**report**

I have created a todo list web application using HTML, JavaScript, Spring MVC, and RESTful APIs as part of Sprint 2 Day 5.

The frontend is built using index.html, which is placed under src/main/webapp. This file contains JavaScript code that uses let, const, and var to demonstrate data types and hoisting. I have defined functions like addTask(), fetchTodos(), and deleteTask() to handle task operations. I used arrays to store the list of tasks, and applied control flow constructs like if conditions and forEach loops to render and manage them. Object usage was also explored in earlier versions by structuring each task as an object with properties.

On the backend, I created a class TodoController.java under src/main/java/com/nisum/controller. It acts as a Spring RestController with REST endpoints:

GET /api/todo – returns the current task list  
POST /api/todo – adds a new task  
DELETE /api/todo/{index} – deletes a task at the given index

I used an in-memory List of Strings to manage tasks. To allow JavaScript in the frontend to interact with the backend, I added CrossOrigin on the controller. I also added produces = "application/json" to the GetMapping to resolve a 406 error during API testing.

The application was configured using standard Spring XML configuration files — web.xml and spring-servlet.xml — located under WEB-INF. I set up component scanning for the controller package and enabled Spring MVC annotation support.

I used Maven for project management and packaged the app as a WAR. Required dependencies like spring-webmvc, javax.servlet-api, and jstl were added in pom.xml. I compiled and deployed the WAR using Smart Tomcat on port 8081 with the context path /assignment5.

After deployment, I tested the application by opening [http://localhost:8081/assignment5/index.html](http://localhost:8081/assignment5/index.html" \t "_new). The web page loaded successfully. I tested functionality by adding and deleting tasks. All fetch requests (GET, POST, DELETE) worked correctly and returned expected responses.

I also used Chrome DevTools to verify that JavaScript fetch calls were hitting the backend. The Network tab showed 200 OK status for all REST API interactions. I directly visited the API endpoint as well ([http://localhost:8081/assignment5/api/todo](http://localhost:8081/assignment5/api/todo" \t "_new)) to verify the JSON response.

Overall, I have successfully implemented all six required JavaScript concepts — data types, functions, control flow, hoisting, objects, and arrays — and integrated them into a working Spring MVC REST application that is deployed and fully testable.

Output:















