IS113 - Web Application Development

Week 12

Sections G4/G5

K. J. Shim

Agenda Today

- Passing control across pages
- Passing data across pages
- Authenticating users

Go to your webroot

- Go to eLearn -> Content -> Session 12 -> In Class -> **week12.zip**
- Unzip **week12.zip** such that you have the following directory structure locally:

```
webroot  -
         |
         |-- is113
         |         |
         |         |-- week12
         |         |         |
         |         |         |-- control
         |         |         |        |-- first.php
         |         |         |        |-- second.php
         |         |         |-- data
         |         |         |        |-- page1.php
         |         |         |        |-- page2.php
         |         |         |-- session
         |         |         |        |-- response.php
         |         |         |        |-- evil.php
         |         |         |        |-- good.php
```

Continued…

```
...
        |-- is113
        |        |
        |        |-- week12
        |        |        |
        |        |        |-- school
        |        |        |        |-- database
        |        |        |        |      |-- create.sql
        |        |        |        |-- include
        |        |        |        |      |-- AccountDAO.php
        |        |        |        |      |-- ConnectionManager.php
        |        |        |        |      |-- Grade.php
        |        |        |        |      |-- GradeDAO.php
        |        |        |        |-- login.php
        |        |        |        |-- process_login.php
        |        |        |        |-- logout.php
        |        |        |        |-- home.php
        |        |        |        |-- testAccountDAO.php
        |        |        |        |-- testGradeDAO.php
```

Continued…

```
...
        |-- is113
        |       |
        |       |-- week12
        |       |       |
        |       |       |-- labtest
        |       |       |       |-- database
        |       |       |       |       |-- create.sql
        |       |       |       |-- include
        |       |       |       |       |-- ConnectionManager.php
        |       |       |       |       |-- Grade.php
        |       |       |       |       |-- GradeDAO.php
        |       |       |       |-- Congrats.php
        |       |       |       |-- search.php
        |       |       |       |-- SeeYouAgain.php
        |       |       |       |-- testGradeDAO.php
        |       |       |       |-- verify.php
        |       |       |
        |       |       |-- authenticate
        |       |       |       |-- password.php
        |       |       |
        |       |       |-- fanclub
        |       |       |       |-- ...
```

Exercise 1 control

Passing Control Across Pages

- Method 1: Form Submission

```
<html>
  <body>
    <form action='second.php'>
            <input type="submit“/>
      </form>
  </body>
</html>
```

first.php

Passing Control Across Pages

- Method 2: Hyperlink

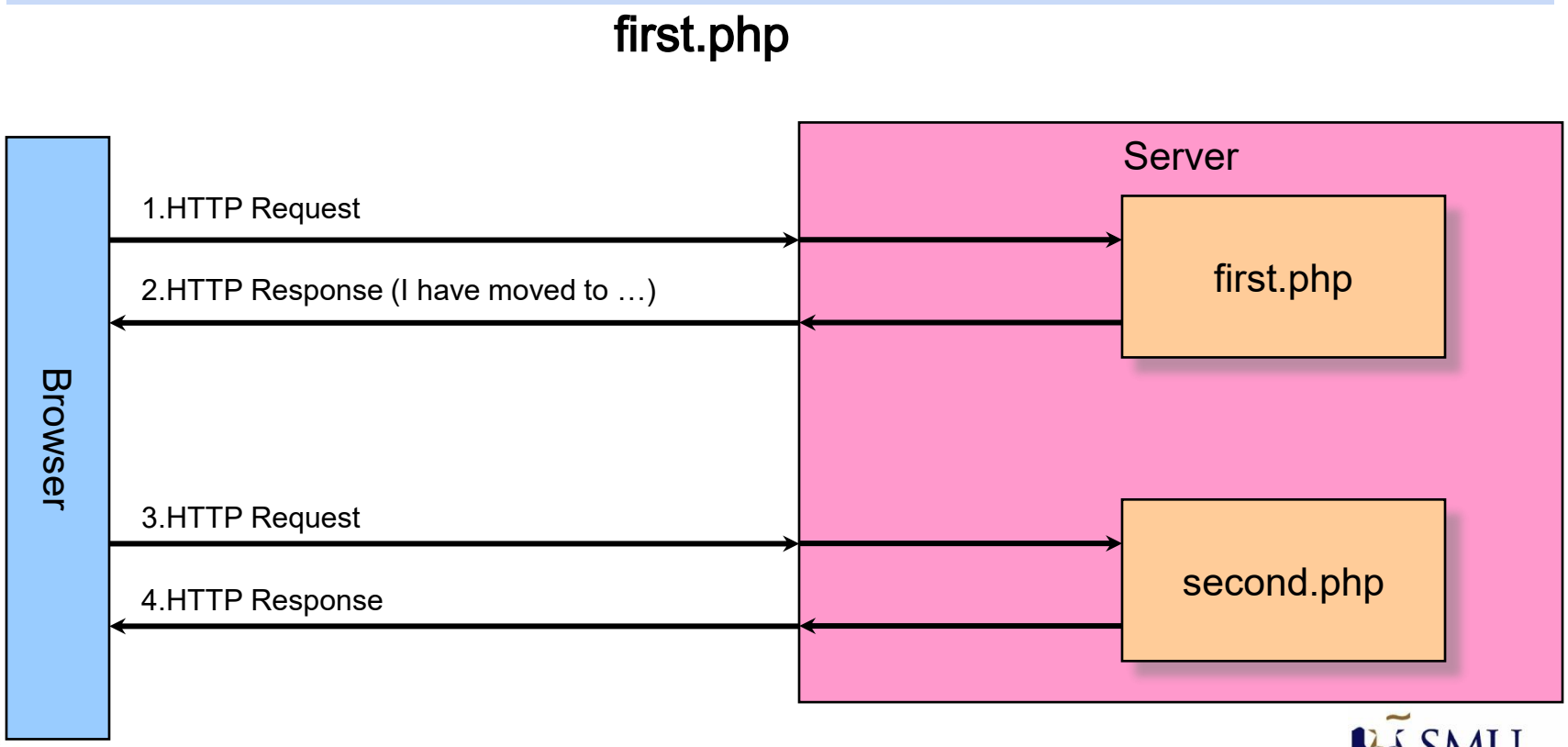
```
<html>
  <body>
    <a href='second.php'>Go to another page</a>
  </body>
</html>
```

first.php

Passing Control Across Pages

- Method 3: **Automatic** page redirection

```
header("Location: second.php");
exit;
```



Exercise 2 data

Passing Data Across Pages

- Method 1: Through form fields, including **hidden** fields

page1.php

Name: Bob
Age: 25

page2.php

Hidden Fields

```
<html><body>
   <form method="post" action="page2.php">
      Name: <input type="text" name="name">
            <input type="hidden" name="age" value="25">
      <input type="submit" value="Next">
   </form>
</body></html>
```

page1.php

```
<?php
   echo "Name: " . $_POST["name"];
   echo "<br>";
   echo "Age: " . $_POST["age"];
?>
```

page2.php

Passing Data Across Pages

- Method 2: Through URL

```
<html>
  <body>
    <a href="page2.php?name=Bob&age=25">
    View Details
    </a>
  </body>
</html>
```

page1.php

```
<?php
    echo “Name:$name<br>Age: $age”;
?>
```

page2.php

Passing Data Across Pages

- Method 3: Using **HTTP Session**
- What is **HTTP Session**?
  - Stores data shared between a user and a website (e.g. eLearn, iBanking, etc.)
  - Data available **across multiple pages**
  - Data will automatically be reset after a period of time

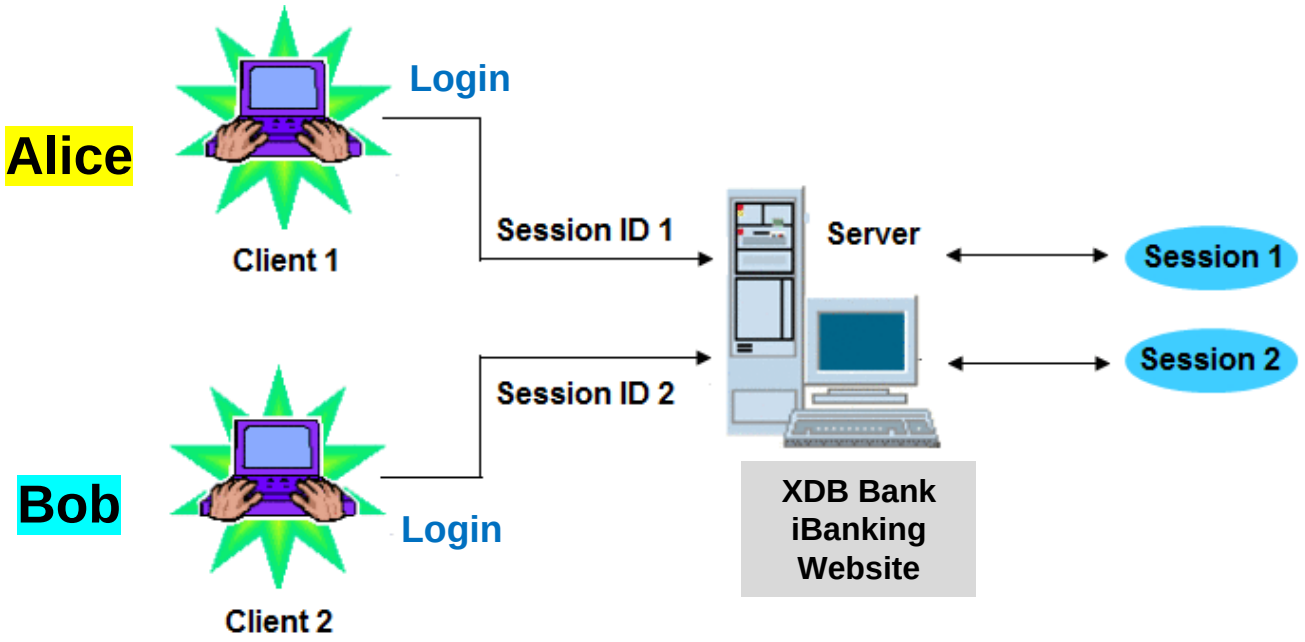
HTTP Session

- Why we need it?
  - Identify a user across multiple web pages
  - Pass data between web pages in the same site
- How to use HTTP Session?
  - Call session_start()
    - Initialize a session OR
    - Resume an existing session *(if a session already exists)*
  - Use **$_SESSION** superglobal to add new key-value pairs into an HTTP Session
- **Note:** Make sure session_start() is called before accessing $_SESSION superglobal variables

HTTP Session: Clearing Contents

- Session would be cleared automatically after some period of time has lapsed
- What can we do to clear it earlier?
- On **server side** (i.e. PHP file):
  - We can set $_SESSION to an empty array, or
  - We can use unset($_SESSION[<key>])
    - e.g., unset($_SESSION["count"])

HTTP Session – iBanking "Login"



- Each web user receives a **unique Session ID** from the iBanking **web server**.
- **Session ID** is a way for the **web server** to uniquely identify "Alice" apart from "Bob"… and other users.

Image Ref: https://www.sitesbay.com/php/php-session-in-php

HTTP Session – iBanking "Logout"

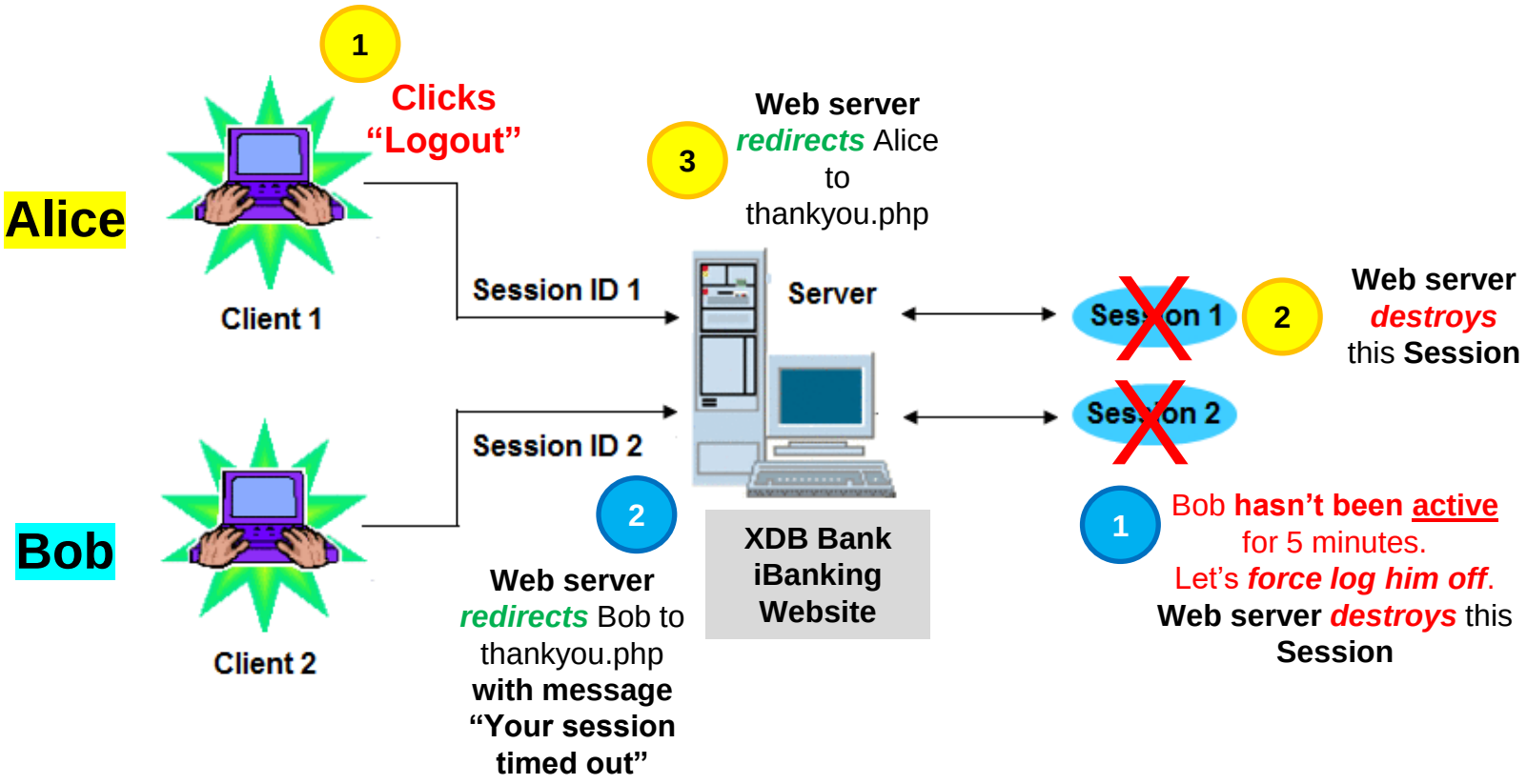


Image Ref: https://www.sitesbay.com/php/php-session-in-php

Session Management – PHP Functions

Session Function	What it does
`session_start()`	1) If there is NO session existing → **Create a new session** 2) If a session already exists → **Resume** the current session (continue to use it)
`session_id()`	Return the **Session ID** of the **current session**
`unset($_SESSION['a'])`	**Remove** the specified **Session Variable** (Example to the left) Remove a **Session Variable** whose **key** is **a**
`session_unset()`	**Remove** ALL **Session Variables**. You can also do the following (same effect): `$_SESSION = []`
`session_destroy()`	**Remove** (un-register) the current session. → Session ID is removed from the web server. → ***This does NOT unset Session Variables***

HTTP Session

page1.php

Name: Bob
Age: 25

page2.php

HTTP Session

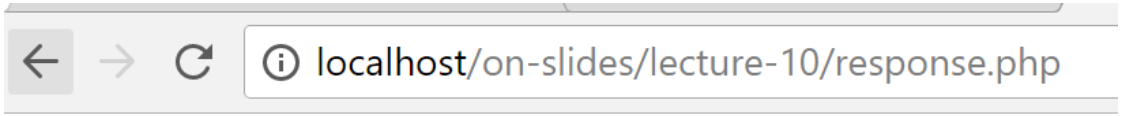
```
<?php
     session_start();
     $_SESSION['age'] = 25; // Create a new Session Variable
?>
<html><body>
   <form method="post" action="page2.php">
      Name: <input type="text" name="name"/>
      <input type="submit" value="Next"/>
   </form>
</body></html>
```

page1.php

```
<?php  session_start();  ?>
<html><body>
<?php
   echo "Name: " . $_POST["name"]; // Retrieved from Form Submission
   echo "<br>";
   echo "Age: " . $_SESSION["age"]; // // Retrieve from Session
?>
</body></html>
```

page2.php

HTTP Session: Another Example



You have accessed the page 1 times

response.php

You have accessed the page 2 times

response.php

HTTP Session: Another Example

```
<?php
session_start();
if(!isset($_SESSION["count"])){
    $_SESSION["count"] = 0;
}
$_SESSION["count"]++;
echo "You have accessed the page ".$_SESSION["count"]." times";
?>
```

response.php

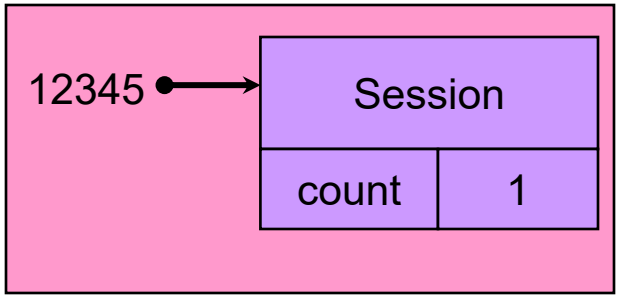
HTTP Session: Another Example

1. Client sends HTTP request

2. Server creates a session and generates a unique session id

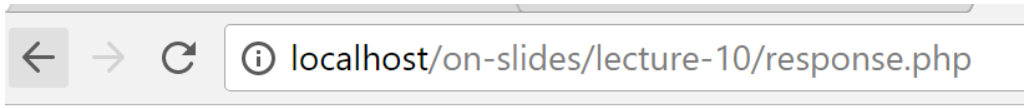
3. Session id is returned to the client

HTTP/1.1 200 OK
Connection: Keep-Alive
Date: Sun, 3 Jan 2018 11:02:15 GMT
Set-Cookie:PHPSESSID=12345;path=/



Client

Server



You have accessed the page 1 times

HTTP Session: Another Example

Client

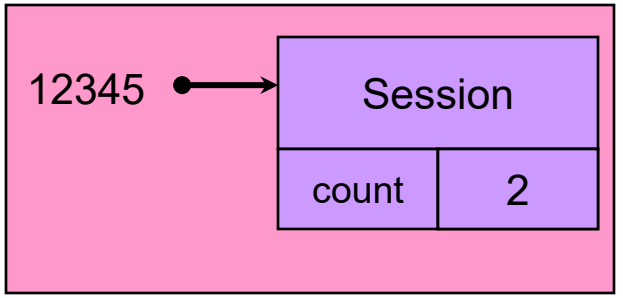
4. Client sends another HTTP request

GET / HTTP/1.1
Host: blue.smu.edu.sg
Cookie: PHPSESSID=12345

6. Server sends response.

5. Server retrieves the session and use value stored in the session during previous request.

6. Increment count.



Server

You have accessed the page 2 times

HTTP Session: Clearing Contents

(1)

You have launched 1 nuke bombs so far!

