SMART CLASSROOM & TIMETABLE SCHEDULER

1. Overview

- In modern educational institutions, the process of creating class timetables and managing classroom allocations remains one of the most challenging administrative tasks. With multiple courses, limited classroom resources, and diverse faculty schedules, manually preparing a timetable often leads to inefficiency, human errors, and scheduling conflicts.

- The *Smart Classroom & Timetable Scheduler* project aims to eliminate these challenges through the integration of Artificial Intelligence (AI) and automation. The system is designed to simplify and optimize timetable creation, improve classroom management, and ensure the effective use of resources in colleges and universities.

2. Motivation

- Traditional methods of timetable management rely heavily on manual work and static spreadsheets, which require considerable time and effort from faculty coordinators. Often, these methods lead to overlapping lectures, unassigned classrooms, and uneven faculty workloads.

- This inspired our team —**Ponakal Bhuvan Teja**, **Ajaz** and **Sachin Mamadapur** — to create a smart solution that can handle this process more efficiently. Our motivation was to develop an automated system capable of generating accurate and conflict-free timetables, while also providing a foundation for smart classroom functionalities.

3. Objective

- ➢The main objectives of the *Smart Classroom & Timetable Scheduler* project are:
- ➢To automate the timetable generation process using intelligent algorithms.
- ➢Allocate classrooms and faculty based on availability, avoiding schedule conflicts.
- ➢To integrate smart classroom features such as:
    - ✓IoT-enabled attendance systems
    - ✓Smart board scheduling
    - ✓Automated alerts for schedule changes
- ➢To provide a user-friendly interface for administrators, faculty, and students.
- ➢To ensure optimal resource utilization within the institution.

4.Implementation and Working

- The Smart Classroom & Timetable Scheduler is implemented using **web technologies (HTML, CSS, and JavaScript)**.

- It provides both **manual** and **Excel-based** input methods to create optimized class timetables automatically. The system dynamically allocates lectures and lab sessions without overlapping teacher schedules.

❑ Front-End Design (HTML + CSS):

- ➢ HTML defines the structure of the user interface, including input forms, tables, and output displays.
- ➢ CSS is used for styling — providing a clean, modern, and user-friendly layout with proper alignment an color coding (e.g., yellow cells for lab sessions).

❑ Logic and Functionality (JavaScript):

- ➢ JavaScript handles all core logic such as reading Excel files, validating inputs, generating random but non-conflicting schedules, and displaying output dynamically.
- ➢ The code uses the **XLSX.js** library to read Excel sheets and extract teacher–subject–credit data.
- ➢ Each subject's lectures and labs are assigned automatically across the weekdays (Monday–Friday).
- ➢ It ensures that no teacher or class is double-booked in the same time slot.

❑ Data Flow:

- ➢ **Input:** Class name, teacher name, subject, credits, and lab info (manual or Excel).
- ➢ **Processing:** Schedule generation based on available time slots and credit hours.
- ➢ **Output:** A color-coded timetable displayed in a structured table format.

➢ Working Steps

1.User enters **class name** and uploads teacher–subject data (via Excel or manual form).

2.The system reads and validates the data.

3.Timetable is automatically generated for 5 days (Monday–Friday) with 6 periods each day.

4.Lecture and **Lab** sessions are placed intelligently without clashes.

1.The output timetable is displayed in tabular form for easy viewing and verification.

➢ Technologies Used

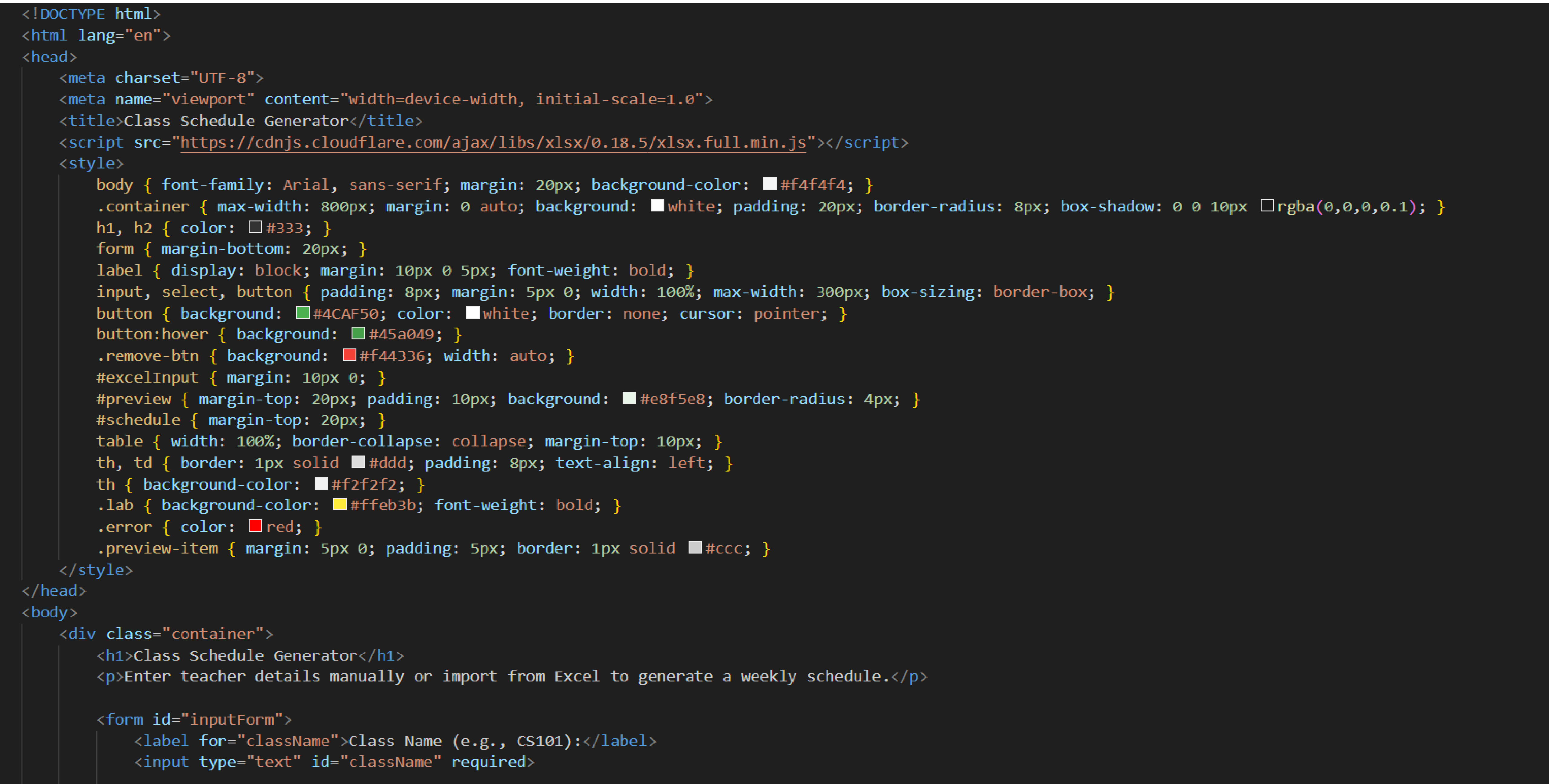
•**HTML5** – User interface and page structure

