WAD I Lab Test 2 (2 hours) [30 marks]

General Instructions:

- You can refer to any offline resources already on your laptop, but you must disable all networking and Bluetooth connections during the test. You must not communicate with anyone via any means during the test.
- Just before the test, you will be given instructions by the invigilator as to how to obtain resource files required for the lab test and how to submit your solutions.
- **No questions** will be entertained during the test. If necessary, **make your own assumptions** and **document them as comments** in the submitted HTML or PHP files**.**
- You are allowed to use only standard PHP classes and functions in your solutions – do not use any third-party libraries.
- Use meaningful names for classes, methods, functions and variables, as well as indent your code correctly. Otherwise, you may attract a penalty of up to **20%** of your score for the corresponding question.
- You **MUST** include your name as author in the comments of all your submitted source files. Failure to do so WILL attract a penalty of up to **20%** of your score for the corresponding question.

  For example, if your registered name is "FAN Bing Bing" and email ID is fan.bingbing.2020, include the following comment at the beginning of each source file you write.

```
<!--
      Name:  FAN Bing Bing
      Email: fan.bingbing.2020
-->
```

- You may wish to comment out the parts in your code which cause errors. But commented code will not be marked.
- Unless otherwise stated, you can assume that user inputs are in the correct format.
- This lab test has **3 questions** (**14 pages**): Question 1 and 3 of this lab test do not involve a database. **Question 2 involves a database and you need to import a given SQL file.**
- Instructions are given for WAMP users considering default setting – the setting that we support for this course. If you are using MAMP, you may need to make necessary changes by yourself (e.g., modifying connection information in `ConnectionManager.php`).

DO NOT TURN OVER UNTIL YOU ARE TOLD TO DO SO

Question 1: Food Court (Difficulty Level: */*/*) **[9 marks]**

Given:

```
● q1/
   o classes/
      ▪ Member.php          (TO BE COMPLETED)
      ▪ Sale.php            (TO BE COMPLETED)
      ▪ SaleLineItem.php    (DO NOT MODIFY THE CODE)
      ▪ Stall.php           (DO NOT MODIFY THE CODE)
      ▪ StallDAO.php        (DO NOT MODIFY THE CODE)
   o common.php             (DO NOT MODIFY THE CODE)
   o test1.php              (DO NOT MODIFY THE CODE)
   o test2.php              (DO NOT MODIFY THE CODE)
   o view.php               (TO BE COMPLETED)
```

Part A (3 marks) – Difficulty Level (*)

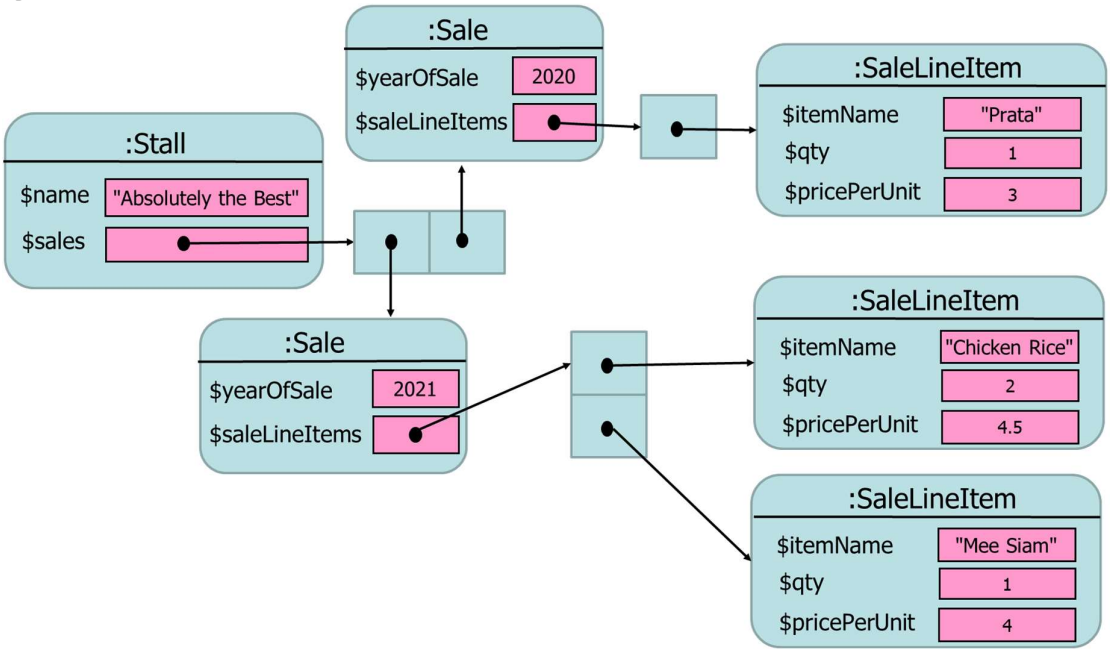
Complete **`view.php`** such that it displays the following webpage:

