5COSC024W Server-side Web Development Mock ICT02 – Questions & Full Answers

Please note that some of these answers are OVERLY detailed, just to give you an idea of some of the aspects you could consider when tackling the questions. Such a high level of detail is NOT expected in your answers!

Question 1

Referring to the PHP code from line 19 to line 21.
Explain in detail what this PHP code does.

[5 Marks]

- Line 19 creates a local variable called $SQL1 and assigns to it an INSERT INTO query to create a new record in the orders table.
- The INSERT INTO SQL query inserts 2 data items in that table: the user id currently in the session and the system current date and time.
- Line 21 executes the SQL query for our database connection which results in the actual insertion in the orders table provided no errors were made.

Question 2

Referring to the PHP code from line 19 to line 21.
Explain what `$_SESSION['c_userid']` is (line 20), where it gets its value from and how this value is used here.

[5 Marks]

- `$_SESSION['c_userid']` refers to the id of the user as stored in the session variable.
- This was created on the login_process.php when the user successfully logged into the system.
- The user ID was stored in this session variable so that it could be retrieved here on checkout.php to assign the right order to the right user.
- This is possible because user id is a Foreign Key (FK) in the orders child table that references the Primary Key (PK) user id of the Users parent table.
- Indeed, the tables orders and users are interconnected with a relationship with a cardinality of one-to-many. One user can place many orders, but an order is for one and only one user.
- Consequently, orders is the child table and carries userId as a FK that references the PK userId of its parent table Users.

Question 3

Referring to the PHP code from line 19 to line 21.
Explain what `$currentdatetime` is (line 20), where it gets its value from and how this value is used here.

[5 Marks]

- $currentdatetime is a local variable of type date.
- It stores the system date and time as currently captured by the PHP `date` function.
- It uses the format ('Y-m-d H:i:s') so that it can be accepted by the MySQL Database and inserted into the orders table.

Question 4

Referring to the PHP code from line 23 to line 31.
Explain in detail what this PHP code does.

[5 Marks]

- Line 23 checks whether the SQL query error detector returns the value of zero. If it does, this means that the new order specified on line 19 and 20 was successfully added on line 21.
- Line 25 creates a local variable called $SQL2 and assigns to it a SELECT query to RETRIEVE the highest order number (max) i.e. the latest order number for the user currently logged in this session (as the orders table has an auto_increment on orderNo).

- The value of the orderNo for the new order just placed by the user currently logged in needs to be retrieved so that it can be used to allocate the order lines to this specific order just generated.
- This is because orderNo is a FK in the order_line that references orderNo as a PK in the Orders table.
- Indeed, the tables orders and order_line are interconnected with a relationship with a cardinality of one-to-many. One order has many order lines, but an order line is for one and only one order.
- Consequently, order_line is the child table and carries orderId as a FK that references the PK orderId of its parent table orders.
- On line 25 `Max(orderNo)` is aliased as `maxno` so that we can refer to it in the array of records `results2` created on line 28.

Question 5

Referring to the PHP code from line 23 to line 31.

Explain in detail what `$exeSQL2` and `$results2` are (lines 27 and 28) and how they are used here.

[5 Marks]

- `$exeSQL2` is an execution variable (a Boolean) that contains the outcome of the execution of the SQL query.
- `$results2` is an array of records that is populated using the mysql_fetch_array function.
- It contains one value (the highest order no for the logged in user) since only one attribute `max(orderNo)` and one row are retrieved by the SQL query. Only one attribute is listed in the SQL query and the query uses the MAX aggregating SQL function, which by definition always retrieves only one row.
- The value is assigned to a local variable `$latestorderno` on line 29 which is displayed as an order reference number on line 31.

Question 6

Referring to the PHP code from line 23 to line 31.

`maxno` appears on line 29 inside the square brackets. Should `orderNo` be used instead? Explain.

[5 Marks]

- `maxno` is an alias used in the query on line 25 to rename the attribute retrieved by the aggregating function `max(orderNo)`.
- It is necessary to use an alias when referencing the value in the square bracket when reading from the array of records.
- Line 29 refers to this alias in the statement `$results2['maxno']`.
- This is what will get the value of the highest order no and assign it to the local variable `$laterstorderno` so that it can be used as a FK value in the INSERT INTO query on line 53.