•**CSS3** – Styling and layout

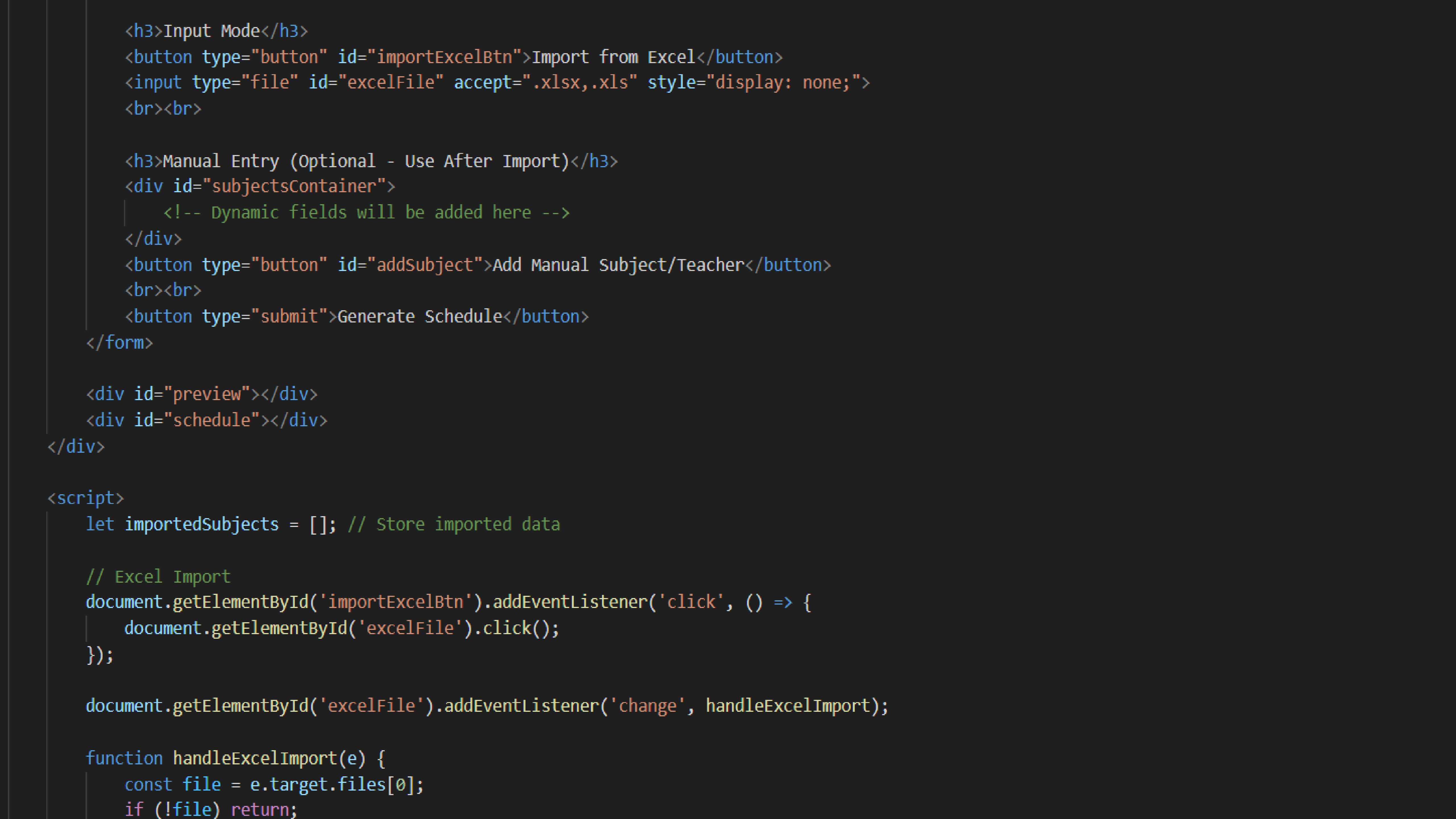
•**JavaScript (ES6)** – Logic, validation, and dynamic scheduling

•**XLSX.js** – Excel file processing library

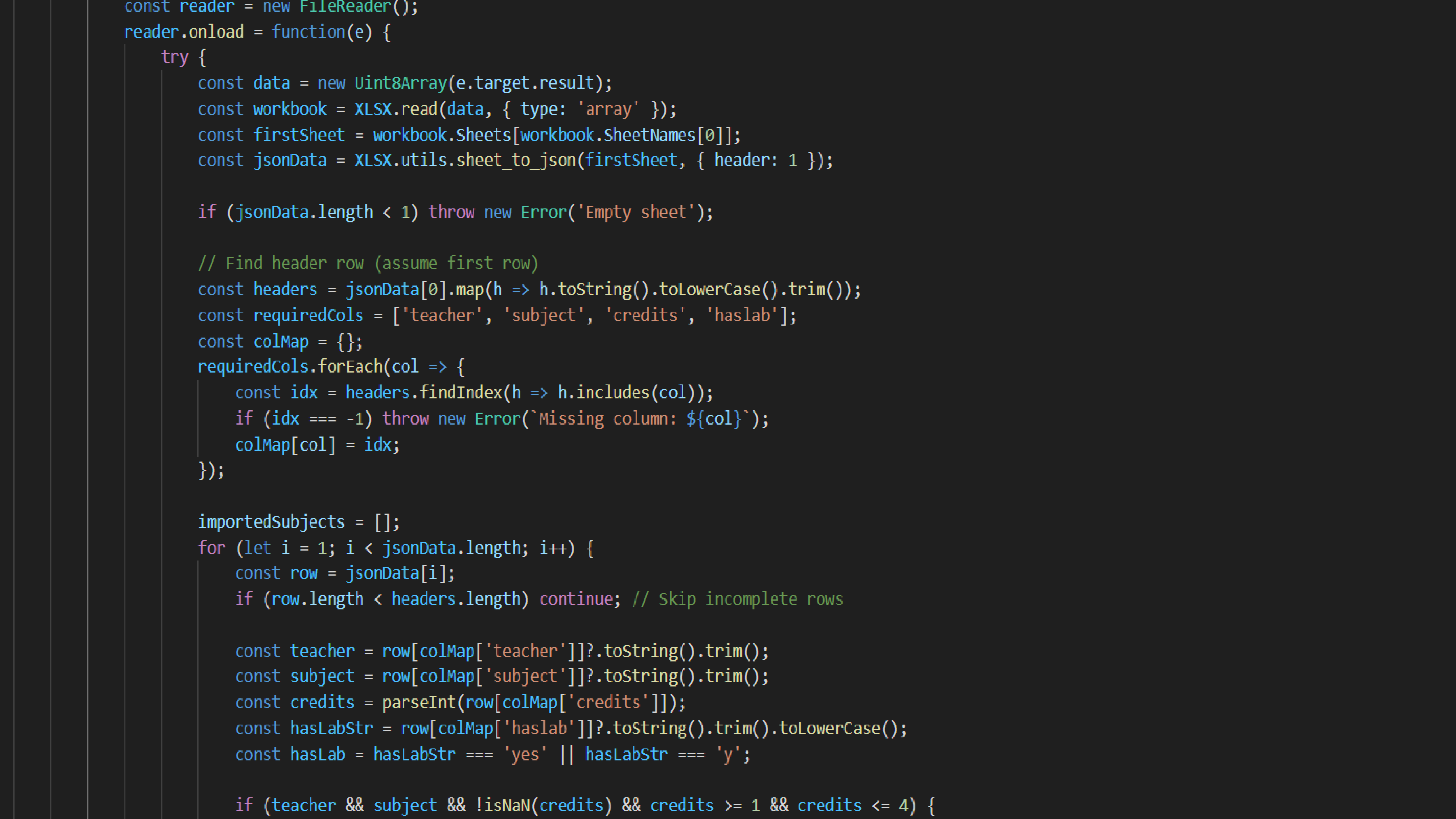
5.Source code



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Class Schedule Generator</title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; background-color: #f4f4f4; }
        .container { max-width: 800px; margin: 0 auto; background: white; padding: 20px; border-radius: 8px; box-shadow: 0 0 10px rgba(0,0,0,0.1); }
        h1, h2 { color: #333; }
        form { margin-bottom: 20px; }
        label { display: block; margin: 10px 0 5px; font-weight: bold; }
        input, select, button { padding: 8px; margin: 5px 0; width: 100%; max-width: 300px; box-sizing: border-box; }
        button { background: #4CAF50; color: white; border: none; cursor: pointer; }
        button:hover { background: #45a049; }
        .remove-btn { background: #f44336; width: auto; }
        #excelInput { margin: 10px 0; }
        #preview { margin-top: 20px; padding: 10px; background: #e8f5e8; border-radius: 4px; }
        #schedule { margin-top: 20px; }
        table { width: 100%; border-collapse: collapse; margin-top: 10px; }
        th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
        th { background-color: #f2f2f2; }
        .lab { background-color: #ffeb3b; font-weight: bold; }
        .error { color: red; }
        .preview-item { margin: 5px 0; padding: 5px; border: 1px solid #ccc; }
    </style>
</head>
<body>
    <div class="container">
        <h1>Class Schedule Generator</h1>
        <p>Enter teacher details manually or import from Excel to generate a weekly schedule.</p>

