Reset button:

Graduate Employment Survey Results - NTU, NUS, SIT, SMU & SUTD

Search employment statistics by University

University:

Filter Reset

Summary Statistics

University	Avg Employment Rate of University	Avg Salary (S$) of University
National University of Singapore	93.62	3,632.48
Singapore Management University	95.26	4,283.14
Singapore Institute of Technology	92.48	3,223.32
Singapore University of Technology and Design	93.87	3,865.33
Nanyang Technological University	87.76	3,072.96

Detail Statistics

Record ID	Year	University	School	Degree	Emplo Ra
1	2017	National University of Singapore	Faculty of Science	Bachelor of Science (Pharmacy) ##	99.1
2	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration	94.9
3	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration (Hons)	98.2
4	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration (Accountancy)	97.2
5	2017	National University of Singapore	NUS Business School	Bachelor of Business Administration (Accountancy) (Hons)	100
6	2017	National University of Singapore	School of Computing	Bachelor of Computing (Communications And Media) **	na
7	2017	National University of Singapore	School of Computing	Bachelor of Computing (Computational Biology) **	na
8	2017	National University of Singapore	School of Computing	Bachelor of Computing (Computer Science)	93.5
9	2017	National University of Singapore	School of Computing	Bachelor of Computing (Electronic Commerce) **	na
10	2017	National University of Singapore	School of Computing	Bachelor of Computing (Information Security) **	na

Question 6: Maintain a Restaurant Menu (**)

Given:

- `q6/model`
  - `ConnectionManager.php, Food.php` (**complete**)
  - `FoodDAO.php` ***(partial)***
- `q6/`
  - `common.php` (**complete**)
  - `maintain_menu.php` (**partial**)
  - `edit.php` ***(partial)***
  - `delete.php` ***(partial)***
  - `setup.sql` ***(run this before you start)***

This exercise allows you to maintain the Food Menu at IS113 Kiosk. This application makes use of three php programs to provide an interface for the user to maintain the set of data in the database. The primary key for each food item is defined as the Stock-Keeping-Unit (SKU in short) of type integer. The use of an SKU will identify a unique record from the database. The database also keeps track of the description, category and price of the SKU.

Part A: Complete "FoodDAO.php"

Complete functions in FoodDAO.php to perform the following:

- retrieve all records of food items that are offered, sorted by SKU. The method will return an indexed array of `Food` objects.
- retrieve a single record identified by the SKU the user input. The method will return a `Food` object.
- update a single record in the database for the SKU selected by the user The method will return a status.
- delete a single record in the database for the SKU selected by the user The method will

Part B: Complete "maintain_menu.php", “edit.php”, “delete.php”

- *The `maintain_menu.php` will serve as the landing page of the application. There are two parts. At the top, it will display a snapshot of what are the food items available in the database, together with the description and prices. For each record, it will have a link should the user want to edit or delete the record.*

- *After that, there is an interface which will allow the user to create new food items.*

- *The following would be the behavior when `maintain_menu.php` is opened in the web browser:*

Food Menu at IS113 Kiosk

SKU	Description	Category	Price		
101	Fish and Chips	Main	$15.80	Edit	Delete
102	Beef Steak	Main	$20.90	Edit	Delete
103	Noodle	Main	$10.70	Edit	Delete
201	Fries	Side	$10.70	Edit	Delete
202	Salad	Side	$10.70	Edit	Delete
301	Orange Juice	Drink	$5.70	Edit	Delete
302	Apple Juice	Drink	$6.80	Edit	Delete

Add New Food Item

SKU :	
Description :	
Category :	
Price :	

Add Item

- *A user can add a new food item by keying in all the values and click on **Add Item**.*

Add New Food Item

SKU :	401
Description :	Ice cream
Category :	Dessert
Price :	9.90

Add Item

- *Upon successful creation of the new record. The menu is refreshed and the successful statement is provided. The*

Food Menu at IS113 Kiosk

SKU	Description	Category	Price		
101	Fish and Chips	Main	$15.80	Edit	Delete
102	Beef Steak	Main	$20.90	Edit	Delete
103	Noodle	Main	$10.70	Edit	Delete
201	Fries	Side	$10.70	Edit	Delete
202	Salad	Side	$10.70	Edit	Delete
301	Orange Juice	Drink	$5.70	Edit	Delete
302	Apple Juice	Drink	$6.80	Edit	Delete
401	Ice cream	Dessert	$9.90	Edit	Delete

Add New Food Item

SKU :	
Description :	
Category :	
Price :	

Add Item

Food item : **Ice cream** inserted into the menu.

- *If the user enters a set of values but with the same SKU. The error message will be shown.*

For example, if the user enters

Add New Food Item

SKU :	401
Description :	Apple Pie
Category :	Dessert
Price :	2.50

Add Item

Upon clicking Add Item, the screen will show

Food Menu at IS113 Kiosk

SKU	Description	Category	Price		
101	Fish and Chips	Main	$15.80	Edit	Delete
102	Beef Steak	Main	$20.90	Edit	Delete
103	Noodle	Main	$10.70	Edit	Delete
201	Fries	Side	$10.70	Edit	Delete
202	Salad	Side	$10.70	Edit	Delete
301	Orange Juice	Drink	$5.70	Edit	Delete
302	Apple Juice	Drink	$6.80	Edit	Delete
401	Ice cream	Dessert	$9.90	Edit	Delete

Add New Food Item

SKU :	
Description :	
Category :	
Price :	

Add Item

Error in creating food item : Apple Pie. Check your data.

- *To edit a record, click on the Edit link. The following will be the display. The user will only be able to update the description, category and price. To complete the update, click on 'Update Info'. A link is provided at the end of the page to return to the landing page.*

Update Food Item

SKU :	101
Description :	Fish and Chips
Category:	Main
Price :	15.80

Update Info

Click here to return to Main Page

- *A successful update will show the following :*

Update Food Item

SKU :	101
Description :	Fish and Chips II
Category:	Main New
Price :	20.80

Update Info

Food item : **Fish and Chips II** updated successfully.

Click here to return to Main Page

- *To delete a record, click on the Delete link from `maintain_menu.php`. The details of the item will be displayed. Click on Confirm to delete the item.*

Delete Item

SKU :	101
Description :	Fish and Chips II
Category:	Main New
Price :	20.80

Confirm

Click here to return to Main Page

- *A successful delete will show the following :*

Delete Item

SKU :	101
Description :	Fish and Chips II
Category:	Main New
Price :	20.80

Confirm

Delete was successful!

Click here to return to Main Page

Question 7: Blog Posts (**)

Given:

- `q7/database`
  - `create.sql`
- `q7/model`
  - `ConnectionManager.php`
  - `Post.php`
  - `PostDAO.php`
- `q7/`
  - `common.php`
  - `add.html`
  - `add.php`
  - `delete.php`
  - `edit.php`
  - `update.php`
  - `display.php`

Read and use the given **`create.sql`** to understand and create the necessary database and tables for this question.

`create.sql`

```
...
create table post (
    id integer auto_increment primary key,
    create_timestamp datetime,
    update_timestamp datetime,
    subject varchar(100),
    entry text,
    mood varchar(30)
);
...
```

- **id** is an internal (to MySQL database table post) ID
  - It is auto-generated and auto-incremented by the MySQL database.
  - Users or you (developer) do NOT need to manually add **id** data when insert new rows.

- **create_timestamp**
  - Indicates the datetime of data (row) insertion.
  - Any new blog posts being added to the **post** table via your web application will have **CURRENT_TIMESTAMP** as the default value. See **CURRENT_TIMESTAMP** below.
  - For more information, please check out: https://www.w3schools.com/sql/func_mysql_current_timestamp.asp
  - For example:

```
INSERT INTO post
                    (
                        create_timestamp,
                        update_timestamp,
                        subject,
                        entry,
                        mood
                    )
                VALUES
                    (
                        CURRENT_TIMESTAMP,
                        CURRENT_TIMESTAMP,
                        'I hate school',
                        'I do not want to go to school',
                        'Sad'
                    )
```

- **update_timestamp**
  - Indicates the timestamp of data (row) update.
  - Any new blog posts being added to the **post** table via your web application will have **CURRENT_TIMESTAMP** as the default value. See **CURRENT_TIMESTAMP** above.
  - When a particular **post** is updated (e.g. subject/entry/mood change) via your web application, your code must also update **update_timestamp** by setting it to **CURRENT_TIMESTAMP**.
  - This way, we can capture in the MySQL database … when was the last time a particular **post** was updated.

Part A (Difficulty: **)

Edit **`display.php`** such that it:

- Uses **`PostDAO` object** to query the database table **`post`** via public method **`getAll()`**, which returns an Indexed Array of **`Post` objects**.

- Receives an Indexed Array of **`Post` objects** and displays the posts' details in an HTML table.

`display.php`

My Blog Posts

ID	Create Timestamp	Last Update Timestamp	Subject	Edit Link	Delete Link
1	2019-01-23 22:00:00	2019-01-23 22:00:00	Term Starts Again	Edit	Delete
2	2019-01-25 23:59:02	2019-01-25 23:59:02	I Suck	Edit	Delete
3	2019-01-29 09:15:00	2019-01-29 09:15:00	My Puppy	Edit	Delete
4	2019-02-05 21:00:00	2019-02-05 21:00:00	CNY Homework	Edit	Delete
5	2019-02-14 13:12:00	2019-02-14 13:25:00	My First Love	Edit	Delete

Add a New Blog Post

1. The table column **"Edit Link"** must display a HyperLink to page **`edit.php`**.
    a. The HyperLink URL will look like this: **`edit.php?id=1`**
    b. Clicking on this link will make a new HTTP GET request to **`edit.php`** with one parameter with the name **`id`**. The value (e.g. **`1`** in the above example) is a particular post's **id** (as retrieved from the database). Your code can obtain this **id** from each **`Post` object** via public Getter method **`getID()`**.