Stall Name	Count of Sales
Absolutely the Best	2
Best Food	1
Come! Get Good Food!	2

Figure 1-1: Expected output of **`view.php`**

Instructions:

1. Obtain an indexed array (i.e., list) of **`Stall`** objects from **`StallDAO`** by calling **`getStalls()`**. The following is the memory state diagram describing <u>first</u> **`Stall`** object in the indexed array returned by **`getStalls()`**:



2. Process the indexed array of **`Stall`** objects to display the output shown in **Figure 1-1**.

Part B (3 marks) – Difficulty Level (*)

Complete **`Sale.php`**

Instructions:

1. Complete function **`getDollarsReceived()`**:

    a. Iterate through all the **`SaleLineItem`** objects in attribute **`$saleLineItems`**
    b. For each **`SaleLineItem`** object, compute the amount of dollars that is received based on the quantity sold and the price per unit.
    c. Return the sum of dollars received over all **`SaleLineItem`** objects

2. Use **`test1.php`** to check. It displays the following expected output:

```
Correct answer: 13

Your answer: 13
```

Figure 1-2: Expected output of **`test1.php`**

Part C (3 marks) – Difficulty Level (*)

Complete **`Member.php`** following the class diagram below:

Member
- $name
+ __construct($name) + getName() + setName($name)

Use **`test2.php`** to check. It displays the following expected output:

```
Andy
Bob
```

Figure 1-3: Expected output of **`test2.php`**

Question 2: Lunar Air-Conditioning (Difficult level: **//**)** **[15 marks]**

Given:

```
• q2/
  o CreateDB.sql                    (DO NOT MODIFY THE CODE)
  o common.php                      (DO NOT MODIFY THE CODE)
  o classes/
      ▪ ConnectionManager.php       (DO NOT MODIFY THE CODE)
      ▪ User.php                    (DO NOT MODIFY THE CODE)
      ▪ Aircon.php                  (DO NOT MODIFY THE CODE)
      ▪ Request.php                 (DO NOT MODIFY THE CODE)
      ▪ UserDAO.php                 (DO NOT MODIFY THE CODE)
      ▪ AirconDAO.php               (TO BE COMPLETED)
      ▪ RequestDAO.php              (TO BE COMPLETED)
  o images/
      ▪ lunar.jpg
      ▪ user1.png
      ▪ user2.png
      ▪ manager1.png
  o login.php                       (TO BE COMPLETED)
  o client_view.php                 (TO BE COMPLETED)
  o manager_view.php                (TO BE COMPLETED)
  o update_request_status.php       (TO BE COMPLETED)
```

Overview: We are building a web application for Lunar Air-Conditioning Services Pte. Ltd. to manage their customers and services. The application allows a customer/client to log into the system to view his/her aircon conditions and makes service requests (e.g., maintaining or repairing), and it allows a manager to view and accept or reject requests. Your tasks are to complete the PHP files such that the clients and managers can view aircon/request information and make operations.

In **`ConnectionManager.php`**, verify that the username, password, port number are correct.