Question 7

Referring to the PHP code from line 41 and from line 51 to line 55.

Briefly explain the PHP code on line 41.

[5 Marks]

- The foreach loop on line 41 splits the session array between the key (referred to here as `$indexprodid`) and the value (referred to here as `$quantityinbasket`).
- This allows to read the values stored in the session array as key-value pair.
- For each iteration:
  - the id of the selected product gets retrieved and stored in `$indeprodid`.
  - the required number of items for the selected product gets retrieved and stored in `$quantityinbasket`.

Question 8

Referring to the PHP code from line 41 and from line 51 to line 55.

Explain what `$SQL4` is (line 51) and what it is used for.

[5 Marks]

- Line 51 creates a local variable called $SQL4 and assign to it an INSERT INTO query to create a new record in the order_line table.
- For each key-value pair in the basket session array it inserts a new record in the order_line table: the order no of the current order, the product id of the selected product, the number of items specified for the selected product and the sub total.

Question 9

Referring to the PHP code from line 41 and from line 51 to line 55.

Explain what `$latestorderNo`, `$indexprodid`, `$quantityinbasket` and `$subtotal` are (line 53 and 54), where they get their values from and how these values are used.

[5 Marks]

- orderNo gets the value of the number of the order retrieved from the database `$latestorderno` (see line 29). Therefore, `$latestorderno` is a local variable of type integer that contains the latest order no retrieved from the database on line 29.
- prodId gets its value from the key of the basket session array for this present iteration `$indeprodid`. Therefore, $indexprodid is a local variable of type integer that contains the key of the session array.
- The quantity ordered gets its value from the value stored inside basket the session array for this present iteration `quantityinbasket`. Therefore, `$quantityinbasket` is a local variable of type integer that contains the value of the session array.
- The subtotal gets the value from the calculation operated on line 49. Therefore, `$subtotal` is a local variable of type decimal that contains the result of the calculation price x required number of items.

Question 10

Referring to the PHP code from line 101 to line 121.

Explain in detail what this PHP code does and clearly describe the web page it generates.

[5 Marks]

- Line 107 creates a local variable of type string called $SQL and assigns to it a SQL query that retrieves all the details from album table ordered by album code.
- Line 108 executes this query using the mysqli_query function for our database connection and the output of this execution is stored in the local variable $exeSQL.
- This query multiple records i.e. as many as there are albums.
- Line 110 creates an array of records called $results, populates it with the result of the execution of the SQL query using the mysqli_fetch_array function and iterates through it.
- For every iteration through the array of records $results it will display the following:
    - The album title as an anchor with a query string u_id attached to it that passes through the album code using the GET method.
    - The artist and the year of release as plain text.

These 3 elements are separated by || for formatting purposes. This is a quick illustration of the output:

The Wall?u_id=354 || Pink Floyd || 1979
Wu-Tang Forever?u_id=770|| Wu-Tang Clan || 1997
Abbey Road?u_id=912 || The Beatles || 1969

Question 11

Referring to the PHP code on line 112.

A reference to **followingpage.php** (`href=followingpage.php`) appears on line 112 of the **initialpage.php**.

Write the PHP code for the **followingpage.php** file according to the specification below:

When clicking on a link located on initialpage.php, the user should be able to access followingpage.php which should read from the database table Album_Copy and display a list of items related to the one item selected by the user on initialpage.php.

[5 Marks]

1st Answer: Two separate queries

```
<?php
include("db.php");
echo "<html>";
echo "<body>";
echo "<h1>FOLLOWING PAGE</h1>";

$albumcode = $_GET['u_id'];

$SQL1 = " SELECT    albumTitle, artist, yearOfRelease
             FROM Album
             WHERE albumCode = ".$albumcode;

$exeSQL1 = mysqli_query($conn,$SQL1) or die (mysqli_error($conn));

$arraya = mysqli_fetch_array($exeSQL1);

echo "<p><b>Album Title: ".$arraya['albumTitle']."</b>";
echo "<br>Artist: ".$arraya['artist'];
echo "<br>Year Of Release: ".$arraya['yearOfRelease']."</p>";

$SQL2 = " SELECT  copyCode, purchaseDate, conditions
             FROM Album_Copy
             WHERE albumCode = ".$albumcode;

$exeSQL2 = mysqli_query($conn,$SQL2) or die (mysqli_error($conn));

while ($arrayc = mysqli_fetch_array($exeSQL2))
{
       echo "<p>Copy code: ".$arrayc['copyCode'];
       echo "<br>Purchase Date: ".$arrayc['purchaseDate'];
       echo "<br>Condition: ".$arrayc['conditions']."</p>";
}
?>
```

