Deakin University

Database Fundamentals

OnTrack Submission

Interaction with a database via an User Interface

Submitted By:
Sathiyanarayanan Senthil Kumar
s223789819
2023/09/24 21:59

Tutor:
Mehul Warade

Outcome	Weight
Fundamental concepts of database	♦♦♦♦♦
Relational Database Modelling	♦♦♦♦♦

This task is to develop a user interface for the DB that was created in the previous tasks. I used HTML, CSS, JavaScript and NodeJS to implement the front-end. By completing this task, I got to know how a webpage can be connected to a MySQL server.

September 24, 2023

User interface and Embedded SQL

1. Introduction:

This essay explains how a user interface can be designed to add data to the “BOOKINGS.COM” DB that was designed and developed in the previous tasks. For creating the user interface, HTML, Bootstrap, JavaScript was used to design the front-end. To host the webpage, NodeJS was used with express framework. And for the database, MySQL DB was used.

2. Hosting the webpage and connecting to MySQL:

2.1 Code:

```
//Using express framework to host the webpage
let express = require('express');
let app = express();
var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

const mysql = require('mysql2');

//Setting port number to 3000
function normalizePort(val) {
    var port = parseInt(val, 10);

    if (isNaN(port)) {
        return val;
    }

    if (port >= 0) {
        return port;
    }
    return false;
}

app.use(express.static('Public'));
//Parse URL encoded data from HTTP
app.use(express.urlencoded({ extended: false }));

// Connecting to the MySQL database
const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'mysqlpwd',
    database: 'bookings_com_db',
    port: 3306
});
```

```
db.connect((err) => {
    if (err) {
        console.error('Error connecting to the database: ' +
err.stack);
        return;
    }
    console.log('Connected to the database as ID ' + db.threadId);
});

app.post('/submit', (req, res) => {
    const { fname, lname, phone, email, ccrd, cexp } = req.body;

    const insertQuery = `INSERT INTO CUSTOMER (CUST_FNAME,
CUST_LNAME, CUST_PHONE, CUST_EMAIL, CUST_CREDIT_CARD,
CUST_CREDIT_EXP) VALUES (?, ?, ?, ?, ?, ?)`;
    const values = [fname, lname, phone, email, ccrd, cexp];

    db.query(insertQuery, values, (err, result) => {
        if (err) {
            console.error('Error inserting data: ' + err);
            return res.status(500).send('Error inserting data');
        }
        console.log('Data inserted successfully');
        res.send('Data inserted successfully');
    });
});


app.listen(port, function () {
    console.log(`Web server running at: http://localhost:${port}`)
    console.log("Type Ctrl+C to shut down the web server")
})
```

2.2 Explanation:

For hosting the webpage, NodeJS was used with express framework.

- **Require Express and Create an Express Application:** First, the code imports the Express.js library and creates an instance of an Express application, which will be used to define routes and handle HTTP requests.
- **Normalize Port:** The function normalizePort is used to set the port for the server. It first checks if there's a port provided in the environment variable process.env.PORT. If not, it defaults to port 3000. The port value is then set on the Express application.
- **Static File Serving and URL Encoding:** These lines configure the Express application to serve static files from a directory named 'Public'. This is useful for serving client-side assets like HTML, CSS, and JavaScript files. The express.urlencoded middleware is used for parsing URL-encoded data from HTTP requests.

- **Create a MySQL Connection:** This code establishes a connection to a MySQL database. It provides the database host, username, password, database name, and port information. This connection will be used to interact with the database.
- **Handle Database Connection:** This code attempts to connect to the MySQL database. If there's an error, it will be logged. Otherwise, it prints a success message along with the thread ID.
- **Handle POST Requests:** This route handles HTTP POST requests to '/submit'. It extracts data from the request body, constructs an SQL query to insert data into a 'CUSTOMER' table, and executes the query using the MySQL connection. If there's an error during the insertion, an error response is sent. If the data is inserted successfully, a success response is sent.
- **Start the Server:** Finally, the code starts the Express web server, listening on the port defined earlier. It also logs a message indicating the server's address and instructs how to shut it down.

