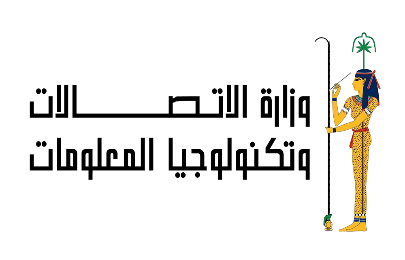
**A logo of a graduate cap and globe

AI-generated content may be incorrect.**

***Digital Egypt Pioneers Initiative - DEPI***

***Project Title: Academix – Learning Management System (LMS)***

**Team Members**

**.1 Abdelrahman Saad Hamouda Hassan**

**.2 Mazen Mostafa Abdelaziz Elbarody**

**3. Mohamed Sherif Kamel Hassan**

**.4 Moataz Mohammed Mohammed Mohammed**

**.5 Ashraf medhat monir khalifa**

# 

# **Project Planning & Management**

## **1.1Project Proposal**

### **Overview**

### The **LMS Project** is a comprehensive **Learning Management and Student Information System (LMS/SIS)** designed as a digital solution for **universities and educational institutions** to manage their academic environments efficiently. It unifies learning, administration, and communication in one platform for **students, instructors, and administrators**.

### Students can access materials, track progress, and receive **AI-generated quizzes and personalized support**. Instructors can create and manage courses, upload content, assess students, and communicate through an intuitive dashboard. Administrators oversee user management, academic data, reporting, and system operations.

### Equipped with **AI-powered tools** like a **quiz generator** and **course-scoped assistant**, the system enhances learning quality, automates repetitive tasks, and supports personalized education. Built for **scalability, security, and usability**, the LMS empowers universities to streamline processes, improve engagement, and provide accessible, data-driven learning experiences for all users anytime and anywhere.

### **Objectives:**

* Facilitate access to education anytime and anywhere.
* Organize course management with tools to create, edit, and delete content.
* Enable students to track progress, take assessments.
* Support instructors with tools to upload content, monitor learners, and provide feedback.
* Strengthen administration through reports and role-based access control.
* Enhance user experience with intuitive UI/UX design.
* Ensure security and reliability with safe login and data backups.

### 

### **Scope:**

The scope of the *Academix – Learning Management System (LMS)* includes the design and development of an integrated digital platform that supports online learning. The system will provide functionalities for students, instructors, and administrators.

### 

### **Included (Frontend):**

* User authentication (Login, Reset Password)
* Dashboard with progress tracking
* Course catalog and course details
* Assignment listing and submission
* Grades overview
* Announcements.
* Account settings and profile management

### 

### **Included (Backend):**

* User management (students, instructors, admins)
* Course and content management (create, update, delete)
* Assignment submission handling & grading system
* Database integration with SQL Server
* Reporting and analytics (progress, grades, usage)
* Notification and messaging system
* Role-based access control (RBAC)
* Security (data protection, backups, authentication)

## 

## **Project Plan (Gantt Chart Summary)**

|  |  |  |
| --- | --- | --- |
| Phase | Task | Duration |
| Planning & Analysis | Project proposal, requirements gathering, scope definition | Week 1 |
| Design | UI/UX design, wireframes, database schema design | Weeks 2 – 3 |
| Frontend Development | HTML, CSS, JS, Bootstrap implementation | Weeks 4 – 5 |
| Backend Development & AI Integration | C#, SQL Server integration, API development | Weeks 6 – 7 |
| Testing & QA | Unit testing, integration testing, bug fixing | Week 8 |
| Deployment | Hosting setup, deployment to server, final documentation | Week 9 |

### **Milestones**

1. Project Initiation & Proposal – Approving objectives, scope, and requirements *(Week 1)*
2. System Design – UI/UX design, wireframes, and database schema *(Weeks 2–3)*
3. Frontend Development – Implementing core pages (Login, Dashboard, Courses, Assignments) *(Weeks 4–5)*
4. Backend Development – Building database, APIs, and system logic *(Weeks 6–7)*
5. System Integration & Testing – Connecting frontend with backend, performing QA *(Week 8)*

### 

### **Deliverables**

|  |  |
| --- | --- |
| Delivery Date | Deliverable |
| Project Proposal Document | End of Week 1 |
| UI/UX Design Package | End of Week 3 |
| Frontend Implementation | End of Week 5 |
| Backend System | End of Week 7 |
| Testing Reports | Week 8 |

## **1.3 Task Assignment & Roles**

|  |  |  |
| --- | --- | --- |
| Team Member | Role | Responsibilities |
| Moataz Mohammed Mohammed | Frontend Developer | - Building user interface with HTML/CSS/JS/Bootstrap  - Responsive design  - Ensuring accessibility |
| Ashraf Medhat Monir Khalifa | Frontend Developer | - Building user interface with HTML/CSS/JS/Bootstrap  - Responsive design  - Ensuring accessibility |
| Mohamed Sherif Kamel Hassan | Backend Developer | - Server-side logic using C# (.NET)  - Database design & queries (SQL Server)  - API development |
| Mazen Mostafa Abdelaziz Elbarody | QA / Tester | - Functional testing  - Security & performance testing  - Bug reporting & validation |
| Abdelrahman Saad Hamouda Hassan | Backend Developer | - Server-side logic using C# (.NET)  - Database design & queries (SQL Server)  - API development |