You will have three data tables after importing **`CreateDB.sql`**

- **`accounts`**: user information
  - **`username`**
  - **`password`**
  - **`fullname`**
  - **`role:`** role of the user (*client* or *manager*)
- **`aircons`**: details of aircons
  - **`aircon_id`**: ID of the aircon
  - **`username`**: username of aircon user
  - **`name:`** name of the aircon, often refers to the name of the room the aircon is installed in
  - **`location:`** address of the aircon
  - **`last_request_date:`** date of the last service request
  - **`last_request_status:`** status of the last request of the aircon (*completed, pending, rejected, or accepted*)
- **`requests`**: requests from users sent to managers
  - **`id`**: request ID
  - **`aircon_id`**: corresponding aircon ID
  - **`request_date`**: date of the request

- **`status:`** status of the request (*completed, pending, rejected, or accepted*)

Figure 2-1 below visualizes how the three data tables are connected:

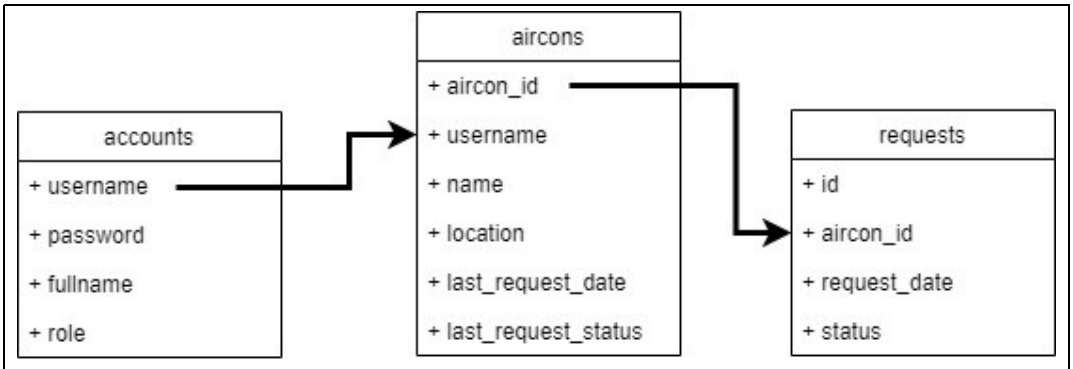


Figure 2-1: Connections between the three data tables.

You are given three users (two clients and one manager). For the sake of simplicity, the passwords are the same as the usernames, and **they are stored in the database as plaintext (<u>they are NOT hashed</u>)**. You are also provided users' photos whose names are the same as the usernames. The details of the three users are given below.

Username	Password	Full name	Role	Images
`user1`	`user1`	`Barack Obama`	`client`	`user1.png`
`user2`	`user2`	`Donald Trump`	`client`	`user2.png`
`manager1`	`manager1`	`Bill Gate`	`manager`	`manager1.png`

The flow of the web application is given in **Figure 2-2** below.

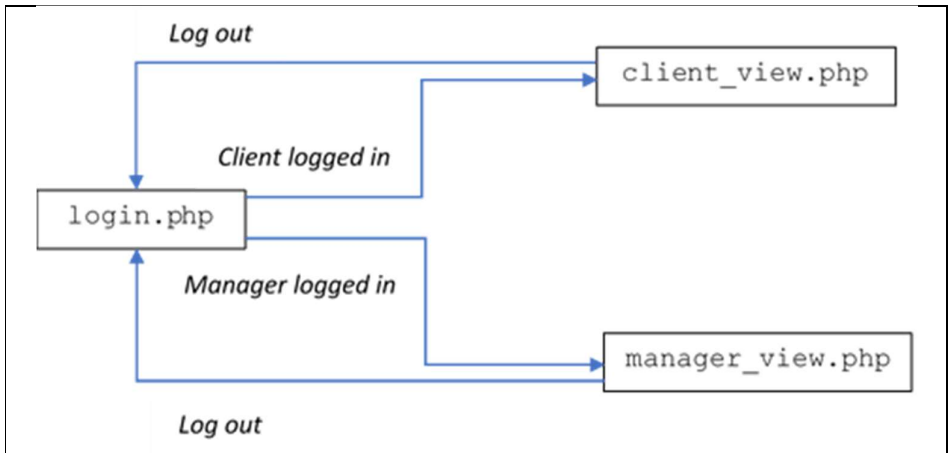


Figure 2-2: Flow of the web application.