You have launched 2 nuke bombs so far!

You have launched 3 nuke bombs so far!

. . .

evil.php

```php
<?php

session_start();

if( !isset($_SESSION['nuke']) ) {
    $_SESSION['nuke'] = 0;
}

// Launch Nuke
$_SESSION['nuke']++;

echo "You have launched " . $_SESSION['nuke'] . " nuke bombs so far!";

?>
```

(2)

```php
<?php

session_start();

// Before unsetting Session Variable 'nuke'
var_dump($_SESSION);

// Remove Session Variable 'nuke'
unset($_SESSION['nuke']);

// After unsetting Session Variable 'nuke'
var_dump($_SESSION);
```

good.php

`unset($_SESSION['nuke'])` removes the "nuke" Session Variable created by **`evil.php`**

(3)

You have launched 1 nuke bombs so far!

You have launched 2 nuke bombs so far!

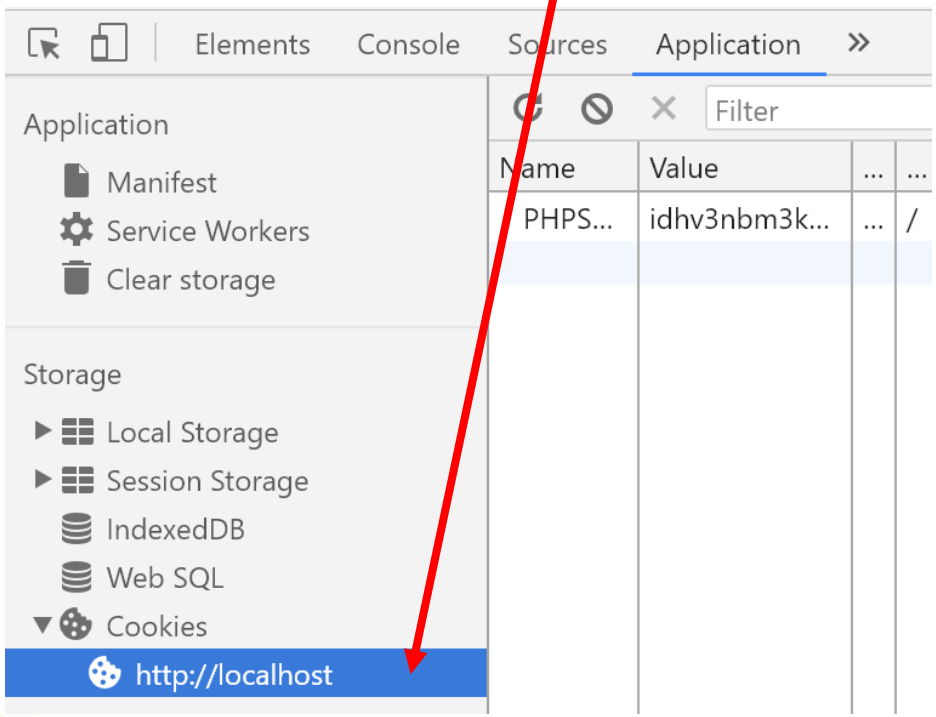
You have launched 3 nuke bombs so far!

. . .

evil.php

HTTP Session: Clearing Contents

- On **client side** (i.e., web browser):
  - Session id can be forced to be cleared using, e.g., Chrome Dev Tools (Ctrl+Shift+I)
  - Right click and press **Clear**

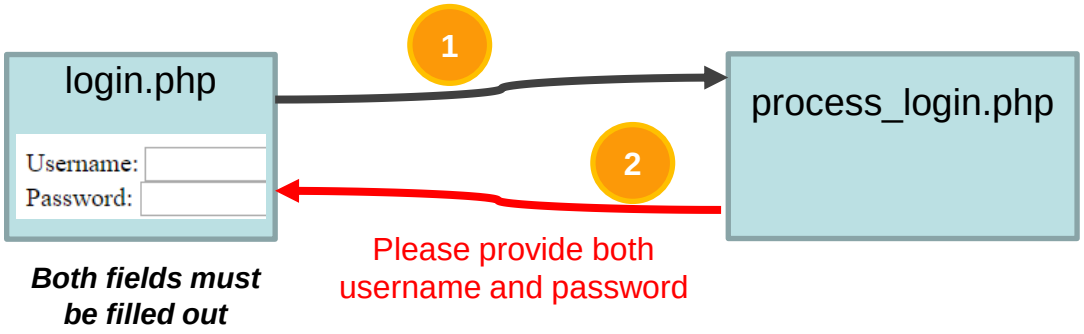


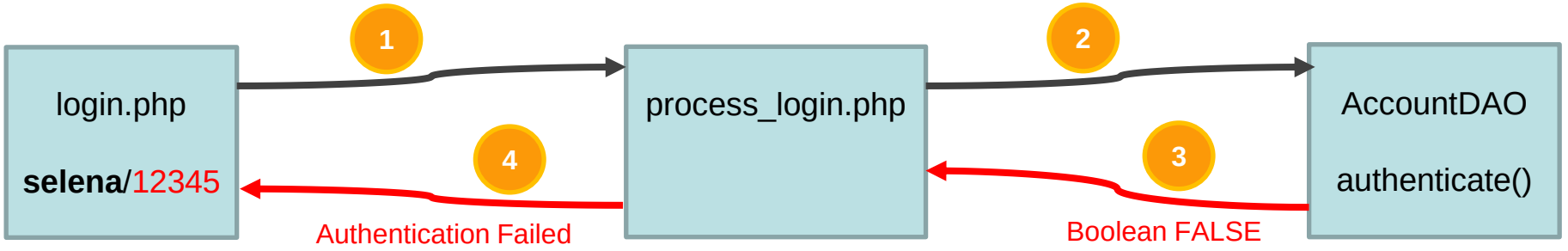
Exercise 3
school

/is113/week12/school/database/create.sql

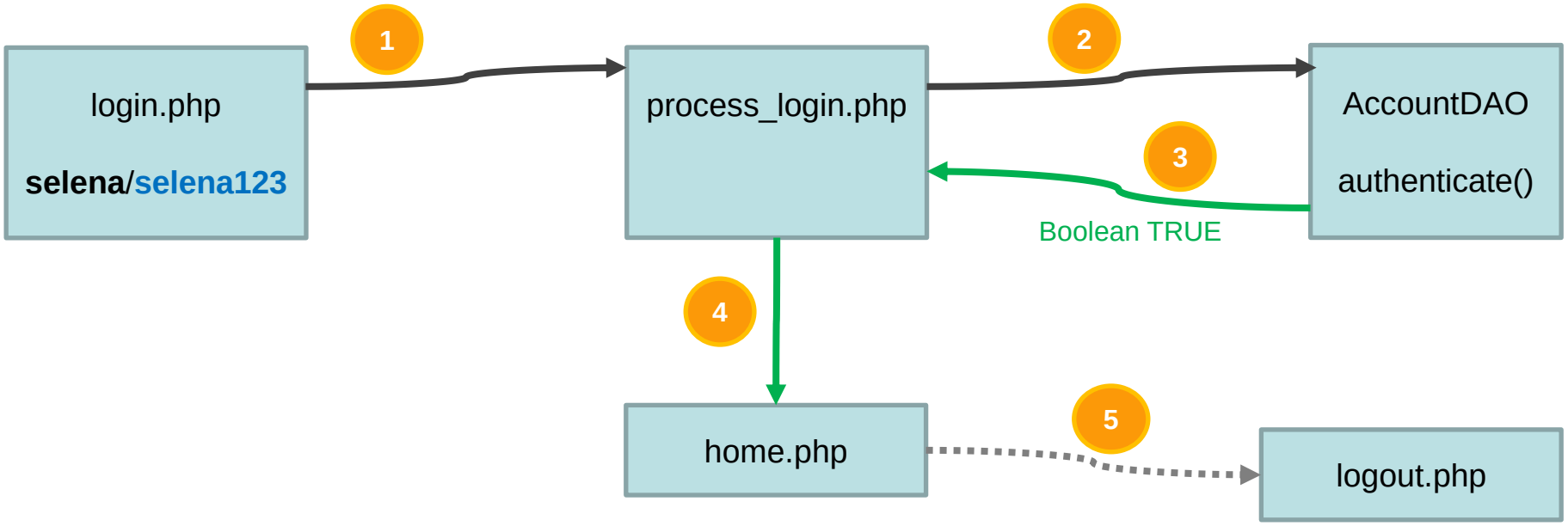
- Copy the file's content into **PHPMyAdmin → SQL**
- Run **all queries**
- Verify that you have:
  - Schema: **week12school**
  - Tables:
    - **account**
    - **grade**

User Navigation Flow





User Navigation Flow



/is113/week12/school/include/AccountDAO.php

1

- Complete **authenticate()** method
- Test the method using **/is113/week12/school/testAccountDAO.php**

```php
require_once 'include/AccountDAO.php';

$dao = new AccountDAO();

// Test Case 1 - selena/selena123 (Correct username/password)
echo '<hr>';
$username = 'selena';
$password = 'selena123';
echo "<h1>Authenticate $username/$password (TRUE)</h1>";
var_dump( $dao->authenticate($username, $password) );

// Test Case 2 - selena/12345 (Incorrect username/password)
echo '<hr>';
$username = 'selena';
$password = '12345';
echo "<h1>Authenticate $username/$password (FALSE)</h1>";
var_dump( $dao->authenticate($username, $password) );
```

