Functional Specification Document (FSD)

Project Name: LMS Website (Learning Management System) **Version:** 1.0 **Date:** December 31, 2025

1. Introduction

This Functional Specification Document (FSD) outlines the system architecture, user roles, and detailed feature set of the LMS Website. The application is a comprehensive Learning Management System designed to facilitate online education through a robust platform connecting students, instructors, and administrators.

2. System Overview

The LMS is a web-based application built on the **MERN Stack** (MongoDB, Express.js, React.js, Node.js). It features a responsive design, secure authentication, and distinct workflows for different user personas.

User Roles

1. **Student**: The primary end-user who consumes content, purchases courses, and takes assessments.
2. **Instructor**: Content creators who publish courses, manage curriculum, and create quizzes.
3. **Administrator**: Superusers who manage the platform, approve content/instructors, and handle marketing tools like coupons and newsletters.

3. Functional Requirements

3.1 Authentication & Authorization Module

The system routes all users through a secure authentication layer using JWT (JSON Web Tokens).

- **User Registration (`/signup`)**:
  - Users can register as students providing basic details.
  - **Feature**: `Signup1` page captures user credentials.
- **User Login (`/login`)**:
  - Secure login with email and password.
  - **Feature**: `Login1` returns a JWT token stored client-side for session management.
- **Password Management**:
  - **Forgot Password**: Users can request a password reset via email.
  - **OTP Verification**: The system sends an OTP (`/verify-otp`) to verify identity before resetting credentials.
  - **Reset Password**: Users can set a new password (`/set-new-password`) after successful verification.

3.2 Student Module

Features available to logged-in students.

3.2.1 Course Discovery & Interaction

- **Home Page (`/`)**: Displays featured courses, categories, and success stats.
- **Course Listing (`/courses`)**: A catalog view of all available courses with filtering options.
- **Course Details (`/course/:id`)**: Detailed landing page for a specific course showing interface, curriculum preview, instructor info, and pricing.
- **Wishlist (`/wishlist`)**: Users can save courses for later consideration without adding them to the cart.

3.2.2 Shopping Cart & Checkout

- **Cart Management (`/cart`)**:
  - Add/Remove courses.
  - **Duplicate Protection**: The system prevents adding the same course twice (logic in `App.jsx`).
  - **Data Persistence**: Cart state is synced with `localStorage` to persist across sessions.
- **Checkout & Payment (`/payment`)**:
  - Secure integration for processing course purchases.
  - **Coupon Application**: Users can apply discount codes (`AdminCoupons` context) to reduce the total price.

3.2.3 Learning Environment

- **My Learning (`/my-learning`)**: Dashboard showing enrolled courses.
- **Course Player**:
  - **Lecture Video (`/lecture/:lectureId`)**: Video player for course content.
  - **Resources (`/resource/:id`)**: Access to supplementary files (PDFs, code zips) linked to lectures.
- **Quizzes (`/quiz`)**:
  - Interactive multiple-choice quizzes to test knowledge.
  - Results calculation and feedback.

3.2.4 User Profile & Communication

- **Profile Management**: Users can edit personal information (`/profile`, `/settings`).
- **Purchase History**: View past transactions.
- **Messages**: Internal messaging system for communication.

3.3 Instructor Module

Features for content creators (`/teach`, `/instructor-dashboard`).

3.3.1 Course Creation Workflow

Instructors do not publish directly; they create "Pending Courses" which require Admin approval.

- **Create Course (`/create`)**:
  - Initialize a new course draft.
- **Course Builder (`/dashboard/:pendingCourseId`)**:
  - **Curriculum Management**: Add sections, lectures, and upload videos.
  - **Edit Capabilities**: Modify course title, description, thumbnail, and pricing.
  - **Draft Resume**: The system auto-saves progress to `localStorage` to prevent data loss during creation.

- **Quiz Creation (`/createQuiz`)**: Instructors can design assessments attached to courses.

3.3.2 Instructor Dashboard

- **Overview**: View status of submitted courses (Pending/Approved).
- **Analytics**: (Implied) View enrollment stats and earnings.

3.4 Admin Module

Central control panel (`/admin`).

3.4.1 Content Management

- **Course Approval (`/preview/pending-course/:pendingCourseId`)**:
  - Admins review "Pending Courses" submitted by instructors.
  - Admins can Approve (publish to catalog) or Reject submissions.
- **Instructor Management (`/admin/instructors`)**: View and manage the list of registered instructors.

3.4.2 Marketing Tools

- **Coupon Management (`/admin/coupons`)**:
  - Create and delete discount codes.
  - Set discount percentages and validity.
- **Newsletter System (`/admin/newsletter`)**:
  - **Drafting**: simple HTML editor for email content.
  - **Sending**: Dispatches emails to all subscribed users (`AdminNewsletterDetail`).

3.5 Common Features

- **Network Status**: System logs network connectivity status (`logNetworkStatus`).
- **Notifications**: Real-time alerts for system events (Stubbed in `Notifications` page).
- **Contact Us (`/contact`)**: Public form for inquiries.

---

4. Technical Specifications

4.1 Frontend Architecture

- **Framework**: React.js with Vite.
- **Routing**: `react-router-dom` v6 with dynamic routes (e.g., `/course/:id`).
- **State Management**: React `Context API` (CartContext) combined with local state (`useState`, `useReducer`).
- **Styling**: Vanilla CSS (`App.css`) and Tailwind CSS.
- **HTTP Client**: `axios` (implied via `api.js` utility) with interceptors for auth tokens.

4.2 Backend Architecture

- **Server**: Node.js with Express.js.
- **Database**: MongoDB (Mongoose ODM).
  - **Models**: `User`, `Course`, `PendingCourse`, `Video`, `Quiz`, `Question`, `Coupon`, `Newsletter`.

- **File Handling**: `uploadRoutes` for handling video/image assets (likely Multer/Cloudinary integration).
- **Security**: `authRoutes` handling Login/Signup/OTP.

4.3 Data Flow

1. **Course Publication**:
   - Instructor -> `PendingCourse` Model -> Admin Review -> `Course` Model (Public).
2. **Purchase**:
   - User -> Cart -> Payment Gateway -> `User.enrolledCourses` update.

End of Document