Part A (5 marks) – Difficulty Level (**)

A.1. Complete **`login.php`** allowing the clients and managers to login and view their aircons. In this module, the user inputs his/her **`username`** and **`password`** to log into the system. **Figure 2-3** shows the webpage when the entered **`username`** does not exist in the database, and **Figure 2-4** shows the webpage when the **`username`** exists but the **`password`** is not valid. You can use **`get($username)`** from **`UserDAO.php`** to retrieve the user information.

You may assume that the user always enters some non-whitespace characters in the Username input field.

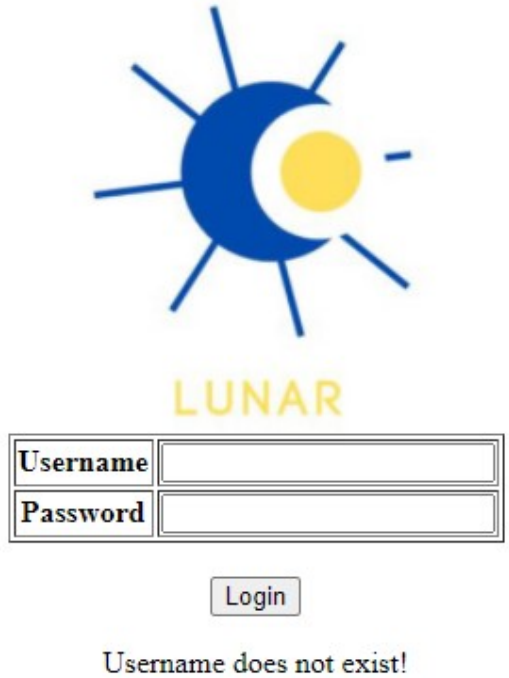


Figure 2-3: The **`username`** does not exist.

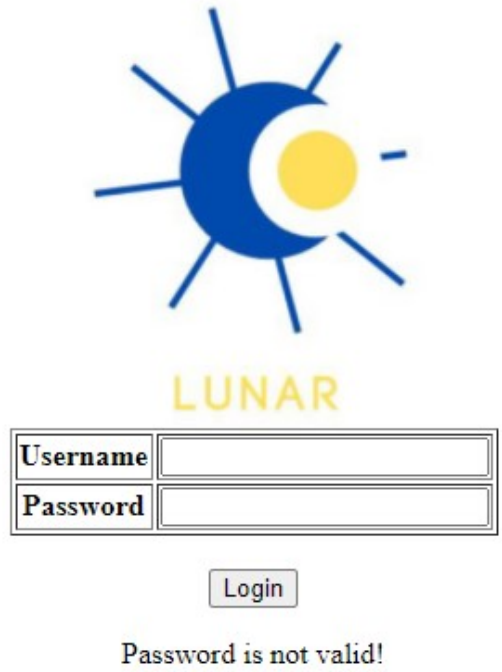


Figure 2-4: The **`username`** exists, but the **`password`** is not valid.

If the **`username`** and **`password`** combination is correct, then:

- If the user role is **`client`**, redirect the user to **`client_view.php`**
- If the user role is **`manager`**, redirect the user to **`manager_view.php`**

A.2. Modify **`client_view.php`** and **`manager_view.php`**: If the user has not logged in but tries to access **`client_view.php`** or **`manager_view.php`**, then redirect the user to **`login.php`**. If a client logged in but tries to access **`manager_view.php`**, then redirect to **`client_view.php`**. Otherwise, if a manager logged in but tries to access **`client_view.php`**, then redirect to **`manager_view.php`**.

Hint:

- Use to put things at the center of the page
- You can put an object inside a session variable

Part B (4 marks) – Difficulty Level (**)

Complete **`client_view.php`** such that a client can view his/her aircons.

If the user is a **`client`**, the webpage displays the user photo, full name, and a table showing the details of the aircons under the corresponding **`username`**, as shown in **Figure 2-5** below. The user's photo has the same name as the username and ends with "`.png`" extension and is located in Folder "`q2/images`". You may need to use function **`getAll()`** in **`AirconDAO.php`** to get necessary information to display the table.