This code sets up a basic Node.js web server with Express.js, connects to a MySQL database, and provides an endpoint for inserting data into the database when a POST request is received.

3. Designing the front-end with HTML, Bootstrap and validating the form with JavaScript:

3.1 Code:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Customer details</title>
    <meta charset="utf-8">
    <meta name="sathiyanarayanan" content="SIT772">
    <meta name="description" content="Filling a database">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootst
rap.min.css"
        rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
        crossorigin="anonymous">

</head>

<body>

    <header class="container-fluid bg-dark text-white">
        <h1 class="p-5 text-center">Bookings.com</h1>
    </header>
    <main class="container ">
        <h3 class="py-4">Customer Details:</h3>
```

```
        <p>Please fill the following details and click Submit to
register your details to our database!</p>

        <form action="/submit" name="customer" id="customer"
onsubmit="return Validation()" method="post">
            <div class="row mb-2">
                <label class="col-md-2 col-form-label"
for="fname"><strong>First name:</strong></label>
                <div class="col-md-7">
                    <input type="text" class="form-control"
id="fname" placeholder="Enter your fname..." name="fname">
                </div>
                <div id="fnamemsg"></div>
            </div>
            <div class="row mb-2">
                <label class="col-md-2 col-form-label"
for="phone"><strong>Last name:</strong></label>
                <div class="col-md-7">
                    <input type="text" class="form-control"
id="lname" placeholder="Enter your lname..." name="lname">
                </div>
                <div id="lnamemsg"></div>
            </div>
            <div class="row mb-2">
                <label class="col-md-2 col-form-label"
for="phone"><strong>Phone Number:</strong></label>
                <div class="col-md-7">
                    <input type="number" class="form-control"
id="phone" placeholder="xxxxxxxxxx" name="phone">
                </div>
                <div id="phonemsg"></div>
            </div>
            <div class="row mb-2">
                <label class="col-md-2 col-form-label"
for="email"><strong>Email ID:</strong></label>
                <div class="col-md-7">
                    <input type="email" class="form-control"
id="email" placeholder="something@email.com" name="email">
                </div>
                <div id="emailmsg"></div>
            </div>
            <div class="row mb-2">
                <label class="col-md-2 col-form-label"
for="ccrd"><strong>Credit card number:</strong></label>
                <div class="col-md-7">
                    <input type="number" class="form-control"
id="ccrd" placeholder="xxxx xxxxxx xxxxxx" name="ccrd">
                </div>
```

```
                <div id="ccrdmsg"></div>
            </div>
            <div class="row mb-2">
                <label class="col-md-2 col-form-label"
for="cexp"><strong>Credit card Expiry date:</strong></label>
                <div class="col-md-7">
                    <input type="date" class="form-control"
id="cexp" name="cexp">
                </div>
                <div id="cexpmsg"></div>
            </div>
            <br>
            <hr>
            <div class="row py-4">
                <div class="col-md-2"></div>
                <div class="col-md-5 px-5 justify-content-center
">
                    <button type="submit" class="btn btn-
primary">Submit</button>
                </div>
                <div class="col-md-5 px-5 justify-content-center">
                    <button type="reset" class="btn btn-
primary">Reset</button>
                </div>
            </div>
        </form>
    </main>
    <br><br><br>

    <footer class="footer bg-dark text-light text-center py-3
fixed-bottom">
        2023 Copyright bookings.com
    </footer>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstra
p.bundle.min.js"
        integrity="sha384-
ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
        crossorigin="anonymous"></script>

    <script>
        function Validation() {

            const fname = document.getElementById("fname").value;
            const fnameMsg = document.getElementById("fnamemsg");
            const lname = document.getElementById("lname").value;
            const lnameMsg = document.getElementById("lnamemsg");
```