2. The table column **"Delete Link"** must display a HyperLink to page **`delete.php`**.
    a. The HyperLink URL will look like this: **`delete.php?id=1`**
    b. Clicking on this link will make a new HTTP GET request to **`delete.php`** with one parameter with the name **`id`**. The value (e.g. **`1`** in the above example) is a particular post's **id** (as retrieved from the database). Your code can obtain this **id** from each **`Post` object** via public Getter method **`getID()`**.

3. "**Add a New Blog Post**" at the bottom of the page is a HyperLink to page **`add.html`**.

Part B (Difficulty: **) EDIT

In `display.php`, suppose that the user clicks on the **first post**'s **"Edit"** HyperLink.

ID	Create Timestamp	Last Update Timestamp	Subject	Edit Link
1	2019-01-23 22:00:00	2019-01-23 22:00:00	Term Starts Again	Edit

The user is taken to **`edit.php`** with a particular **id**, e.g. **1** (this is the **ID** of the **first post** in my local MySQL database table **`post`**).

Edit `edit.php` such that it:

- Retrieves the value of the parameter **id** from HTTP GET request.
- Takes this **id** value and calls **`PostDAO`** object's public method **`get($id)`**. This method is partially implemented in **`PostDAO.php`**.
    - **Complete this method** such that it retrieves a row from the database table **`post`** where the **id** column value matches that of the method parameter **`$id`**.
    - If a matching row is found in table **`post`**, this method retrieves all column data and create a new **`Post` object**. This **`Post` object** is then returned to **`edit.php`**.
- **`edit.php`** takes this **`Post` object** and displays the post's details as shown below:

`edit.php?id=1`

Create Timestamp: 2019-01-23 22:00:00
Last Update Timestamp: 2019-01-23 22:00:00
Subject: Term Starts Again
Entry:
Another term started omg i wanna die alrdy...
Mood: Sad
Update Info

Click here to return to Main Page

- The user should be able to key in new data for:
    - **subject**
    - **entry**
- The user should be able to select new **mood** (drop-down list).

Upon keying in or selecting new data:

- The user clicks on the SUBMIT button "Update Info".
- The form will submit to **`update.php`** via HTTP POST method.

Part C (Difficulty: **) UPDATE

*(Continuing with **Part B** example)*

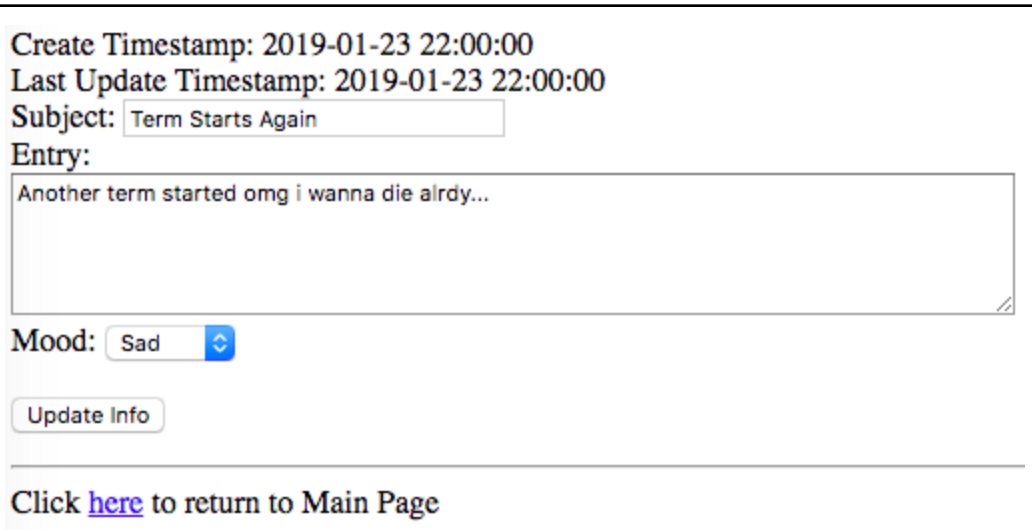
Suppose that the same user clicks on "Update Info" SUBMIT button. It submits to **`update.php`**.

Edit `update.php` such that it:

- Retrieves the following from HTTP POST request.
  - **id**
  - **subject**
  - **entry**
  - **mood**
- Calls **`PostDAO`** object's public method **`update($id, $subject, $entry, $mood)`**.
  - This method is defined in **`PostDAO.php`**. This method is partially implemented.
  - **Complete this method** such that:
    - It updates the table (**`post`**) **row** where the row's **id** column value matches that of the method parameter **`$id`**. It must update **update_timestamp, subject, entry, mood**.
    - If the query executes successfully, then the method will return **Boolean True**.
    - Else, it will return **Boolean False**.
    - How do you check if **query** ran **successfully**?
      - See what **`$stmt->execute()`** returns. Does it return a Boolean value?
      - Query failed?
        - Try **var_dump**-ing **`$stmt->errorinfo()`** and see what it shows.

BEFORE editing

`edit.php?id=1`



AFTER editing the "subject" text and "entry" text

`edit.php?id=1`

Create Timestamp: 2019-01-23 22:00:00
Last Update Timestamp: 2019-01-23 22:00:00
Subject: Term Starts Again OMGG!!!
Entry:
Another term started omg i wanna die aIrdy... I want to travel and enjoy life. School is terrible.

Mood: Sad

Update Info

Click here to return to Main Page

AFTER clicking on the "Update Info" SUBMIT button in `edit.php`

`update.php`

Update was successful!

Click here to return to Main Page

AFTER clicking on the HyperLink here

`display.php` shows the updated "Subject" text for the first post (1st row of the HTML table)
Do you also notice **"Last Update Timestamp"** reflects a new **timestamp**?

`display.php`

My Blog Posts

ID	Create Timestamp	Last Update Timestamp	Subject	Edit Link	Delete Link
1	2019-01-23 22:00:00	2019-03-20 08:14:57	Term Starts Again OMGG!!!	Edit	Delete
2	2019-01-25 23:59:02	2019-01-25 23:59:02	I Suck	Edit	Delete
3	2019-01-29 09:15:00	2019-01-29 09:15:00	My Puppy	Edit	Delete
4	2019-02-05 21:00:00	2019-02-05 21:00:00	CNY Homework	Edit	Delete
5	2019-02-14 13:12:00	2019-02-14 13:25:00	My First Love	Edit	Delete

Add a New Blog Post

Click on **Edit** HyperLink and let's check to see if all details were updated correctly.
`edit.php` correctly displays all new data.

`edit.php?id=1`

Create Timestamp: 2019-01-23 22:00:00
Last Update Timestamp: 2019-03-20 08:14:57
Subject: Term Starts Again OMGG!!!
Entry:
Another term started omg i wanna die alrdy... I want to travel and enjoy life. School is terrible.
Mood: Sad

Update Info

Click here to return to Main Page

Part D (Difficulty: **) DELETE

Remember the guy that asked me out on Valentine's Day?
DUH! He dumped me. So, I want to erase him from my blog FOREVER! -_-;

In `display.php`, suppose that the user clicks on the **"Delete"** HyperLink.

5	2019-02-14 13:12:00	2019-02-14 13:25:00	My First Love	Edit	Delete

The user is taken to **`delete.php`** with a particular **id**, e.g. **5** (this is the **ID** of the **last post** in my local MySQL database table **`post`**).

Edit `delete.php` such that it:

- Retrieves the value of the parameter **id** from HTTP GET request.
- Takes this **id** value and calls **`PostDAO`** object's public method **`get($id)`**.
    - **You should have completed this method's implementation in Part B**.
- **`delete.php`** takes the **`Post` object** returned by **`get($id)`**. It displays the post's details as shown below:

`delete.php?id=5`

Subject	My First Love
Entry	A very handsome boy gave me roses for Vday and asked me out!
Mood	Happy
Create Timestamp	2019-02-14 13:12:00
Update Timestamp	2019-02-14 13:25:00

Confirm Delete

Click here to return to Main Page

- The user clicks on the SUBMIT button "Confirm Delete".
- The form will submit to **`delete.php`** via HTTP POST method.
- The form has ONE (1) **hidden** input inside the FORM:

```
<input type='hidden' name='id' value='5'>
```

- It is hidden - such that it does not display in the web browser.
    - *View Source will show the above HTML though.*
- Hidden input fields are submitted as part of form submission.

Upon “Confirm Delete”, the page displays:

`delete.php`

Delete was successful

Click here to return to Main Page

So what’s going on in **`delete.php`**?

- It uses a **`PostDAO`** object to call its public method **`delete($id)`**. This method is partially implemented.
  - **Complete this method** such that it deletes a row from the database table **`post`** where the **id** column value matches that of the method parameter **`$id`**.
  - If the query executes successfully, then the method will return **Boolean True**.
  - Else, it will return **Boolean False**.
  - How do you check if **query** ran **successfully**?
    - See what **`$stmt->execute()`** returns. Does it return a Boolean value?
    - Query failed?
      - Try **var_dump**-ing **`$stmt->errorinfo()`** and see what it shows.
- Upon successful delete, display:
  - **Delete was successful**
- Upon unsuccessful delete, display:
  - **Delete was NOT successful**

Now, let’s go see if the old memory of the heartbreaker is really GONE!!!

`delete.php`

Delete was successful

Click here to return to Main Page

Click on here HyperLink.

`display.php` no longer lists my stupid love blog post. :-(
The post (with ID 5) is gone from the database table **`post`** permanently.