2nd Answer: One JOIN query

```
<?php
include("db.php");
echo "<html>";
echo "<body>";
echo "<h1>FOLLOWING PAGE</h1>";

$albumcode = $_GET['u_id'];

$SQL = "SELECT a.albumTitle AS albumTitle, a.artist AS artist,
         a.yearOfRelease AS yor, ac.copyCode AS copyCode,
         ac.purchaseDate AS purchaseDate, ac.conditions AS conditions
         FROM Album a JOIN Album_Copy ac
         ON a.albumCode = ac.albumCode
         AND a.albumCode = ".$albumcode;

$exeSQL = mysqli_query($conn,$SQL) or die (mysqli_error($conn));

while ($array = mysqli_fetch_array($exeSQL))
{
             echo "<p><b>Album Title: ".$array['albumTitle']."</b>";
             echo "<br>Artist: ".$array['artist'];
             echo "<br>Year Of Release: ".$array['yor'];
             echo "<br>Copy code: ".$array['copyCode'];
             echo "<br>Purchase Date: ".$array['purchaseDate'];
             echo "<br>Condition: ".$array['conditions']."</p>";
}
?>
```

Question 12

Referring to the JavaScript code from line 201 to line 204 and from line 210 to line 215.
Explain how this JavaScript code creates a MongoDB database connection using environment variables.

[5 Marks]

Overall, the express app **only listens for requests after successfully connecting to DB** to avoid situations where app runs without necessary DB connection.

- The code on line 201 loads and imports the **Express framework** into the app, which allows to create simple applications using Node.js. It does so by creating an immutable `express` variable and uses the Node function called `require()` to load the express module and then assigns it to the variable just created.
- The code on line 202 creates an **Express application**. It does so by creating an immutable `app` variable and uses the Express function called `Express()` to return an object that represents the app and then assigns it to the variable just created.
- The code on line 203 loads and imports the **dotenv module** into the app, which allows to access environment variables. It does so by creating an immutable `dotenv` variable and uses the Node function called `require()` to load the dotenv module and then it assigns it to the variable just created.
- The code on line 204 loads and imports the **mongoose module** into the app, which allows to connect to a MongoDB database. It does so by creating an immutable `mongoose` variable and uses the Node function called `require()` to load the express module and then assigns it to the variable just created.
- The code on line 210 uses the `mongoose.connect` method to initiate the MongoDB connection.
- The `mongoose.connect` method uses the connection string DB_CONNECT as retrieved from .env file.
- `DB_CONNECT` is the connection string stored in the secure .env file that enables the database connection.
- .env is an external file that allows the secure storing of environment variables externally for enhanced security.
- The code on line 211 uses the `.then()` **promise handler** to handle successful connection.
- In case the database connection is successful:
    - A 1st message is logged on the console to indicate that the connection is operational.
    - The server on port 3000 is started and listened and a 2nd message is is logged on the console to indicate that the server is up and running.
- The code on line 215 uses the `.catch()` **rejection handler** to handle failed connection.
- In case the database connection is unsuccessful: an error message is logged on the console.

Question 13

Referring to the JavaScript code from line 201 to line 202 and from line 217 to line 225.
Explain how the GET and POST HTTP requests are handled in this JavaScript code and what output this JavaScript code produces.

[5 Marks]

- The code on line 201 loads and imports the **Express framework** into the app, which allows to create simple applications using Node.js. It does so by creating an immutable `express` variable and uses the Node function called `require()` to load the express module and then assigns it to the variable just created.
- The code on line 202 creates an **Express application**. It does so by creating an immutable `app` variable and uses the Express function called `Express()` to return an object that represents the app and then assigns it to the variable just created.

