# Learning Management System (LMS) Documentation

## Table of Contents

1. [Introduction](#introduction)
2. [Features](#features)
3. [Technical Stack](#technical-stack)
4. [System Architecture](#system-architecture)
5. [Database Schema and Relationships](#database-schema-and-relationships)
6. [API Documentation](#api-documentation)
7. [Setup and Installation](#setup-and-installation)
8. [Configuration](#configuration)
9. [Usage Guide](#usage-guide)
10. [Security Features](#security-features)
11. [Troubleshooting](#troubleshooting)
12. [FAQ](#faq)
13. [Contributing](#contributing)
14. [License](#license)

## Introduction

The Learning Management System (LMS) is a comprehensive web-based platform designed to manage academic activities, student progress, and faculty-student interactions. Built with Flask and JavaScript, it provides a robust and user-friendly interface for educational institutions.

## Features

### Student Features

* **Dashboard**
  + Real-time course progress tracking
  + Upcoming class schedule
  + Assignment deadlines
  + Attendance statistics
  + Grade overview
  + Personalized notifications
  + Quick access to recent activities
* **Attendance**
  + View attendance records
  + Track attendance percentage
  + Receive attendance notifications
  + Request leave applications
  + View attendance trends
* **Assignments**
  + Submit assignments online
  + Track submission status
  + View feedback and grades
  + Download assignment materials
  + Request deadline extensions
* **Exams**
  + View exam schedule
  + Access exam results
  + Download hall tickets
  + View exam guidelines
  + Check exam room allocation
* **Courses**
  + Access course materials
  + View course syllabus
  + Track course progress
  + Download lecture notes
  + Access recorded lectures
* **Social Platform**
  + Interact with peers
  + Share study materials
  + Join study groups
  + Create discussion threads
  + Share announcements

### Faculty Features

* **Student Management**
  + View student profiles
  + Track student progress
  + Manage student records
  + Generate student reports
  + Send notifications to students
* **Course Management**
  + Create and update courses
  + Upload course materials
  + Set course schedules
  + Manage course enrollment
  + Track course completion
* **Assignment Management**
  + Create assignments
  + Grade submissions
  + Provide feedback
  + Set deadlines
  + Track submission status
* **Attendance**
  + Mark attendance
  + Generate reports
  + Track attendance trends
  + Export attendance data
  + Manage leave requests
* **Grade Management**
  + Submit grades
  + Generate grade reports
  + Track student performance
  + Calculate final grades
  + Export grade sheets
* **Analytics**
  + View class performance
  + Generate progress reports
  + Track attendance trends
  + Analyze student performance
  + Generate statistical reports

## Technical Stack

### Backend

* **Framework**: Flask (Python)
* **Database**: JSON-based storage
* **Authentication**: Custom JWT-based system
* **API**: RESTful architecture
* **Caching**: Redis (optional)
* **Task Queue**: Celery (optional)

### Frontend

* **HTML5**: Semantic markup
* **CSS3**: Responsive design
* **JavaScript**: Interactive features
* **AJAX**: Asynchronous data loading
* **Bootstrap**: UI components
* **jQuery**: DOM manipulation

### Development Tools

* **Version Control**: Git
* **Code Editor**: VS Code recommended
* **Browser**: Chrome/Firefox for development
* **Testing**: pytest, Jest
* **CI/CD**: GitHub Actions
* **Documentation**: Swagger/OpenAPI

## System Architecture

### Backend Architecture

#### Flask Application Structure

# app.py - Main Application Entry Point  
from flask import Flask, jsonify, request  
from flask\_cors import CORS  
import json  
from datetime import datetime, timedelta  
import re  
  
app = Flask(\_\_name\_\_)  
CORS(app)  
  
# Database Models  
class User:  
 def \_\_init\_\_(self, id, username, password, role):  
 self.id = id  
 self.username = username  
 self.password = password  
 self.role = role  
  
class Course:  
 def \_\_init\_\_(self, id, name, faculty\_id, schedule):  
 self.id = id  
 self.name = name  
 self.faculty\_id = faculty\_id  
 self.schedule = schedule  
  
# Authentication Middleware  
def authenticate\_token(token):  
 try:  
 # Verify JWT token  
 payload = jwt.decode(token, SECRET\_KEY, algorithms=['HS256'])  
 return payload  
 except:  
 return None  
  
# API Routes  
@app.route('/api/login', methods=['POST'])  
def login():  
 data = request.get\_json()  
 # Authentication logic  
 return jsonify({"token": token, "user": user\_data})  
  
@app.route('/api/dashboard', methods=['GET'])  
@require\_auth  
def get\_dashboard():  
 # Dashboard data retrieval  
 return jsonify(dashboard\_data)

#### Data Storage

# data\_manager.py - Data Management  
class DataManager:  
 def \_\_init\_\_(self):  
 self.data\_path = "data/"  
   
 def load\_data(self, filename):  
 with open(f"{self.data\_path}{filename}", 'r') as f:  
 return json.load(f)  
   
 def save\_data(self, filename, data):  
 with open(f"{self.data\_path}{filename}", 'w') as f:  
 json.dump(data, f, indent=4)