```
            const phone = document.getElementById("phone").value;
            const phoneMsg = document.getElementById("phonemsg");
            const email = document.getElementById("email").value;
            const emailMsg = document.getElementById("emailmsg");
            const ccrd = document.getElementById("ccrd").value;
            const ccrdMsg = document.getElementById("ccrdmsg");
            const cexp = document.getElementById("cexp").value;
            const cexpMsg = document.getElementById("cexpmsg");

            let cfname, clname, cphone, cemail, cccrd, ccexp;

            if (fname.length === 0) {
                fnameMsg.className = 'col-md-3';
                fnameMsg.classList.add("text-danger");
                fnameMsg.innerHTML = "<em>You did not enter your
fname</em>";
                cfname = false;
            }
            else {
                fnameMsg.className = 'col-md-3';
                fnameMsg.classList.add("text-success");
                fnameMsg.innerHTML = "Valid";
                cfname = true;
            }

            if (lname.length === 0) {
                lnameMsg.className = 'col-md-3';
                lnameMsg.classList.add("text-danger");
                lnameMsg.innerHTML = "<em>You did not enter your
lname</em>";
                clname = false;
            }
            else {
                lnameMsg.className = 'col-md-3';
                lnameMsg.classList.add("text-success");
                lnameMsg.innerHTML = "Valid";
                clname = true;
            }

            if (phone.length === 0) {
                phoneMsg.className = 'col-md-3';
                phoneMsg.classList.add("text-danger");
                phoneMsg.innerHTML = "<em>You did not enter your
phone number</em>";
                cphone = false;
            }
            else if (phone.length != 10) {
                phoneMsg.className = 'col-md-3';
```

```
                phoneMsg.classList.add("text-danger");
                phoneMsg.innerHTML = "<em>Phone number should be
10 digits long</em>";
                cphone = false;
            }
            else {
                phoneMsg.className = 'col-md-3';
                phoneMsg.classList.add("text-success");
                phoneMsg.innerHTML = `Valid`;
                cphone = true;
            }

            if (email.length === 0 || email.indexOf('@') == -1) {
                emailMsg.className = 'col-md-3';
                emailMsg.classList.add("text-danger");
                emailMsg.innerHTML = "<em>Enter vaild email
address.</em>";
                cemail = false;
            }
            else {
                emailMsg.className = 'col-md-3';
                emailMsg.classList.add("text-success");
                emailMsg.innerHTML = "Valid";
                cemail = true;
            }

            if (ccrd.length === 0) {
                ccrdMsg.className = 'col-md-3';
                ccrdMsg.classList.add("text-danger");
                ccrdMsg.innerHTML = "<em>Enter a vaild credit card
number.</em>";
                cccrd = false;
            }
            else {
                ccrdMsg.className = 'col-md-3';
                ccrdMsg.classList.add("text-success");
                ccrdMsg.innerHTML = "Valid";
                cccrd = true;
            }

            if (cexp === "" || isNaN(Date.parse(cexp))) {
                cexpMsg.className = 'col-md-3';
                cexpMsg.classList.add("text-danger");
                cexpMsg.innerHTML = "<em>Enter a vaild
date.</em>";
                ccexp = false;
            }
            else {
```

```
                cexpMsg.className = 'col-md-3';
                cexpMsg.classList.add("text-success");
                cexpMsg.innerHTML = "Valid";
                ccexp = true;
            }

            if (cfname && clname && cemail && cphone && cccrd &&
ccexp) {
                return true;
            }
            else {
                event.preventDefault();
                return false;
            }