- Overall, the code from line 219 to line 221 sets a route, listens to HTTP GET requests and responds by rendering an Embedded JavaScript (EJS) template created externally to display the front-end of the app.
- The code on line 219 uses the `get()` **method** to define a route handler (a function) to process the HTTP GET request.
- The forward slash `/` specifies path where the callback function is invoked, in this case the root of the site.
- `Req` contains the information (e.g. URL parameters) about the **GET request** made to the server from the client when entering the URL, in our case http://localhost:3000 (no parameters).
- `Res` refers to the **response** sent back to the client from the server.
- `=>` is the callback function executed when the GET request is sent to the root URL of our web app, in our case http://localhost:3000

- The response on line 220 renders the EJS template called todo.ejs which displays the front-end of the Web app i.e. the HTML form to enter a new item in the to-do list.

- Overall, the code from line 223 to line 225 sets a route, listens to HTTP POST requests and responds by logging the value captured in the form in the console.
- The code on line 219 uses the `post()` **method** to define a route handler (a function) to process the HTTP POST request.
- The forward slash `/` specifies path where the callback function is invoked, in this case the root of the site.
- `Req` contains the information (e.g. content of the form) about the **POST request** made to the server from the client when clicking on the form submit button.
- `Res` refers to the **response** sent back to the client from the server.
- `=>` is the callback function executed when the POST request is sent to the root URL of our web app, in our case http://localhost:3000
- The response on line 224 logs a message on the console that consists of the actual data value posted through the form.

Question 14

Referring to the JavaScript code from line 201 to line 205 and from line 227 to line 238.
Explain how this JavaScript code implements the CREATE operation (of the CRUD operations) and how a new document is added into MongoDB.

[5 Marks]

- The code on line 201 loads and imports the **Express framework** into the app, which allows to create simple applications using Node.js. It does so by creating an immutable `express` variable and uses the Node function called `require()` to load the express module and then assigns it to the variable just created.
- The code on line 202 creates an **Express application**. It does so by creating an immutable `app` variable and uses the Express function called `Express()` to return an object that represents the app and then assigns it to the variable just created.
- The code on line 203 loads and imports the **dotenv module** into the app, which allows to access environment variables. It does so by creating an immutable `dotenv` variable and uses the Node function called `require()` to load the dotenv module and then it assigns it to the variable just created.
- The code on line 204 loads and imports the **mongoose module** into the app, which allows to connect to a MongoDB database. It does so by creating an immutable `mongoose` variable and uses the Node function called `require()` to load the express module and then assigns it to the variable just created.
- The code on line 205 loads and imports a **mongoose model** called TodoTask (located in the TodoTask.js file). It does so by creating an immutable `TodoTask` variable and uses the Node function called `require()` to load the model then assigns it to the variable just created.
- The model uses the schema defined in `TodoTask.js` to ensure that documents adhere to a defined structure, including data types, validation, and more.

- The code on line 227 uses the `post()` **method** to define an asynchronous route handler (a function) to process the HTTP GET request.
- The forward slash `/` specifies path where the callback function is invoked, in this case the root of the site.
- `Req` contains the information (e.g. content of the form) about the **POST request** made to the server from the client when clicking on the form submit button.
- `Res` refers to the **response** sent back to the client from the server.
- `=>` is the callback function executed when the POST request is sent to the root URL of our web app, in our case http://localhost:3000
- The code on line 228 creates a new instance `todoTask` of the model `TodoTask` i.e. it creates a new document `todoTask` to be added to the MongoDB database based on the schema defined in `TodoTask.js`
- The response that is sent back on line 229 sets the actual data value posted through the form to the `content` field of the created document.
- The code on lines 231 to 237 consists of an asynchronous try-catch block that waits for completed operations to proceed:
    - Save and redirect if the save operation is successful.
    - Send an error message and redirect if the save operation is not successful

Question 15

Node question.

Explain how JavaScript, traditionally a client-side scripting language, can be used for back-end Web development as part of the Node.js runtime environment.