        <form id="inputForm">
            <label for="className">Class Name (e.g., CS101):</label>
            <input type="text" id="className" required>
```

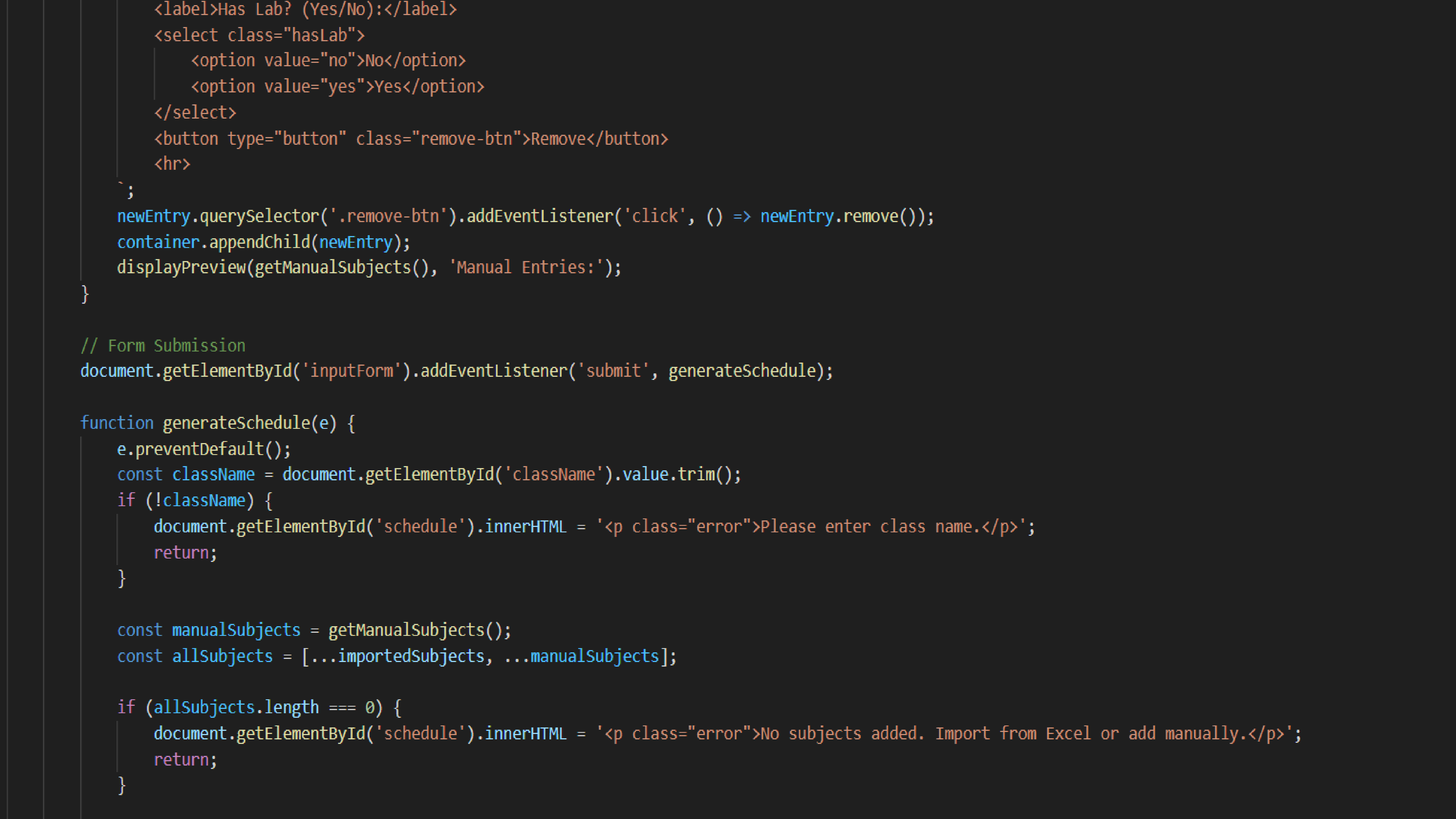


```
            <h3>Input Mode</h3>
            <button type="button" id="importExcelBtn">Import from Excel</button>
            <input type="file" id="excelFile" accept=".xlsx,.xls" style="display: none;">
            <br><br>

            <h3>Manual Entry (Optional - Use After Import)</h3>
            <div id="subjectsContainer">
                <!-- Dynamic fields will be added here -->
            </div>
            <button type="button" id="addSubject">Add Manual Subject/Teacher</button>
            <br><br>
            <button type="submit">Generate Schedule</button>
        </form>

        <div id="preview"></div>
        <div id="schedule"></div>
    </div>

    <script>
        let importedSubjects = []; // Store imported data

        // Excel Import
        document.getElementById('importExcelBtn').addEventListener('click', () => {
            document.getElementById('excelFile').click();
        });

        document.getElementById('excelFile').addEventListener('change', handleExcelImport);

        function handleExcelImport(e) {
            const file = e.target.files[0];
            if (!file) return;
```

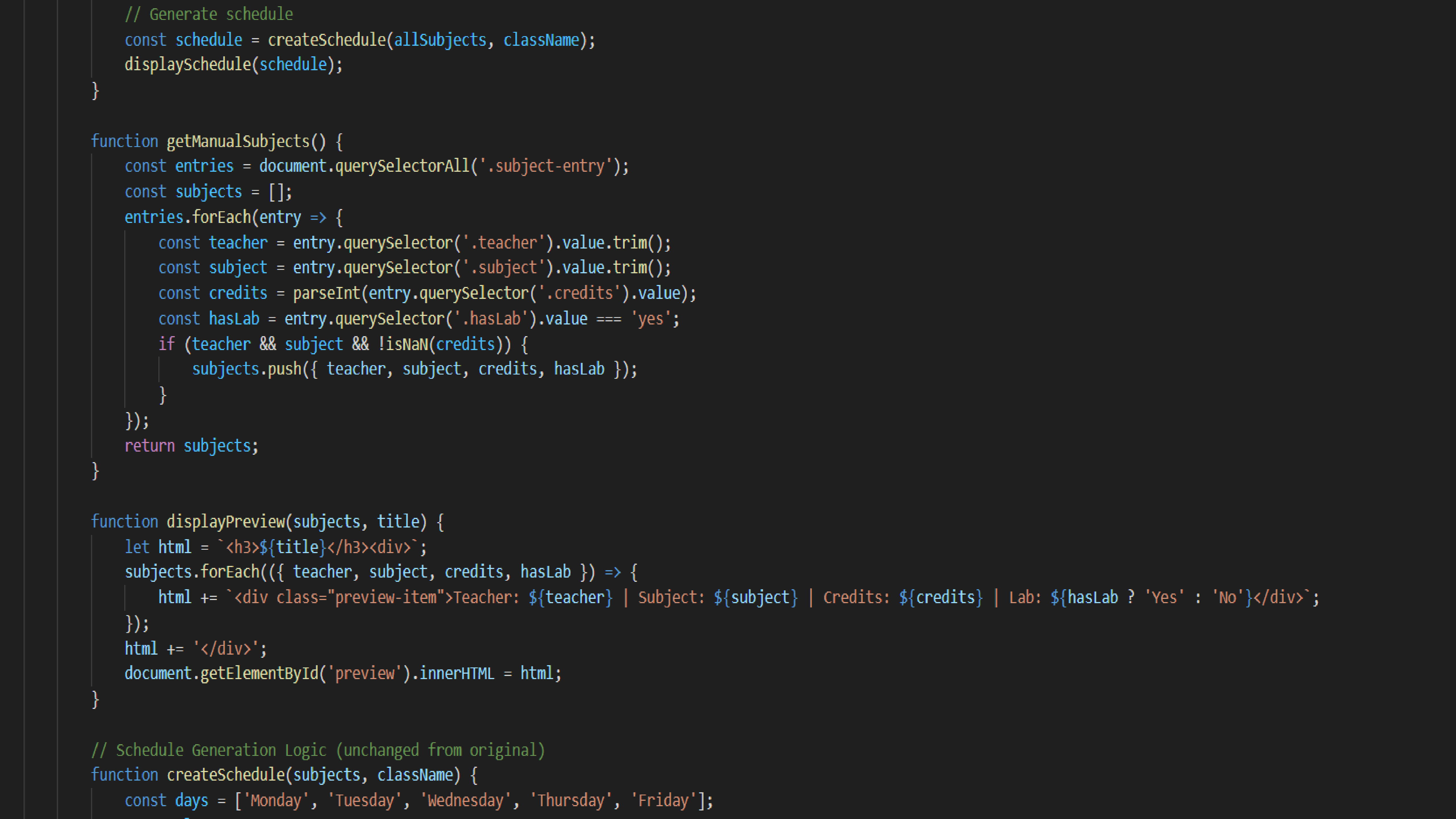


```
            const reader = new FileReader();
            reader.onload = function(e) {
                try {
                    const data = new Uint8Array(e.target.result);
                    const workbook = XLSX.read(data, { type: 'array' });
                    const firstSheet = workbook.Sheets[workbook.SheetNames[0]];
                    const jsonData = XLSX.utils.sheet_to_json(firstSheet, { header: 1 });

                    if (jsonData.length < 1) throw new Error('Empty sheet');

                    // Find header row (assume first row)
                    const headers = jsonData[0].map(h => h.toString().toLowerCase().trim());
                    const requiredCols = ['teacher', 'subject', 'credits', 'haslab'];
                    const colMap = {};
                    requiredCols.forEach(col => {
                        const idx = headers.findIndex(h => h.includes(col));
                        if (idx === -1) throw new Error(`Missing column: ${col}`);
                        colMap[col] = idx;
                    });

                    importedSubjects = [];
                    for (let i = 1; i < jsonData.length; i++) {
                        const row = jsonData[i];
                        if (row.length < headers.length) continue; // Skip incomplete rows

                        const teacher = row[colMap['teacher']]?.toString().trim();
                        const subject = row[colMap['subject']]?.toString().trim();
                        const credits = parseInt(row[colMap['credits']]);
                        const hasLabStr = row[colMap['haslab']]?.toString().trim().toLowerCase();
                        const hasLab = hasLabStr === 'yes' || hasLabStr === 'y';

                        if (teacher && subject && !isNaN(credits) && credits >= 1 && credits <= 4) {
```

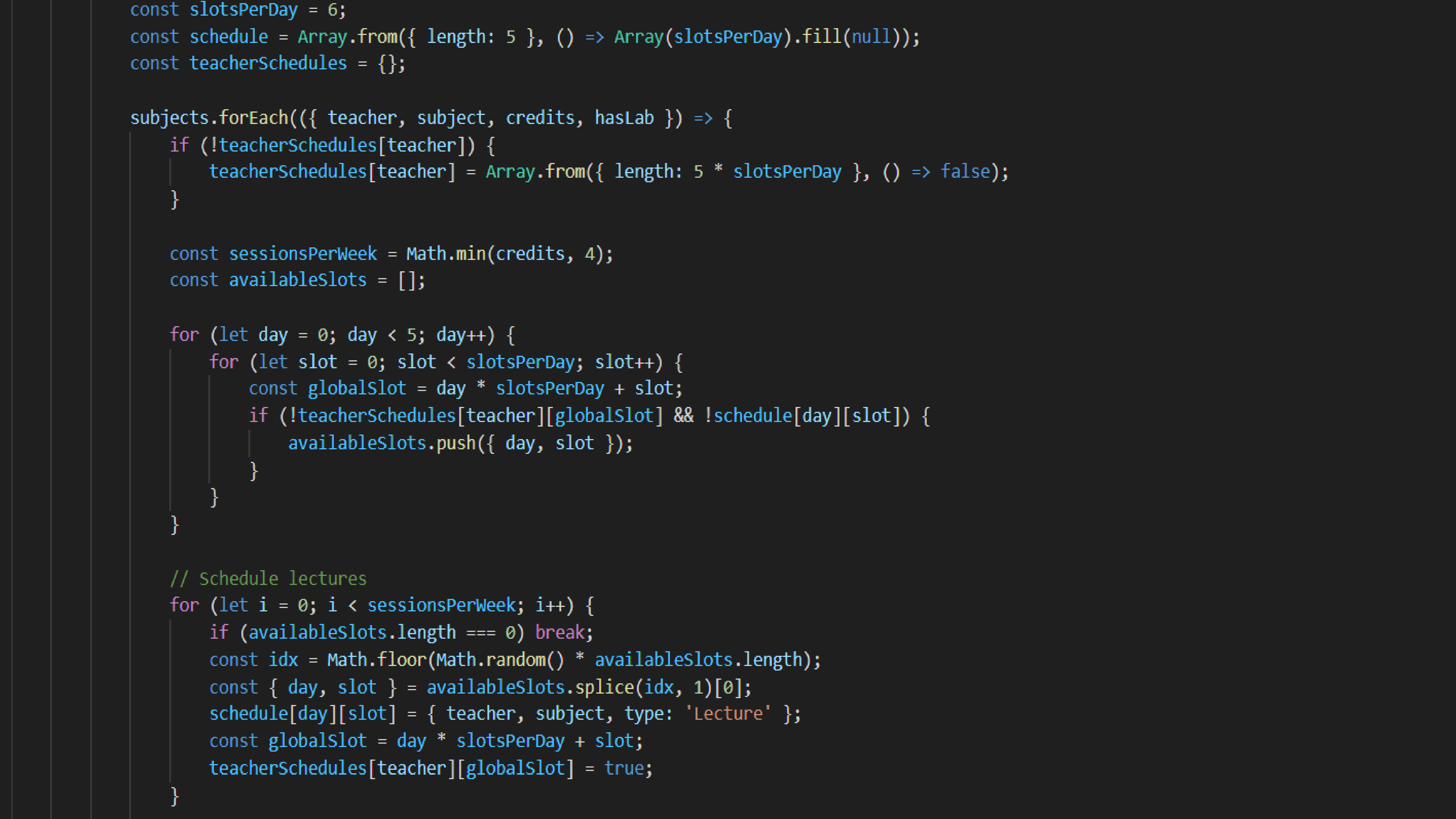
```
                    importedSubjects.push({ teacher, subject, credits, hasLab });
                }
            }

            if (importedSubjects.length === 0) throw new Error('No valid data found in Excel');

            // Clear manual form
            document.getElementById('subjectsContainer').innerHTML = '';
            displayPreview(importedSubjects, 'Imported from Excel:');
            e.target.value = ''; // Reset file input

        } catch (error) {
            document.getElementById('preview').innerHTML = `<p class="error">Error importing Excel: ${error.message}. Please check format.</p>`;
            importedSubjects = [];
        }
    };
    reader.readAsArrayBuffer(file);
}

// Manual Form Functions
document.getElementById('addSubject').addEventListener('click', addSubjectEntry);
function addSubjectEntry() {
    const container = document.getElementById('subjectsContainer');
    const newEntry = document.createElement('div');
    newEntry.className = 'subject-entry';
    newEntry.innerHTML = `
        <label>Teacher Name:</label>
        <input type="text" class="teacher" required>
        <label>Subject:</label>
        <input type="text" class="subject" required>
        <label>Credits (1-4):</label>
        <input type="number" class="credits" min="1" max="4" required>
```



```
                <label>Has Lab? (Yes/No):</label>
                <select class="hasLab">
                    <option value="no">No</option>
                    <option value="yes">Yes</option>
                </select>
                <button type="button" class="remove-btn">Remove</button>
                <hr>
            `;
            newEntry.querySelector('.remove-btn').addEventListener('click', () => newEntry.remove());
            container.appendChild(newEntry);
            displayPreview(getManualSubjects(), 'Manual Entries:');
        }

        // Form Submission
        document.getElementById('inputForm').addEventListener('submit', generateSchedule);

        function generateSchedule(e) {
            e.preventDefault();
            const className = document.getElementById('className').value.trim();
            if (!className) {
                document.getElementById('schedule').innerHTML = '<p class="error">Please enter class name.</p>';
                return;
            }

            const manualSubjects = getManualSubjects();
            const allSubjects = [...importedSubjects, ...manualSubjects];

            if (allSubjects.length === 0) {
                document.getElementById('schedule').innerHTML = '<p class="error">No subjects added. Import from Excel or add manually.</p>';
                return;
            }
```



```javascript
            // Generate schedule
            const schedule = createSchedule(allSubjects, className);
            displaySchedule(schedule);
        }

        function getManualSubjects() {
            const entries = document.querySelectorAll('.subject-entry');
            const subjects = [];
            entries.forEach(entry => {
                const teacher = entry.querySelector('.teacher').value.trim();
                const subject = entry.querySelector('.subject').value.trim();
                const credits = parseInt(entry.querySelector('.credits').value);
                const hasLab = entry.querySelector('.hasLab').value === 'yes';
                if (teacher && subject && !isNaN(credits)) {
                    subjects.push({ teacher, subject, credits, hasLab });
                }
            });
            return subjects;
        }

        function displayPreview(subjects, title) {
            let html = `<h3>${title}</h3><div>`;
            subjects.forEach(({ teacher, subject, credits, hasLab }) => {
                html += `<div class="preview-item">Teacher: ${teacher} | Subject: ${subject} | Credits: ${credits} | Lab: ${hasLab ? 'Yes' : 'No'}</div>`;
            });
            html += '</div>';
            document.getElementById('preview').innerHTML = html;
        }

        // Schedule Generation Logic (unchanged from original)
        function createSchedule(subjects, className) {
            const days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'];
```

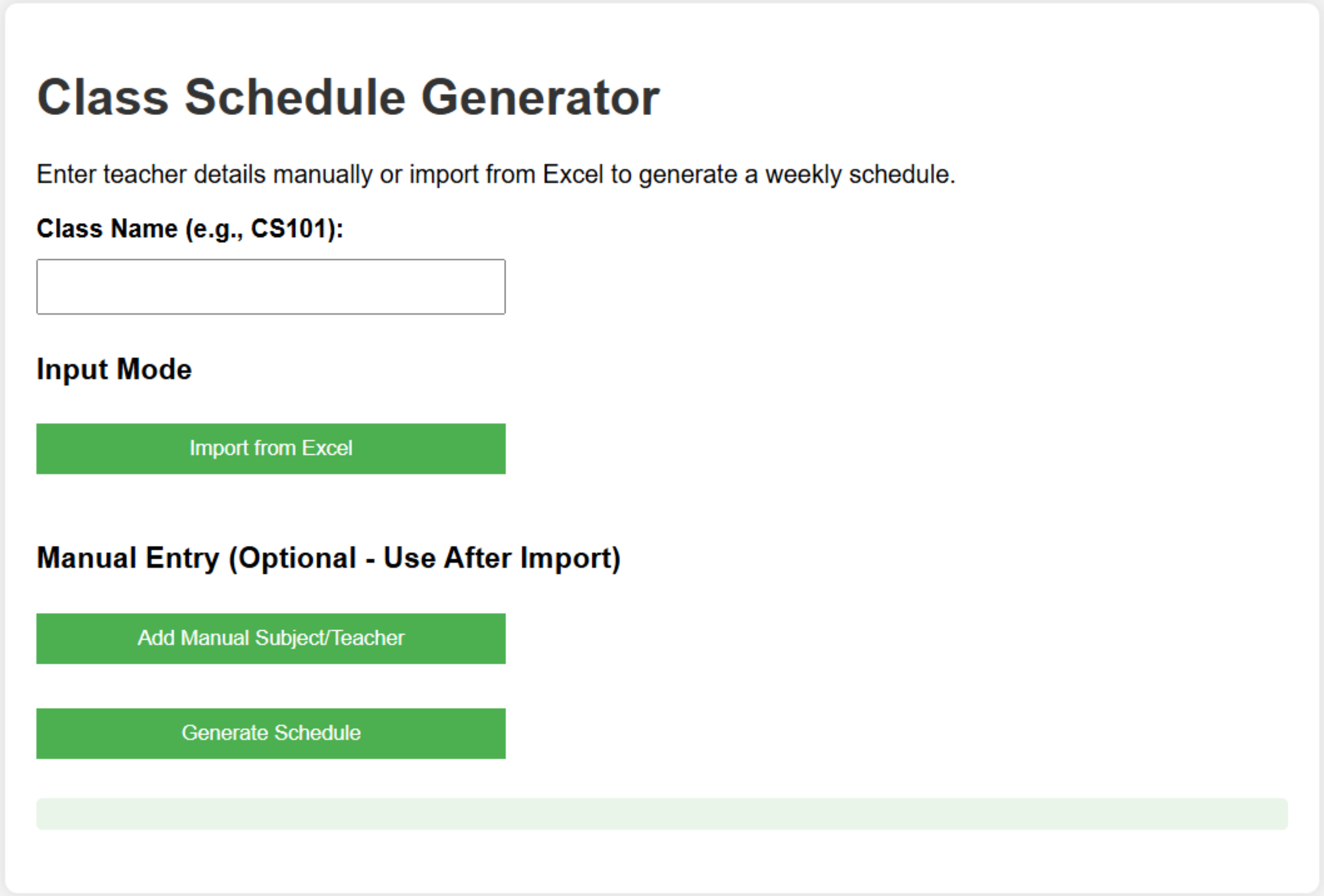


```
            const slotsPerDay = 6;
            const schedule = Array.from({ length: 5 }, () => Array(slotsPerDay).fill(null));
            const teacherSchedules = {};

            subjects.forEach(({ teacher, subject, credits, hasLab }) => {
                if (!teacherSchedules[teacher]) {
                    teacherSchedules[teacher] = Array.from({ length: 5 * slotsPerDay }, () => false);
                }

                const sessionsPerWeek = Math.min(credits, 4);
                const availableSlots = [];

                for (let day = 0; day < 5; day++) {
                    for (let slot = 0; slot < slotsPerDay; slot++) {
                        const globalSlot = day * slotsPerDay + slot;
                        if (!teacherSchedules[teacher][globalSlot] && !schedule[day][slot]) {
                            availableSlots.push({ day, slot });
                        }
                    }
                }

                // Schedule lectures
                for (let i = 0; i < sessionsPerWeek; i++) {
                    if (availableSlots.length === 0) break;
                    const idx = Math.floor(Math.random() * availableSlots.length);
                    const { day, slot } = availableSlots.splice(idx, 1)[0];
                    schedule[day][slot] = { teacher, subject, type: 'Lecture' };
                    const globalSlot = day * slotsPerDay + slot;
                    teacherSchedules[teacher][globalSlot] = true;
                }
```

```
        // Schedule lab
        if (hasLab && availableSlots.length >= 2) {
            const idx = Math.floor(Math.random() * availableSlots.length);
            let { day, slot } = availableSlots[idx];
            if (slot + 1 < slotsPerDay && !schedule[day][slot + 1]) {
                schedule[day][slot] = { teacher, subject, type: 'Lab (Part 1)' };
                schedule[day][slot + 1] = { teacher, subject, type: 'Lab (Part 2)' };
                const globalSlot1 = day * slotsPerDay + slot;
                const globalSlot2 = day * slotsPerDay + slot + 1;
                teacherSchedules[teacher][globalSlot1] = true;
                teacherSchedules[teacher][globalSlot2] = true;
                availableSlots.splice(idx, 1);
                const nextIdx = availableSlots.findIndex(s => s.day === day && s.slot === slot + 1);
                if (nextIdx > -1) availableSlots.splice(nextIdx, 1);
            }
        }
    });

    return { className, days, slotsPerDay, schedule };
}

// Display Schedule (unchanged)
function displaySchedule({ className, days, slotsPerDay, schedule }) {
    let html = `<h2>Weekly Schedule for ${className}</h2>
                <table>
                    <tr><th>Time Slot</th>`;
    days.forEach(day => html += `<th>${day}</th>`);
    html += '</tr>';

    const timeSlots = ['9:00-10:00', '10:00-11:00', '11:00-12:00', '12:00-1:00', '1:00-2:00', '2:00-3:00'];
```

```
            for (let slot = 0; slot < slotsPerDay; slot++) {
                html += `<tr><td>${timeSlots[slot]}</td>`;
                days.forEach((_, day) => {
                    const entry = schedule[day][slot];
                    if (entry) {
                        const className = entry.type.includes('Lab') ? 'lab' : '';
                        html += `<td class="${className}">${entry.subject}<br>(${entry.teacher}, ${entry.type})</td>`;
                    } else {
                        html += '<td>-</td>';
                    }
                });
                html += '</tr>';
            }

            html += '</table>';
            document.getElementById('schedule').innerHTML = html;
        }
    </script>
</body>
</html>
```

6.User Interface of Smart Timetable Website



Class Schedule Generator

Enter teacher details manually or import from Excel to generate a weekly schedule.

Class Name (e.g., CS101):

Input Mode

Manual Entry (Optional - Use After Import)

➢ Overview:

The user interface of the **Smart Timetable Scheduler** is designed to be **simple, responsive, and user-friendly**, allowing both students and faculty to interact easily.

The website interface includes the following key sections:

➤ **Input Section:** Users can either manually enter teacher and subject details or import data directly from an Excel file.

➤ **Preview Panel:** Displays all the entered or imported information clearly before generating the timetable.

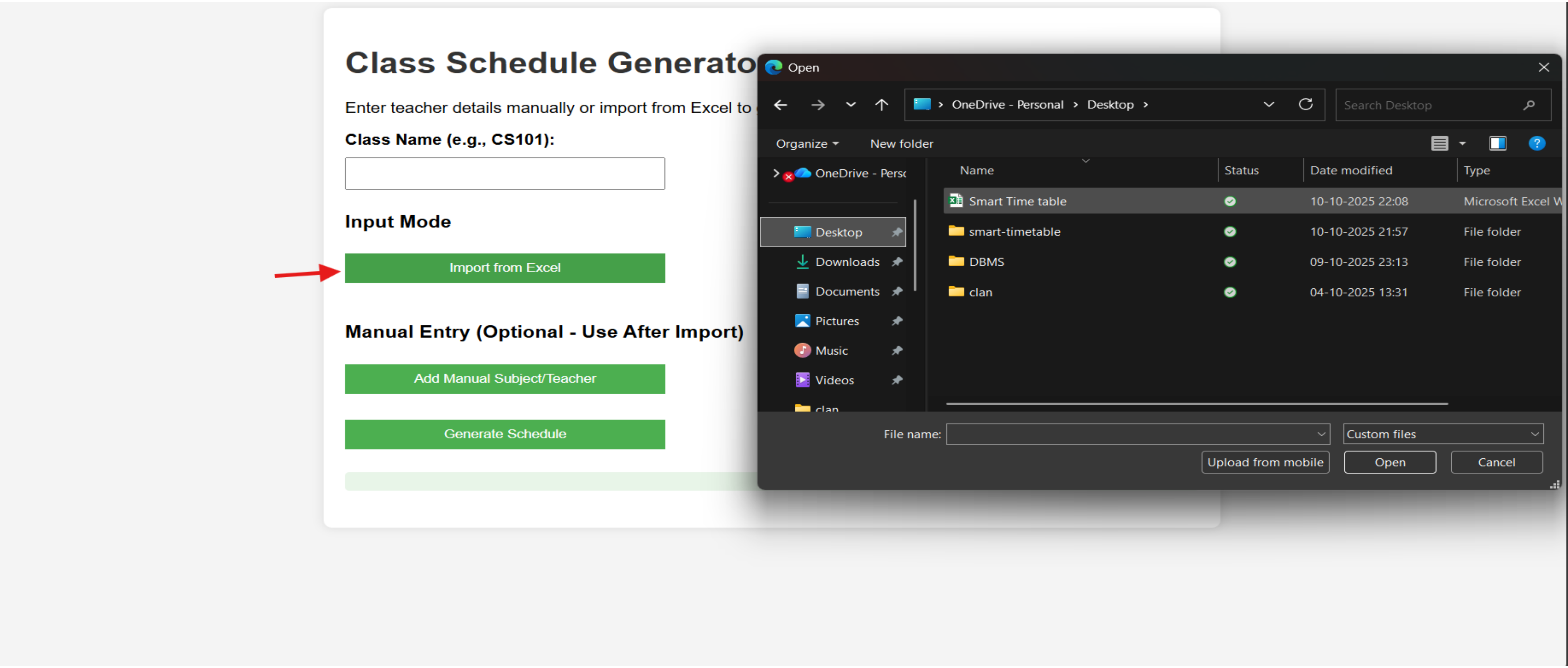
➤ **Generate Schedule Button:** Automatically creates a timetable with proper slot allocation for each subject and lab.

➤ **Output Display:** The final weekly schedule is shown in a **table format** with separate columns for days and time slots.

The clean layout, color-coded lab sessions, and error-free design make it easy for users to view and modify schedules efficiently.

7.Input Module

➢ Data Input through Excel File



Class Schedule Generato

Enter teacher details manually or import from Excel to

Class Name (e.g., CS101):

Input Mode

Import from Excel

Manual Entry (Optional - Use After Import)

Add Manual Subject/Teacher

Generate Schedule

Open

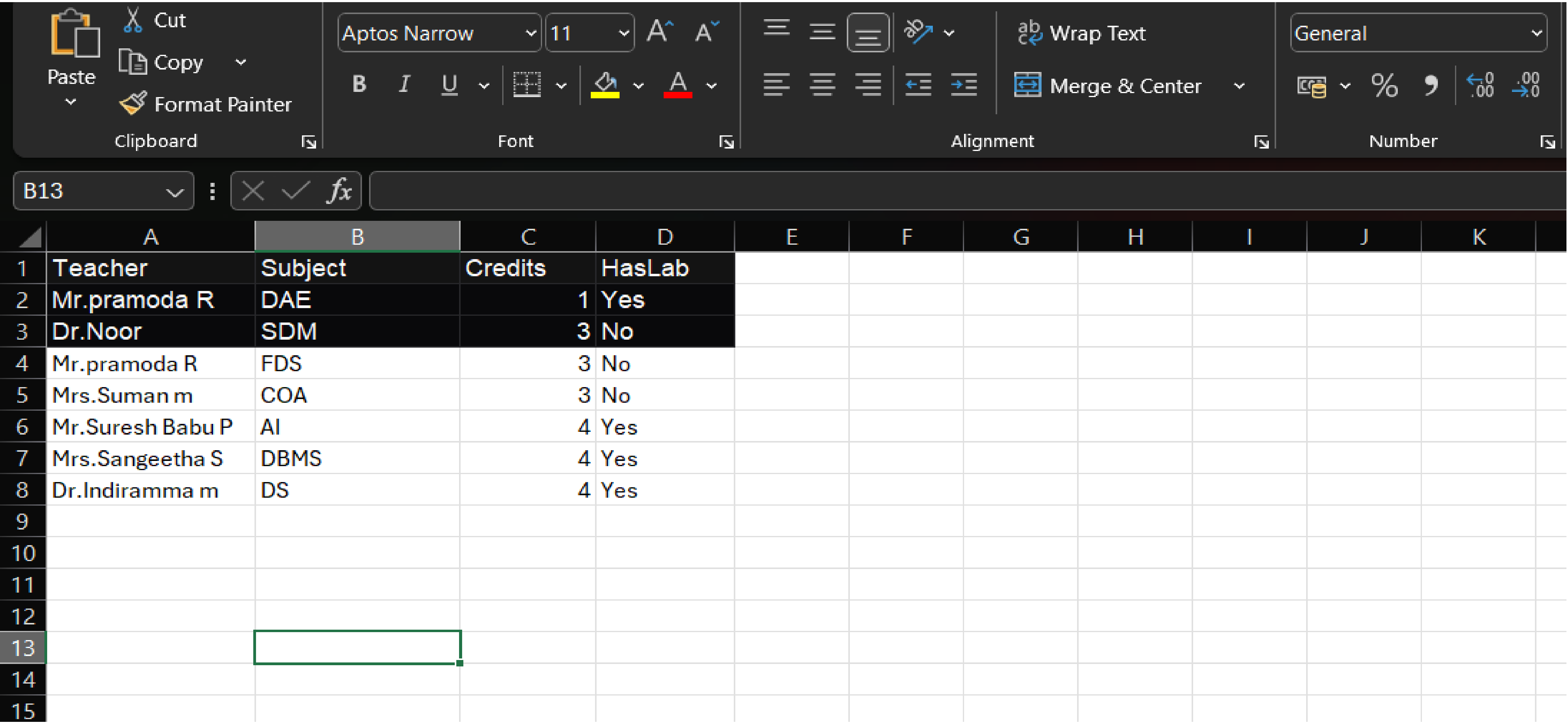
OneDrive - Personal › Desktop ›

Search Desktop

Organize ▾ New folder

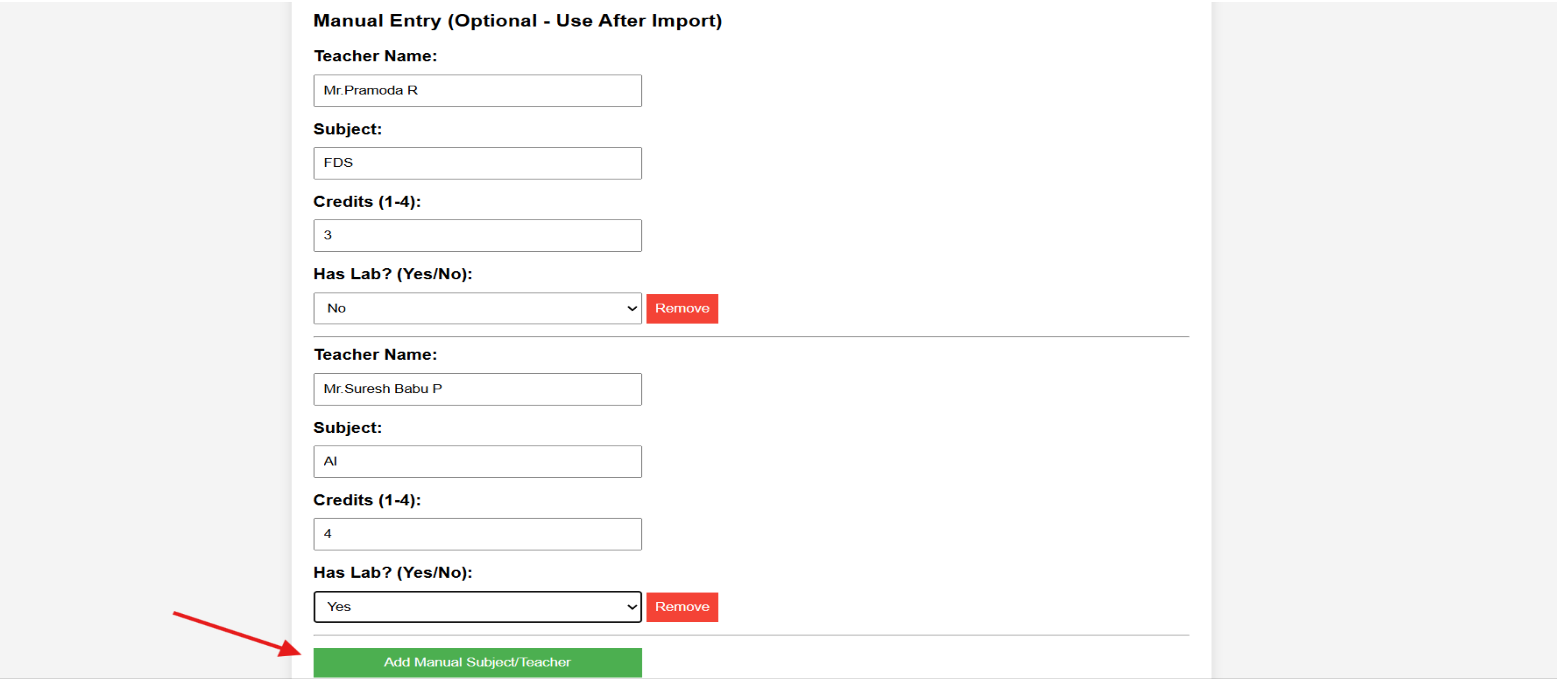
Name	Status	Date modified	Type
