

# Web Application Development Fundamentals

## Workshop Objective

To develop (design and implement) a simple and secure To-Do List web application, using HTML, CSS, JavaScript, Node.js, and SQLite, with an emphasis on secure coding practices.

### Topics Covered:

- **Secure Web Development Basics:** Discuss common security threats such as SQL injection and XSS (Cross-Site Scripting).
- **Best Practices:** Highlight critical security principles, like input validation, error handling, and secure HTTP methods.
- **Project Overview:** Describe the functionality and goals of the To-Do List application, outlining its core components.

## Part 1: Backend and Database Setup

### A. Set Up Node.js and Express

- **Objective:** Initialize a Node.js project and configure Express for HTTP requests.
- **Instructions:**
  - For help to start <https://www.w3schools.com/nodejs/>
  - Create a new project directory for Node.js.
  - Use **npm init** to configure the project's **package.json**.
  - Install Express to manage server routing and requests.

### B. Set Up SQLite Database

- **Objective:** Configure SQLite to store tasks and define a secure database structure.
- **Instructions:**
  - Install SQLite and set up a connection within the Node.js project.
  - Design the database schema with a todos table that includes:
    - id: A unique integer identifier for each task.
    - task: A text field for task descriptions.
    - completed: A boolean to indicate task completion status.

### C. Build REST API Endpoints

- **Objective:** Develop CRUD (Create, Read, Update, Delete) API endpoints for tasks.
- **Instructions:**
  - Define an API route to fetch all tasks from the todos table.
  - Create a POST route to add tasks, ensuring input validation.

- Set up a DELETE route for task deletion.
- **Security Tip:** Avoid exposing sensitive data and use appropriate HTTP status codes for error handling.
- **Challenge:** Encourage students to understand the role of CORS (Cross-Origin Resource Sharing) and why it might be necessary.

## Part 2: Frontend Design and Integration

### A. Build the HTML and CSS for the To-Do List

- **Objective:** Create a basic user interface for the application.
- **Instructions:**
  - Set up HTML elements: a title, input field for new tasks, a submission button, and a list area for displaying tasks.
  - Use CSS to style a simple, visually appealing layout.

### B. Connect the Frontend with Backend Using JavaScript

- **Objective:** Enable interaction between the frontend and backend API endpoints.
- **Instructions:**
  - Use JavaScript's fetch to make HTTP requests to the backend.
  - Write functions to:
    - Add tasks by sending data to the backend.
    - Fetch tasks from the backend and display them in the HTML list.
    - Delete tasks by calling the DELETE endpoint and updating the UI accordingly.
  - Remember to sanitizing user input to prevent XSS attacks.

## Part 3: Testing and Review

### Testing the Application

- **Objective:** Ensure that each feature functions correctly and securely.
- **Instructions:**
  - Test functionality: adding, viewing, and deleting tasks.
  - Test edge cases, like adding empty tasks or overly long task descriptions.
- **Security Review:**
  - Identify and address vulnerabilities (e.g., SQL injection, XSS).
  - Verify proper HTTP status codes, like 200 for success and 400 for bad requests.