[5 Marks]

- **JavaScript** was initially a browser client-scripting language. This means that it was hosted by the browser and could be used to:
  - Capture of user's input.
  - Make of HTTP requests.
  - Process instructions to output HTML and CSS.
- The **runtime Node.js environment** was introduced to allow JavaScript to
  - Be used as utility scripts to configure system and to read, write and manipulate files.
  - Send and receive data over the network.
  - Implement efficient web servers and attach application code to it.
  - Make and serve HTTP requests.
- This allowed JavaScript:
  - to be uncoupled from browser and to run on the server, including on cloud-based servers.
  - to use the same language both on the server and client.
  - to be used as part of several front-end frameworks (e.g., React, Vue.js, Angular) and back-end frameworks (e.g., Express.js, Next.js, Sails.js
  - to be used as part of several stacks e.g., MEAN stack (MongoDB, Express, Angular, Node) and MERN stack (MongoDB, Express, React, Node).

Question 16

Node question.

Explain what the **Node.js** is and discuss this technology in relation to coding, speed & performance, Web Servers, databases, frameworks, and application domains.

[5 Marks]

- **Node.js** is a JavaScript host runtime environment. It allows JavaScript to:
  - Be used as utility scripts to configure system and to read, write and manipulate files.
  - Send and receive data over the network.
  - Implement efficient web servers and attach application code to it.
  - Make and serve HTTP requests.
- **Coding**
  - Only one language for back-end and front-end.
  - Higher amount of code.
- **Speed & Performance**
  - Faster due to asynchronous execution.
  - Node.js sends a query to computer file system.
  - Node.js works on next request without waiting for previous.
  - When file system has opened and read requested file, the server sends content back to the client.
- **Web Server**
  - Can be used to set up and run its own **web server**.
  - Easy to do with Express, Koa, etc
- **Databases**
  - Happily supports relational DBs.
  - Happily supports NoSQL DBs.
- **Frameworks**
  - Multiple frameworks to support back-end, front-end and full-stack.
  - E.g. Express, Meteor, koa.js, sails.js
- **Application Domains**
  - Highly scalable server-side solutions.
  - Real-time apps like chat applications, blogs, video streaming applications.
  - Dynamic single-page apps e.g. portfolios, etc.

Question 17

Node question.

Define what the following packages are and explain what their respective roles are in the development of a Node-driven back-end web app: **Nodemon, Express, Ejs, dotenv, and Mongoose.**

[5 Marks]

- **Nodemon**:
  - Package that monitors for any changes in the source code & automatically restarts the server.
  - It automatically restarts the Node app when file change in directory are detected.

- **Express**
  - Minimal and flexible Node.js web app framework.
  - It facilitates the rapid development of Node-based Web apps.
  - It provides a robust set of features to develop web apps.
    - It sets up handlers to respond to different HTTP requests.
    - It performs different actions based on HTTP Method and URL
    - It sets common web app settings such as connection port and template locations.
    - Dynamically renders HTML Pages based on passing arguments to templates.

- **Ejs**
  - One of the most popular template engines for JavaScript.
  - It enables to rapidly build web applications that are split into different components.
  - It embeds JavaScript code in a reusable web template to generate HTML.
  - It replaces the variables in a template file with actual values and displays this value to the client.
  - It retains the syntax of HTML while allowing data interpolation.

- **dotenv**
  - Module that loads configuration environment variable from separate .env file.

- **Mongoose**
  - Object Data Modelling (ODM) library for MongoDB and Node.js.
  - It acts as a front-end to MongoDB.
  - It manages relationships between data and provides schema validation.
  - It is used to translate between objects in code & representation of objects in MongoDB.
  - It provides a document storage and query system very similar to JSON.

Question 18

MongoDB Atlas Question.

Explain what a **Database-as-a-Service (DBaaS),** what it is used for and what the benefits of using it are.

[5 Marks]

- **Database-as-a-Service (DBaaS)**
  - Fully-managed cloud database.
  - It allows the setting up, configuration, deployment, managing & scaling of a database on the cloud.
  - It uses cloud public service provider e.g., AWS (Amazon), Azure (MS) or GCP (Google).
  - No need for physical hardware, software updates, and configuration.