Welcome, Barack Obama

Logout

Your aircons

ID	Name	Location	Date of Last Request	Status of Last Request
u1_aircon1	Bed Room 1	Clementi Ave 2, Blk351	2020-08-09	rejected
u1_aircon2	Bed Room 2	Clementi Ave 2, Blk351	2020-05-11	pending
u1_aircon3	Bed Room 3	Clementi Ave 2, Blk351	2020-03-04	accepted
u1_aircon4	Living Room 1	Clementi Ave 2, Blk351	2020-01-10	completed

Figure 2-5: Details of the aircons of **`user1`**.
(This is the client's view after logging in)

If the user clicks on Button **`Logout`**, then redirect the user to **`login.php`**. If the user has already logged out, he/she will need to log in again to be able to view **`client_view.php`**.

Part C (6 marks) – Difficulty Level (**)

Complete **`manager_view.php`** such that the manager can view the requests.

- If the user is a manager, display user's photo, full name, and a table showing the details of the requests as shown in **Figure 2-6**.
- If the user clicks on Button **`Logout`**, then redirect the user to **`login.php`**. If the user has already logged out, he/she will need to log in again to be able to view **`manager_view.php.`**
- For each **request** whose **status** is **"pending"**, the last column of the table row (**"Action"**) is a **hyperlink** to **`update_request_status.php`**.
  - **`update_request_status.php`** expects a **parameter** called **id**, which corresponds to the request id, e.g., **`update_request_status.php?id=1`**.
  - Complete **`update_request_status.php`** such that clicking on the **hyperlink** results in the execution of the following THREE (3) tasks:
    1. Use **`RequestDAO`** and call its **`updateStatus($id, $status)`** method to update the corresponding request's status from **"pending"** to **"accepted"**.
       - Complete the implementation of **`updateStatus($id, $status)`** method.
    2. Use **`AirconDAO`** and call its **`updateLastRequestStatus($aircon_id, $status)`** method to update the corresponding aircon's status from **"pending"** to **"accepted"**.
       - Complete the implementation of **`updateLastRequestStatus($aircon_id, $status)`** method.
    3. Upon completing Tasks #1 and #2 above, forward the user back to **`manager_view.php`**.

Welcome, Bill Gate

Logout

Service requests

ID	Aircon ID	Location	Request Date	Status	Action
1	u1_aircon2	Clementi Ave 2, Blk351	2020-05-11	pending	Accept this request
2	u1_aircon3	Clementi Ave 2, Blk351	2020-03-04	accepted	
3	u2_aircon2	29 International Business Park Rd	2020-05-11	pending	Accept this request
4	u2_aircon3	29 International Business Park Rd	2020-03-04	accepted	

Figure 2-6: Details of the service requests viewed by the manager.
*(This is the manager's view when the database is newly loaded (from **`createDB.sql`**)*

See next page for the manager's view AFTER clicking on "Accept this request" (hyperlink) for the request with ID = 1.

Welcome, Bill Gate

Logout

Service requests

ID	Aircon ID	Location	Request Date	Status	Action
1	u1_aircon2	Clementi Ave 2, Blk351	2020-05-11	accepted	
2	u1_aircon3	Clementi Ave 2, Blk351	2020-03-04	accepted	
3	u2_aircon2	29 International Business Park Rd	2020-05-11	pending	Accept this request
4	u2_aircon3	29 International Business Park Rd	2020-03-04	accepted	

Figure 2-7: Details of the service requests viewed by the manager.
(This is the manager's view AFTER the manager clicked on "Accept this request" hyperlink for request with ID = 1)

Subsequently, the manager clicks on "Accept this request" hyperlink for the request with ID = 3.

Welcome, Bill Gate

Logout

Service requests

ID	Aircon ID	Location	Request Date	Status	Action
1	u1_aircon2	Clementi Ave 2, Blk351	2020-05-11	accepted	
2	u1_aircon3	Clementi Ave 2, Blk351	2020-03-04	accepted	
3	u2_aircon2	29 International Business Park Rd	2020-05-11	accepted	
4	u2_aircon3	29 International Business Park Rd	2020-03-04	accepted	

Figure 2-8: Details of the service requests viewed by the manager.
(This is the manager's view AFTER the manager clicked on "Accept this request" hyperlink for request with ID = 3)

Question 3: Study Groups (Difficulty Level */***)** **[6 marks]**

Given:

- q3/
  - classes/
    - BusyTime.php (DO NOT MODIFY THE CODE)
    - Student.php (DO NOT MODIFY THE CODE)
    - StudentDAO.php (DO NOT MODIFY THE CODE)
    - StudyGroup.php (DO NOT MODIFY THE CODE)
  - images/
    - check1.png
    - check2.png
  - common.php (DO NOT MODIFY THE CODE)
  - show_availability.php (TO BE COMPLETED)
  - show_groups.php (TO BE COMPLETED)
  - start.html (DO NOT MODIFY THE CODE)
  - test_show_availability.php (DO NOT MODIFY THE CODE)
  - test_show_groups.php (DO NOT MODIFY THE CODE)

Overview: We build a web application that puts students in a course into study groups based on some constraints.