Smart Time table	✓	10-10-2025 22:08	Microsoft Excel W
smart-timetable	✓	10-10-2025 21:57	File folder
DBMS	✓	09-10-2025 23:13	File folder
clan	✓	04-10-2025 13:31	File folder

➢ Sample Excel File



	A	B	C	D
1	Teacher	Subject	Credits	HasLab
2	Mr.pramoda R	DAE	1	Yes
3	Dr.Noor	SDM	3	No
4	Mr.pramoda R	FDS	3	No
5	Mrs.Suman m	COA	3	No
6	Mr.Suresh Babu P	AI	4	Yes
7	Mrs.Sangeetha S	DBMS	4	Yes
8	Dr.Indiramma m	DS	4	Yes

➢ Manual Data Input (Optional Mode):



Manual Entry (Optional - Use After Import)

Teacher Name: Mr.Pramoda R

Subject: FDS

Credits (1-4): 3

Has Lab? (Yes/No): No

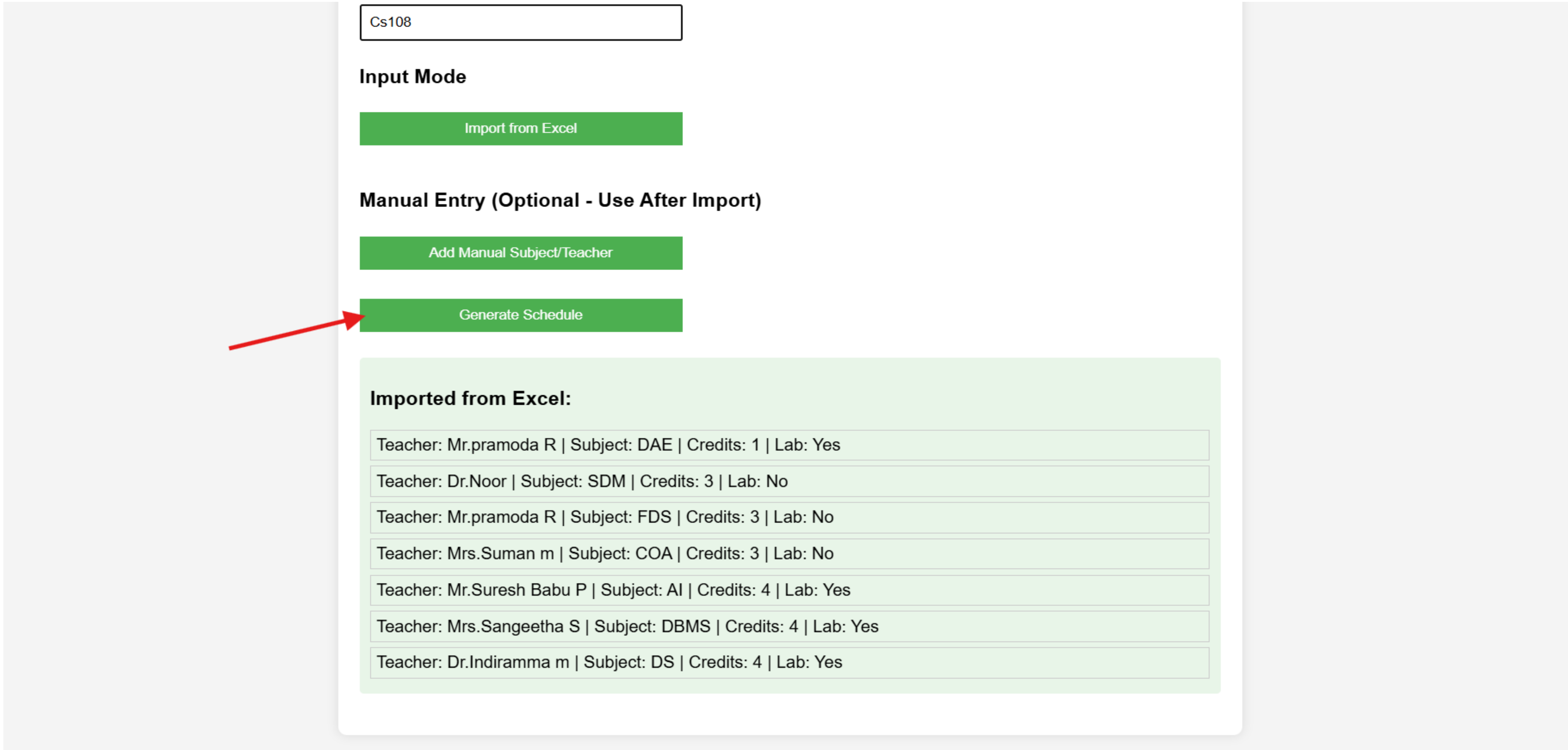
Teacher Name: Mr.Suresh Babu P

Subject: AI

Credits (1-4): 4

Has Lab? (Yes/No): Yes

➢ After Importing A file press generate Shedule



Cs108

Input Mode

Manual Entry (Optional - Use After Import)

Imported from Excel:

Teacher: Mr.pramoda R | Subject: DAE | Credits: 1 | Lab: Yes

Teacher: Dr.Noor | Subject: SDM | Credits: 3 | Lab: No

Teacher: Mr.pramoda R | Subject: FDS | Credits: 3 | Lab: No

Teacher: Mrs.Suman m | Subject: COA | Credits: 3 | Lab: No

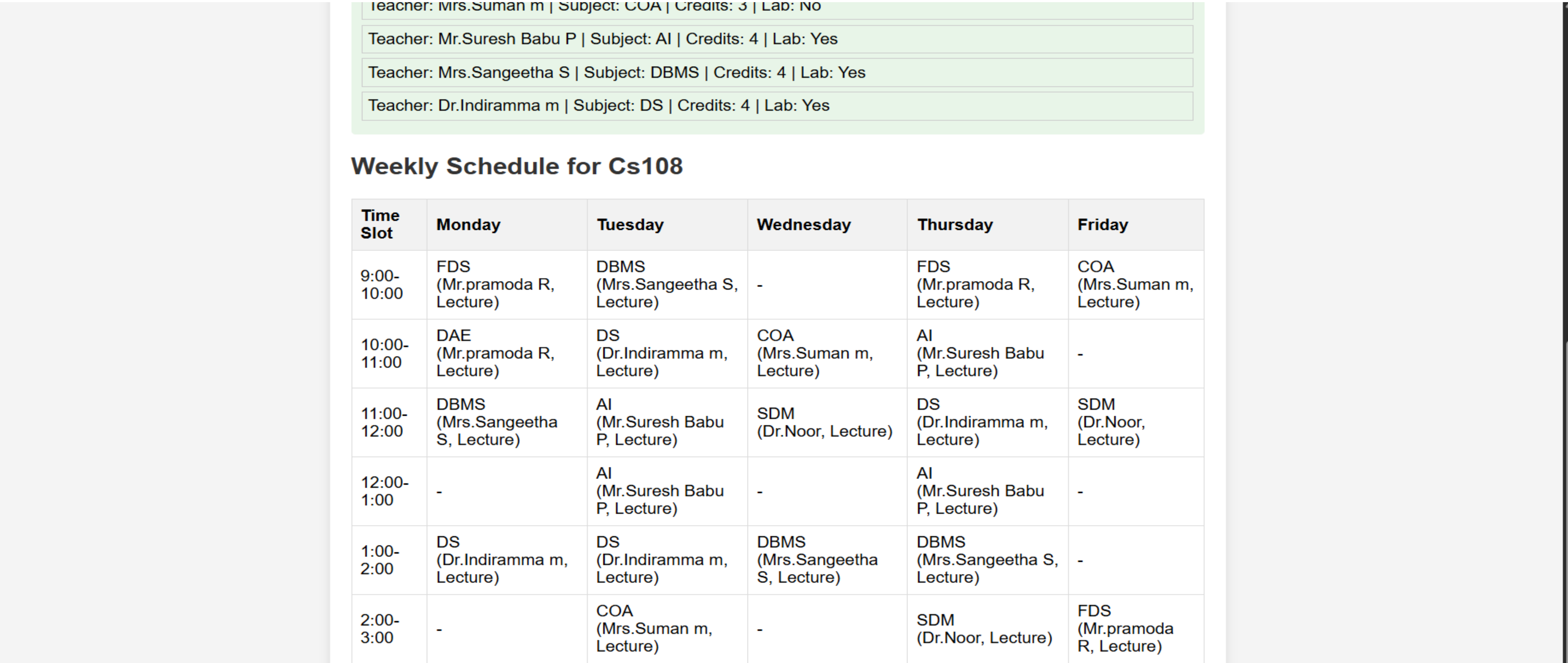
Teacher: Mr.Suresh Babu P | Subject: AI | Credits: 4 | Lab: Yes

Teacher: Mrs.Sangeetha S | Subject: DBMS | Credits: 4 | Lab: Yes

Teacher: Dr.Indiramma m | Subject: DS | Credits: 4 | Lab: Yes

8.Sample Output

- It will generate the Time Table based on the credits we have given.



Teacher: Mrs.Suman m | Subject: COA | Credits: 3 | Lab: No

Teacher: Mr.Suresh Babu P | Subject: AI | Credits: 4 | Lab: Yes

Teacher: Mrs.Sangeetha S | Subject: DBMS | Credits: 4 | Lab: Yes

Teacher: Dr.Indiramma m | Subject: DS | Credits: 4 | Lab: Yes

Weekly Schedule for Cs108

Time Slot	Monday	Tuesday	Wednesday	Thursday	Friday
9:00-10:00	FDS (Mr.pramoda R, Lecture)	DBMS (Mrs.Sangeetha S, Lecture)	-	FDS (Mr.pramoda R, Lecture)	COA (Mrs.Suman m, Lecture)
10:00-11:00	DAE (Mr.pramoda R, Lecture)	DS (Dr.Indiramma m, Lecture)	COA (Mrs.Suman m, Lecture)	AI (Mr.Suresh Babu P, Lecture)	-
11:00-12:00	DBMS (Mrs.Sangeetha S, Lecture)	AI (Mr.Suresh Babu P, Lecture)	SDM (Dr.Noor, Lecture)	DS (Dr.Indiramma m, Lecture)	SDM (Dr.Noor, Lecture)
12:00-1:00	-	AI (Mr.Suresh Babu P, Lecture)	-	AI (Mr.Suresh Babu P, Lecture)	-
1:00-2:00	DS (Dr.Indiramma m, Lecture)	DS (Dr.Indiramma m, Lecture)	DBMS (Mrs.Sangeetha S, Lecture)	DBMS (Mrs.Sangeetha S, Lecture)	-