It includes the following **benefits**:

- **Cost savings**
  - Predictable periodic costs based on used resources, no need for additional capacity.
- **Scalability – up and down**
  - Quickly & easily provision additional storage & computing capacity at run time.
- **Simpler, cheaper management**
  - DB management done by cloud provider, lightens the admin burden on IT staff.
- **Rapid development and faster time-to-market**
  - Devs use DB capabilities to integrate with their apps without need to request access.

- **Data and application security**
  - Cloud provider offers enterprise security e.g., default encryption & access mgmt controls.
- **Reduced risk**
  - Cloud provider offers guaranteeing amount of uptime; excess downtime compensated.
- **Software quality**
  - Cloud provider offers wide variety of highly configurable DBaaS options.

Question 19

MongoDB and Mongoose question.
Define what the following things are and explain how they are used in the development of a Node/MongoDB CRUD Web app: **MongoDB document, MongoDB collection, mongoose schema,** and **mongoose model.**

[5 Marks]

- **JSON**
  - Open standard file and data interchange format to store and transmit data consisting of attribute–value pairs and arrays.
  - Language-independent data format.
  - Originally derived from JavaScript.

- **JSON Documents**
  - Documents consist of name-value pairs.
  - Names are strings.
  - Values can be numbers, strings, nulls, arrays, or objects.

- **BSON: Binary JSON**
  - Binary representation to store data in JSON format, optimized for speed, space, and flexibility.
  - It offers the capacity to query and manipulate objects by specific keys inside the JSON/BSON document, even in nested documents many layers deep.
  - It provides more sophisticated numeric types.
  - It includes additional non-JSON-native data types e.g., dates and binary objects.

- **MongoDB Document**
  - Record in MongoDB.
  - Similar to JSON objects.
  - Data structure: field & value pairs.
  - Values of fields may include other docs, arrays, and arrays of docs.

- **MongoDB Collection**
  - Gathering of documents.
  - Analogous to tables in relational databases.
  - It can be created implicitly or explicitly.
  - It provides documentation validation rules.
  - Documents in a collection do not need to have the same schema.
  - Changing the structure of a document will update the structure of the collection.

- **Mongoose Schema**
  - Data structure of documents to be created as enforced by application layer.
  - It specifies the fields stored in each doc along with validation requirements & default values.

- **Mongoose Model**
  - Wrapper for the Mongoose schema.
  - Provides provides an interface for the database to create, read , update, delete records (CRUD).
  - A schema is "compiled" into a model using mongoose.model().
  - The model is then mapped to documents using the model's schema definition.
  - The model represents a collection of documents that follows the structure specified by the schema.
  - Documents are thus individual instances attached to a model.

Question 20

MongoDB Connection question.

Explain how **environment variables** and the **Dotenv** package are used to create a connection between Node.js and MongoDB**.**

[5 Marks]

- **Environment variables**
  - Environment Variables are variables that are set outside source code, often through a cloud provider or an Operating System.
  - They are used to securely and conveniently configure sensitive elements that don't change often e.g. DB connection strings, URLs, authentication keys, and passwords.
  - They are supported by Node and are accessible via the dotenv module.
- **Dotenv**
  - Dotenv is a lightweight npm package, automatically loads environment variable from a .env file.
  - The .env file needs to be at the top level of the app file structure.
  - The .env file can store database-related environment variables and is easy to change.
  - The .env file is accessible through the process.env.KEY pattern.
- For example:
  - The code on line 203 loads and imports the **dotenv module** into the app, which allows to access environment variables. It does so by creating an immutable `dotenv` variable and uses the Node function called `require()` to load the dotenv module and then it assigns it to the variable just created.
  - The code on line 204 loads and imports the **mongoose module** into the app, which allows to connect to a MongoDB database. It does so by creating an immutable `mongoose` variable and uses the Node function called `require()` to load the express module and then assigns it to the variable just created.
  - The code on line 210 uses the `mongoose.connect` method to initiate the MongoDB connection.
  - The `mongoose.connect` method uses the connection string DB_CONNECT as retrieved from .env file.
  - `DB_CONNECT` is the connection string stored in the secure .env file that enables the database connection.
  - .env is the external file that allows the secure storing of environment variables externally for enhanced security.

END OF DOCUMENT