Training Center Management System

Project Requirement Specification Document

1. Project Overview

Project Title: Training Center Management System
Project Type: Web-based Application
Domain: Education Management

1.1 Purpose

The Training Center Management System is designed to streamline the operations of a training center by providing a comprehensive platform for managing courses, students, trainers, attendance, assignments, and leave management through a role-based access system.

1.2 Scope

This system will cater to three types of users: Administrators, Trainers, and Students. Each role will have specific functionalities and access levels to ensure proper management and security.

1.3 Objectives

- Digitize training center operations
- Implement role-based access control
- Provide real-time dashboards and statistics
- Streamline student enrollment and management
- Facilitate assignment and leave management
- Generate comprehensive reports and analytics

2. Technical Requirements

2.1 Frontend Technologies

- **HTML5**: Structure and markup
- **CSS3**: Styling and layout
- **JavaScript**: Client-side functionality and AJAX requests
- **Bootstrap**: Responsive design framework

2.2 Backend Technologies

- **Python**: Server-side programming language
- **Flask**: Web framework for Python
- **NumPy**: Numerical computing for statistics
- **Pandas**: Data manipulation and analysis
- **Matplotlib**: Data visualization and chart generation

2.3 Database

- **MySQL**: Relational database management system
- **mysql-connector-python**: Python MySQL database connector

2.4 Communication Protocol

- All frontend-backend communication must be handled through javascript
- RESTful API design principles should be followed
- JSON format for data exchange

3. System Architecture

3.1 Security Requirements

- Password hashing using secure algorithms
- Session management for user authentication
- Role-based access control (RBAC)
- Input validation and sanitization
- SQL injection prevention

4. User Roles and Access Control

4.1 Administrator (Admin)

Highest privilege level with complete system access

4.2 Trainer

Mid-level access for course and student management

4.3 Student

Limited access for personal information and assignments

5. Detailed Functional Requirements

5.1 Administrator Features

5.1.1 Course Management

Requirements:

- Create new courses with the following attributes:
    - Course name
    - Course description
    - Course duration (in weeks/months)
    - Maximum allowed leaves per course
    - Course start and end dates

    - Course status (Active/Inactive)
- Edit existing course details
- Delete courses (with validation checks)
- View all courses with filtering options
- Set course capacity (maximum students)

Acceptance Criteria:

- Admin can create courses with all required fields
- Course duration should be validated (minimum 1 week)
- Leave allowance should be a positive integer
- System should prevent deletion of courses with enrolled students

5.1.2 Trainer Management

Requirements:

- Register new trainers with:
    - Full name
    - Email address (unique)
    - Phone number
    - Specialization areas
    - Auto-generated login credentials
- Assign trainers to specific courses
- Edit trainer information
- Deactivate/activate trainer accounts
- View trainer profiles and assigned courses

Acceptance Criteria:

- System generates secure login credentials for trainers
- Email addresses must be unique across the system
- Trainers can be assigned to multiple courses
- Deactivated trainers cannot access the system

5.1.3 Student Management

Requirements:

- Register students individually through forms:
    - Student name
    - Email address (unique)
    - Phone number
    - Course enrollment
    - Auto-generated login credentials
- Bulk student registration via CSV upload:
    - CSV should contain: name, email, phone, course_name
    - System should validate data and report errors
    - Duplicate email handling
- View all students with course-wise filtering

- Edit student information
- Transfer students between courses
- Deactivate/activate student accounts

Acceptance Criteria:

- CSV upload should validate all required fields
- System should handle duplicate emails gracefully
- Students can only be enrolled in one course at a time
- Error reporting for failed CSV uploads

5.1.4 Attendance Management

Requirements:

- View attendance reports course-wise
- Generate attendance statistics
- Mark attendance for students (bulk operations)
- View individual student attendance history
- Export attendance reports to CSV

Acceptance Criteria:

- Attendance can be marked by date and course
- System should show attendance percentage for each student
- Reports should be filterable by date range