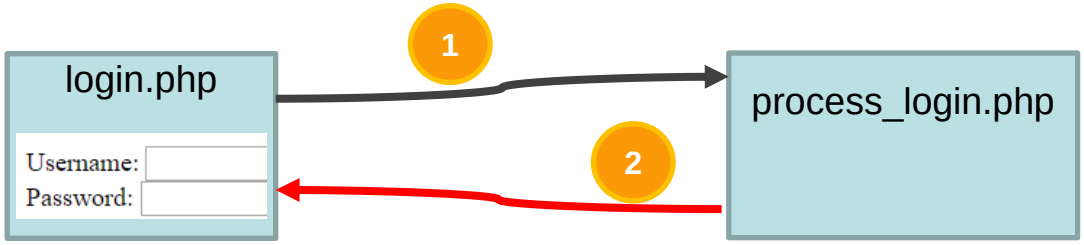
/is113/week12/school/include/GradeDAO.php

- Complete **getGrades()** method
- Test the method using **/is113/week12/school/testGradeDAO.php**

```php
require_once 'include/GradeDAO.php';

$dao = new GradeDAO();

// Test Case 1 - Get all grades
echo '<hr>';
$student_id = 'selena';
echo "<h1>Get all grades for $student_id</h1>";
var_dump( $dao->getGrades($student_id) );

echo '<hr>';
$student_id = 'justin';
echo "<h1>Get all grades for $student_id</h1>";
var_dump( $dao->getGrades($student_id) );
```

/is113/week12/school/process_login.php

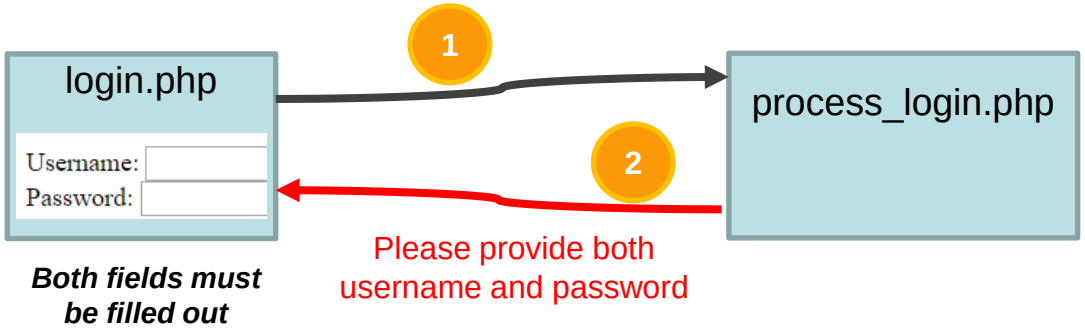
- Implement the following **use case**



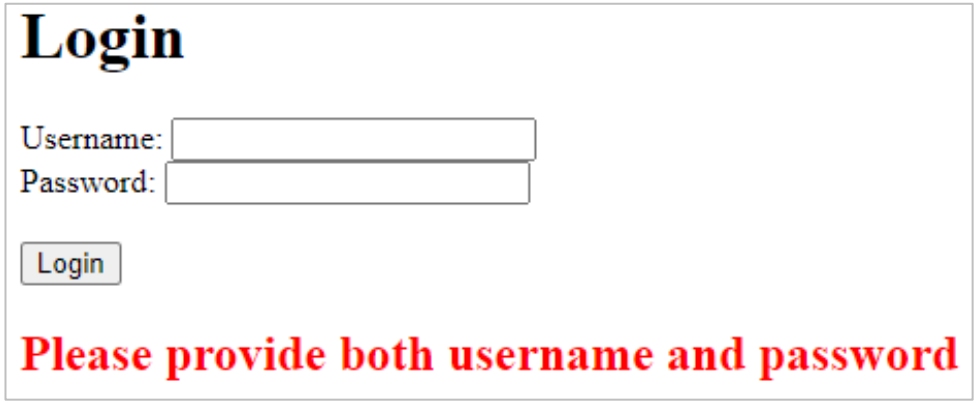
- If the user **fails** to fill in **both input fields** (username & password):
  - Forward the user *back to* **login.php**
- Else
  - **process_login.php** must display both the **username** and the **password**

/is113/week12/school/login.php

- *Continuing from Step 3*



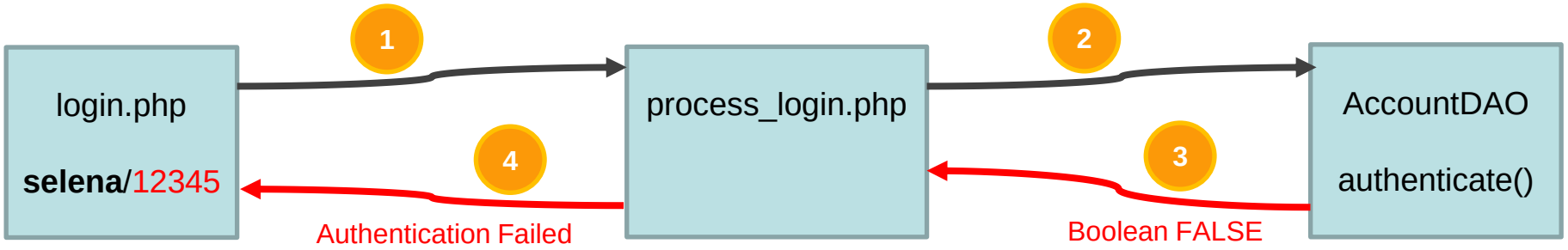
- When the user is *forwarded back* to **login.php** (due to failed authentication), **login.php** must display



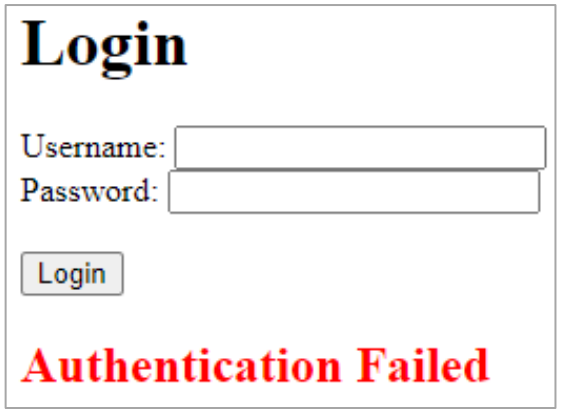
- How can **login.php** distinguish between:
  - User is loading this page for the first time
  - User is here because he/she was *forwarded from* **process_login.php**

/is113/week12/school/process_login.php

- Implement the following **use case**



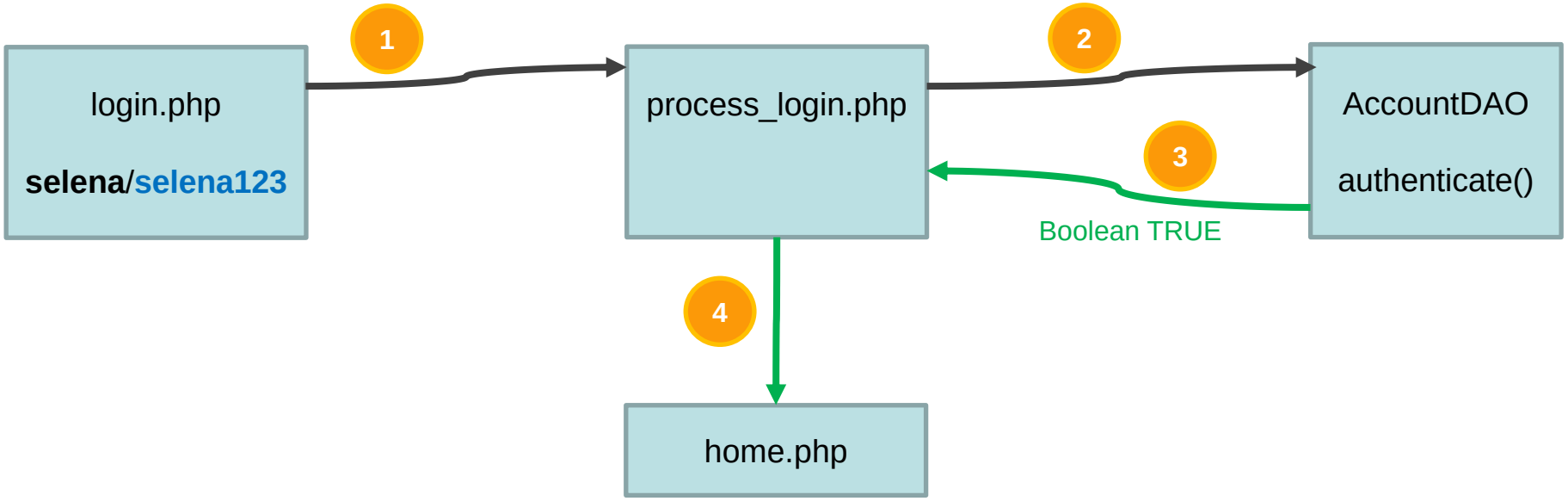
- If authentication **fails**:
  - Forward the user *back to* **login.php**
  - **login.php** must display