## **1.4 Risk Assessment & Mitigation Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Likelihood | Impact | Mitigation Strategy |
| Data Loss on Local Computer (accidental deletion, corruption) | High | Loss of project files | Use cloud storage (Google Drive, OneDrive, GitHub), schedule regular backups |
| Hardware Failure (computer breakdown) | Medium | Work interruption, delays | Design frontend with clear API contracts; use mock data with fetch() placeholders |
| Communication Challenges (Online Study) | Low | Misunderstandings, delays in collaboration | Use clear communication tools (Teams, Zoom, Slack), schedule regular team meetings, maintain shared documents |
| Time constraints | High | High | Prioritize tasks, clear communication, regular progress tracking |

## **1.5. KPIs (Key Performance Indicators)**

|  |  |  |
| --- | --- | --- |
| KPI | Description | Target Value |
| System Uptime | Percentage of time the system is available and running | 99% |
| Response Time | Average page load time and system response | 3 seconds |
| User Adoption Rate | Percentage of students & instructors actively using the system | ≥ 80% within 3 months |
| Assignment Submission Rate | Students submitting assignments on time | ≥ 85% |
| User Satisfaction | Measured via surveys & feedback | ≥ 4/5 |
| Support Tickets Resolution | Percentage of resolved tickets in time | ≥ 90% |

# 

# **2. Literature Review**

**2.1 Feedback & Evaluation**

-The relationship between the quizzes table and the courses table is not accurate

-No table shows any information about semesters

-We made a separate table for roles

-There wasn’t a table containing the formats of notifications

**2.2 Suggested Improvements:**

-Modify the relationship between the quizzes table and the courses table

-Make a table and name it “Academic Terms” to show the details of the semester

-Involve roles in the users table, not in a separate table

-Make a table for Notifications templates

-Add attribute to the student profile table that shows the admission date

**2.3 Final Grading Criteria:**

### **Documentation**

Quality and completeness of the project documentation. This includes system analysis, ERD, data model, design diagrams, grading criteria, and overall structure. Clear writing, proper formatting, and version control are also evaluated **(25 Marks)**

### **Implementation**

Functionality and correctness of the developed LMS system. Code quality, adherence to design specifications, proper use of database relations, backend integration, and clean UI are considered **(35 Marks)**

### **Testing**

Coverage and depth of test cases. Includes functional testing, unit testing, database validation, edge case handling, and evidence of bug fixing. Proper documentation of testing results is expected **(20 Marks)**

### **Presentation**

Clarity, organization, and professionalism of the final presentation. This includes demo quality, communication skills, visual design of slides, teamwork, and ability to answer questions confidently **(20 Marks)**

# 

# **3. Requirements** **Gathering**

## **3.1 Stakeholders Analysis:**

### **1. Admin**

The admin is responsible for the overall supervision of the system. They manage users, monitor data, update general settings, and have full access to all sections to ensure smooth and organized operations.

### **2. Staff**

Staff members handle the daily operational tasks within the system. They enter data, follow up on requests or bookings, and communicate with other users. Their role is essential for maintaining efficient workflows.

### **3. Instructors**

They offer content or services through the platform (e.g., courses, sessions, or products). They manage their profiles, upload content or schedule sessions, and engage with users.

### **4. Technical Support Team**

This team is responsible for solving technical issues, maintaining the system, and ensuring that the platform runs smoothly. They intervene in case of malfunctions and work on regular system updates.

### **5. IT Department**

The IT team develops and maintains the database and system infrastructure. They handle information security, backups, and permission management to ensure system stability and data protection.

### **6**. **Students**

Students are one of the main user groups in the system. They enroll in courses, access learning materials, submit assignments, take quizzes, and track their academic progress. Their interaction with the system is central to the learning process.

## 

## **3.2 Use case and user story scenarios:**

3.2.1 use case:

1-Student submits assignment

A diagram of a course

AI-generated content may be incorrect.

2- Instructor creates course

A diagram of a course

AI-generated content may be incorrect.

3- Administrator change roles

A diagram of a system

AI-generated content may be incorrect.

A diagram of a person

AI-generated content may be incorrect.

**3.2.2 user story:**

1- Instructor can create a quiz

**A screenshot of a diagram

AI-generated content may be incorrect.**

2- Administrator can generate reports

A screenshot of a computer

AI-generated content may be incorrect.

## 

## **3.3 Functional Requirements**

### **User Authentication and Authorization**

* The system must allow users (students, instructors, and admins) to log in securely using their credentials.
* The system should provide role-based access (e.g., admin, instructor, student).

### **Course Management**

* Instructors can create, update, and delete courses.
* Admins can manage course categories and assign instructors.

### **Lesson and Content Management**

* Instructors can upload learning materials (videos, PDFs, links, assignments, Quizzes).
* Students should be able to view and download course materials.
* The system must support various file formats.

### 

### **Quiz and Assessment Management**

* Instructors can create and manage quizzes for their courses.
* Students can attempt quizzes and view their results.
* The system automatically grades objective questions (e.g., MCQs).

### **Enrollment and Academic Term Management**

* Students can take courses based on available academic terms.
* Admins can define and manage academic terms.

### **Grading and Reporting**

* The system should calculate grades automatically and display results to students.
* Instructors can modify or override grades if needed.
* Admins can generate performance reports for students and instructors.

### **Notifications and Announcements**

* The system should send notifications to students about upcoming quizzes, new materials, or course updates.
* Instructors can post announcements visible to enrolled students.

### **User Profile Management**

* Each user can view and update their profile information.
* The system should store details like name, email, role, and activity logs.

## 

## **3.4 Non-Functional Requirements**

### **.1Performance**

* The system should handle up to 500 concurrent users without performance degradation.
* Page load time should not exceed 3 seconds under normal usage.

### **.2Scalability**

* The system should be designed to scale horizontally (e.g., by adding more servers) as user numbers grow.

### **.3Security**

* All passwords must be stored securely (e.g., hashed and salted).
* User sessions should expire after a period of inactivity.
* The system must use HTTPS to secure data transmission.

### **.4Usability**

* The interface should be intuitive and consistent across all pages.
* Navigation should be simple for all user types.

### **.5Reliability and Availability**

* The system should be available 99% of the time.
* Regular backups must be scheduled to prevent data loss.

### **.6Maintainability**

* The system should be modular to allow easy updates or feature additions.
* Documentation should be maintained for both developers and administrators.

### **.7Compatibility**

* The system must work on major browsers (Chrome, Edge, Firefox).
* The interface should be responsive and accessible on mobile devices.

# **4. System Analysis & Design**

## **4.1 Problem Statement & Objectives**

**Problem Statement:**

Educational institutions often struggle with managing online learning processes in a centralized and secure manner. The core problem is the lack of a unified system that allows instructors to publish content and manage assessments, enables students to access learning materials and track their progress, and provides administrators with full control over users and reporting. Existing solutions either complicate the process or inadequately meet the fundamental requirements.

**Objectives:**

* To provide a unified and easy-to-use e-learning platform.
* To enable instructors to create and manage course content and assessments.
* To enable students to access courses, track their academic progress, and obtain certificates.

**4.2 Use Case Diagram & Descriptions**

**Actors:**

* **Student:** Can view courses, take courses, view lessons, submit assignments, take quizzes, view grades, update profile.
* **Instructor:** Can create and manage courses, manage lessons and content, create and manage quizzes and assignments, grade student submissions, post announcements.
* **Administrator:** Can manage all users (create, update, delete), manage all courses, define academic terms, generate system-wide reports, manage system settings.

**4.3 Sample Use Case Descriptions:**

**Use Case: Enroll in a Course**

Actor: Student

Description: A student views the available courses for the current academic term and selects one to enroll in.

**Use Case: Grade an Assignment**

Actor: Instructor

Description: An instructor views a list of submitted assignments for their course, selects one, assigns a grade, and provides feedback.

A diagram of a person

AI-generated content may be incorrect.

**Software Architecture:**

**Architecture Style:** The system will follow the Model-View-Controller (MVC) pattern. This separates the application's data model, user interface, and control logic, promoting organized code, easier maintenance, and parallel development.

## **High-Level Design:**

* **Presentation Layer (View):** Comprises HTML, CSS, JavaScript, and Bootstrap pages. This layer handles user interaction and displays data received from the server.
* **Application Layer (Controller):** Built using C# with ASP.NET Core. This layer processes HTTP requests, contains the business logic, interacts with the data layer, and sends responses back to the presentation layer.
* **Data Layer (Model):** Consists of a Microsoft SQL Server database. This layer is responsible for storing and retrieving all system data, including user information, courses, grades, and announcements.

**4.4 Database Design & Data Modeling**

**4.4.1 ER Diagram (Entity-Relationship Diagram):**

**Entities (Tables):** Users, Roles, AcademicTerms, Courses, Enrollments, Lessons, Assignments, Submissions, Quizzes, Questions, Grades, Announcements, Notification Templates.

## **Key Relationships:**

* A User has one Role (Student, Instructor, Admin). (Incorporating the feedback: RoleId is a foreign key in the Users table).
* An AcademicTerm has many Courses.
* A Course belongs to one AcademicTerm and is taught by one Instructor (a User with Instructor role).
* A Student (a User) can have many Enrollments in many Courses.
* A Course has many Lessons, Assignments, and Quizzes.
* A Quiz has many Questions.
* A Student can have many Submissions for Assignments and Quizzes.
* A Submission can have one Grade.

**4.4.2 Logical & Physical Schema:**

**Users Table:** UserId (PK, int), Username (nvarchar), Email (nvarchar), PasswordHash (nvarchar), RoleId (FK, int), FirstName (nvarchar), LastName (nvarchar), AdmissionDate (date, for students), DateJoined (datetime).

**AcademicTerms Table:** TermId (PK, int), TermName (nvarchar; e.g., 'Fall 2025'), StartDate (date), EndDate (date).

**Courses Table**: CourseId (PK, int), CourseName (nvarchar), Description (text), InstructorId (FK to Users), TermId (FK to AcademicTerms).

**Enrollments Table:** EnrollmentId (PK, int), StudentId (FK to Users), CourseId (FK to Courses), EnrollmentDate (datetime).

**Lessons Table:** LessonId (PK, int), CourseId (FK to Courses), Title (nvarchar), ContentUrl (nvarchar), OrderNumber (int).

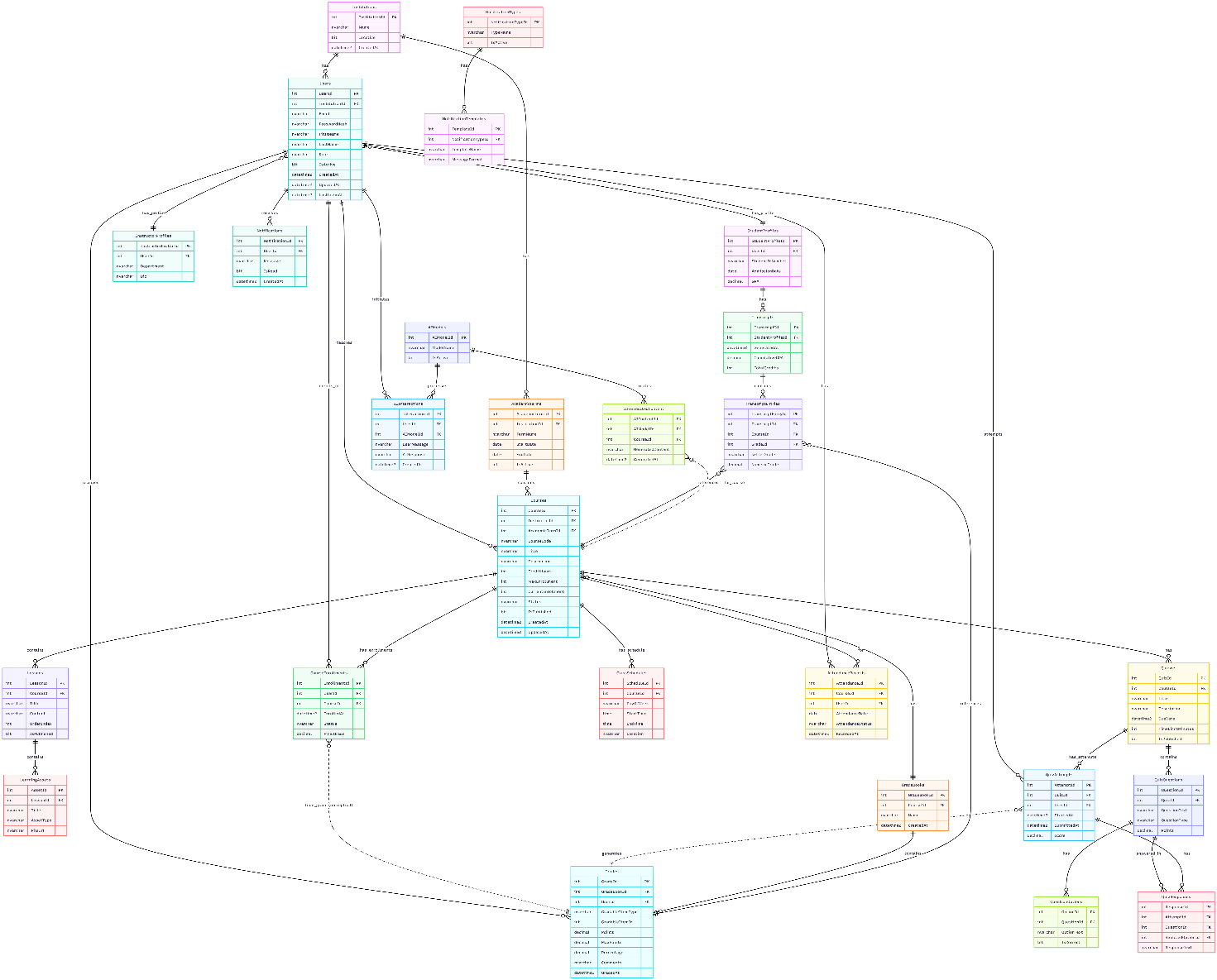
**Assignments Table:** AssignmentId (PK, int), CourseId (FK to Courses), Title (nvarchar), Description (text), DueDate (datetime).

**Quizzes Table:** QuizId (PK, int), CourseId (FK to Courses), Title (nvarchar), TimeLimit (int).

**Submissions Table:** SubmissionId (PK, int), AssignmentId (FK to Assignments, nullable), QuizId (FK to Quizzes, nullable), StudentId (FK to Users), SubmissionDate (datetime), FileUrl (nvarchar), AnswerText (text).

**Grades Table:** GradeId (PK, int), SubmissionId (FK to Submissions), Score (decimal), Feedback (text), GradedById (FK to Users, Instructor), GradedDate (datetime).