5.1.5 Leave Management

Requirements:

- View all leave applications with status
- Approve or reject leave applications
- Add comments while approving/rejecting
- View leave statistics course-wise
- Send notifications to students about leave status

Acceptance Criteria:

- Leave applications should show student details and reason
- System should prevent students from exceeding leave limits
- Status changes should be logged with timestamps

5.1.6 Admin Dashboard

Requirements:

- Total number of courses (active/inactive)
- Total number of students per course

- Total number of trainers
- Pending leave applications count
- Recent system activities
- Course-wise enrollment statistics (charts)
- Monthly attendance trends (charts)

Acceptance Criteria:

- Dashboard should load within 3 seconds
- Charts should be interactive and responsive
- Statistics should update in real-time

5.2 Trainer Features

5.2.1 Topic Management

Requirements:

- Create topics for assigned courses:
  - Topic name
  - Topic description
  - Course association
  - Topic sequence/order
- Edit topic details
- Delete topics (with validation)
- Reorder topics within a course

Acceptance Criteria:

- Trainers can only create topics for their assigned courses
- Topics should have logical sequencing
- Deletion should check for existing assignments

5.2.2 Assignment Management

Requirements:

- Create assignments linked to topics:
  - Assignment title
  - Assignment description
  - Due date and time
  - Topic association
  - Assignment type (individual/group)
- Edit assignment details
- Delete assignments
- View assignment submission status
- Grade assignments (optional)

Acceptance Criteria:

- Due dates should be future dates
- Students should be notified of new assignments
- Assignment status should be tracked (submitted/pending)

5.2.3 Student Overview

Requirements:

- View all students enrolled in trainer's courses
- See individual student profiles
- View student attendance records
- Track student assignment completion rates
- View student leave history

Acceptance Criteria:

- Information should be course-specific
- Real-time data updates
- Export student reports to PDF/CSV

5.2.4 Trainer Dashboard

Requirements:

- Course-wise statistics:
  - Number of topics created
  - Number of assignments per topic
  - Assignment completion rates
  - Student performance metrics
- Recent activities
- Upcoming assignment deadlines
- Charts showing:
  - Topic-wise assignment distribution
  - Student completion rates
  - Performance trends

Acceptance Criteria:

- Dashboard should be responsive across devices
- Charts should be generated using Matplotlib
- Data should be accurate and up-to-date

5.3 Student Features

5.3.1 Authentication and Profile

Requirements:

- Login using provided credentials

- View personal profile information
- Change password functionality
- Update contact information (limited fields)

Acceptance Criteria:

- Secure login with session management
- Password should be encrypted
- Profile updates should be validated

5.3.2 Assignment Management

Requirements:

- View assignments organized by topics
- See assignment details and due dates
- Download assignment files (if any)
- Submit assignments (file upload)
- Track assignment submission status
- View grades/feedback (if provided)

Acceptance Criteria:

- Assignments should be sorted by due date
- File upload should support multiple formats
- Submission should be timestamped
- Late submissions should be clearly marked

5.3.3 Leave Management

Requirements:

- Apply for leave with:
  - Leave start date
  - Leave end date
  - Reason for leave
  - Supporting documents (optional)
- View leave application history
- Check leave application status
- View remaining leave balance
- Cancel pending leave applications

Acceptance Criteria:

- Leave dates should be validated
- System should check leave balance before approval
- Students should receive status notifications

5.3.4 Student Dashboard

Requirements:

- Personal statistics:
  - Total leaves taken vs. allowed
  - Remaining leave balance
  - Total assignments received
  - Assignments completed/pending
  - Overall attendance percentage
- Recent activities
- Upcoming assignment deadlines
- Leave application status
- Performance charts

Acceptance Criteria:

- Dashboard should provide clear visual indicators
- Charts should show trends over time
- Information should be accurate and real-time

6. Non-Functional Requirements

6.1 Performance Requirements

- Page load time should not exceed 3 seconds
- Database queries should be optimized
- AJAX requests should respond within 2 seconds
- System should handle 100 concurrent users

