Supporting Documentation - Online Learning Platform (Word Count: 995)

W3C Compliance

The website has been validated using the W3C Markup Validation Service and CSS Validation Service. All HTML templates were checked by validating the rendered HTML output for each page. All CSS files were validated using the W3C CSS Validation Service. No errors were detected in either validation process.

Feature Explanation: Contact Form System

I've chosen to explain the contact form system as it demonstrates both client-side and server-side implementation with database integration.

Client-Side Components

HTML Structure (contact.ejs)

- Form built with semantic HTML elements
- Input fields with appropriate labels and placeholders
- Error message spans for validation feedback

Form Validation (main.js)

- JavaScript validation occurs on form submission
- Validates name (required), email (pattern matching), subject (required), and message (10+ characters)
- Displays error messages without submitting when validation fails

- Clears previous error messages on resubmission

CSS Styling

- Provides visual feedback through error/success messages
- Responsive design for different screen sizes

Server-Side Components

Route Handler (index.mjs)

- Express router with two routes:
  - GET: Renders the contact form template
  - POST: Processes form submissions
- The POST handler:
  - Extracts form data using `req.body`
  - Performs server-side validation
  - Inserts valid data into the database
  - Renders the contact page with feedback messages

Database Integration

- Stores submissions in the SQLite contacts table
- Uses prepared statements to prevent SQL injection
- Example: `await db.run('INSERT INTO contacts (name, email, subject, message) VALUES (?, ?, ?, ?)', [name, email, subject, message])`

Data Flow

1. User completes the form

2. Client-side JavaScript validates inputs
3. Valid data is sent to server via POST
4. Server validates data again
5. Data is stored in the database
6. Server returns success/error feedback to user

This implementation ensures data integrity through validation at both client and server levels.

Legal, Ethical, and Accessibility Considerations

Legal Considerations

Data Protection

- Contact form collects only necessary information
- No personal data shared with third parties
- Data stored securely in the database

Image Usage

- All images sourced from royalty-free services (Pexels, Unsplash)
- Image credits provided in site footer
- Currently using placeholder references for development

Copyright

- All content is original and created specifically for this platform

Accessibility Considerations

Semantic HTML

- Uses semantic elements throughout (header, nav, main, section, footer)
- Improves screen reader compatibility and SEO

Alt Text for Images

- All images include descriptive alt text
- Enables visually impaired users to understand image content

Color Contrast

- Color scheme maintains sufficient contrast ratios
- Primary colors used consistently for interactive elements
- Helps users with visual impairments or color blindness

Form Accessibility

- Form fields have associated labels
- Clear error messages
- Required fields clearly indicated

Responsive Design

- Website adapts to different screen sizes
- Content accessible on mobile devices without horizontal scrolling

Security Considerations

Input Validation

- Validation at both client and server sides
- Prevents processing or storing malicious data

SQL Injection Prevention

- Prepared statements for all database queries
- User inputs never directly concatenated into SQL queries

XSS Prevention

- EJS templating engine escapes output by default
- HTML special characters in user inputs converted to entity equivalents

Error Handling

- Detailed errors logged server-side but not exposed to users
- Generic error messages displayed to prevent information leakage

Database Structure (ERD)

```
+---------------+       +----------------+       +---------------+
| courses       |       | instructors    |       | sessions      |
+---------------+       +----------------+       +---------------+
| id            |       | id             |       | id            |
| name          |       | name           |       | title         |
| description   |       | bio            |       | description   |
| duration      |       | subjects       |       | date          |
| instructor_id +-------+ email          |       | time          |
+---------------+       +----------------+       | course_id     |
                                                 +---------------+
+---------------+
| contacts      |
+---------------+
| id            |
| name          |
| email         |
| subject       |
| message       |
| submitted_at  |
+---------------+
```

Table Relationships:

- **courses**: Contains course information with foreign key to instructors
- **instructors**: Stores instructor details
- **sessions**: Stores live sessions with foreign key to courses
- **contacts**: Stores contact form submissions (standalone table)

This database structure maintains relational integrity between courses, instructors, and sessions while supporting the platform's core requirements.