Web and Internet Programming Lab Report: HTML Table and Form Design Implementation

Student ID: 2022100000027

1 Objective(s)

- To design and develop a comprehensive HTML form with multiple input types including text fields, radio buttons, checkboxes, dropdown menus, file uploads, and textarea elements for collecting structured user data
- To implement form validation techniques using HTML5 validation attributes such as required, pattern, maxlength, and input type specifications to ensure data integrity and user input accuracy
- To create structured data presentation using HTML tables with proper use of table elements, colspan/rowspan attributes, and styling to display timetable/schedule information in an organized format
- To develop user-friendly web interfaces incorporating visual elements, proper spacing, alignment, and interactive features like submit/reset buttons to enhance user experience and form usability

2 Problem Analysis/Background Theory

In this lab, we explored fundamental HTML concepts for creating interactive web pages through forms and structured data presentation using tables. HTML forms enable comprehensive user data collection through various input elements including `<input>` with different types (text, email, radio, checkbox, file), `<textarea>` for multi-line text, `<select>` for dropdown menus, and `<label>` for accessibility and user experience.

Form validation was implemented using HTML5 attributes such as `required`, `pattern`, and `maxlength` to ensure data integrity before submission. HTML tables were utilized to present structured schedule information using `<table>`, `<tr>` for rows, `<th>` for headers, and `<td>` for data cells, with `colspan` and `rowspan` attributes to create complex grid layouts for timetable presentation.

Understanding these core HTML elements and their attributes is essential for creating well-structured, accessible, and user-interactive web applications that can effectively collect user input and display organized information.

3 Tools/Environment Used

Code Editor:

- Visual Studio Code (VS Code) - Free, lightweight, and feature-rich code editor with extensive HTML, CSS, and JavaScript support

VS Code Extensions:

- Live Server - Creates a local development server with live reload functionality for real-time preview of HTML changes

- Error Lens - Enhances error visualization by displaying diagnostic messages inline, making syntax errors and warnings more prominent
- Prettier - Automatic code formatter that ensures consistent code styling and formatting across HTML, CSS, and JavaScript files
- GitHub Copilot - AI-powered coding assistant that provides intelligent code suggestions and completions for faster development
- Auto Rename Tag - Automatically renames paired HTML tags when editing
- HTML End Tag Labels - Automatically labels HTML end tags
- Bracket Pair Colorizer - Color-codes matching brackets for better code readability

Web Browser:

- Microsoft Edge - Primary browser for testing and debugging with built-in Developer Tools (DevTools) including Elements panel, Console, Network inspector, and Performance analyzer

Operating System:

- Windows 11 Pro - Professional edition providing enhanced security features, business-grade tools, and optimized performance for development workflows

This development environment provides a comprehensive setup for HTML web development with modern tools that enhance productivity, code quality, and debugging capabilities suitable for academic lab work and professional web development projects.

4 Code Implementation

4.1 Routine Table Implementation (routine.html)

The `routine.html` file demonstrates advanced HTML table implementation for presenting a weekly class schedule using complex table structure with `colspan` and `rowspan` attributes. The implementation creates a grid-based timetable where the first row contains time slot headers from **08:00 to 20:00** in 30-minute intervals, utilizing `rowspan="2"` attributes to create proper table structure. It employs `<th>` elements for both time headers and day labels (SUN through SAT), while `<td>` elements contain course information including course codes (CSE472.3, CSE361.2, etc.), time ranges (16:30–18:30), and room locations (Location: SEU609, Location: SEU527) using symbols or text for visual enhancement.

The implementation uses `colspan` attributes to span course entries across multiple time slots, accurately representing the duration of each class session, while empty cells represent free periods in the schedule. The table includes proper border styling (`border="1"`), center alignment (`align="center"`), and cell padding (`cellpadding="5"`) for improved visual presentation and readability, creating an organized and functional timetable that effectively communicates class scheduling information in a structured tabular format.

```html
<!-- HTML code would go here -->
<!DOCTYPE html>
<html>
<head>
    <title>Class Routine</title>
</head>
<body>
    <table border="1" align="center" cellpadding="5">
        <!-- Table content here -->
    </table>
</body>
</html>
```

HTML Code Snippet 1: Summer 2025 Routine

4.2 Form Implementation (form.html)

The `form.html` file implements a comprehensive Student Admission Form using semantic HTML5 structure and modern form validation techniques. The implementation begins with a proper document structure including `<!DOCTYPE>` declaration, meta tags for character encoding and viewport settings, and a centered header section featuring the university logo and form title.

The form utilizes a table-based layout for organizing form fields into logical sections, starting with Personal Information that includes text inputs for first name, last name, etc., with `required` validation attributes and placeholder text for user guidance. The form incorporates diverse input types including email fields with built-in email validation, radio buttons for gender selection, checkboxes for multiple selections, dropdown menus using `<select>` and `<option>` elements, file upload capabilities, `<textarea>` for extended text input, and specialized inputs like date pickers and number fields.

Each form control is properly associated with descriptive labels using the `for` attribute, ensuring accessibility compliance, while validation is enforced through HTML5 attributes such as `required`, `pattern`, `maxlength`, and `minlength` to maintain data integrity before submission.

```
<!-- HTML code would go here -->
<!DOCTYPE html>
<html>
<head>
    <title>Student Admission Form</title>
</head>
<body>
    <form action="#" method="post">
        <!-- Form content here -->
    </form>
</body>
</html>
```

HTML Code Snippet 2: Student Admission Form

5 Output/Demonstration

5.1 Routine Table Output Display (routine.html)

The `routine.html` file produces a well-structured weekly class schedule table that effectively demonstrates advanced HTML table implementation with complex cell spanning and organized data presentation. The output displays a comprehensive timetable grid featuring time slot headers across the top row (08:00 to 20:00 in 30-minute intervals) and day labels (SUN through SAT) in the leftmost column, creating a clear temporal framework for schedule visualization.