6.2 Security Requirements

- All passwords must be hashed using bcrypt or similar
- SQL injection protection through parameterized queries
- XSS protection through input sanitization
- CSRF protection for form submissions
- Session timeout after 30 minutes of inactivity

6.3 Usability Requirements

- Responsive design for mobile and desktop
- Intuitive navigation structure
- Error messages should be user-friendly
- Form validation with real-time feedback
- Consistent UI/UX across all pages

6.4 Reliability Requirements

- System uptime of 99%
- Database backup procedures
- Error logging and monitoring
- Graceful error handling

7. User Interface Requirements

7.1 General UI Guidelines

- Clean and professional design
- Bootstrap-based responsive layout
- Consistent color scheme and typography
- Loading indicators for AJAX requests
- Success/error message notifications

7.2 Navigation Structure

- Role-based navigation menus
- Breadcrumb navigation for deep pages
- Quick access to frequently used features
- Search functionality where applicable

7.3 Forms and Data Entry

- Client-side and server-side validation
- Clear field labels and placeholders
- Required field indicators
- Progress indicators for multi-step processes

8. Testing Requirements

8.1 Unit Testing

- Test all API endpoints
- Test database operations
- Test authentication and authorization
- Test file upload functionality

8.2 Integration Testing

- Test complete user workflows
- Test role-based access control
- Test AJAX functionality
- Test CSV upload process

8.3 User Acceptance Testing

- Admin workflow testing
- Trainer workflow testing
- Student workflow testing
- Cross-browser compatibility

Database Design

9. Entity Relationship Diagram Description

The database consists of the following main entities with their relationships:

9.1 Tables Structure

9.1.1 Users Table

```
CREATE TABLE users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    full_name VARCHAR(255) NOT NULL,
    phone VARCHAR(15),
    role ENUM('admin', 'trainer', 'student') NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
```

9.1.2 Courses Table

```
CREATE TABLE courses (
    course_id INT PRIMARY KEY AUTO_INCREMENT,
    course_name VARCHAR(255) NOT NULL,
    description TEXT,
    duration_weeks INT NOT NULL,
    max_leaves INT DEFAULT 5,
    start_date DATE,
    end_date DATE,
    max_capacity INT DEFAULT 30,
    is_active BOOLEAN DEFAULT TRUE,
    created_by INT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (created_by) REFERENCES users(user_id)
);
```

9.1.3 Course_Trainers Table (Many-to-Many)

```
CREATE TABLE course_trainers (
    id INT PRIMARY KEY AUTO_INCREMENT,
    course_id INT,
    trainer_id INT,
    assigned_date DATE DEFAULT (CURRENT_DATE),
    is_active BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (course_id) REFERENCES courses(course_id) ON DELETE
CASCADE,
    FOREIGN KEY (trainer_id) REFERENCES users(user_id) ON DELETE CASCADE,
    UNIQUE KEY unique_course_trainer (course_id, trainer_id)
);
```

9.1.4 Students Table

```
CREATE TABLE students (
    student_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT UNIQUE,
    course_id INT,
    enrollment_date DATE DEFAULT (CURRENT_DATE),
    is_active BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (course_id) REFERENCES courses(course_id) ON DELETE SET
NULL
);
```

9.1.5 Topics Table

```
CREATE TABLE topics (
    topic_id INT PRIMARY KEY AUTO_INCREMENT,
    topic_name VARCHAR(255) NOT NULL,
    description TEXT,
    course_id INT,
    trainer_id INT,
    sequence_order INT DEFAULT 1,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (course_id) REFERENCES courses(course_id) ON DELETE
CASCADE,
    FOREIGN KEY (trainer_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

9.1.6 Assignments Table