2:00-3:00	-	COA (Mrs.Suman m, Lecture)	-	SDM (Dr.Noor, Lecture)	FDS (Mr.pramoda R, Lecture)

9.Code Reference & Inspiration

- Code implemented using **HTML, CSS, and JavaScript** for front-end logic.
- **ExcelJS/XLSX library** used for importing Excel files.
- Inspired by **real-world timetable management challenges** in colleges and universities.
- Algorithm designed to handle **lecture and lab scheduling**, avoiding conflicts.
- Manual and automatic input options ensure **flexibility and ease of use**.
- UI design inspired by **user-friendly web applications** with clear previews and error handling.
- Logical structure references **basic scheduling algorithms** and randomization for slot allocation.
- Designed for **scalability and future enhancements** like AI-based optimization and smart classroom integration
- “Inspired by real-world scheduling problems and online tutorials for XLSX integration and dynamic DOM manipulation”.

10.Conclusion

- The **Smart Class Schedule Generator** project successfully addresses the challenges of manual timetable creation in educational institutions. By automating the scheduling process, it minimizes errors and prevents conflicts in faculty availability, classroom allocation, and lab scheduling.

- The system provides a **flexible input mechanism**, allowing both manual entry of subjects and teachers, as well as bulk import from Excel files. This flexibility ensures that administrators can easily manage and update schedules without repetitive effort.

- **Lab and lecture scheduling** is intelligently handled, ensuring consecutive slots for labs where needed and proper distribution of lectures across the week. This helps in optimizing classroom and faculty utilization.

- The project showcases the **practical use of web technologies**—HTML, CSS, and JavaScript—demonstrating how front-end development can be combined with logical algorithms to solve real-world academic problems.

- A **preview feature** allows administrators to quickly verify and review the entered data before generating the schedule, reducing the likelihood of errors and saving time.

- Overall, the system enhances **efficiency, transparency, and accuracy** in timetable management. It lays a foundation for future enhancements such as AI-based optimization, IoT integration for smart classrooms, and automated notifications to students and faculty.