- Else
  - **process_login.php** must display **Authentication Valid**

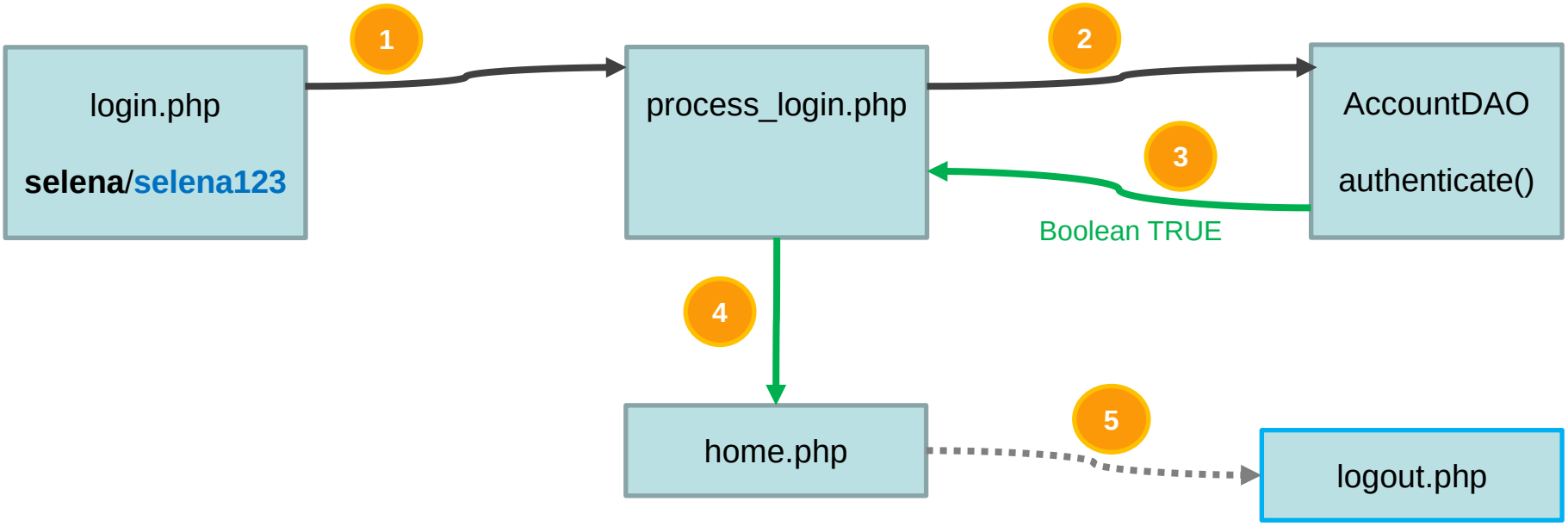
/is113/week12/school/process_login.php

- Implement the following **use case**



- If authentication **succeeds**:
  - Forward the user to **home.php**

/is113/week12/school/logout.php



- **logout.php** must perform (in the following sequence):
  - **Remove** ALL Session Variables
  - **Remove** the **current session** (destroy it)
  - Forward the user *back to* **login.php**

/is113/week12/school/home.php

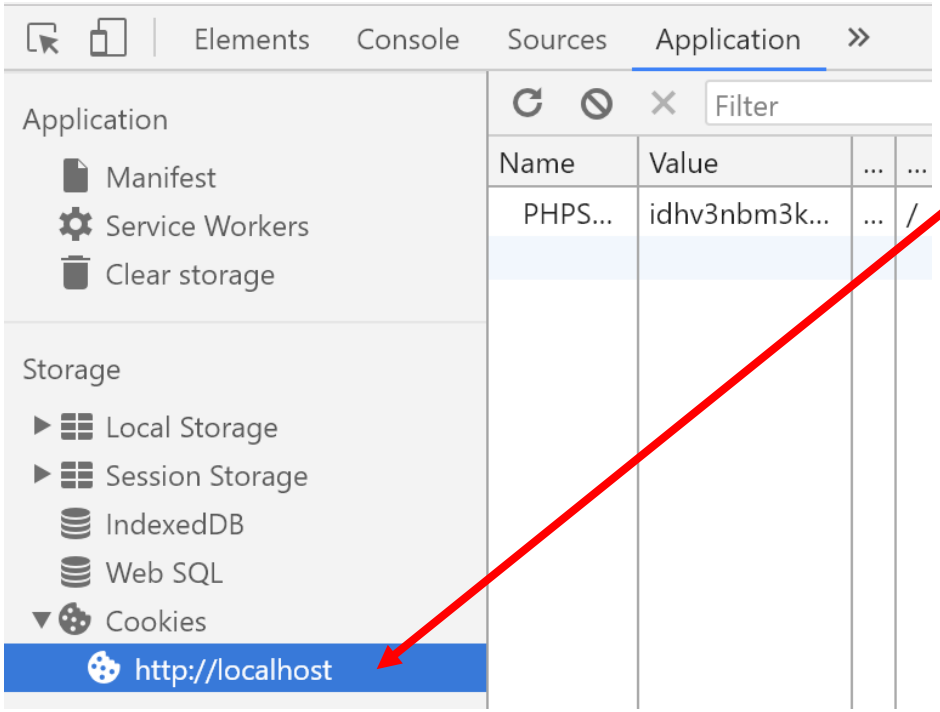
- Complete **home.php** such that:
  - When a user visits the page **without successful authentication**, it *forwards the user back to* **login.php**.
  - When a **successfully authenticated** user visits the page, it must:
    - Retrieve the user's **grades** (from the 'grade' database table)
    - Display the grades in an HTML table
    - For example, below shows user **selena**'s **grades**

You are logged in as **selena** | Log Out

ID	Title	Grade
IS111	Introduction to Programming	C
IS112	Introduction to Databases	B-
IS113	Introduction to Web	C+
IS210	Advanced Programming	C-
IS230	Advanced Web	D+
ECON100	Introduction to Economics	C

/is113/week12/school/home.php

- How do I **test** the scenario of an **un-authenticated user** attempting to access **home.php**?
- There are **2 options**:
  - Go to **logout.php** (which would clear all session variables AND destroy the current session *if any*)
  - Go to **Chrome → More Tools → Developer Tools → Application → Cookies**



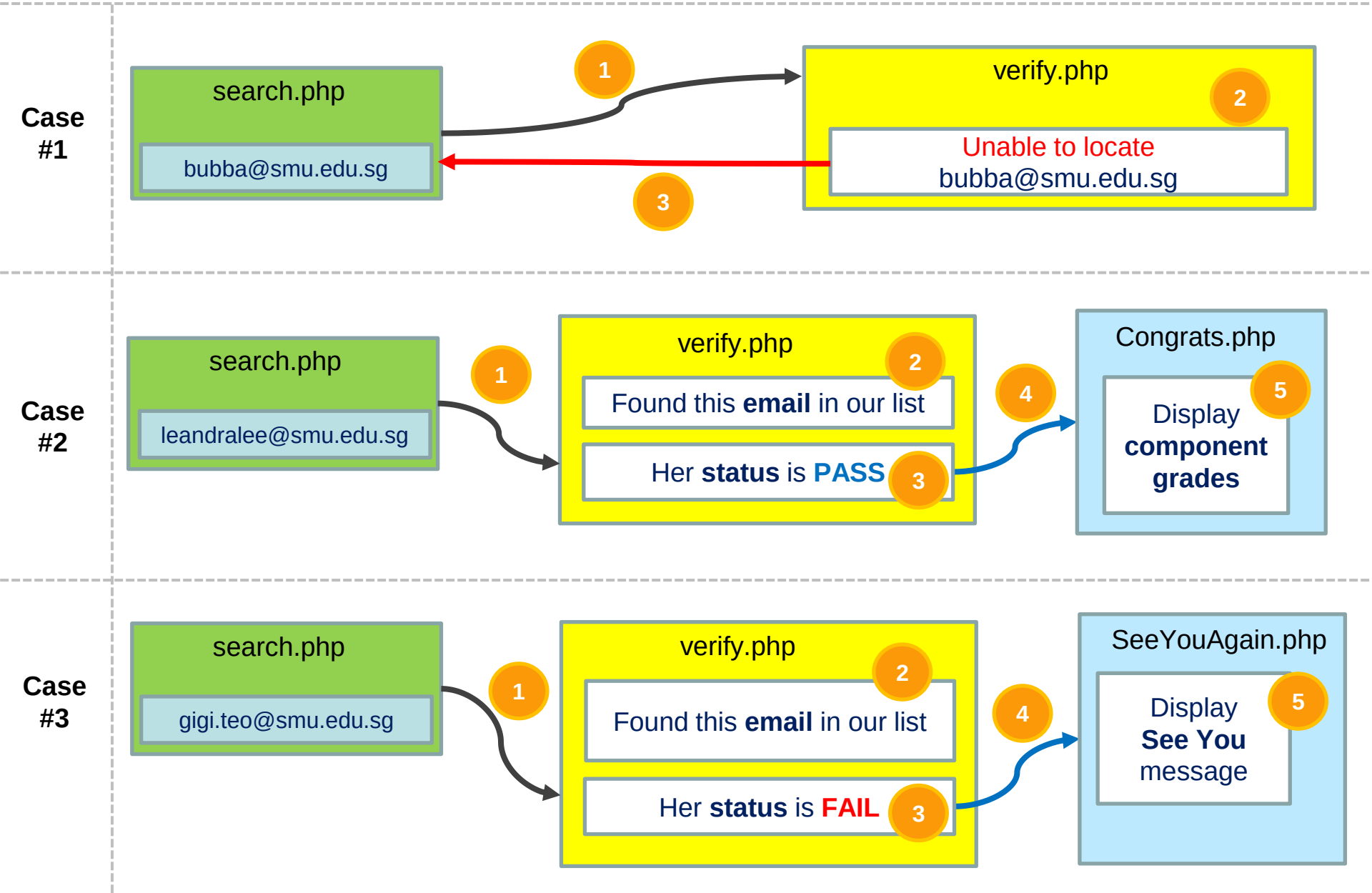
Click on the item under **Cookies** and click **Clear**