`display.php`

My Blog Posts

ID	Create Timestamp	Last Update Timestamp	Subject	Edit Link	Delete Link
1	2019-01-23 22:00:00	2019-03-20 08:14:57	Term Starts Again OMGG!!!	Edit	Delete
2	2019-01-25 23:59:02	2019-01-25 23:59:02	I Suck	Edit	Delete
3	2019-01-29 09:15:00	2019-01-29 09:15:00	My Puppy	Edit	Delete
4	2019-02-05 21:00:00	2019-02-05 21:00:00	CNY Homework	Edit	Delete

Add a New Blog Post

Part E (Difficulty: **) INSERT

I found a new eye candy. He is my new boyfriend. His name is Justin. I can't wait to write about him!!!

In `display.php`, suppose that the user clicks on the **"Add a New Blog Post"** HyperLink.

`display.php`

My Blog Posts

ID	Create Timestamp	Last Update Timestamp	Subject	Edit Link	Delete Link
1	2019-01-23 22:00:00	2019-03-20 08:14:57	Term Starts Again OMGG!!!	Edit	Delete
2	2019-01-25 23:59:02	2019-01-25 23:59:02	I Suck	Edit	Delete
3	2019-01-29 09:15:00	2019-01-29 09:15:00	My Puppy	Edit	Delete
4	2019-02-05 21:00:00	2019-02-05 21:00:00	CNY Homework	Edit	Delete

Add a New Blog Post

The user is taken to **`add.html`** and fills out the form with new details.

`add.html`

Add a New Blog Post

Subject: My new boyfriend Justin

Entry:

Justin is the best boyfriend EVAAA. He has big eyes. He has the best smile in the world. He is just so perfect in every way. I am gonna marry him.

Mood: Happy

Submit New Post

Click here to return to Main Page

- The user clicks on the SUBMIT button "Submit New Post".
- The form will submit to **`add.php`** via HTTP POST method.
- *Have a look at the HTML inside **`add.html`**. You will see some fun **JavaScript** script for form validation.*

So what's going on in **`add.php`**?

- It uses a **`PostDAO`** object to call its public method **`add($subject, $entry, $mood)`**. This method is partially implemented.
  - **Complete this method** such that it **inserts a NEW ROW** into the database table **`post`**.
  - The SQL query string is provided for you inside the method.
  - If the query executes successfully, then the method will return **Boolean True**.
  - Else, it will return **Boolean False**.
  - How do you check if **query** ran **successfully**?
    - See what **`$stmt->execute()`** returns. Does it return a Boolean value?
    - Query failed?
      - Try **var_dump**-ing **`$stmt->errorinfo()`** and see what it shows.
- Upon successful insertion, display:
  - **Insertion was successful**
- Upon unsuccessful insertion, display:
  - **Insertion was NOT successful**

`add.php`

Insertion was successful

Click here to return to Main Page

Now, let's go see if my new post about my new love Justin is listed!
Click on here HyperLink (above).

`display.php` shows the latest list of posts.
The new post has an auto-generated (by MySQL) ID of 6.

`display.php`

My Blog Posts

ID	Create Timestamp	Last Update Timestamp	Subject	Edit Link	Delete Link
1	2019-01-23 22:00:00	2019-03-20 08:14:57	Term Starts Again OMGG!!!	Edit	Delete
2	2019-01-25 23:59:02	2019-01-25 23:59:02	I Suck	Edit	Delete
3	2019-01-29 09:15:00	2019-01-29 09:15:00	My Puppy	Edit	Delete
4	2019-02-05 21:00:00	2019-02-05 21:00:00	CNY Homework	Edit	Delete
6	2019-03-20 08:42:09	2019-03-20 08:42:09	My new boyfriend Justin	Edit	Delete

Add a New Blog Post

Click on Edit HyperLink (ID 6) and make sure that all post details are correctly displayed in **`edit.php`**.

Question 8: Club (**)

1. Use `wad_club.sql` to create the required database schema and tables
2. Take a look at all the given PHP files.
3. Implement the methods in `PersonDAO.php` and `ClubDAO.php` according to the comments in the files.
4. Edit `index.php, view_club.php, add_members.php` and `add_members_process.php` such that it works according to the page flows described below.

Description of page flows:

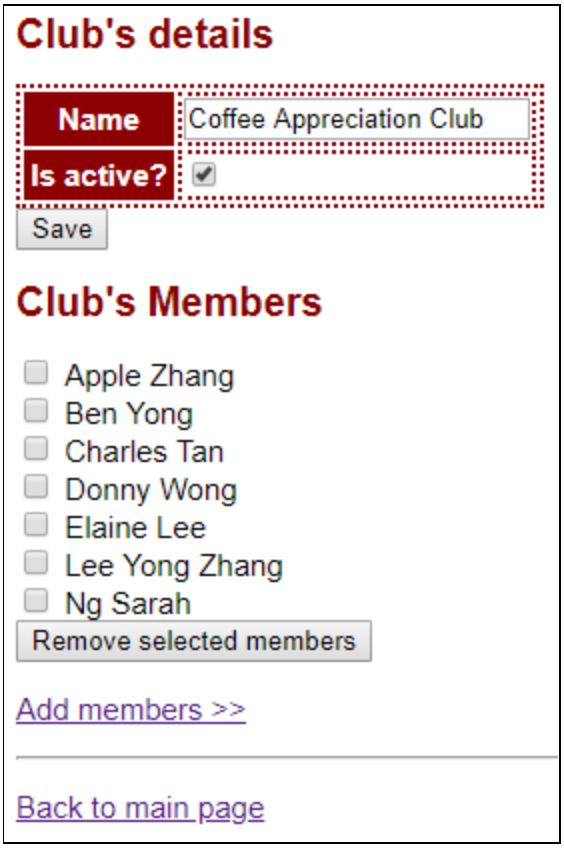
1. Upon first loading of `index.php`, it displays a table of all clubs' details

Name	Is Active?
Coffee Appreciation Club	Active
PHP Supporters	Active
Python Gourment Club	Inactive

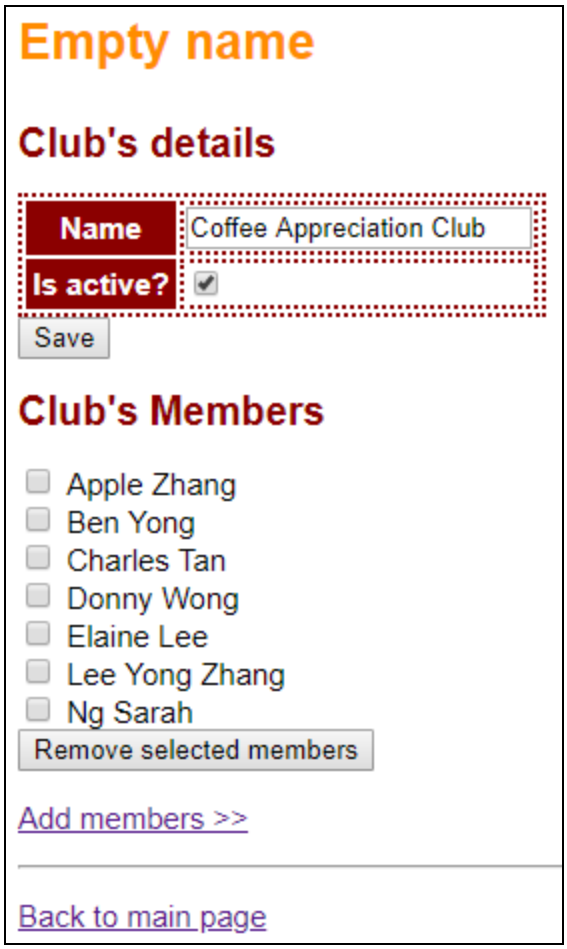
2. Click on a club's name to go to `view_club.php` to view that club's details. E.g. if user clicks on 'Coffee Appreciation Club', `view_club.php` will display:
    a. The club's details in a table:
        i. Row 1 has a text field that has the club's name by default.
        ii. Row 2 has a checkbox. If the club is active, the checkbox is ticked. Otherwise, the checkbox is not ticked.
    b. `'Save'` button.

    c. The names of the club's members; one name per line. Clicking on a member's name will tick its corresponding checkbox.
    d. '`Remove selected members`' button.

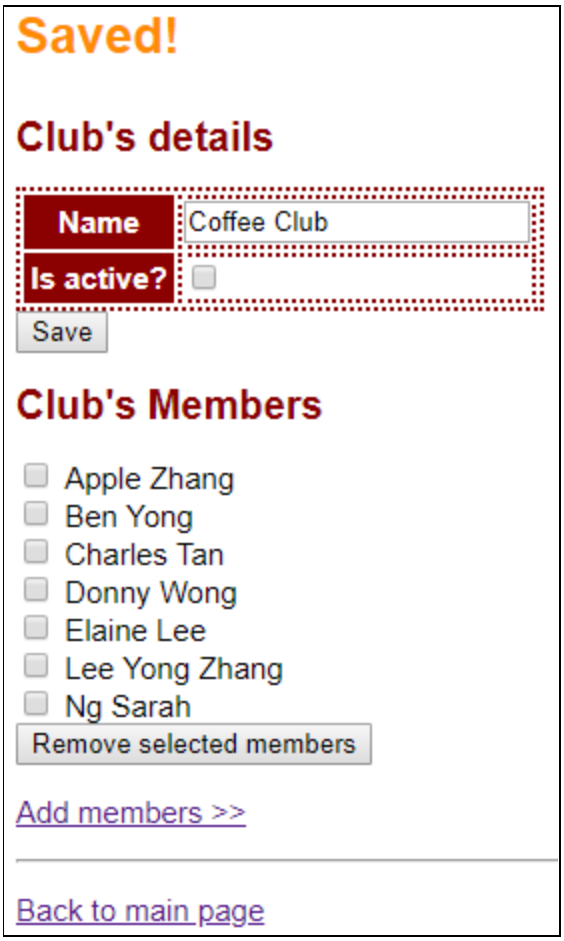
    e. '`Add members >>`' links to `add_members.php`.



