**In Class Assignment I:**

**Answer to Q1.**

<form id="loginForm" action="login" method="POST">

<input type="text" id="username" name="username">

<input type="password" id ="password" name="password" />

<input type="submit" value="SignIn" onclick="handleLogin()" /></form>

</form>

<script>

function handleLogin() {

var password = document.getElementById("username").value;

// preprocess password here

}

</script>

Given below is the code that can be used tested on w3schools

<!DOCTYPE html>

<html>

<head>

<script src="https://cdn.jsdelivr.net/npm/md5-js-tools@1.0.2/lib/md5.min.js"></script>

</head>

<body>

<h2>HTML Forms</h2>

<form action="/action\_page.php">

<label for="fname">First name:</label><br>

<input type="text" id="fname" name="fname" value="John"><br>

<label for="lname">Last name:</label><br>

<input type="text" id="lname" name="lname" value="Doe"><br><br>

<input type="submit" value="Submit"onclick="handleLogin()" >

</form>

<script>

function handleLogin() {

var password = document.getElementById("lname").value;

var hash = MD5.generate(password);

document.getElementById("lname").value = hash;

}

</script>

<p>If you click the "Submit" button, the form-data will be sent to a page called "/action\_page.php".</p>

</body>

</html>

The other option is to utilize onsubmit event of the form.

<form action="/action\_page.php" onsubmit="handleLogin()" >

Yes another option is to associate an event listener

<!DOCTYPE html>

<html>

<head>

<script src="https://cdn.jsdelivr.net/npm/md5-js-tools@1.0.2/lib/md5.min.js"></script>

</head>

<body>

<h2>HTML Forms</h2>

<form action="/action\_page.php" >

<label for="fname">First name:</label><br>

<input type="text" id="fname" name="fname" value="John"><br>

<label for="lname">Last name:</label><br>

<input type="text" id="lname" name="lname" value="Doe"><br><br>

<input type="submit" value="Submit" id="myBtn" >

</form>

<script>

const element = document.getElementById("myBtn");

element.addEventListener("click", handleLogin);

function handleLogin() {

var password = document.getElementById("lname").value;

var hash = MD5.generate(password);

document.getElementById("lname").value = hash;

}

</script>

<p>If you click the "Submit" button, the form-data will be sent to a page called "/action\_page.php".</p>

</body>

</html>

**Answer to Q2:**

<form action="main" method="POST">

<input type='submit' name="nav" value='upload' />

<input type='submit' name="nav" value='search' />

<input type='submit' name="nav" value='gallery' />

<input type='submit' name="nav" value='logout' />

</form>

//in servlet

String nav = request.getParameter("nav");

if (nav == null) {

//no button has been selected

} else if (nav.equals("upload")) {

//upload button was pressed

Response.sendRedirect(“upload”);

} else if (nav.equals("gallery")) {

//gallery button was pressed

} else if (nav.equals("search")) {

//search button was pressed

} else if (nav.equals("logout")) {

//logout button was pressed

} else {

//someone has altered the HTML and sent a different value!

}

As opposed HTTP Post Request to the respective servlets, in the revised approach an HTTP Post will be sent to MainServlet which will respond with 302 status code causing another pair of HTTP Request and Response between browser and the UploadServlet.

**Answers to Q3 are similar to the reference code involving JDBC calls already posted on D2L.**

**Answer to Q4:**

In the HTML page, add the following:

<div id="index" style="display:block">

<p id="listView">List Files Here</p>

<img id="myImg" src="" width="107" height="98">

<button onclick="handleFileListing()">List</button>

</div>

Revise handleFileListing function as follows:

function handleFileListing() {

const xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

alert(this.responseText);

document.getElementById("myImg").src = this.responseText;

}

};

xhttp.open("GET", "/triviaapi/trivia/Id=1");

xhttp.send();

}

The reference code for the server side is similar to the code referencing JDBC calls and already posted on D2L.

**In Class Assignment II**

Answer to Q1 (a):

public class UploadAsyncTask extends AsyncTask {

protected void onPostExecute(String result) {

System.out.println(result);

}

protected String doInBackground() {

return new UploadClient().uploadFile();

}

}

**Answer to Q1(b)**

Declare semaphore in the UploadServerThread class as follows:

private static Semaphore sem = new Semaphore(3);

call acquire() method on the semaphore object in the beginning of the run method.

Call release() method on the semaphore object at the end of the run method.

Declare semaphore in the UploadServer class as follows:

public static Semaphore sem = new Semaphore(3);

call acquire() method on the semaphore object in the beginning of the while loop in the UploadServer class.

Call release() method on the static semaphore object towards the end (i.e. after client socket is closed) in the run() method of the UploadServerThread class.

**Answer to Q2**

Q3. Refer to the ThreadSafety class in the ThreadSafety.zip attached to the Midterm Guidelines folder to answer the following questions.

1. [2 marks] What are all possible outputs of the following java code?

T1 :1, T2: 0

T2: 1, T1: 0

T1: 1, T2:1

T2:1, T1: 1

Additionally T1:0 and T2:0 is also possible due to the timing of system.out.println calls in the two threads

1. [1 marks] What are the possible output(s) of a revised ThreadSafety class listed below.

public class ThreadSafety implements Runnable{

int shared = 2;

public static void main(String[] args) {

ThreadSafety ts = new ThreadSafety();

Thread t1 = new Thread(ts, "T1");

t1.start();

Thread t2 = new Thread(ts, "T2");

t2.start();

}

public void run() {

synchronized(this) {

int copy = shared;

try {

Thread.sleep((int)(Math.random() \* 5000));

} catch (InterruptedException e) { }

shared = copy - 1;

System.out.println(Thread.currentThread().getName() + ": " +shared);

}

}

}

T1 :1, T2: 0

T2: 1, T1: 0

1. [1 mark] Among the possible outputs of part(a) and (b) which ones would you consider as thread safe?

Outputs of (b) are thread safe.

1. [2 marks] The following code is an attempt to utilize wait() and notify() to synchronize threads T1 and T2 in such a way that T2 always runs before T1, thus always guaranteeing the following output:

T2: 1

T1: 0

However, there are bugs in the code that are not letting achieve the above result. Fix the code to guarantee the above output.

public class ThreadSafety implements Runnable{

int shared = 2;

public static void main(String[] args) {

ThreadSafety ts = new ThreadSafety();

Thread t1 = new Thread(ts, "T1");

t1.start();

Thread t2 = new Thread(ts, "T2");

t2.start();

}

public void run() {

synchronized(this) {

if (Thread.currentThread().getName().contains("T1")) {

try {

this.wait();

} catch (InterruptedException e) { }

}

int copy = shared;

try {

Thread.sleep((int)(Math.random() \* 10000));

} catch (InterruptedException e) { }

shared = copy - 1;

System.out.println(Thread.currentThread().getName() + ": " +shared);

this.notifyAll();

}

}

}

**In Class Assignment III**

**Reference Code already provided**