Exercise 4
labtest

/is113/week12/labtest/database/create.sql

- Copy the file's content into **PHPMyAdmin → SQL**
- Run **all queries**
- Verify that you have:
  - Schema: **week12LT1**
  - Table: **grade**

User Navigation Flow



/is113/week12/labtest/*

- Complete:
  - search.php
  - verify.php
  - Congrats.php
  - SeeYouAgain.php

Test Case 1

search.php	search.php
Find Lab Test 1 Results bubba@smu.edu.sg Search My Lab Test 1 Grade	**Find Lab Test 1 Results** e.g. gigi.teo@smu.edu.sg Search My Lab Test 1 Grade **Unable to locate bubba@smu.edu.sg**

Test Case 2

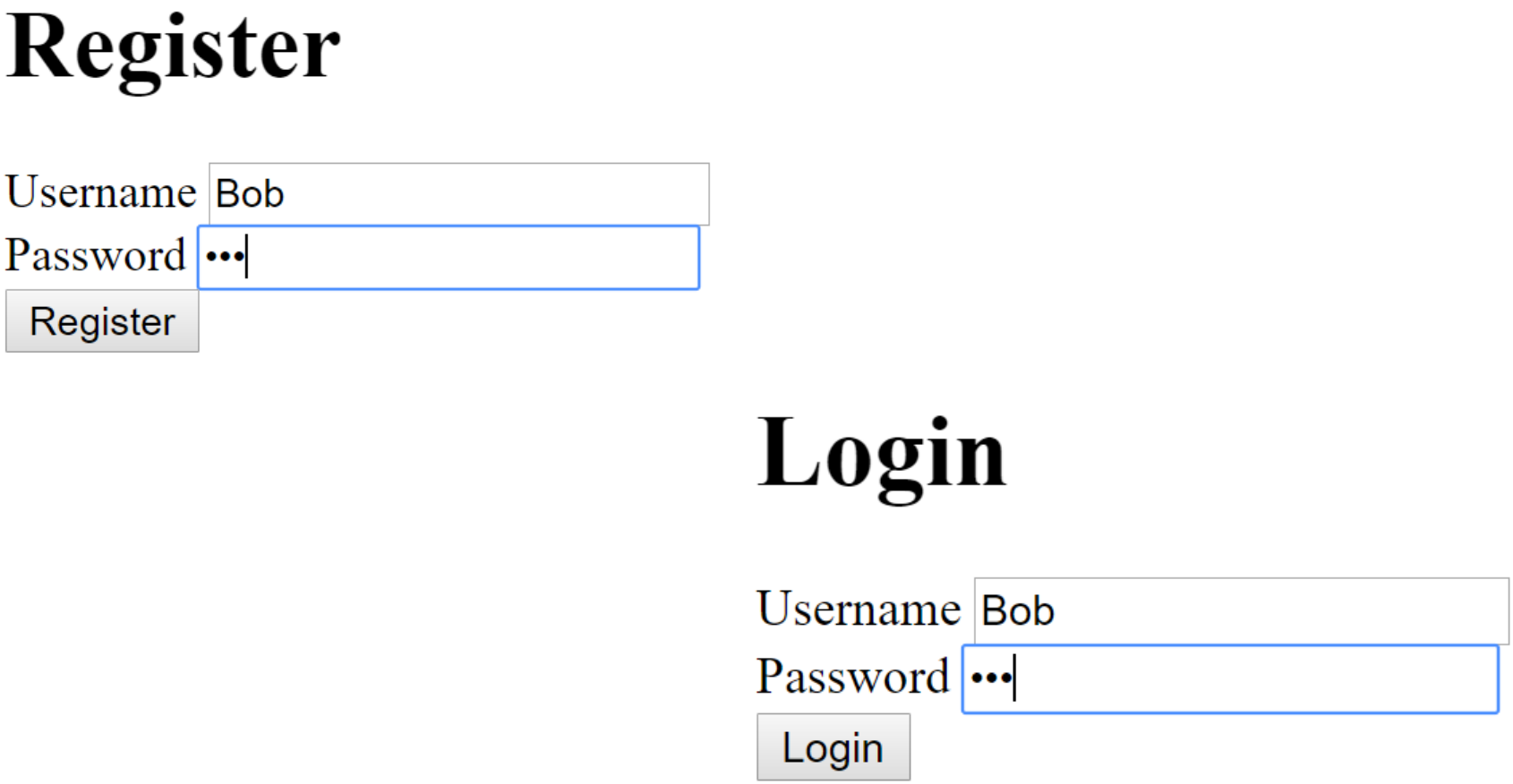
search.php	Congrats.php
Find Lab Test 1 Results leandralee@smu.edu.sg [Search My Lab Test 1 Grade]	**Congratulations! You passed Lab Test 1!** **Grade for leandralee@smu.edu.sg** Q1 | Q2 | Q3 | Total 7 | 8 | 1 | 16 Back to Search

Test Case 3

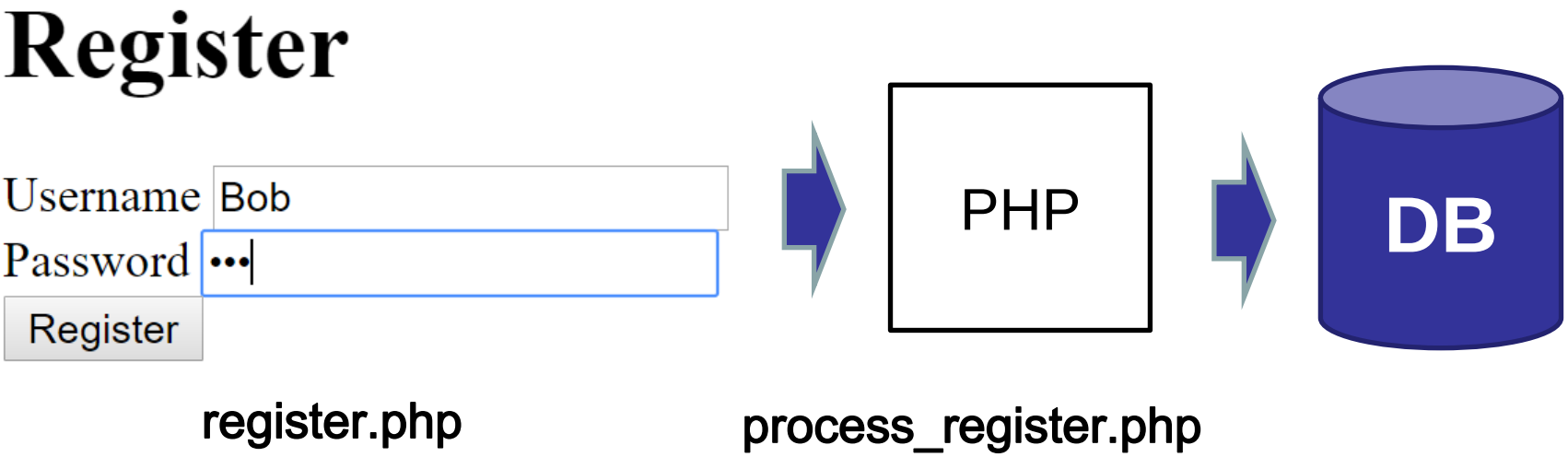
search.php	SeeYouAgain.php
Find Lab Test 1 Results gigi.teo@smu.edu.sg [Search My Lab Test 1 Grade]	**OH MY GOD! You failed Lab Test 1!** **See you again NEXT YEAR!** **Your email gigi.teo@smu.edu.sg has been added to the repeat students list!** Back to Search

Exercise 5 authenticate

Authenticating Users



Authenticating Users



- The user creates an account and his/her password is stored in the database.
- For **security reasons**, we do not want to store plain text password in the database.

Solution: Password Hashing

- A **one-way** transformation on a password
  - Turn the password into another string
- Password can be transformed to its hashed string
  - But, **not the other way round**
- We want to store **hashed** password in the DB

Password Hashing

```
<html><body>
   <h1>Register</h1>
   <form method="post" action="process_register.php">
      Username <input type="text" name="username"/><br/>
      Password <input type="password" name="password"/><br/>
      <input type="submit" value="Register"/>
   </form>
</body></html>
```

