# Learning Management System (LMS) Project Specification

## 1. Project Overview

This document outlines the scope, functional requirements, and technical specifications for the development of a foundational Learning Management System (LMS). The application will facilitate online learning and course management, supporting three core user roles: Admin, Trainer, and Trainee.

## 2. Functional Requirements

### 2.1 Trainee Role

The Trainee role is designed to enable learners to interact with available courses. Key functionalities include:

* **Course Catalog:** View a list of all available courses.
* **Enrollment:** Register and enroll in specific courses.
* **Class Booking:** Book a seat in scheduled classes associated with enrolled courses.
* **Payment Processing:** Complete payment for course enrollment.

### 2.2 Trainer Role

The Trainer role provides the necessary tools for instructors to manage their classes and schedules. Key functionalities include:

* **Class Creation & Management:** Create, update, and manage the details of individual classes.
* **Schedule Management:** Define class times, dates, and capacity.
* **Class Status Control:** Open and close classes to manage trainee booking availability.

### 2.3 Admin Role

The Admin role is responsible for the overall management and oversight of the system. Key functionalities include:

* **User Management:** Create, view, edit, and delete user accounts.
* **Role Management:** Assign and modify the roles of users (Admin, Trainer, Trainee).
* **Reporting:** Generate basic reports on system usage and course statistics.
* **Content Management:** Manage the creation and details of courses within the system.

## 3. Technical Specifications

### 3.1 Architecture

The application will follow a standard client-server architecture with a clear separation between the frontend and backend.

### 3.2 Technology Stack

* **Backend:** ASP.NET Core 8 Web API will be used to build a robust and scalable backend.
* **Frontend:** The client-side application will be developed using Vanilla JavaScript, HTML5, and CSS3, ensuring a lean and high-performance user interface.
* **Database:** SQLite will serve as the relational database, providing a simple, file-based solution suitable for the project's scope.

### 3.3 Security

* **Authentication:** User authentication will be handled by **ASP.NET Core Identity**, providing a secure framework for user registration and login.
* **Authorization:** **JSON Web Tokens (JWT)** will be utilized for a modern, stateless authorization mechanism, ensuring secure communication between the frontend and backend and controlling access to specific API endpoints based on user roles.