User Navigation Flow: `start.html -> show_groups.php -> show_availability.php`

In **`start.html`**, we specify some constraints on the study groups that are to be created.

In **Figure 3-1**, we specify that we want to divide students in a course into study groups of sizes 2 to 3, where at least one student in a group has a GPA of 3.5 or above.

Create Study Groups

Minimum group size: 2

Maximum group size: 3

There is at least one student in the group with GPA of at least: 3.5

Get Groups

Figure 3-1: `start.html`

When the "Get Groups" submit button of **`start.html`** is clicked, the web application will create and display the created study groups in **`show_groups.php`**.

In **Figure 3-2**, 50 students are put into 17 study groups: G1 to G17. For each group, we have the names of the students who are assigned to it. For example, for G15 we have Julia Q., Julia J., and Sarah H. Each student is assigned to 1 study group.

Note: You can assume that all students have unique names.

[Complete this in Part A]

G1	G2	G3	G4	G5
Dan B. Chris X. Dan T.	Bob B. Sarah U. Bob L.	Chris T. Ann A. Dan D.	Chris E. Bob O. Ann L.	Sarah J. Emily I. Dan C.
G6	**G7**	**G8**	**G9**	**G10**
Emily X. Chris D. James O.	James M. Ann S. James B.	James S. Emily L. Julia G.	Jill Q. James G. Bob E.	Jill B. Bob N. Chris A.
G11	**G12**	**G13**	**G14**	**G15**
Kane S. Jill Y. Ann K.	Ann Q. Bob H. James K.	Emily G. James A. Sarah G.	Jill J. Ann X. Ann R.	Julia Q. Julia J. Sarah H.
G16	**G17**			
Chris G. Jill D. Jill U.	James L. Bob J.			

Figure 3-2: `show_groups.php`

When one of the hyperlinks (e.g., G15) in **`show_groups.php`** is clicked, **`show_availability.php`** shows common time slots of students in the corresponding study group. For simplicity, a student has the same time constraints each week, and we only consider Mon-Fri.

In **Figure 3-3**, we show what happened when the G15 hyperlink is clicked:

- Names of **unavailable** students are shown; e.g., Julia J. is unavailable from 9-11 am on Monday.
- Common available time slots are indicated by check marks:
  - ✅ indicates the common slots with the longest continuous time (i.e., the Tue slot in **Figure 3-3** that spans 5 hours)
  - ✔ indicates the other common slots.

[Complete this in Part B]

	Mon	Tue	Wed	Thu	Fri
9-10	Julia J.		Julia J.	Julia Q. Julia J. Sarah H.	Julia Q.
10-11					
11-12	✔	✅			
12-13					
13-14	Julia Q.		Julia Q. Sarah H.	Julia Q.	✔
14-15		Julia Q.			
15-16		Julia Q. Julia J.			
16-17		Sarah H.			

Figure 3-3: `show_availability.php`

Part A (2 marks) – Difficulty Level (***)

Complete **`create_study_groups($students, $min_size, $max_size, $min_gpa)`** in **`show_groups.php`**. This function takes as parameters an indexed array of `Student` objects (`$students`) and 3 constraints (`$min_size`, `$max_size`, `$min_gpa`). It returns an indexed array of `StudyGroup` objects (or `null`).