        }
    </script>
</body>

</html>
```

3.2 Explanation:

The high-level breakdown of the code used to design the form webpage is explained below:

1. `<!DOCTYPE html>`: This is the document type declaration, which tells the web browser that this is an HTML5 document.

2. `<html lang="en">`: This tag defines the root of the HTML document and specifies that the document is in English ("en").

3. `<head>`: This section contains metadata and links to external resources. Here's what's included in the head section:

- **`<title>`:** Specifies the title of the web page, which is displayed in the browser's title bar or tab.
- **`<meta charset="utf-8">`:** Sets the character encoding to UTF-8, ensuring that special characters are displayed correctly.
- **`<meta name="sathiyanarayanan" content="SIT772">`:** This meta tag contains custom information but should typically be used for specifying the author or other relevant metadata.
- **`<meta name="description" content="Filling a database">`:** Provides a brief description of the web page.
- **`<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">`:** Specifies the viewport settings for responsive design.
- **`<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" ...>`:** Links to an external CSS stylesheet (Bootstrap) to apply styles to the web page.

4. `<body>`: This is the main content of the web page. Here's what's included in the body section:

- **`<header>`:** Defines a header section. It includes a "Bookings.com" title.

- **`<main>`:** Contains the main content of the web page. It includes a form for collecting customer details.
- **`<form>`:** This form element specifies the action attribute ("/submit") to which the form data will be sent when submitted. It also includes an `onsubmit` attribute calling a JavaScript function named "Validation()" to validate the form before submission.
    - Inside the form, there are several form fields for collecting customer information, such as first name, last name, phone number, email, credit card number, and credit card expiry date.
    - Finally, there are "Submit" and "Reset" buttons for form submission and resetting form fields.

5. `<footer>`: At the end of the page, there's a footer section with copyright information for "bookings.com."

6. `<script>`: JavaScript code is included at the end of the HTML document. It defines a function named "Validation()" that is called when the form is submitted. This function validates the form fields and displays validation messages based on user input. If all the validations pass, the form is submitted; otherwise, it prevents form submission and displays error messages.

- The function starts by declaring and initializing variables for various form fields (e.g**., `fname`, `lname`, `phone`, `email`, `ccrd`, and `cexp`**) by retrieving their values from the HTML document.
- It also declares variables for corresponding error message elements (e.g**., `fnameMsg`, `lnameMsg`, `phoneMsg`**, etc.) by selecting them from the HTML document using `document.getElementById()`.
- Several boolean variables (e.g., **`cfname`, `clname`, `cphone`,** etc.) are declared to track whether the validation for each field passes or fails.
- The function then performs **validation for each form field**. If a field's input is deemed invalid, it:
    - Sets the error message element's class to style it (e.g., "text-danger" for red text).
    - Updates the error message to provide feedback to the user.
    - Sets the corresponding boolean variable to `false` to indicate validation failure.
- If a field's input is valid, it:
    - Sets the error message element's class to style it (e.g., "text-success" for green text).
    - Updates the error message to indicate that the input is valid.
    - Sets the corresponding boolean variable to `true` to indicate validation success.
- After validating all fields, the function checks the boolean variables (`cfname`, `clname`, `cphone`, etc.) to see if all the validations have passed.
- If all validations are successful (i.e., all boolean variables are `true`), the function returns `true`, allowing the form to be submitted.
- If any of the validations fail (i.e., any boolean variable is `false`), the function calls `event.preventDefault()` to prevent the form from being submitted and returns `false`.

This JavaScript function validates the form fields for the customer's first name, last name, phone number, email, credit card number, and credit card expiry date. It provides user-friendly error messages and prevents the form submission if any of the validations fail. This client-side validation helps ensure that the user provides valid data before the form is submitted to the server.

4. Summary:

In summary, this HTML document creates a web page for capturing customer details, validates the input data using JavaScript, and provides a responsive design using Bootstrap styles. The form data is intended to be submitted to the "/submit" endpoint, handled by a server-side script to insert the customer's data into MySQL database.

5. Explanation video link:

https://youtu.be/X5mRS33QxJY