#### Authentication System

# auth.py - Authentication System  
import jwt  
from datetime import datetime, timedelta  
  
class AuthManager:  
 def \_\_init\_\_(self):  
 self.secret\_key = "your-secret-key"  
   
 def generate\_token(self, user\_id, role):  
 payload = {  
 'user\_id': user\_id,  
 'role': role,  
 'exp': datetime.utcnow() + timedelta(hours=24)  
 }  
 return jwt.encode(payload, self.secret\_key, algorithm='HS256')  
   
 def verify\_token(self, token):  
 try:  
 return jwt.decode(token, self.secret\_key, algorithms=['HS256'])  
 except:  
 return None

### Frontend Architecture

#### HTML Structure

<!-- dashboard.html - Student Dashboard -->  
<!DOCTYPE html>  
<html>  
<head>  
 <title>Student Dashboard</title>  
 <link rel="stylesheet" href="styles.css">  
</head>  
<body>  
 <div class="dashboard-container">  
 <header>  
 <h1>Student Dashboard</h1>  
 <nav>  
 <ul>  
 <li><a href="#courses">Courses</a></li>  
 <li><a href="#assignments">Assignments</a></li>  
 <li><a href="#attendance">Attendance</a></li>  
 </ul>  
 </nav>  
 </header>  
 <main>  
 <section id="courses">  
 <!-- Course content -->  
 </section>  
 <section id="assignments">  
 <!-- Assignment content -->  
 </section>  
 </main>  
 </div>  
 <script src="dashboard.js"></script>  
</body>  
</html>

#### JavaScript Implementation

// dashboard.js - Dashboard Functionality  
class Dashboard {  
 constructor() {  
 this.token = localStorage.getItem('token');  
 this.userId = localStorage.getItem('userId');  
 this.initializeDashboard();  
 }  
   
 async initializeDashboard() {  
 try {  
 const response = await fetch('/api/dashboard', {  
 headers: {  
 'Authorization': `Bearer ${this.token}`  
 }  
 });  
 const data = await response.json();  
 this.renderDashboard(data);  
 } catch (error) {  
 console.error('Error loading dashboard:', error);  
 }  
 }  
   
 renderDashboard(data) {  
 // Render dashboard components  
 this.renderCourses(data.courses);  
 this.renderAssignments(data.assignments);  
 this.renderAttendance(data.attendance);  
 }  
}  
  
// Initialize dashboard when DOM is loaded  
document.addEventListener('DOMContentLoaded', () => {  
 new Dashboard();  
});

#### CSS Styling

/\* styles.css - Main Stylesheet \*/  
:root {  
 --primary-color: #3498db;  
 --secondary-color: #2ecc71;  
 --text-color: #333;  
}  
  
.dashboard-container {  
 max-width: 1200px;  
 margin: 0 auto;  
 padding: 20px;  
}  
  
header {  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 margin-bottom: 30px;  
}  
  
nav ul {  
 display: flex;  
 list-style: none;  
 gap: 20px;  
}  
  
nav a {  
 text-decoration: none;  
 color: var(--text-color);  
 font-weight: 500;  
}  
  
section {  
 margin-bottom: 40px;  
 padding: 20px;  
 background: #fff;  
 border-radius: 8px;  
 box-shadow: 0 2px 4px rgba(0,0,0,0.1);  
}

## Database Schema and Relationships

### Entity Relationship Diagram (ERD)

+---------------+ +----------------+ +----------------+  
| User | | Course | | Assignment |  
+---------------+ +----------------+ +----------------+  
| id |<----->| id |<----->| id |  
| username | | name | | title |  
| password | | code | | description |  
| email | | faculty\_id | | course\_id |  
| role | | schedule | | due\_date |  
| department | | credits | | max\_marks |  
| semester | | status | | status |  
+---------------+ +----------------+ +----------------+  
 ^ ^ ^  
 | | |  
 | | |  
+---------------+ +----------------+ +----------------+  
| Attendance | | Enrollment | | Submission |  
+---------------+ +----------------+ +----------------+  
| id | | id | | id |  
| student\_id | | student\_id | | student\_id |  
| course\_id | | course\_id | | assignment\_id |  
| date | | semester | | submission\_date|  
| status | | grade | | file\_path |  
| remarks | | status | | marks |  
+---------------+ +----------------+ +----------------+  
 ^ ^ ^  
 | | |  
 | | |  
+---------------+ +----------------+ +----------------+  
| Grade | | Material | | Notification |  
+---------------+ +----------------+ +----------------+  
| id | | id | | id |  
| student\_id | | course\_id | | title |  
| course\_id | | description | | file\_path |  
| assignment\_id | | upload\_date | | type |  
| marks | | type | | message |  
| grade | | upload\_date | | status |  
| feedback | | upload\_date | | created\_at |  
| read\_at | | upload\_date | | read\_at |  
+---------------+ +----------------+ +----------------+

## API Documentation

[API Documentation section remains the same as in the previous update…]

## Setup and Installation

[Setup and Installation section remains the same as in the previous update…]

## Configuration

[Configuration section remains the same as in the previous update…]

## Usage Guide

[Usage Guide section remains the same as in the previous update…]

## Security Features

[Security Features section remains the same as in the previous update…]

## Troubleshooting

[Troubleshooting section remains the same as in the previous update…]

## FAQ

[FAQ section remains the same as in the previous update…]

## Contributing

[Contributing section remains the same as in the previous update…]

## License

[License section remains the same as in the previous update…]