**NotificationTemplates Table:** TemplateId (PK, int), TemplateName (nvarchar), Subject (nvarchar), Body (text).

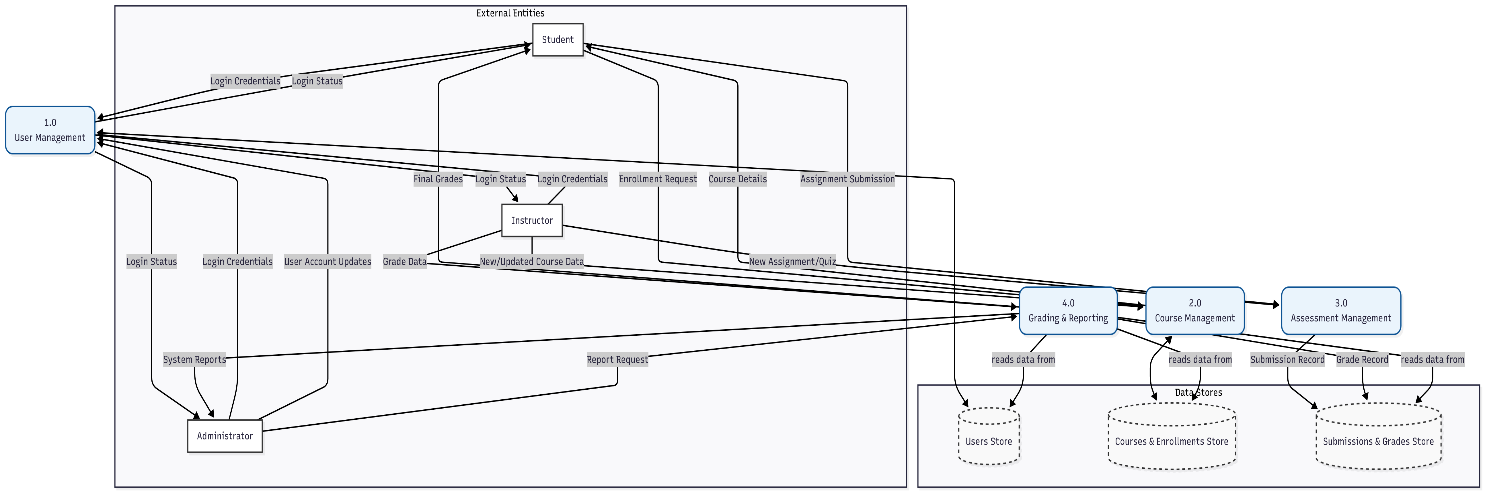


* 1. **Data Flow & System Behavior**

**4.5.1 DFD (Data Flow Diagram)**

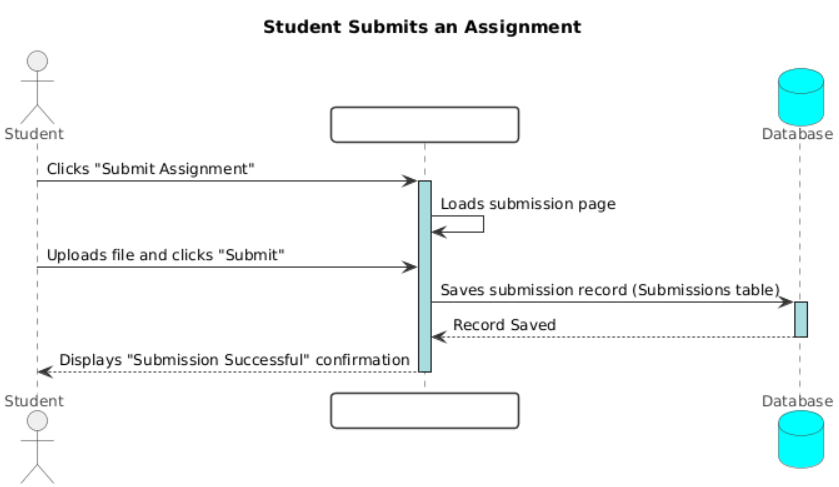
External Entities: Student, Instructor, Administrator. Process: Academix LMS System.

Data Flows: Login Credentials, Course Enrollment Request, Course Data, Assignment Submission, Grades, Reports.



**4.5.2 Sequence Diagrams:**

1. **Student Submits an Assignment:**

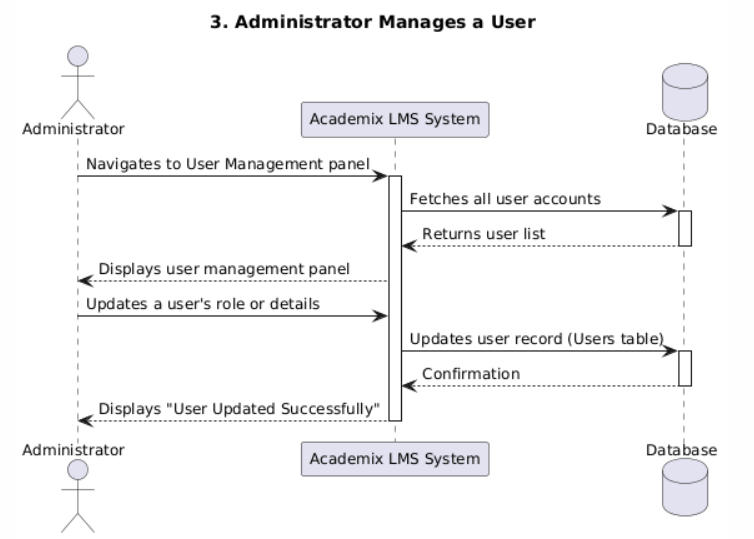


1. **Instructor Grades an Assignment**

A diagram of a assignment

AI-generated content may be incorrect.

1. **Administrator Manages a User**



**4.5.3 Activity Diagrams:**

**1. Student Submitting an Assignment**A diagram of a course

AI-generated content may be incorrect.

1. **Instructor Grading an Assignment**

**A diagram of a course

AI-generated content may be incorrect.**

**4.5.4 State Diagram:**

1. **Assignment Lifecycle**

**A diagram of a student's process

AI-generated content may be incorrect.**

**4.5.5 Class Diagram**

**Main Classes:**

User (Attributes: UserId, Username, Email, PasswordHash; Methods: Login(), UpdateProfile())

Student (Inherits from User; Attributes: AdmissionDate; Methods: SubmitAssignment(), ViewGrades())

Instructor (Inherits from User; Methods: CreateCourse(), GradeSubmission())

Course (Attributes: CourseId, CourseName, Description; Methods: AddLesson())

Assignment (Attributes: AssignmentId, Title, DueDate; Methods: Submit())

Grade (Attributes: GradeId, Score, Feedback)

Relationships: Association (e.g., Instructor teaches Course), Inheritance (Student and Instructor inherit from User), Composition (Course contains Lesson).

A screenshot of a computer

AI-generated content may be incorrect.

* 1. **UI/UX Design & Prototyping**

**4.6.1 Wireframes & Mockups**

**Key Screens:** Login Page, Student Dashboard (showing enrolled courses, upcoming deadlines), Instructor Dashboard (showing taught courses, to-grade list), Course Detail Page (with lessons list, assignments, announcements), Assignment Submission Page, Admin User Management Panel.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Design Tool**: Figma or Adobe XD will be used to create high-fidelity mockups.

**4.6.2 UI/UX Guidelines:**

**Design Principles**: Clarity, Consistency, Efficiency.

**Color Scheme:**

Primary Color: Blue (#2E5AAC) — conveys trust and calm.

Supporting Colors: Neutral grays for backgrounds and text.

State Colors: Green (#28A745) for success, Red (#DC3545) for errors or destructive actions.

The system also includes a Light and Dark Mode palette for user comfort in different lighting environments.

**Typography**:

Primary Font: "Inter", sans-serif — chosen for its modern, clean appearance and readability across screen sizes.

Font Hierarchy: Headings range from 24–32px, while body text uses 16px for readability.

**Accessibility:**

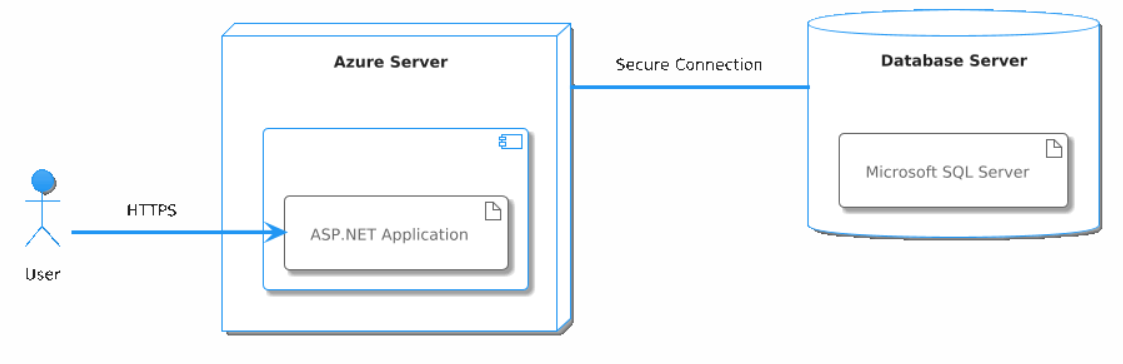
Ensure all interactive elements are keyboard navigable. Provide sufficient color contrast (minimum WCAG AA standard). Use alt text for all informative images.

* 1. **System Deployment & Integration**

**4.7.1 Technology Stack**

* Frontend: HTML5, CSS3, JavaScript, Bootstrap 5.
* Backend: C#, ASP.NET Core MVC.
* Database: Microsoft SQL Server.
* Version Control: Git, GitHub.