Instructions:

1. Put the **`Student`** objects in **`$students`** into groups based on the **`$min_size`**, **`$max_size`**, and **`$min_gpa`** constraints.
   a. Each group should be put inside a **`StudyGroup`** object.
   b. Each group must contain at least **`$min_size`** members and at most **`$max_size`** members.
   c. Each group must contain at least one member whose GPA is at least **`$min_gpa`**.
   d. Each student must be put into one and only one **`StudyGroup`** object.
   e. You can assume that **`$min_size`** and **`$max_size`** are positive integers, and **`$min_gpa`** is a float (≥0.0 and ≤4.0). Remember, PHP performs automatic type conversion.
   f. There could be 0, 1, or more possible groupings of students that satisfy the 3 constraints -- **`$min_size`**, **`$max_size`**, and **`$min_gpa`**.

2. If the constraints cannot be met (i.e., there is 0 possible grouping that satisfies the 3 constraints) return **`null`**, otherwise return an indexed array of **`StudyGroup`** object(s).

3. If there are **multiple** possible groupings of students (i.e., the students can be put into multiple study group arrangements, each satisfying the 3 constraints), return any **one** of them.

4. Use **`test_show_groups.php`** to test your code. The expected output is shown in **Figure 3-4**.

TC1:

Status: Pass

TC2:

Status: Pass

TC3:

Status: Pass

Figure 3-4: Expected output of **`test_show_groups.php`**

Note:

1. After inputting the constraints shown in **Figure 3-1** to **`start.html`**, the table shown in your **`show_groups.php`** may be different from the one shown in **Figure 3-2**. This can be the case even when your function is correct as there can be **multiple** possible groupings of students that match the 3 constraints. And for such cases, your function only returns **one** of them.
2. We will use a separate test script containing different test cases to grade your submission.

Part B (4 marks) – Difficulty Level (***)

Complete **`display_timetable($students)`** in **`show_availability.php`**. It `echo`-es a HTML table (e.g., see **Figure 3-3**) that shows common time slots of `Student` objects in the input indexed array `$students`.

Instructions:

1. For each of the 8 x 5 time blocks – 8 time blocks of 1 hour each in a day; 5 days a week – identify the **`Student`** objects in **`$students`** who are unavailable.
    a. Use **`getTimetable()`** method of **`Student`** to get an indexed array of **`BusyTime`** objects
    b. Each **`BusyTime`** object specifies the day and time for which a student is unavailable

2. Display the results in a table (see **Figure 3-3**):
    a. Show the students who are unavailable in each cell.
    b. Merge *consecutive* cells within a single day that have the same set of unavailable students. For example, in **Figure 3-3**, for Monday, we merge 9-10 and 10-11 cells, as the unavailable student is the same (i.e., Julia J.).
    c. For cells for which all students are available, mark them with "check" marks.
        i. Use **`check1.png`** (✅) to indicate the longest consecutive time blocks for which all in **`$students`** are available. Set its width to be 50 px.
        ii. Use **`check2.png`** (✔) to indicate other time blocks for which all students are available. Set its width to be 50 px.
        iii. Following b. merge consecutive cells within a single day for which all students are available.

3. Use **`test_show_availability.php`** to test your code. The expected output is shown in **Figure 3-5.**

Note:

1. We will use a separate test script containing different test cases to grade your submission.
2. Do not hard code the expected output shown in **Figure 3-5**.

TC1:

	Mon	Tue	Wed	Thu	Fri
9-10	✓	Jill U.	Jill U.	✓	Bob H. Jill U.
10-11					
11-12	Jill U.				
12-13					
13-14	✓	Bob H.	✓	Jill U.	✓
14-15		Jill U.			
15-16					
16-17					

TC2:

	Mon	Tue	Wed	Thu	Fri
9-10	✓	✓	✓	James S. Bob J.	James S. Bob N.
10-11					
11-12					
12-13					
13-14	James S.		Bob N.	James S. Bob N.	✓
14-15					
15-16		Bob N.			
16-17		James S.			

TC3:

	Mon	Tue	Wed	Thu	Fri
9-10	Chris G. Jill D.	Chris G. Jill D. Jill U.	Jill U.	Chris G. Jill D.	Jill D. Jill U.
10-11					
11-12	Jill U.				
12-13					
13-14	Chris G.	Jill D.	Chris G.	Chris G. Jill D. Jill U.	✓
14-15		Jill U.			
15-16		Chris G. Jill U.			
16-17					

Figure 3-5: Expected output of **`test_show_availability.php`**

- END -