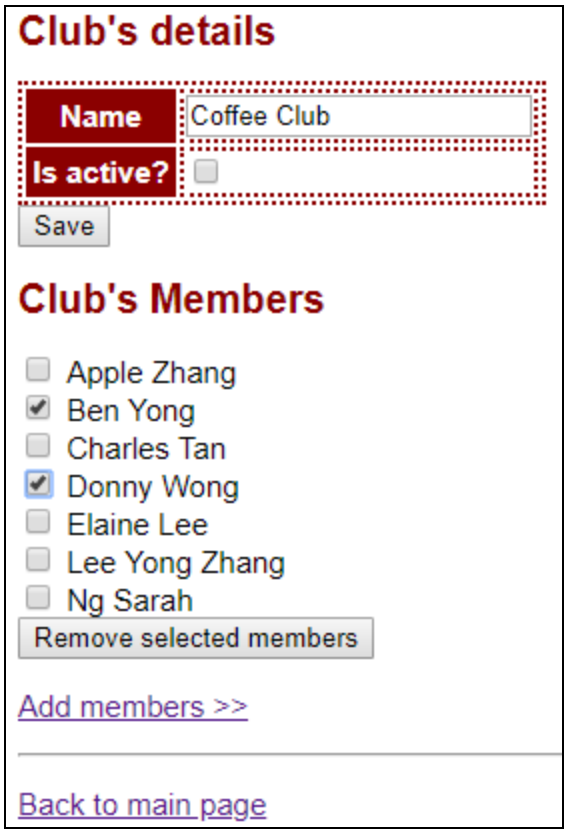
3. When he clicks the ‘Save’ button, the form is submitted back to `view_club.php` via HTTP POST.
4. If the submitted club’s name is empty or has only white spaces, display
    a. ‘`Empty name`’ and
    b. the club’s original details should be shown in the form.



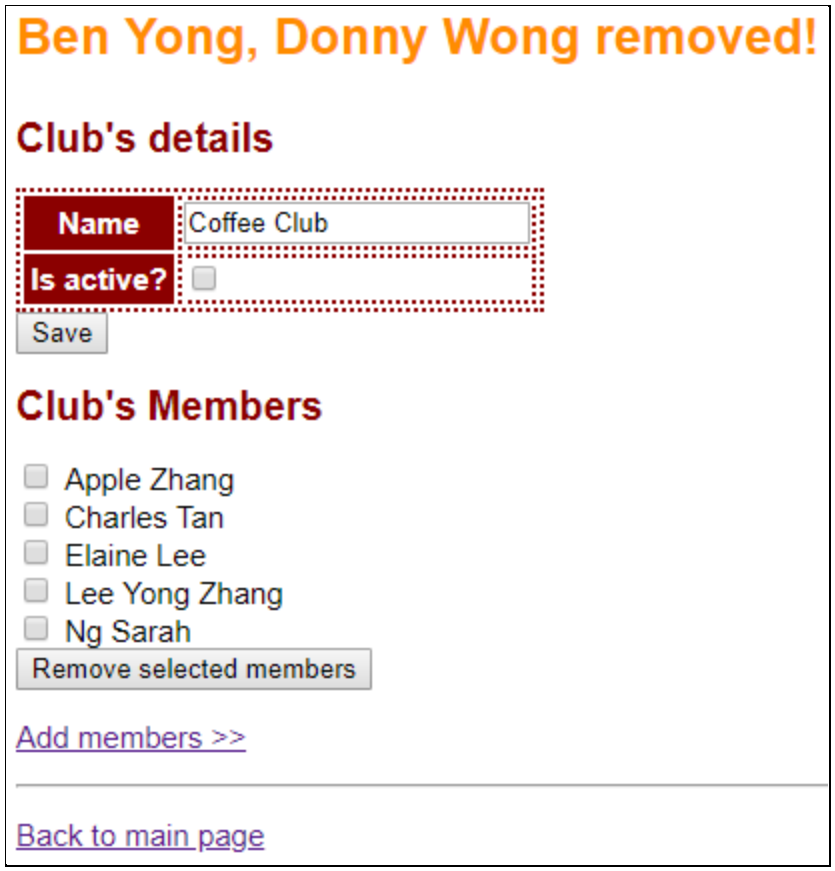
5. Otherwise, save the updated details for the club, display
    a. '`Saved!`' and
    b. the club's updated details should be shown in the form.



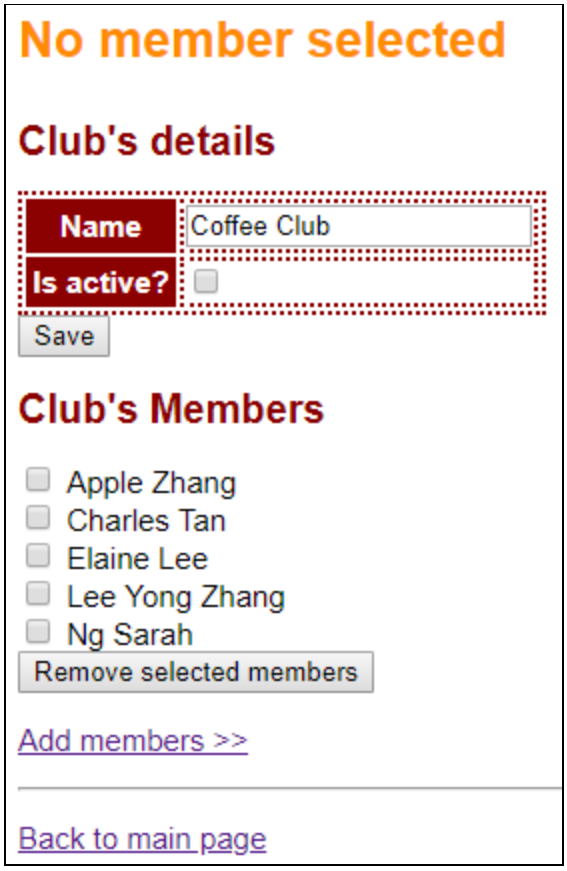
6. If user ticks one of more club's members and click 'Remove selected members' button, remove the selected members and display '*member 1's name, member 2's name, … removed!*'.
   a. E.g. user selects 'Ben Yong' and 'Donny Wong' and submits the form.



   b. The 2 members are gone from the list of members displayed. Message 'Ben Yong, Donny Wong removed!' is shown.

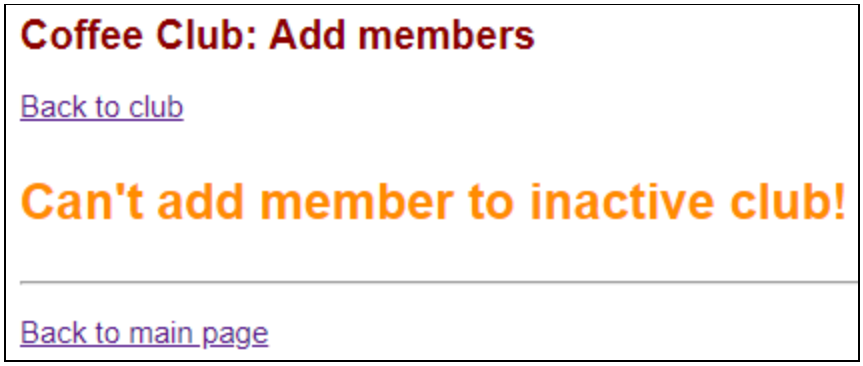


7. However, if user never select any member and clicks ‘Remove selected members’ button, display ‘`No member selected`’.

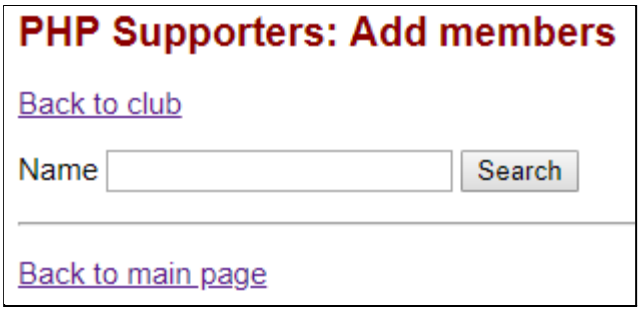


8. Click on link '`Add members >>`' that links to `add_members.php` which allows user to add members to the club. Page `add_members.php` displays
    a. Heading 1 '*`Club's Name`*`: Add members`'.
    b. '`Back to club`' links back `view_club.php` to display the details of the club.

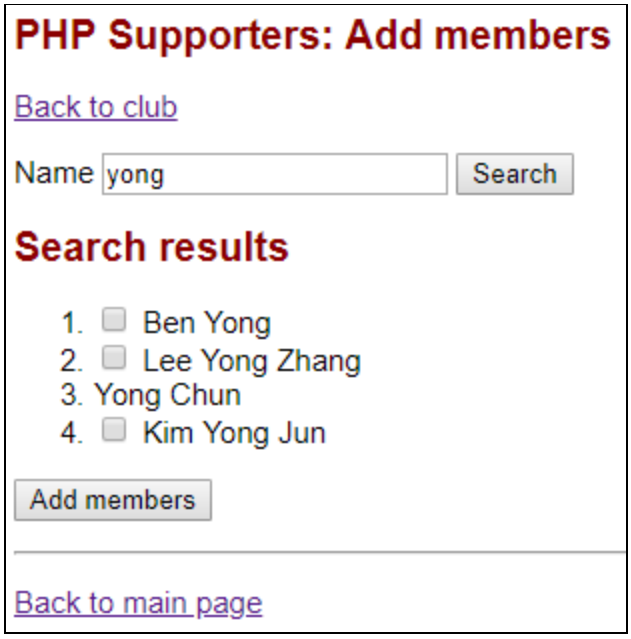
9. For inactive clubs (e.g. 'Coffee Club'), page `add_members.php` displays '`Can't add member to inactive club!`'.