**4.7.2 Deployment Diagram**

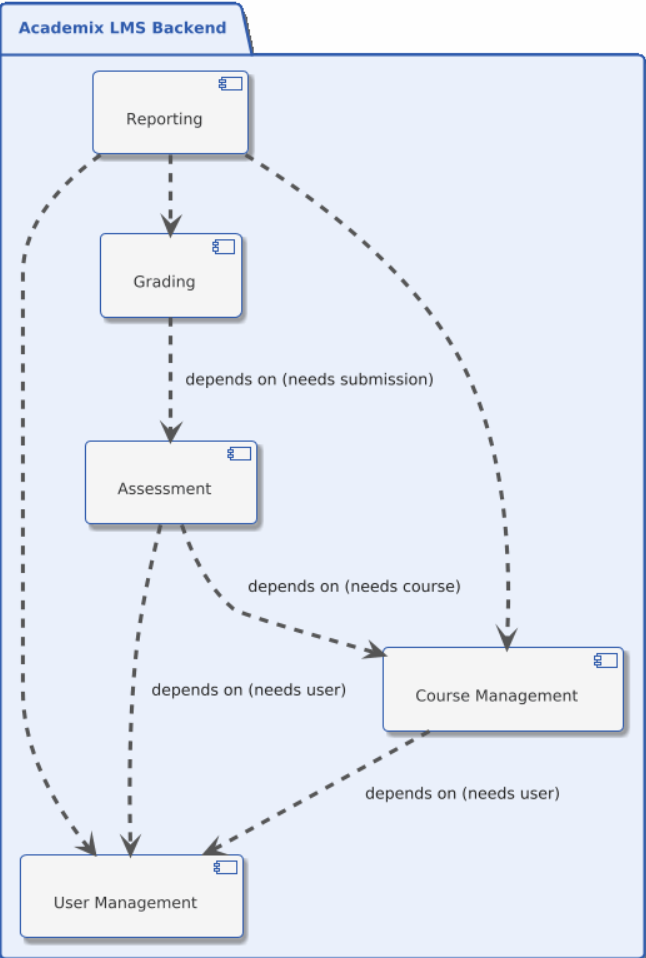
****

**4.7.3 Component Diagram**

## **High-Level Components:**

* User Management Component: Handles authentication and user profiles.
* Course Management Component: Handles courses, lessons, and enrollment.
* Assessment Component: Handles assignments, quizzes, and submissions.
* Grading Component: Handles grading logic and feedback.
* Reporting Component: Generates reports for administrators.

Dependencies: The Assessment Component depends on the User Management Component (to identify students) and the Course Management Component (to link assignments to courses).



* 1. **Additional Deliverables**

## **4.8.1 API Documentation**

After Web API is developed, documentation will be provided using Swagger/OpenAPI.

**Example Endpoint:** POST /api/assignments/{id}/submit

**Description:** Submits an assignment for a student. **Parameters:** Assignment ID, Student ID, Submission File. **Response:** Success confirmation or error message.

**4.8.2 Testing & Validation**

**Unit Testing**: xUnit or NUnit for testing individual backend methods (e.g., grade calculation logic).

**Integration Testing**: Testing the interaction between the frontend, backend API, and database.

**User Acceptance Testing (UAT) Plan:** A set of test scenarios will be given to a select group of students, instructors, and administrators to ensure the system meets their needs before final deployment.

**4.8.3 Deployment Strategy**

**Hosting Environment**: A cloud platform such as Microsoft Azure. The web app will be hosted on Azure App Services, and the database on Azure SQL Database.

**Deployment Pipelines:** Use GitHub Actions for CI/CD (Continuous Integration/Continuous Deployment). Automate build and deployment processes upon pushing code to the main branch.

**Scaling Considerations:** Azure App Services and Azure SQL Database can be scaled vertically (more powerful resources) and horizontally (more instances) based on user load

# **5. Implementation (Source Code & Execution)**

## **5.1 Source Code**

1. **Structured & Well-Commented Code**

* Writing clean and well-structured code with clear comments explaining the purpose of each module/function and any non-trivial logic, including important components such as User Management and Courses.

1. **Coding Standards & Naming Conventions**

* LMS: Define consistent naming for all tables/models such as Course, Lesson, Module, Enrollment, Assignment, and Submission to standardize the codebase and database.

1. **Modular Code & Reusability**

* Divide the application into small, reusable components/classes and functions, ensuring proper Separation of Concerns.

1. **Security & Error Handling**

* Protect course content resources, validate file size/type for uploads, encrypt passwords and manage sessions, enforce login attempt limits, and verify user roles.

## **5.2 Version Control & Collaboration**

1. **Version Control Repository**

* Host the project on a version control platform (GitHub/GitLab/Bitbucket) and provide the repository link (public or private according to policy).

1. **Branching Strategy**

* Use separate branches for each major feature (e.g., feature/assessment-engine, feature/scorm-integration).

1. **Commit History & Documentation**

* Attach test scenarios to each PR related to learning flows (e.g., student enrollment, assignment submission, automated grading).

1. **CI/CD Integration (if applicable)**

* Not applicable

## 

## **5.3 Deployment & Execution**

**README File (should include)**

1. **Installation steps:**

* Download the project and open it using Visual Studio.
* Restore NuGet Packages.
* Build the project.

1. **System requirements (Hardware/Software):**

* Operating System: Windows 10/11
* Software:
  + - .NET 6/7 SDK
    - SQL Server 2019 or later
    - Visual Studio 2022
* Hardware:
  + - RAM: 8 GB or higher
    - Processor: Intel i5 or higher

1. **Configuration instructions:**

* Update the appsettings.json file to set the database connection string.
* Apply database migrations.

1. **Execution guide:**

* Run locally: Launch the project in Visual Studio (F5).
* Deployed version: (Link will be provided after deployment).

1. **API documentation (if applicable):**

* (Swagger/Endpoints details will be added after implementation).

