PHP PROJECT WORK

You will write only the database and the HTML codes inside the log books; the image you see above is the interface of the project work (Student Study Planner)

Submission of project: 9th September to the Lecture's email

Submission of logo book: 11th September to your course rep.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Logbook: Student Study Planner Web Application

1. Main Objective

To design and implement a dynamic, client-side web application that allows students to effectively manage their study schedule by adding, viewing, and tracking academic tasks.

2. Specific Learning Outcomes

- To apply core front-end web technologies (HTML, CSS, JavaScript) in an integrated project.
- To implement dynamic DOM manipulation based on user interaction.
- To utilize the browser's localStorage API for persistent data storage, simulating a database.
- To understand and implement Create, Read, and Delete (CRD) operations within a single-page application.

3. Hardware

- Standard PC or Laptop
- Minimum 4GB RAM
- Stable Internet Connection (for loading external libraries/fonts if used)

4. Software

- Web Browser (Chrome, Firefox, Edge)
- Code Editor (VS Code, Sublime Text)

- Version Control System (Git)

5. Other Materials

- UI Design Mockup (as provided in the initial project image)
- W3C HTML & CSS Validation Services

6. Flowchart (textual)

1. User opens planner page → 2. Fills Subject, Task, Due Date → 3. Submits form → 4. Client validates inputs → 5a. (Demo) Save to localStorage and update task list OR 5b. (Production) Send POST to server API → 6. Server inserts into database tasks table → 7. Respond success → 8. Client refreshes list view → End.

7. Code

Per your instruction, the logbook will include only the database (SQL) and the HTML code.

A — Database (SQL)

```
-- Create a database for the study planner

CREATE DATABASE IF NOT EXISTS study_planner;

USE study_planner;


-- Create tasks table

CREATE TABLE IF NOT EXISTS tasks (

    id INT AUTO_INCREMENT PRIMARY KEY,

    subject VARCHAR(100) NOT NULL,

    task TEXT NOT NULL,

    due_date DATE NOT NULL,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);
```

```
-- Sample inserts
INSERT INTO tasks (subject, task, due_date)
VALUES
('Mathematics', 'Revise calculus integration techniques', '2025-09-05'),
('Physics', 'Complete lab report on kinematics', '2025-09-07');

-- Example: select tasks ordered by due date
SELECT id, subject, task, due_date, created_at
FROM tasks
ORDER BY due_date ASC;
```

B — HTML (single-file, enhanced from your interface)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Student Study Planner</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f7fe;
      display: flex;
```

```
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  margin: 0;
  padding: 20px;
  box-sizing: border-box;
}
.planner-container {
  background-color: white;
  padding: 2rem;
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0,0,0,0.1);
  width: 100%;
  max-width: 650px;
}
h1 {
  text-align: center;
  color: #4361ee;
  margin-bottom: 1rem;
}
.form-grid {
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 1rem;
```

```
}
.form-group { margin-bottom: 1rem; }
label { display:block; margin-bottom: .5rem; font-weight: bold; color:#333; }
input, textarea, button {
  width:100%;
  padding:12px;
  border:1px solid #ddd;
  border-radius:6px;
  font-size:1rem;
  box-sizing:border-box;
}
textarea { min-height:80px; resize:vertical; grid-column: 1 / -1; }
button {
  padding:14px;
  background-color:#4361ee;
  color:white;
  border:none;
  border-radius:6px;
  font-size:1.05rem;
  cursor:pointer;
  transition: background-color .2s ease;
}
button:hover { background-color:#3a56d4; }
.btn-row { display:flex; gap:.5rem; }
```

```
    .tasks {
      margin-top: 1.5rem;
    }
    table {
      width:100%;
      border-collapse: collapse;
      margin-top:.5rem;
    }
    th, td {
      padding:.6rem;
      border:1px solid #eee;
      text-align:left;
      font-size:.95rem;
    }
    th { background:#f0f4ff; color:#213556; }
    .small { font-size:.85rem; color:#666; }
    .empty { text-align:center; padding:1rem; color:#777; }
    @media (max-width:600px) {
      .form-grid { grid-template-columns: 1fr; }
      textarea { grid-column: auto; }
    }
  </style>
</head>
<body>
```

```
  <div class="planner-container">
    <h1>Student Study Planner</h1>

    <form id="study-planner-form" aria-label="Study planner form">
      <div class="form-grid">
        <!-- Subject Input -->
        <div class="form-group">
          <label for="subject">Subject</label>
          <input type="text" id="subject" name="subject" placeholder="e.g.
Mathematics" required>
        </div>

        <!-- Due Date Input -->
        <div class="form-group">
          <label for="due-date">Due Date</label>
          <input type="date" id="due-date" name="due-date" required>
        </div>