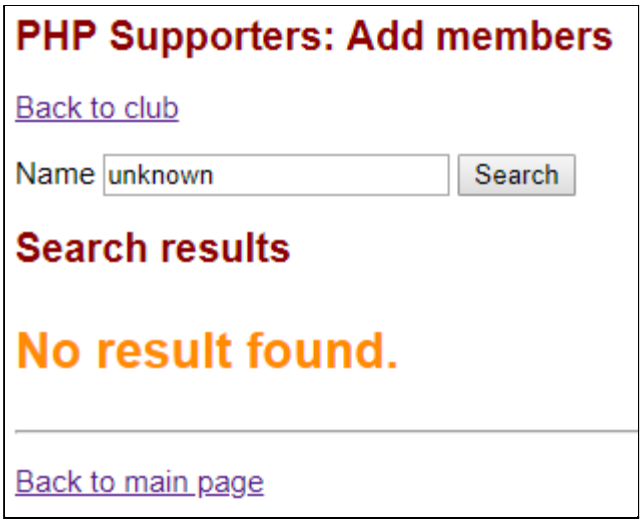
10. For active clubs (e.g. 'PHP Supporters'), page `add_members.php` displays at ext field that allows user to specify a name and a '`Search`' button.



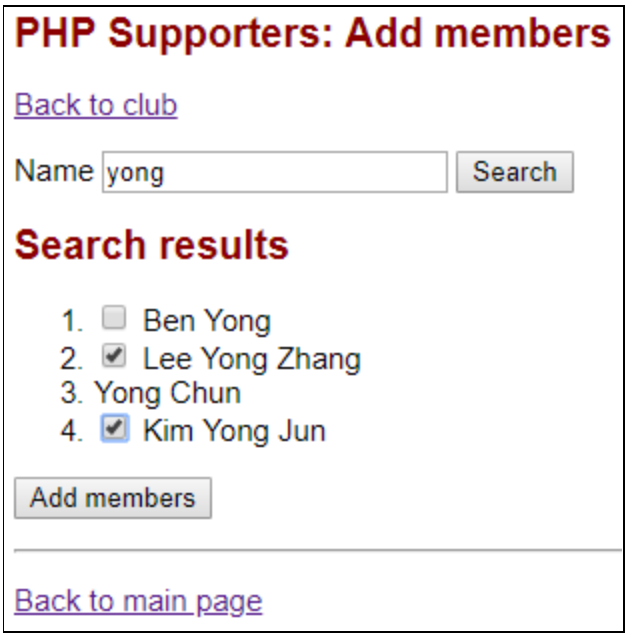
11. Click on '`Search`' button submits the form back to `add_members.php` via HTTP GET. Page `add_members.php` displays
    a. An ordered list of persons' names that contain the text specified by the user, ignoring case.
        i. A checkbox is displayed for those who are NOT members of the club.
        ii. Clicking on the names in the search result will tick its corresponding checkbox.
    b. '`Add members`' button



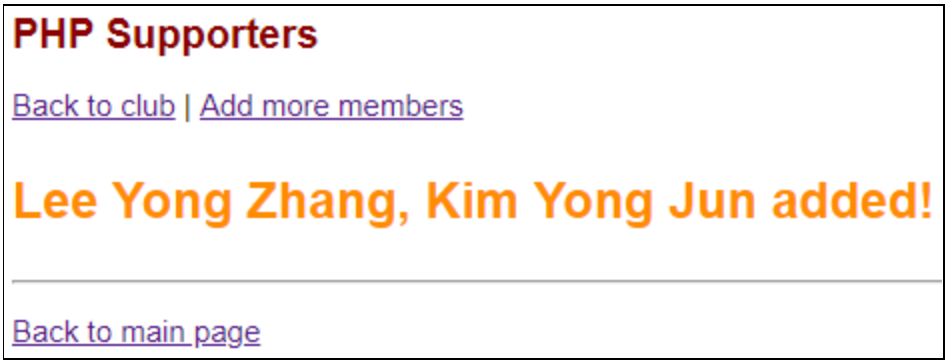
    c. If user leaves 'Name' text field empty, displays all persons (because all names contains the search string aka empty string).
    d. If there is no result (i.e. no one name contains the search string), display '`No result found.`'



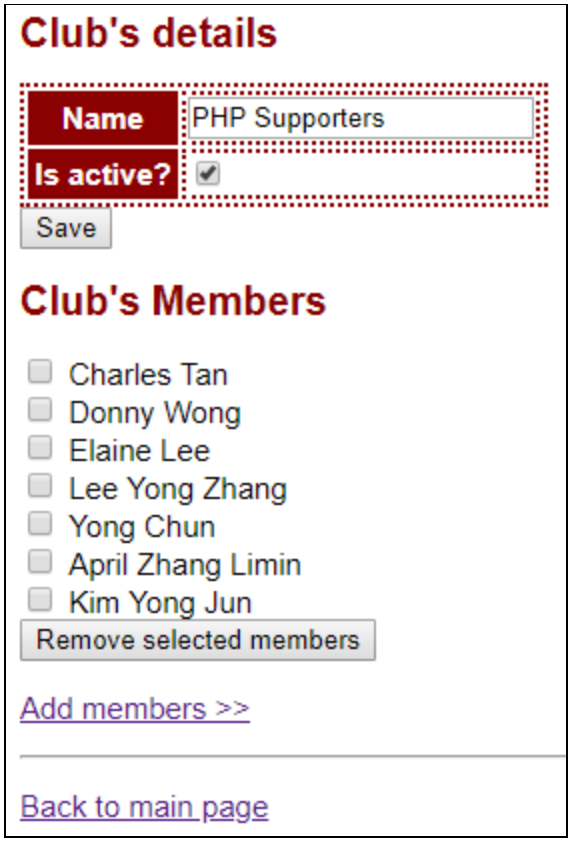
12. When user selects one or more person and clicks '`Add members`' button, the form is submitted to `add_members_process.php` via HTTP POST.



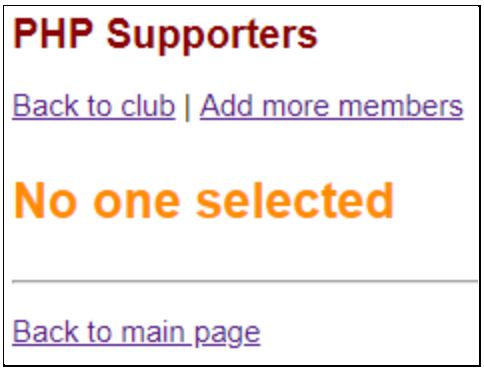
13. Page `add_members_process.php` adds the selected members to the club and displays
    a. Heading 1 '*`Club's Name`*'.
    b. '`Back to club`' links back `view_club.php`
    c. '`Add more members`' links back `add_members.php`
    d. Display message '*`member 1's name, member 2's name, ...`* `added!`'.



e. If you click ‘`Back to club`’, `view_club.php` should display the new added members.



14. If user didn’t select any person and clicks ‘`Add members`’ button, `add_members_process.php` displays ‘`No one selected`’.



Question 9 - Cat

Go to **`cat`** directory**.** Complete the following **Parts A** and **B**.

In the last few weeks, we have been using **Associative Arrays** or **Indexed Arrays** to represent and store "things" such as persons, books, fruits, students, etc.

For example (see **`data.php`** file):

- **`$cats`** is an **Indexed Array** where each cat is represented and stored as an **Associative Array**.
    - The **key** is an **attribute** of a cat (e.g. name, age, gender, status).

```
$cats = [

    // 1st cat
    [
        'name'      => 'Dirty',
        'age'       => 12,
        'gender'    => 'M',
        'status'    => 'A'
    ],
    // 2nd cat
    [
        'name'      => 'Filthy',
        'age'       => 7,
        'gender'    => 'F',
        'status'    => 'A'
    ],
    // 3rd cat
    [
        'name'      => 'Boring',
        'age'       => 3,
        'gender'    => 'M',
        'status'    => 'A'
    ]

    // ... and so on
];
```

