COMSATS University Islamabad, Lahore Campus

Final Examination – FALL 2024

Course Title:	**Web Technologies**	**Course Code:**	**CSC336**	**Credit Hours:**	**3(2,1)**
Instructor/s:	**M. Ali Hassan, M. Usman Akram**	**Program Name:**	**BS (CS/SE)**		
Semester:	[redacted] **Batch** [redacted] **Section:**	[redacted]	**Date:**	**08-01-2025**	
Time Allowed:	**3 Hours**	**Maximum Marks:**		**50**	
Student's Name:	[redacted]	**Reg. No.**	[redacted]		

Question 1: Answer the following questions **[5+5+5 Marks]**

CLO-1; Bloom Taxonomy Level: Understanding

Part A

Explain the differences between server-side rendering (SSR) and client-side rendering (CSR). Additionally, discuss the advantages and disadvantages of each approach.

Part B

Explain the MVC(Model-View-Controller) Architecture with the help of a diagram

Part C

You are given the following specification for a simple web application that manages a library of books:

- The application allows users to:
  - View a list of all books.
  - Add a new book to the library.
  - Edit details of an existing book.
  - Delete a book from the library.
- The application uses:
  - **Express** for handling HTTP requests.
  - **Mongoose** for interacting with a MongoDB database.
  - **EJS** for rendering dynamic views.

Identify the routes and EJS View Files which you would like to implement for above specifications.

Question 2: Answer the following questions **[10+5 Marks]**

CLO-2; Bloom Taxonomy Level: Applying

Part A

"Design an HTML form to collect book details from the user with the following fields:

1. Author (string): Required, must not be empty.
2. Title (string): Required, must not be empty.
3. Genre (string): Optional, but if provided, must be one of the following: ["Fiction", "Non-Fiction", "Fantasy", "Science", "History"].
4. Published **Year** (number): Optional, but if provided, must be a valid 4-digit year (e.g., 2023).

Dynamically bind an onsubmit event handler to the form. This handler should validate the form fields and only allow submission if the form is valid. Otherwise, it should display appropriate error messages."

Part B

Below is a simple js function fetchData which returns a promise and also the usage is provided

```
function fetchData(isSuccessful) {
 return new Promise((resolve, reject) => {
  setTimeout(() => {
   if (isSuccessful) {
    resolve("Data fetched successfully!");
   } else {
    reject("Failed to fetch data.");
   }
  }, 1000); // Simulates a 1-second delay
 });
}

// Usage example
fetchData(true)
 .then((message) => {
  console.log(message); // Output: "Data fetched successfully!"
 })
 .catch((error) => {
  console.error(error); // Output: "Failed to fetch data."
 });
```

create a new function fetchDataCallBackBased which should use the above promise based function and provide a call back based variation

```
// Usage example
fetchDataCallBackBased(true, (error, result) => {
 if (error) {
  console.error("Error:", error); // Output: "Error: Failed to fetch data." (if failed)
 } else {
  console.log("Success:", result); // Output: "Success: Data fetched successfully!" (if successful)
 }
});
```

Question 3: Answer the following questions **[5+5+10 Marks]**

CLO-3; Bloom Taxonomy Level: Applying

Part A

Write an Express.js route that accepts a route parameter type (e.g., "query" or "params") and performs different actions based on the type. If type is "query", it should read two numbers a and b from the query

string, add them, and return the result in JSON format. If type is "params", it should read two numbers from route parameters a and b, multiply them, and return the result in JSON format

Part B

Develop a middleware function in an Express.js application to validate the data in the request body for creating or updating a book.

Each book has the following fields:

- title (string): Required, must not be empty.
- author (string): Required, must not be empty.
- genre (string): Optional, but if provided, must be one of the following: ["Fiction", "Non-Fiction", "Fantasy", "Science", "History"].
- publishedYear (number): Optional, but if provided, must be a valid 4-digit year (e.g., 2023).

The middleware should:

1. Check if the title and author fields are present and not empty.
2. Validate the genre field, if provided, to ensure it matches one of the allowed values.
3. Validate the publishedYear field, if provided, to ensure it is a 4-digit year.
4. Return an error response (status code 400) if any validation fails, specifying the issue.

Also explain how this middleware would be applied to any route

Part C

Suppose you have a model author in mongoose

```
const mongoose = require('mongoose');
// Define Author schema
const authorSchema = new mongoose.Schema({
 name: { type: String, required: true },
 bio: { type: String },
 birthYear: { type: Number, min: 1000, max: 9999 },
});
const Author = mongoose.model('Author', authorSchema);
module.exports = Author;
```

Using Express, implement the following routes:

1. **GET /author/create**
   - This route should render a form to create a new author.
   - If the form was previously submitted with errors, retrieve the submitted data from the session and prepopulate the form with it.
2. **POST /author/create**
   - This route should handle form submissions by:
     - Validating the submitted data.
     - Saving the valid data to the database.
     - Storing any invalid form data in the session if validation fails.
     - Storing success or failure message
     - Redirecting back to the GET /author/create route after storing the data (or in case of validation errors).