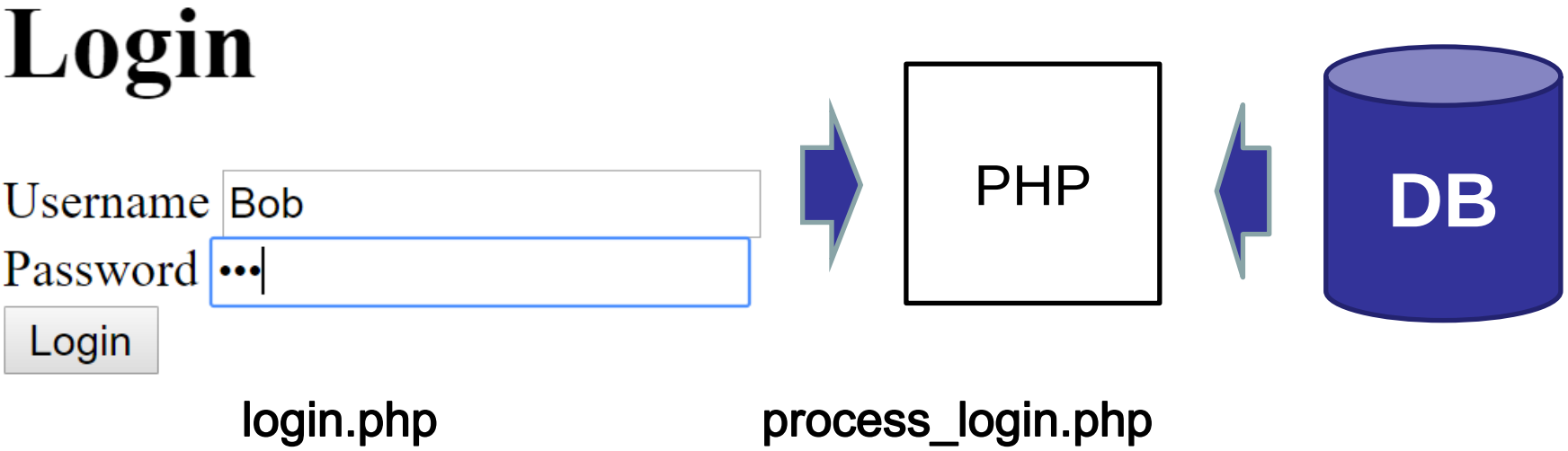
register.php

Password Hashing

```
<?php
    require_once "UserDAO.php";
    $username = $_POST["username"];
    $password = $_POST["password"];
    $hashed = password_hash($password, PASSWORD_DEFAULT);
    $dao = new UserDAO();
    $status = $dao->add($username,$hashed);
    if($status){
        echo "Registered successfully";
    }
    else{
        echo "Failed to register";
    }
?>
```

process_register.php

Login with Password Hashing



- When the user attempts to login:
  - Get the hash of the user's real password from DB
  - It will be checked against the entered password.

Password Hashing

```
<<html><body>
   <h1>Login</h1>
   <form method="post" action="process_login.php">
      Username <input type="text" name="username"/><br/>
      Password <input type="password" name="password"/><br/>
      <input type="submit" value="Login"/>
   </form>
</body></html>
```

login.php

Password Hashing

```
<?php
    require_once "UserDAO.php";
    $username = $_POST["username"];
    $password = $_POST["password"];
    $dao = new UserDAO();
    $hashed = $dao->getHashedPassword($username);
    $status = password_verify($password,$hashed);
    if($status){
        echo "Successful Login";
    }
    else{
        echo "Failed Login";
    }
?>
```

process_login.php

Using Session to Protect Your Pages

- Create a session entry for successful login

```
<?php
    require_once "UserDAO.php";
    $username = $_POST["username"];
    $password = $_POST["password"];
    $dao = new UserDAO();
    $hashed = $dao->getHashedPassword($username);
    $status = password_verify($password,$hashed);
    if($status){
        session_start();
        $_SESSION["user"] = $username;
        echo "Successful Login";
    }
    else{
        echo "Failed Login";
    }
?>
```

process_login.php

Using Session to Protect Your Pages

- For every page that needs to be protected

```
<?php

session_start();

// No session variable "user" => no login
if ( !isset($_SESSION["user"]) ) {

    // redirect to login page
    header("Location: login.php");

    // stop all further execution
    // (if there are statements below)
    exit;
}

?>
```

another_page.php

Using Session to Protect Your Pages

- To avoid repeating code everywhere:

```
<?php
session_start();
if ( !isset($_SESSION["user"]) ) {
    // No session variable "user" =>no login

    // redirect to login page
    header("Location: login.php");
    exit;
}
?>
```

protect.php

```
<?php
require_once "protect.php";
// content below ..
?>
```

another_page.php

/is113/week12/authenticate/password.php

```php
<?php

// User Registration

// Assume we received username/password from Form
$username = 'puppy';
$password = 'puppy123';

// Registration
$hashed_password = password_hash($password, PASSWORD_DEFAULT);
echo "Password: $password <br>";
echo "Hashed password: $hashed_password <br>";
echo "<h2>We will save the following data in MySQL Database Table</h2>";
echo "<h3>Username: $username</h3>";
echo "<h3>Password: $hashed_password</h3>";
```

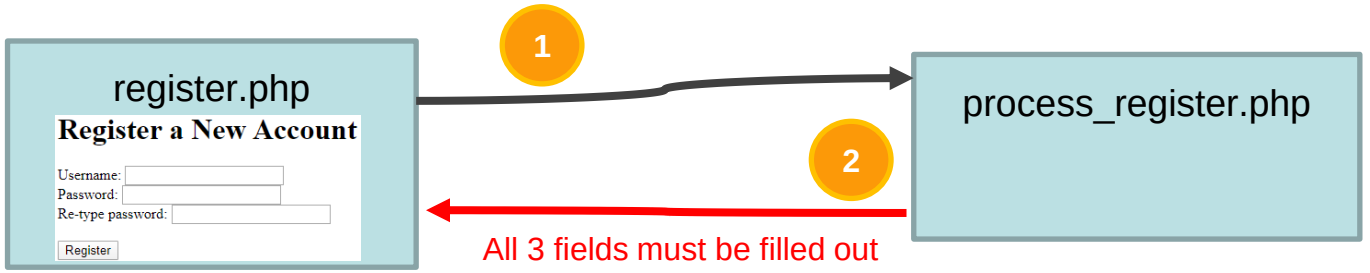
```php
// Authentication
// Next time, user keys in username and password in PLAIN TEXT
// Assume we received username/password from Form
$username = 'puppy';
$password = 'puppy123';

// How do we authenticate the above against the database?
// Especially... the password is in PLAIN TEXT
// How to compare against HASHED password?
// Assume that we retrieved Hashed Password of this user
//    from database
// and it is stored in $hashed_password
$is_same_password = password_verify($password, $hashed_password);

if( $is_same_password ) {
    echo "Authentication Successful";
}
else {
    echo "Authentication Failed";
}
```

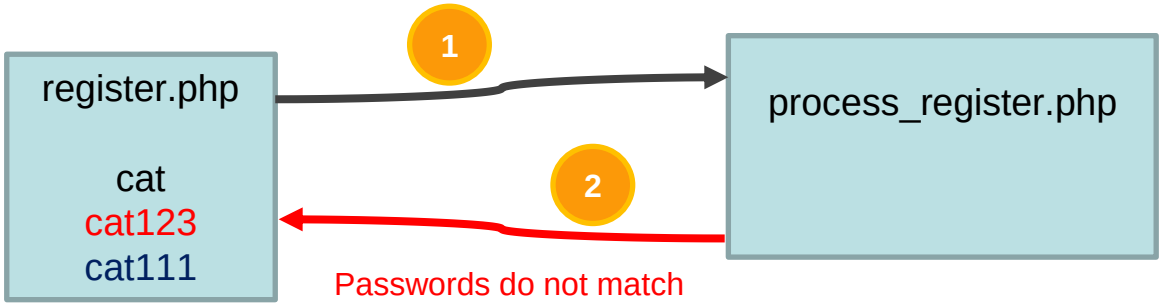
Exercise 6
fanclub
(Try On Your Own - Homework)

/is113/week12/fanclub/database/create.sql

- Copy the file's content into **PHPMyAdmin → SQL**
- Run **all queries**
- Verify that you have:
  - Schema: **week12fanclub**
  - Table: **account**

User Navigation Flow (Register)





