**Requirements Documentation**

**Project Name**: Development of a Basic Learning Management System (LMS) Application  
**Prepared by**: Montassar Boukhadhra

**Date**: 11/09/2025

**Version**: 1.0

**1. Introduction**

This document outlines the functional and technical requirements for the development of a basic Learning Management System (LMS) application for Alarkkan Training Center. The application will have three user roles: Admin, Trainee, and Trainer. The system will be developed using ASP.NET Core 8 Web API for the backend, JavaScript for the frontend, and SQLite for the database. Security will be handled using the Identity Framework for authentication and JWT (JSON Web Tokens) for authorization.

**2. Functional Requirements**

**2.1 User Roles and Access**

1. **Admin**:
   * Can create, update, and delete user accounts (Trainee and Trainer).
   * Can assign roles and permissions to users.
   * Can manage course categories and settings.
2. **Trainee**:
   * Can register for courses.
   * Can book and pay for classes.
   * Can view their course schedule and payment history.
3. **Trainer**:
   * Can schedule, open, and close classes.
   * Can view trainee enrollment for their classes.
   * Can update class details (e.g., time, location).

**2.2 Course Management**

1. **Course Registration**:
   * Trainees can browse available courses and register for them.
   * Trainees can view course details (e.g., description, duration, cost).
2. **Class Booking and Payment**:
   * Trainees can book classes for registered courses.
   * Trainees can make payments for booked classes (basic payment integration).

**2.3 Class Scheduling**

1. **Trainer Class Management**:
   * Trainers can schedule new classes, specifying time, location, and maximum capacity.
   * Trainers can open or close classes for enrollment.
   * Trainers can view enrolled trainees for their classes.

**2.4 User Authentication and Authorization**

1. **Authentication**:
   * Users (Admin, Trainee, Trainer) must log in to access the system.
   * Authentication will be handled using the Identity Framework.
2. **Authorization**:
   * Role-based access control (RBAC) will be implemented using JWT (JSON Web Tokens).
   * Each user role will have specific permissions based on their role.

**3. Non-Functional Requirements**

**3.1 Performance**

1. The application should handle up to 100 concurrent users without significant performance degradation.
2. The system should load course and class details within 2 seconds under normal conditions.

**3.2 Security**

1. **Authentication**:
   * User passwords must be securely hashed and stored.
   * The Identity Framework will handle password hashing and user management.
2. **Authorization**:
   * JWT tokens will be used to enforce role-based access control.
   * Tokens will have a limited lifespan and will be refreshed periodically.
3. **Data Protection**:
   * Sensitive data (e.g., payment information) will be encrypted in transit and at rest.

**3.3 Scalability**

1. The application should be designed to allow for future scaling (e.g., adding more users or courses).
2. The database schema should be normalized to support future enhancements.

**3.4 Usability**

1. The user interface should be intuitive and easy to navigate.
2. The application should provide clear error messages and guidance for user actions.

**4. Technical Specifications**

**4.1 Backend**

1. **Framework**: ASP.NET Core 8 Web API.
2. **Functionality**:
   * API endpoints for user authentication, course management, class scheduling, and payment processing.
   * Integration with the Identity Framework for user authentication.
   * JWT token generation and validation for role-based authorization.

**4.2 Frontend**

1. **Framework**: JavaScript (no additional libraries or frameworks).
2. **Functionality**:
   * User interface for Admin, Trainee, and Trainer roles.
   * Forms for course registration, class booking, and payment.
   * Views for course schedules, enrolled trainees, and payment history.

**4.3 Database**

1. **Database**: SQLite.
2. **Schema**:
   * **Users Table**: Stores user information (e.g., username, password, role).
   * **Courses Table**: Stores course details (e.g., name, description, cost).
   * **Classes Table**: Stores class details (e.g., course ID, trainer ID, time, location).
   * **Enrollments Table**: Stores trainee enrollments in classes.
   * **Payments Table**: Stores payment details for booked classes.

**4.4 Security**

1. **Authentication**:
   * Identity Framework will handle user authentication.
   * Passwords will be hashed using secure algorithms (e.g., bcrypt).
2. **Authorization**:
   * JWT tokens will be used to enforce role-based access control.
   * Tokens will be signed and verified using a secure key.

**5. Assumptions**

1. The application will be developed as a basic prototype without advanced features.
2. The sponsor will provide any necessary clarifications promptly.
3. The development environment and tools are readily available.

**6. Constraints**

1. The application must be delivered within 5 working days.
2. The application will be developed using the specified technologies (ASP.NET Core 8, JavaScript, SQLite).
3. Advanced features such as third-party payment integration or mobile responsiveness are out of scope.

**7. Glossary**

1. **LMS**: Learning Management System.
2. **JWT**: JSON Web Token, a standard for securely transmitting information between parties as a JSON object.
3. **Identity Framework**: A framework in ASP.NET Core for managing user authentication and authorization.

**8. Revision History**

| **Version** | **Date** | **Description** |
| --- | --- | --- |
| 1.0 | 11/09/2025 | Initial Requirements Documentation |