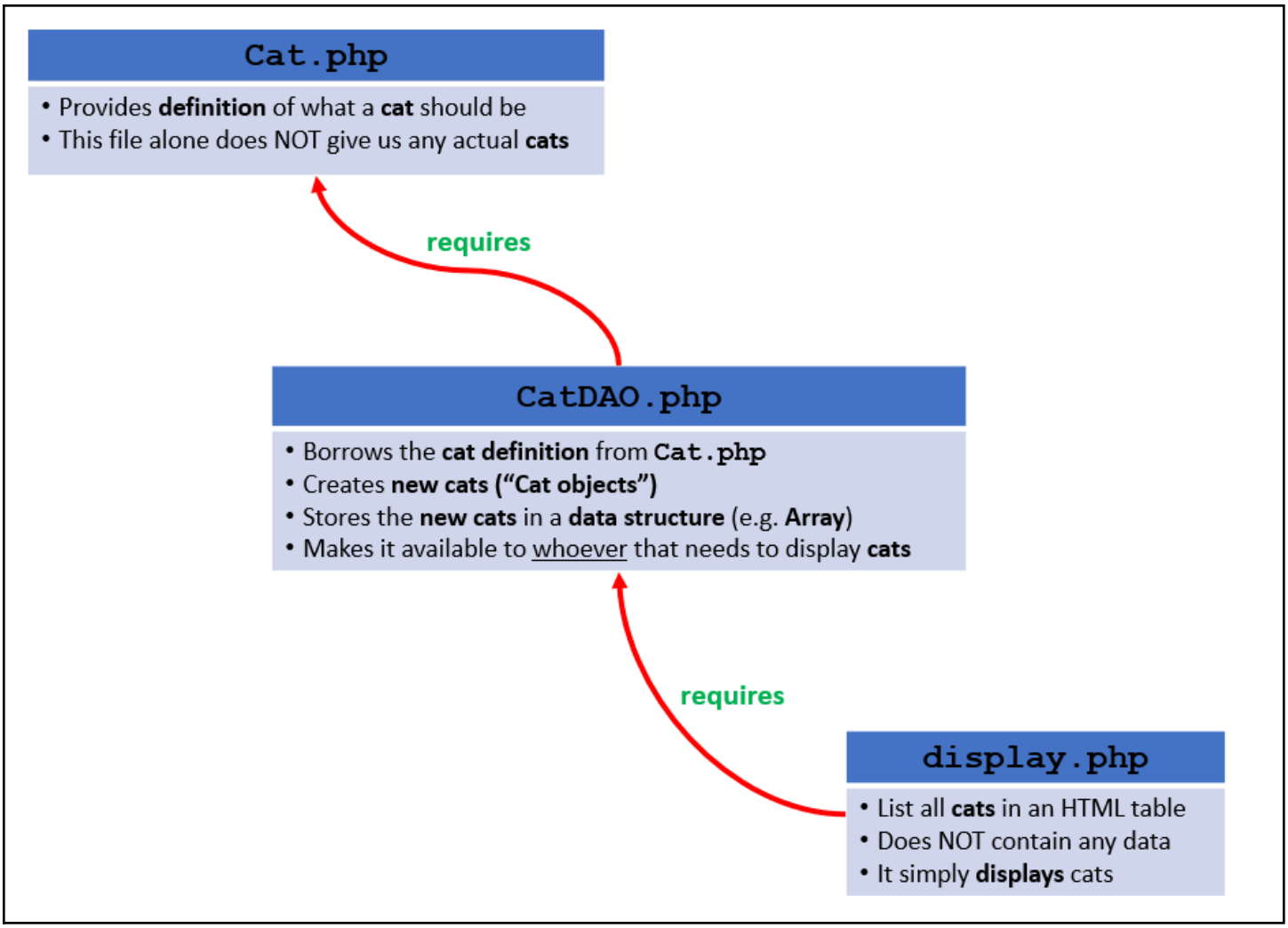
As shown above, we previously created new **cats** and stored them in an **Indexed Array**. Another way to **store** cat information is by using **Classes/Objects**.

Part A (*)

Complete **`Cat.php`** such that:

- **`Cat`** class defines what a cat should be. Every **`Cat`** **object** has the following FOUR (4) attributes:
    - **name** (e.g. 'Dirty')
    - **age** (e.g. 12)
    - **gender** (e.g. 'M' indicating ***male***, 'F' indicating ***female***)
    - **status** (e.g. 'A' indicating ***available***, 'P' indicating ***pending adoption***)
- Implement its constructor so that it takes in values for the FOUR (4) attributes
- Implement **Getter** methods for all of the attributes.

Great! Now that we have **`Cat.php`** that defines what every **cat** should look like, we can use this **definition** to create **new cats**!



We declare a **DAO** (Data Access Object) **class** inside **`CatDAO.php`** file.

`CatDAO.php`

```
<?php
require_once 'Cat.php';

class CatDAO {

    private $cats;

    // constructor
    public function __construct() {
        // Pre-populate static data
        $this->cats = [
            new Cat('Dirty', 12, 'M', 'A'),
            new Cat('Filthy', 7, 'F', 'A'),
            new Cat('Boring', 3, 'M', 'A'),
            new Cat('Needy', 3, 'M', 'P'),
            new Cat('Lazy', 1, 'F', 'P')
```

```
        ];
    }

    // whoever needs $cats, call this method off CatDAO object
    public function getCats() {
        return $this->cats;
    }
}
?>
```

- **`CatDAO`** class's constructor pre-populates **`$cats` Array** with FIVE (5) **`Cat`** objects.
- Via its **`getCats()`** public method, **`CatDAO`** class allows other PHP pages to access the **`$cats` Array**. For instance, later on, **`display.php`** will need to display all the **cats**.

Part B (**)

Complete **`display.php`** file. It must show all cats' information as shown below:

`display.php`

Our Cats

Name	Age	Gender	Status
Dirty	12	M	Available
Filthy	7	F	Available
Boring	3	M	Available
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption

Where can **`display.php`** obtain the information about all the **cats**?

- From **`CatDAO.php`** file!!!
- It **requires** **`CatDAO.php`** file (e.g. **require_once**).
- It needs to **create a new** **`CatDAO`** **object**.
- Using this new **`CatDAO`** **object**, it can call all **public methods** of **`CatDAO`** class.
    - For now, we only have TWO (2) public method **`getCats()`** and the constructor.
    - The **`getCats()`** method will return an **Indexed Array** containing **`Cat`** **objects**.
- Can you now see how… the **concern of data retrieval** (**`CatDAO.php`**) is completely separated from **displaying of data** (**`display.php`**)?

`display.php`

```
<?php
require_once 'CatDAO.php';
$dao = new CatDAO();
$cats = $dao->getCats(); // $cats is an Indexed Array of Cat objects
?>
<html>
<body>
    <h1>Our Cats</h1>
    <table border='1'>
```

```
        <tr>
            <th>Name</th>
            <th>Age</th>
            <th>Gender</th>
            <th>Status</th>
        </tr>
        <?php
            foreach($cats as $cat_object) {
                echo "
                    <tr>
                        <td>
                            {$cat_object->getName()}

        ... and more code below...
```

Question 10 - Cat2

Go to **`cat2`** directory**.** Complete the following **Parts A** and **B**.

Copy the following files from **`<web root>/is113/extra10/cat/`** folder into the current folder **`<web root>/is113/extra10/cat2/`**.

- `Cat.php`
- `CatDAO.php`
- `display.php`

<u>Part A (**)</u>

Edit **`CatDAO.php`** file:

- Implement **`getCatsByStatus($status)`** public method.
- This method takes ONE (1) parameter, **`$status`**, where the valid values are:
  - 'A'
  - 'P'
- Given the parameter value of 'A', it is to:
  - Look for one or more **`Cat`** objects in **`$cats`** **Array** where each cat's **status** is 'A'
  - Insert all matching **`Cat`** objects into an Indexed Array and return it.
- Likewise, for the parameter value of 'P', it is to perform the same but this time, the returned **Indexed Array** will contain all **`Cat`** objects where each cat's **status** is 'P'.
- **Test Cases**
  - In your web browser, open **`test.php`**.
  - There are TWO (2) test cases inside.
  - Each test case must produce correct results. Verify that your new method returns the correct results.

<u>Part B (**)</u>

Edit **`display.php`** file.

When the page loads in a web browser **for the first time**, it must display display all cats.

`display.php`

Our Cats

Name	Age	Gender	Status
Dirty	12	M	Available
Filthy	7	F	Available
Boring	3	M	Available
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption

Filter by Status:
Available ▼

Filter

Next, when the user selects **Available** as the filtering value, and clicks on **Filter SUBMIT button**, the page displays:

`display.php`

Our Cats

Name	Age	Gender	Status
Dirty	12	M	Available
Filthy	7	F	Available
Boring	3	M	Available

Filter by Status:
Available ▼

Filter

NOTE: The page must remember and pre-select the user's form input **"Filter by Status"**. For instance, in the above example, "**Available**" option is pre-selected in the drop-down list.

Next, when the user selects **Pending Adoption** as the filtering value, and clicks on **Filter SUBMIT button**, the page displays:

`display.php`

Our Cats

Name	Age	Gender	Status
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption

Filter by Status:
Pending Adoption ▾

Filter

NOTE: The page must remember and pre-select the user's form input **"Filter by Status"**. For instance, in the above example, "**Pending Adoption**" option is pre-selected in the drop-down list.

Question 11 - Cat3

Go to **`cat3`** directory**.** Complete the following **Parts A** and **B**.

Copy the following files from **`<web root>/is113/extra10/cat2/`** folder into the current folder **`<web root>/is113/extra10/cat3/`**.

- `Cat.php`
- `display.php`

I'm VERY SORRY to share with you... that our cat **Dirty** passed away last night... (a moment of silence please).

Layfoo, the adoption agency head, would like to request you (programmer) to **not display** Dirty's information as he is no longer with us. So, what are you (programmer) now going to do?

- Go open up page **`CatDAO.php`**.
- Go into the **constructor** method.
- Manually remove the line of code creating a new **Cat object** corresponding to **Dirty**.