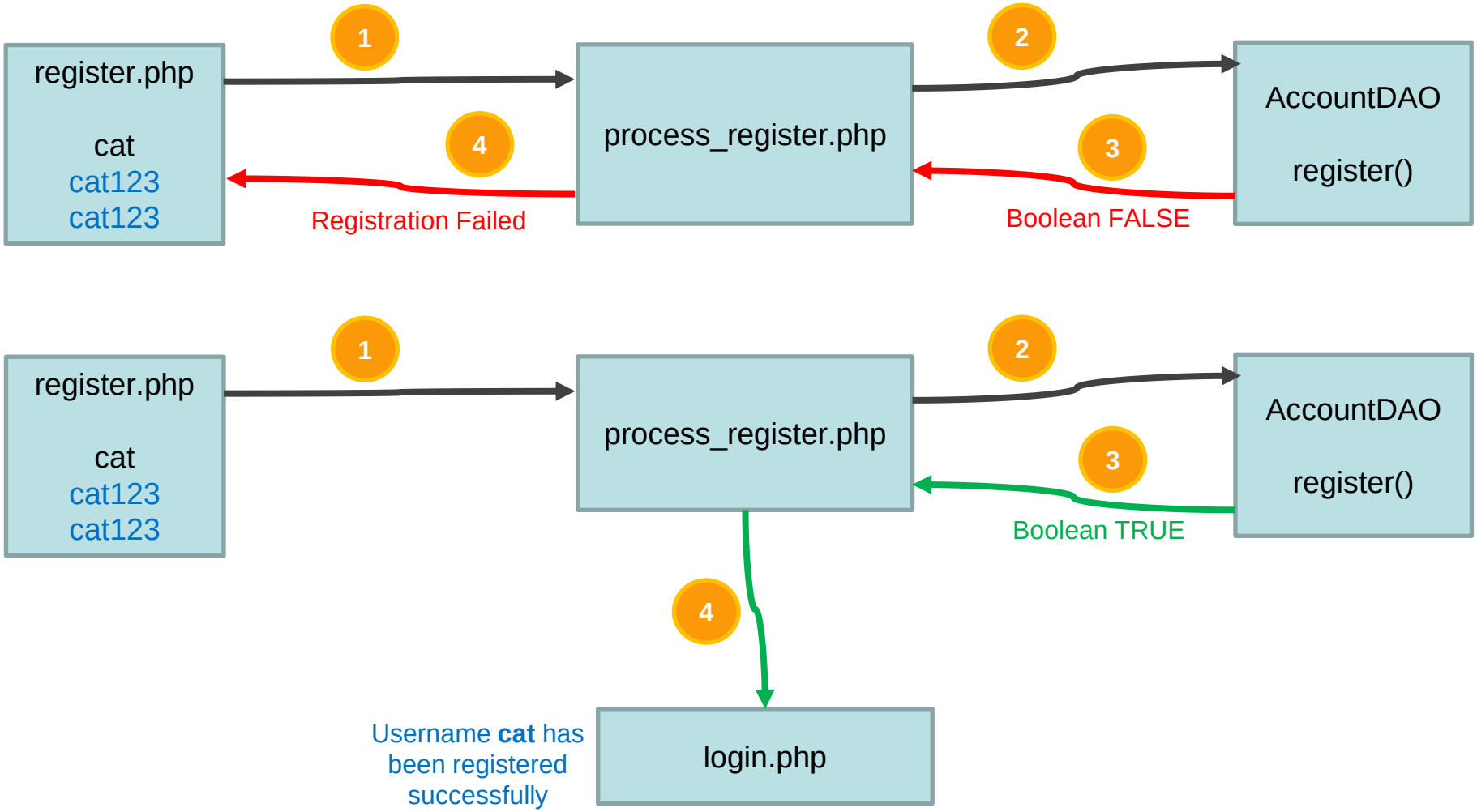
Test Case 1

register.php	register.php
Register a New Account Username: Password: Re-type password: Register	**Register a New Account** Username: Password: Re-type password: Register **All 3 fields must be filled out**
Register a New Account Username: hyunbin Password: •••••••• Re-type password: Register	**Register a New Account** Username: Password: Re-type password: Register **All 3 fields must be filled out**

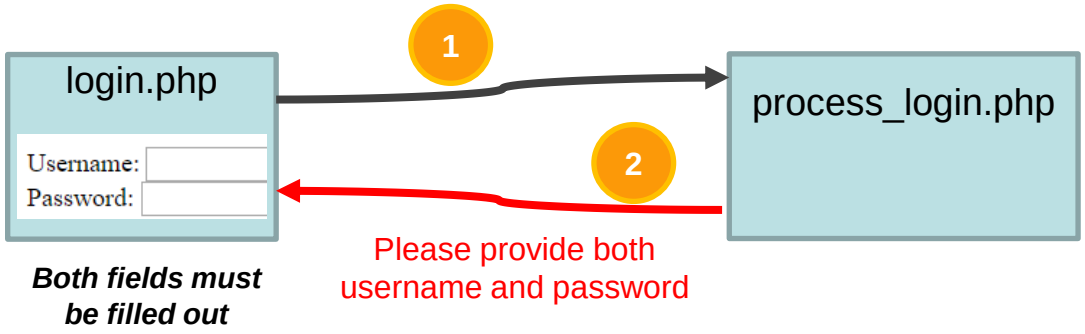
Test Case 2

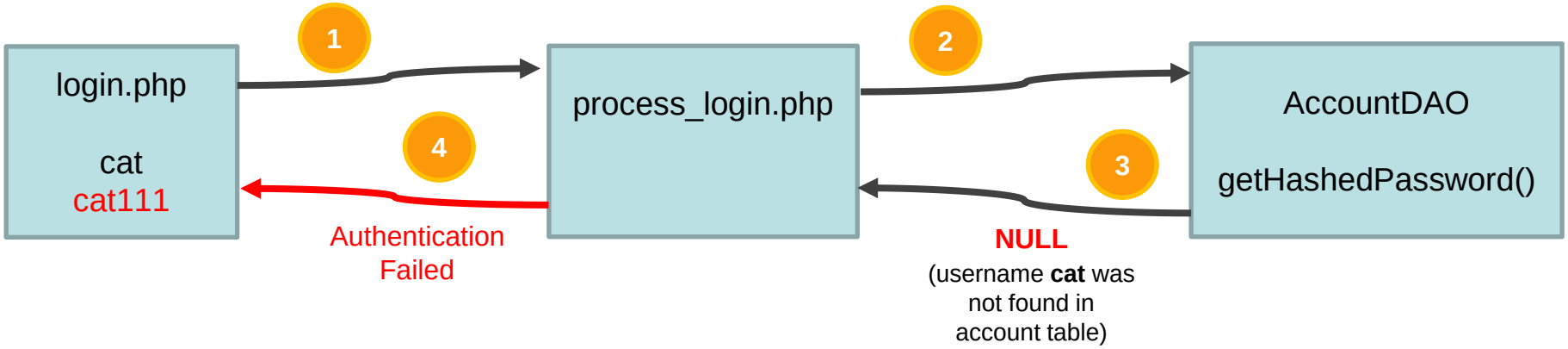
register.php	register.php
Register a New Account Username: hyunbin Password: ••• Re-type password: ••• Register Username: **hyunbin** Password: **123** Re-type Password: **456**	**Register a New Account** Username: Password: Re-type password: Register **Passwords do not match**

User Navigation Flow (Register)

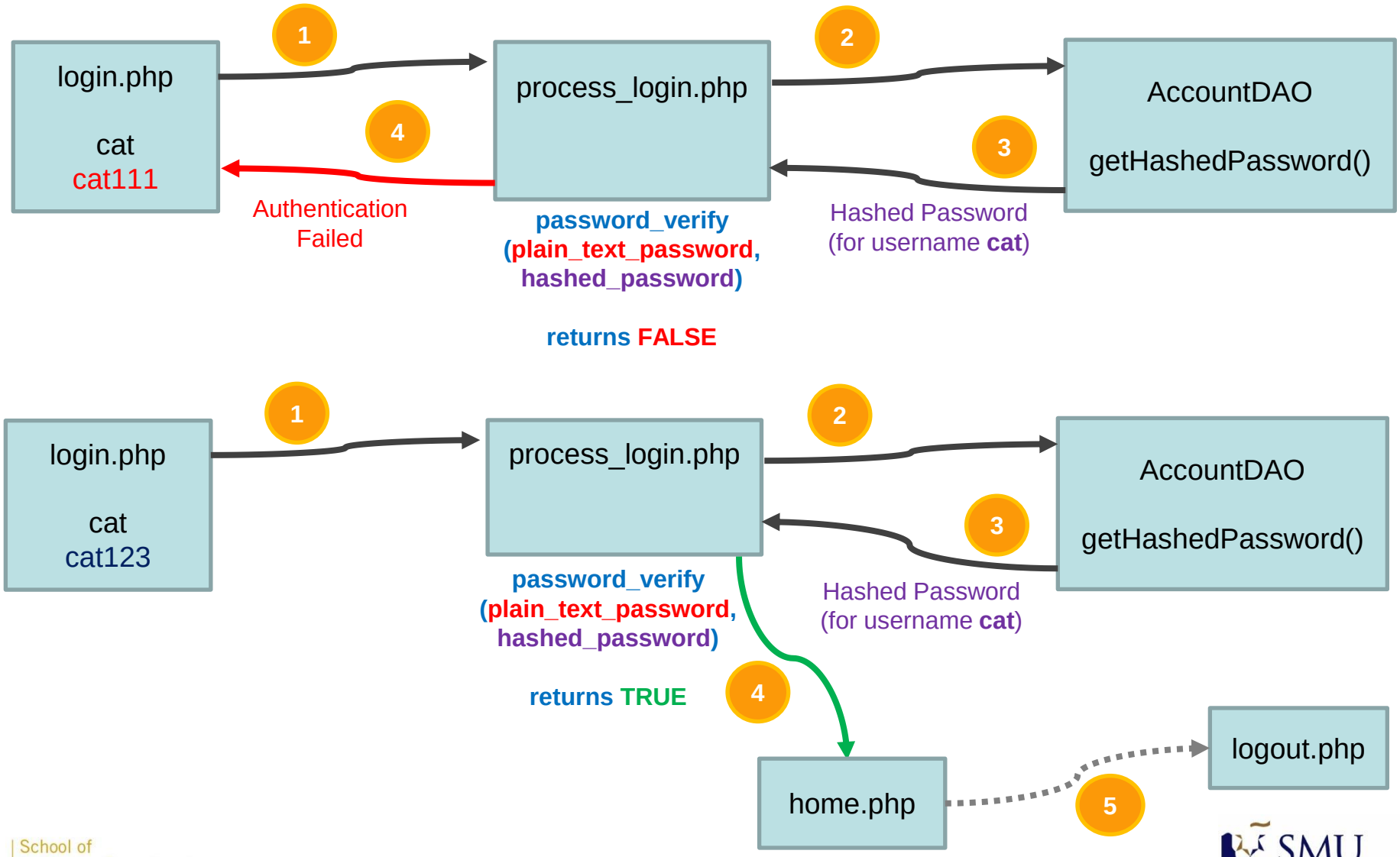


User Navigation Flow (Login)





User Navigation Flow (Login)



/is113/week12/fanclub/*

- Complete:
  - process_register.php
  - register.php
  - login.php
  - process_login.php
  - home.php
  - logout.php

Week 13

- Authenticating users *(hashed password)*
- Trial Lab Test 2

Exercises

- Try **Week 12 Exercises**:
  - https://smu.sg/2021-spring-is113-wk12-doc (Google Doc)
  - Complete
    - **Question 1** (Find Oldest Person)
    - **Question 2** (Shopping Cart)
    - **Question 3** (Login Pages)