```
CREATE TABLE assignments (
    assignment_id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(255) NOT NULL,
    description TEXT,
    topic_id INT,
    created_by INT,
    due_date DATETIME,
    assignment_type ENUM('individual', 'group') DEFAULT 'individual',
    max_points INT DEFAULT 100,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (topic_id) REFERENCES topics(topic_id) ON DELETE CASCADE,
    FOREIGN KEY (created_by) REFERENCES users(user_id)
);
```

9.1.7 Assignment_Submissions Table

```
CREATE TABLE assignment_submissions (
    submission_id INT PRIMARY KEY AUTO_INCREMENT,
    assignment_id INT,
    student_id INT,
    submission_text TEXT,
    file_path VARCHAR(500),
    submitted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    is_late BOOLEAN DEFAULT FALSE,
    grade INT DEFAULT NULL,
    feedback TEXT,
    graded_by INT DEFAULT NULL,
    graded_at TIMESTAMP NULL,
    FOREIGN KEY (assignment_id) REFERENCES assignments(assignment_id) ON
DELETE CASCADE,
    FOREIGN KEY (student_id) REFERENCES students(student_id) ON DELETE
CASCADE,
    FOREIGN KEY (graded_by) REFERENCES users(user_id),
    UNIQUE KEY unique_assignment_student (assignment_id, student_id)
);
```

9.1.8 Attendance Table

```
CREATE TABLE attendance (
    attendance_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    course_id INT,
    attendance_date DATE,
    is_present BOOLEAN DEFAULT FALSE,
    marked_by INT,
    marked_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    notes TEXT,
    FOREIGN KEY (student_id) REFERENCES students(student_id) ON DELETE
CASCADE,
    FOREIGN KEY (course_id) REFERENCES courses(course_id) ON DELETE
CASCADE,
    FOREIGN KEY (marked_by) REFERENCES users(user_id),
    UNIQUE KEY unique_student_date (student_id, attendance_date)
);
```

9.1.9 Leave_Applications Table

```
CREATE TABLE leave_applications (
    leave_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    reason TEXT NOT NULL,
    supporting_document VARCHAR(500),
    status ENUM('pending', 'approved', 'rejected') DEFAULT 'pending',
    applied_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    reviewed_by INT DEFAULT NULL,
    reviewed_at TIMESTAMP NULL,
    admin_comments TEXT,
    FOREIGN KEY (student_id) REFERENCES students(student_id) ON DELETE
CASCADE,
    FOREIGN KEY (reviewed_by) REFERENCES users(user_id)
);
```

9.1.10 Sessions Table (for session management)

```
CREATE TABLE user_sessions (
    session_id VARCHAR(255) PRIMARY KEY,
    user_id INT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP,
    is_active BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

9.1.11 Activity_Logs Table (for audit trail)

```
CREATE TABLE activity_logs (
    log_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    action VARCHAR(255) NOT NULL,
    table_affected VARCHAR(100),
    record_id INT,
    old_values JSON,
    new_values JSON,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ip_address VARCHAR(45),
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

9.2 Database Relationships

1. **Users to Students**: One-to-One (A user can be one student)
2. **Users to Course_Trainers**: One-to-Many (A user can train multiple courses)
3. **Courses to Students**: One-to-Many (A course can have multiple students)
4. **Courses to Topics**: One-to-Many (A course can have multiple topics)
5. **Topics to Assignments**: One-to-Many (A topic can have multiple assignments)
6. **Students to Assignment_Submissions**: One-to-Many
7. **Students to Attendance**: One-to-Many
8. **Students to Leave_Applications**: One-to-Many

9.3 Indexes for Performance

```
-- Performance indexes
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_role ON users(role);
CREATE INDEX idx_students_course ON students(course_id);
CREATE INDEX idx_assignments_topic ON assignments(topic_id);
CREATE INDEX idx_assignments_due_date ON assignments(due_date);
CREATE INDEX idx_attendance_date ON attendance(attendance_date);
CREATE INDEX idx_attendance_student ON attendance(student_id);
CREATE INDEX idx_leaves_student ON leave_applications(student_id);
CREATE INDEX idx_leaves_status ON leave_applications(status);
```