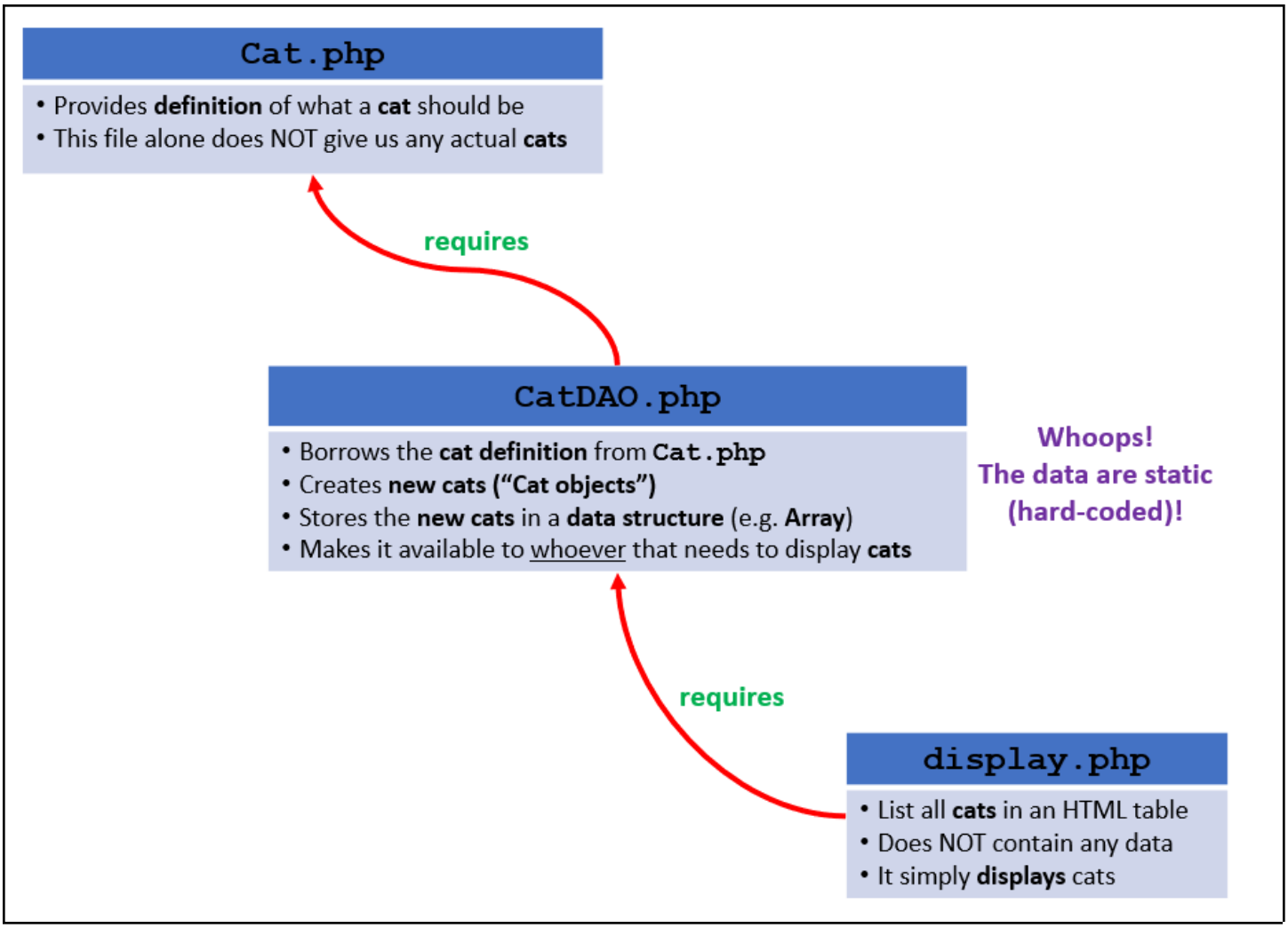
`CatDAO.php`

```
<?php
require_once 'Cat.php';

class CatDAO {

    private $cats;

    // constructor
    public function __construct() {
        // Pre-populate static data
        $this->cats = [
            new Cat('Dirty', 12, 'M', 'A'), // remove this line... Dirty died
            new Cat('Filthy', 7, 'F', 'A'),
            new Cat('Boring', 3, 'M', 'A'),
            new Cat('Needy', 3, 'M', 'P'),
            new Cat('Lazy', 1, 'F', 'P')
        ];
    }
```

WAIT A SECOND…

1. What if there are **users** currently accessing my website? Will they get an error message WHILE I make this code fix?

2. What if **more cats die** in the coming weeks/months? Will I have to change my **code** AGAIN?
3. What if there are **more cats** coming into the agency? Will I have to change my **code** AGAIN?



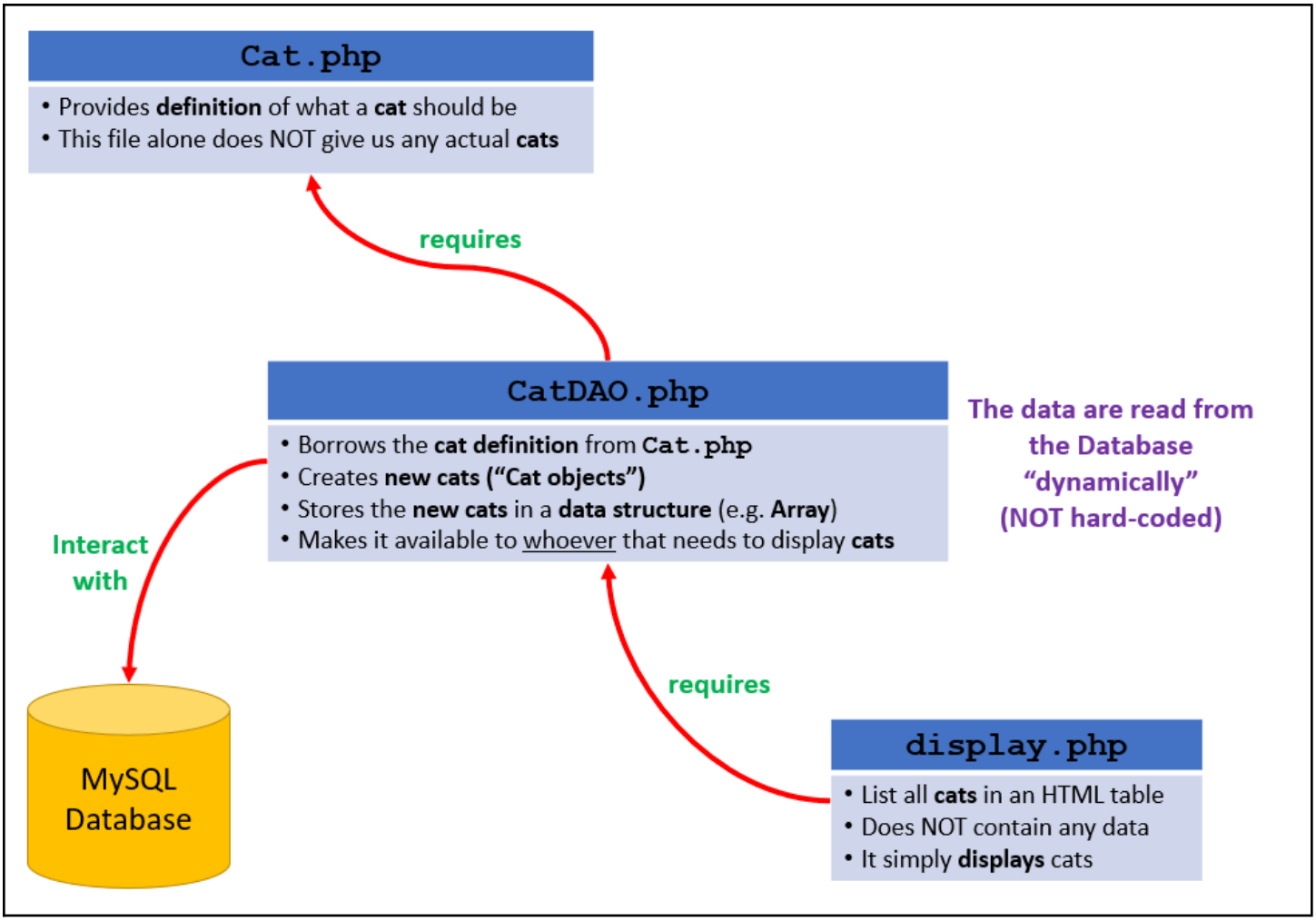
Wow! Our data are **static**! It's **hard-coded**… INSIDE our code (`CatDAO.php`) OMG!!!

Honestly, I (programmer) don't think I can afford taking my website down EVERY TIME … the data need to be updated. Especially given Layfoo's testimonial:

- On average, 3-4 new cats are abandoned and dropped off at his agency office;
- On average, 1-2 cats die every week;
- On average, 2-3 new adoptions happen per month.

Now, we foresee… **frequent data updates**. Definitely, you won't want to keep updating data **hard-coded** INSIDE your PHP code!

So, how do we tackle this problem?



- Above, we completely separated out **data storage** away from PHP code.
- **Data Storage** is now handled exclusively by the **MySQL Database**.
- `CatDAO.php` now can focus on:
  a. **Interacting with the Database**
     - Create (C)
     - Read (R)
     - Update (U)
     - Delete (D)
  b. Store retrieved data (from MySQL Database) into **variables** (e.g. Class objects) so that **other PHP files** can access the data and display, etc.

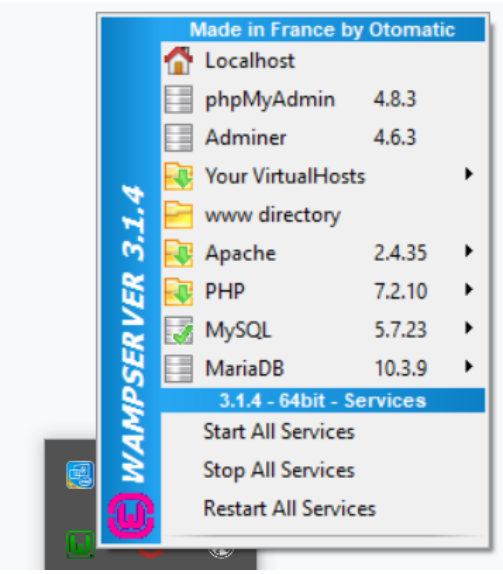
Part A (Creating a Database)

Agency Head Layfoo populates the MySQL database, and he will now maintain the database. you (programmer) do NOT have to worry about updating the data at all.

- During the software (web app) development, you NEVER have your code interact with the **production Database**. What if you make mistakes in your code… that **wipes out** the data in the **production Database**? → DIE ALREADY!!!
- Hence, typically, software developers will **create a replica of the production Database** (if it's too large, then take a subset of it) in their **local development/testing environment**.

Since you're still developing this website for the Cat Adoption Agency… you will have to create a replica of Layfoo's MySQL database… locally on your computer.

1. In your web browser, go to: http://localhost/phpmyadmin/
   a. Alternatively, you can go to **WAMP icon** and **LEFT-CLICK** on the icon.