        <!-- Task Input -->
        <div class="form-group" style="grid-column: 1 / -1;">
          <label for="task">Task</label>
          <textarea id="task" name="task" placeholder="Describe the study task..."
required></textarea>
        </div>
      </div>
```

```
      <div class="btn-row">
        <button type="submit">Add To Planner</button>
        <button type="button" id="clear-all" title="Clear all tasks">Clear All</button>
      </div>
    </form>

    <section class="tasks" aria-live="polite">
      <h2 class="small">Planned Tasks</h2>
      <div id="tasks-container">
        <div class="empty">No tasks yet — add one above.</div>
      </div>
    </section>
  </div>

  <script>
    // Simple client-side persistence demo using localStorage.
    // Replace these with fetch() calls to your server API when integrating with SQL backend.
    const STORAGE_KEY = 'study_planner_tasks_v1';

    function loadTasks() {
      const raw = localStorage.getItem(STORAGE_KEY);
      try {
        return raw ? JSON.parse(raw) : [];
```

```
    } catch (e) {
      console.error('Failed to parse tasks from storage', e);
      return [];
    }
  }

  function saveTasks(tasks) {
    localStorage.setItem(STORAGE_KEY, JSON.stringify(tasks));
  }

  function renderTasks() {
    const container = document.getElementById('tasks-container');
    const tasks = loadTasks();
    if (!tasks.length) {
      container.innerHTML = '<div class="empty">No tasks yet — add one above.</div>';
      return;
    }

    tasks.sort((a,b) => new Date(a.due_date) - new Date(b.due_date));

    let html = '<table aria-describedby="Planned study tasks"><thead><tr><th>Subject</th><th>Task</th><th>Due Date</th><th>Actions</th></tr></thead><tbody>';
    for (const t of tasks) {
```

```
      html += `<tr>
        <td>${escapeHtml(t.subject)}</td>
        <td>${escapeHtml(t.task)}</td>
        <td>${escapeHtml(t.due_date)}</td>
        <td>
          <button data-id="${t.id}" class="delete-btn"
style="padding:.35rem .6rem;border-radius:4px;border:1px solid
#ddd;background:#fff;cursor:pointer">Delete</button>
        </td>
      </tr>`;
    }
    html += '</tbody></table>';
    container.innerHTML = html;

    // Attach delete handlers
    document.querySelectorAll('.delete-btn').forEach(btn =>
      btn.addEventListener('click', () => {
        const id = btn.getAttribute('data-id');
        deleteTask(id);
      })
    );
  }

  function escapeHtml(text) {
    if (!text) return '';
```

```
  return text.replace(/[&<>"']/g, function(m) {
    return {'&':'&amp;','<':'&lt;','>':'&gt;','"':'&quot;',"'":'&#039;'}[m];
  });
}

function addTask(subject, task, dueDate) {
  const tasks = loadTasks();
  const id = Date.now().toString(36) + Math.random().toString(36).slice(2,7);
  tasks.push({ id, subject, task, due_date: dueDate });
  saveTasks(tasks);
  renderTasks();
}

function deleteTask(id) {
  let tasks = loadTasks();
  tasks = tasks.filter(t => t.id !== id);
  saveTasks(tasks);
  renderTasks();
}

document.getElementById('study-planner-form').addEventListener('submit', e => {
  e.preventDefault();
  const subject = document.getElementById('subject').value.trim();
  const task = document.getElementById('task').value.trim();
```

```
    const dueDate = document.getElementById('due-date').value;
    if (!subject || !task || !dueDate) {
      alert('Please fill all fields.');
      return;
    }
    addTask(subject, task, dueDate);
    e.target.reset();
    document.getElementById('subject').focus();
  });

  document.getElementById('clear-all').addEventListener('click', () => {
    if (!confirm('Clear all tasks? This cannot be undone in this demo.')) return;
    localStorage.removeItem(STORAGE_KEY);
    renderTasks();
  });

  // Initialize
  renderTasks();
 </script>
</body>
</html>
```

---

8. Results / Observations

- The HTML form accepts subjects, tasks, and due dates; the demo persists tasks in localStorage and lists them sorted by due date.

- Using the SQL table design above supports multi-user expansions and server-side persistence.
- Client-side demo is immediate and works offline; real DB integration requires a server/API layer.

9. Personal reflection

Building the planner reinforced how small, well-structured schemas map directly to usable UI features. The single-file HTML is great for prototyping; moving to a server-backed API will improve persistence and multi-device sync.

10. Challenges

- Real database persistence requires a server endpoint — this demo uses localStorage to keep the logbook self-contained.
- Handling timezones and date formatting can be tricky when moving from client demo to SQL-backed server.
- Input sanitization and security (e.g., preventing SQL injection and XSS) must be addressed on the server side.

GREATNESS