The table successfully renders course information including course codes (CSE472.3, CSE361.2), specific time ranges (16:30–18:30, 10:00–11:20), and room locations (Location: SEU609, SEU527, SEU613), using visual markers for enhanced readability. The implementation demonstrates proper use of `colspan` attributes to span course entries across multiple time slots, accurately representing class durations, while empty cells effectively indicate free periods in the weekly schedule.

The table output features clean borders, center alignment, appropriate cell padding, and an organized structure that makes it easy to identify class timings, locations, and course conflicts, providing a functional and visually appealing schedule presentation suitable for academic use.

[Image: Sample Output of Routine Table in routine.html]

Figure 1: Sample Output of Routine Table in routine.html

5.2 Form Output Display (form.html)

The `form.html` file renders a comprehensive Student Admission Form interface that demonstrates effective HTML form implementation with proper layout and validation features. The output displays a professionally structured admission form with the university header section containing the institutional logo and form title, followed by organized input fields arranged in a table-based layout for optimal visual presentation.

The form showcases various input types including text fields for personal information (first name, last name), email input with built-in validation, radio button groups for gender selection, checkbox options for multiple selections, dropdown menus for categorical choices, file upload buttons for document submission, and `<textarea>` fields for extended text input.

The form implements real-time HTML5 validation feedback, displaying error messages for required fields, and invalid email formats, while properly formatted labels and placeholder text guide user interaction. The output demonstrates responsive form design with clear visual hierarchy, appropriate spacing, and functional submit/reset buttons that provide immediate user feedback upon interaction.

[Image: Sample Output of Student Admission Form in form.html (half)]

Figure 2: Sample Output of Student Admission Form in form.html (half)

[Image: Sample Output of Student Admission Form in form.html (remaining part)]

Figure 3: Sample Output of Student Admission Form in form.html (remaining part)

6 Discussion & Explanation

This lab solidified my grasp of core HTML concepts by letting me translate abstract tags into tangible, working pages. Building my weekly routine table forced me to think in grids: figuring out exactly where a cell should stretch with `rowspan` or `colspan` felt like solving a miniature puzzle. The first few attempts left gaps and misalignments, but repeated tweaking taught me how the browser's table model flows and how header cells (`<th>`) affect semantics and styling.

Designing the admission form was equally eye-opening. Experimenting with different `<input>` types—text, email, date, file—illustrated how much validation can be delegated to the browser before JavaScript ever runs. I especially appreciated how the `pattern` attribute accepts regular expressions: one concise rule in the markup saved dozens of lines of potential script. Radio buttons and checkboxes highlighted the importance of grouping related controls with `<fieldset>` and `<legend>` for both accessibility and easier styling. Adding a file upload control also emphasized security concerns; it made me think about server-side validation and allowed file types even though this lab stayed strictly on the client side.

Finally, styling both pages underscored the impact of small visual cues on usability. Center-aligned headings, judicious whitespace, and consistency transformed plain markup into an interface that feels professional rather than academic. Overall, the experience moved me from memorizing tag syntax to appreciating how each element collaborates to produce structured, interactive, and accessible content.

7 Conclusion

Through this lab, I significantly improved my ability to create structured HTML content and gained practical experience in building functional web pages. Working with the admission form taught me how different input types serve specific purposes—from simple text fields to complex validation patterns using regular expressions. I discovered that HTML5's built-in validation attributes like `required`, `pattern`, and `maxlength` can handle much of the data integrity work without requiring additional scripting, making forms both user-friendly and robust.

The routine table implementation challenged my understanding of HTML table structure, particularly with `colspan` and `rowspan` attributes. Initially, creating the weekly schedule grid felt overwhelming, but through trial and error, I learned how these spanning attributes work together to create complex layouts that accurately represent time-based data. The process of mapping course information across multiple time slots helped me appreciate how tables can transform raw data into visually organized, readable formats.

Most importantly, this lab reinforced that semantic HTML is the foundation of effective web development. By using proper labels, `<fieldset>`, table headers, and meaningful element structure, both pages became more accessible and easier to style. I gained confidence in designing simple yet functional web pages that incorporate interactive forms, structured data presentation, and user-friendly interfaces. The experience moved me from simply memorizing HTML tags to understanding how they work together to create meaningful, interactive web content that serves real-world purposes in academic and professional contexts.

* * *

Appendix

A.1 Implementation Challenges and Solutions

During the development of the student admission form (`form.html`), one significant challenge encountered was achieving proper alignment and visual organization of form fields using only HTML without CSS styling. HTML form elements by default do not provide sophisticated layout control, making it difficult to create a professional-looking, well-aligned form interface.

Problem Identified: Form fields appeared misaligned and inconsistent when using standard HTML form structure with `<div>` elements. Input fields of different types (text inputs, select dropdowns, textareas) have varying default widths and heights, creating an unorganized visual appearance that negatively impacts user experience. To address this alignment challenge, HTML tables were strategically utilized as a layout mechanism for each form section. Tables were created using `<table>`, `<tr>`, and `<td>` elements.

This table-based approach provided precise control over form field positioning and ensured consistent alignment across different input types. Each section (Personal Information, Address Information, Academic Information, etc.) was contained within its own table structure, creating organized visual groupings that enhance form readability and user navigation.

While modern web development typically uses CSS Flexbox or Grid for layout purposes, this implementation demonstrates effective use of HTML table semantics for structural organization when CSS is not available, showcasing fundamental HTML layout techniques that remain functional across all browser environments.