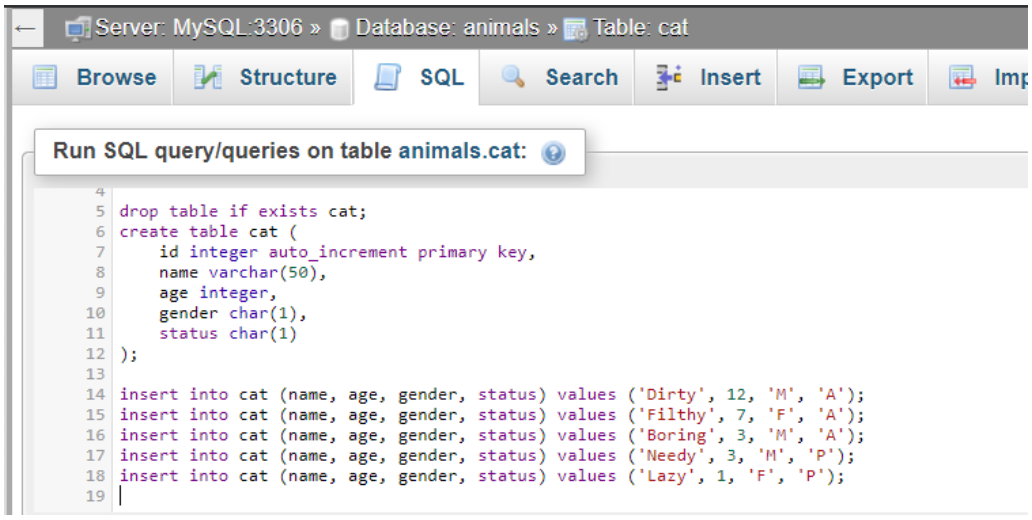


   b. AND click on **phpMyAdmin**. It will pop up a web browser tab/window with the same URL **http://localhost/phpmyadmin**.
   c. If you are using MAMP or something else other than WAMP (or if you have changed WAMP's configuration), then the URL will be different.

2. Sign in with:
   a. Username: **root**
   b. Password: <leave it empty> or something else depending on your setting

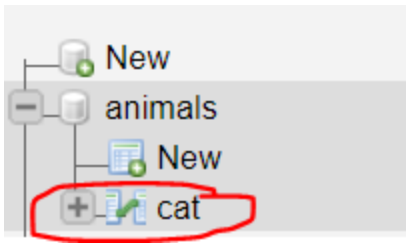
3. Click on **SQL** link at the top menu bar:

4. Open the file `create.sql`. Layfoo provided his MySQL database 'animals' **dump** in this SQL file. Copy the entire content and paste it into **PHPMyAdmin** SQL pane.



5. Click on **Go** button. The queries will be executed.

6. On the **LEFT SIDE** of the window, expand **animals** database and you should see a new table **cat**.



7. Click on **cat** table. You should see FIVE (5) rows of data inside:

	id	name	age	gender	status
Edit Copy Delete	1	Dirty	12	M	A
Edit Copy Delete	2	Filthy	7	F	A
Edit Copy Delete	3	Boring	3	M	A
Edit Copy Delete	4	Needy	3	M	P
Edit Copy Delete	5	Lazy	1	F	P

Our **Database** is now ready!

NOW, we have to find a way to CONNECT to it.

Part B (Connecting to Database)

Open **`CatDAO.php`**. This new **`CatDAO`** is very similar to the previous **`CatDAO`**. Both make available these two **public methods** for OTHER PHP files to call:

- `getCats()`
- `getCatsByStatus($status)`

Both methods return exactly the same things (Indexed Array of Cat objects). Hence, the same **`display.php`** file should work without any coding changes.

But there ARE some changes. Let's have a look.

Connecting to Database

`CatDAO.php` requires another file **`ConnectionManager.php`**.

`ConnectionManager.php`

```
<?php

class ConnectionManager {

    public function connect() {
        $servername = 'localhost';
        $username = 'root';
        $password = '';
        $dbname = 'animals';

        // Create connection
        $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); // if fail,
exception will be thrown

        // Return connection object
        return $conn;
    }
}
```

`ConnectionManager.php` handles **Database Connection** via PHP's **PDO** (PHP Data Object) **extension**.
→ You do NOT need to know the details of **PDO** implementation.
→ You DO need to know how to **configure** the following in **`ConnectionManager.php`**:

- Server/Host name (e.g. **localhost**)
- Username: **Your DB username** ("root" by default)
- Password: **Your DB password** (empty by default)
- DB (instance) name (e.g. In our exercise, it is **animals**)

→ You do NOT need to memorize the code. But you are expected to know how to configure the above.

To use this from another PHP file, we must do the following:

- Import **`ConnectionManager.php`** file (e.g. **require_once**).
- Create a new **`ConnectionManager`** object.
- Off that object, call the public method **`connect()`**.
- This **`connect()`** method will make a connection to the specified Database and **return a PDO object** **`$conn`**.
- We can then use this object **`$conn`** to perform SQL operations (SELECT, INSERT, UPDATE, etc.).

Part C (Retrieving Data from Database)

Let's revisit **`CatDAO.php`**. It provides two **public methods** for OTHER PHP files to call:

- `getCats()`
- `getCatsByStatus($status)`

The same **`display.php`** file from **Question 2** should work without any modification.

Let's have a look at `getCats()`. It consists of SIX (6) steps. In Step 6, it returns an Indexed Array of Cat objects (if any matching rows were found in the database table **cat**).

CatDAO.php | Method getCats()

```
public function getCats() {

        // STEP 1
        // Connect to database 'animals'
        // See 'ConnectionManager.php'
        $connMgr = new ConnectionManager();
        $conn = $connMgr->connect();

        // STEP 2
        // Prepare SQL statement
        $sql = "SELECT name, age, gender, status FROM cat";
        $stmt = $conn->prepare($sql);

        // STEP 3
        // Run SQL
        $stmt->execute();
        $stmt->setFetchMode(PDO::FETCH_ASSOC);
        // Retrieve each row as an Associative Array

        // STEP 4
        // Retrieve query results - ONE ROW AT A TIME
        $cats = [];
        // Initialize an empty (indexed) Array
        // so I can return it to whoever called this function
        // Use WHILE loop to loop through
        while ($row = $stmt->fetch() ) {
            $cat = new Cat(
                                $row['name'],
                                $row['age'],
                                $row['gender'],
                                $row['status']
                            );
            $cats[] = $cat;
        }

        // STEP 5
        // Close DB Connection & SQL Statement
        $stmt = null;
        $conn = null;

        // STEP 6
        // YAY! We're ready to return the array!
        return $cats;
```

```
}
```

Let's have a look at `getCatsByStatus($status)`. It consists of SIX (6) steps. In Step 6, it returns an Indexed Array of Cat objects **whose 'status'** is equal to the **parameter `$status`**.

CatDAO.php | Method getCatsByStatus($status)

```
public function getCatsByStatus($status) {
    // $status == 'A' or 'P'

    // STEP 1
    // Connect to database 'animals'
    // See 'ConnectionManager.php'
    $connMgr = new ConnectionManager();
    $conn = $connMgr->connect();

    // STEP 2
    // Prepare SQL statement
    $sql = "SELECT name, age, gender, status
                FROM cat
                WHERE status = :status ";
    $stmt = $conn->prepare($sql);
    // Parameter binding
    $stmt->bindParam(':status', $status, PDO::PARAM_STR);
    // It binds the value of parameter $status to :status in the SQL statement.

    // STEP 3
    // Run SQL
    $stmt->execute();
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    // Retrieve each row as an Associative Array

    // STEP 4
    // Retrieve query results - ONE ROW AT A TIME
    $cats = [];
    // Initialize an empty (indexed) Array
    // so I can return it to whoever called this function
    // Use WHILE loop to loop through
    while ($row = $stmt->fetch() ) {
        $cat = new Cat(
                $row['name'],
                $row['age'],
                $row['gender'],
                $row['status']
            );
        $cats[] = $cat;
    }

    // STEP 5
    // Close DB Connection & SQL Statement
    $stmt = null;
    $conn = null;

    // STEP 6
    // YAY! We're ready to return the array!
    return $cats;
}
```

Part D (Display Cat Information)

In web browser, open **`display.php`** (you should have copied this file from **cat2** folder into **cat3** folder).

- You do NOT need to make any changes to **`display.php`**.
- The details of **`CatDAO.php`** is well-hidden from **`display.php`**.
- **`display.php`** does NOT need to know that **`CatDAO`** now retrieves data from **MySQL Database**.
- All **`display.php`** needs to know is … what are the public **methods** it can call - in order to retrieve needed data.
- **`CatDAO`** provides two **public methods** that **`display.php`** can call:
  - `getCats()`
  - `getCatsByStatus($status)`

When the page loads in a web browser **for the first time**, it must display display all cats.

`display.php`

Our Cats

Name	Age	Gender	Status
Dirty	12	M	Available
Filthy	7	F	Available
Boring	3	M	Available
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption

Filter by Status:

Available ▾

Filter

Next, when the user selects **Available** as the filtering value, and clicks on **Filter SUBMIT button**, the page displays:

`display.php`

Our Cats

Name	Age	Gender	Status
Dirty	12	M	Available
Filthy	7	F	Available
Boring	3	M	Available

Filter by Status:
Available ▾

Filter

NOTE: The page must remember and pre-select the user's form input **"Filter by Status"**. For instance, in the above example, "**Available**" option is pre-selected in the drop-down list.

Next, when the user selects **Pending Adoption** as the filtering value, and clicks on **Filter SUBMIT button**, the page displays:

`display.php`

Our Cats

Name	Age	Gender	Status
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption

Filter by Status:
Pending Adoption ▾

Filter

NOTE: The page must remember and pre-select the user's form input **"Filter by Status"**. For instance, in the above example, "**Pending Adoption**" option is pre-selected in the drop-down list.