# **Testing & Quality Assurance**

This section outlines the testing plan for the LMS project, focusing on verifying major features and ensuring the system functions correctly across all roles.

## **6.1 Test Cases & Test Plan**

**Student Scenarios**

|  |  |  |
| --- | --- | --- |
| **ID** | **Scenario** | **Expected Result** |
| TC01 | Student logs in with valid credentials | Redirected to dashboard successfully |
| TC02 | Student logs in with invalid credentials | Error message shown: “Invalid credentials” |
| TC03 | Student views course materials | Course content loads correctly |
| TC04 | Student submits assignment | Confirmation message displayed |
| TC05 | Student views grades | Grades displayed accurately |

**Instructor Scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** |  | **Scenario** | **Expected Result** |
| TC06 |  | Instructor logs in | Redirected to instructor dashboard |
| TC07 |  | Instructor creates a course | Course appears in the course list |
| TC08 |  | Instructor grades an assignment | Grade saved and visible to student |
| TC09 |  | Instructor sends announcement | Students receive notification instantly |

**Admin Scenarios**

|  |  |  |
| --- | --- | --- |
| **ID** | **Scenario** | **Expected Result** |
| TC10 | Admin logs in | Redirected to admin dashboard |
| TC11 | Admin updates user role | Change saved successfully |
| TC12 | Admin views reports | Reports displayed correctly |
| TC13 | Admin sends platform announcement | All users receive notification |

**AI & System Scenarios**

|  |  |  |
| --- | --- | --- |
| **ID** | **Scenario** | **Expected Result** |
| TC14 | AI generates quiz | Quiz created for selected topic |
| TC15 | AI assistant answers question | Relevant answer displayed |
| TC16 | Unauthorized access attempt | Redirected to login screen |
| TC17 | High traffic load | System remains stable and responsive |

## **6.2 Automated Testing (Planned)**

* **Postman** for API verification
* **Selenium** for UI interaction tests
* **GitHub Issues** for bug tracking and progress reporting

Contents

[**.1Project Planning & Management** 2](#_Toc210945045)

[**.1.1Project Propos** 2](#_Toc210945046)

[**Overview** 2](#_Toc210945047)

[**Objectives:** 2](#_Toc210945048)

[**Scope:** 2](#_Toc210945049)

[**Included (Frontend):** 2](#_Toc210945050)

[**Included (Backend):** 3](#_Toc210945051)

[**1.2. Project Plan (Gantt Chart Summary)** 3](#_Toc210945052)

[**Milestones** 4](#_Toc210945053)

[**Deliverables** 4](#_Toc210945054)

[**1.3. Task Assignment & Roles** 5](#_Toc210945055)

[**1.4. Risk Assessment & Mitigation Plan** 5](#_Toc210945056)

[**1.5. KPIs (Key Performance Indicators)** 6](#_Toc210945057)

[**2. Literature Review:** 7](#_Toc210945058)

[**.2.1Feedback & Evaluation:** 7](#_Toc210945059)

[**.2.2Suggested Improvements:** 7](#_Toc210945060)

[**.2.3Final Grading Criteria:** 7](#_Toc210945061)

[**Documentation** 7](#_Toc210945062)

[**Implementation** 7](#_Toc210945063)

[**Testing** 7](#_Toc210945064)

[**Presentation** 7](#_Toc210945065)

[.**3Requirements** **Gathering**: 8](#_Toc210945066)

[**.3.1Stakeholders Analysis:** 8](#_Toc210945067)

[**1. Admin** 8](#_Toc210945068)

[**2. Staff** 8](#_Toc210945069)

[**3. Instructors** 8](#_Toc210945070)

[**4. Technical Support Team** 8](#_Toc210945071)

[**5. IT Department** 8](#_Toc210945072)

[**6**. **Students** 8](#_Toc210945073)

[**.3.2Use case scenarios:** 9](#_Toc210945074)

[**.3.3Functional Requirements** 10](#_Toc210945075)

[**1.** **.1User Authentication and Authorization** 10](#_Toc210945076)

[**2.** **Course Management** 10](#_Toc210945077)

[**3.** **Lesson and Content Management** 10](#_Toc210945078)

[**4.** **Quiz and Assessment Management** 10](#_Toc210945079)

[**5.** **Enrollment and Academic Term Management** 10](#_Toc210945080)

[**6.** **Grading and Reporting** 10](#_Toc210945081)

[**7.** **Notifications and Announcements** 10](#_Toc210945082)

[**8.** **User Profile Management** 10](#_Toc210945083)

[**.3.4Non-Functional Requirements** 11](#_Toc210945084)

[**.1Performance** 11](#_Toc210945085)

[**.2Scalability** 11](#_Toc210945086)

[**.3Security** 11](#_Toc210945087)

[**.4Usability** 11](#_Toc210945088)

[**.5Reliability and Availability** 11](#_Toc210945089)

[**.6Maintainability** 11](#_Toc210945090)

[**.7Compatibility** 11](#_Toc210945091)

[**5. Implementation (Source *Code* & Execution)** 12](#_Toc210945092)

[**.5.1Source Code** 12](#_Toc210945093)

[**1.** **Structured & Well-Commented Code** 12](#_Toc210945094)

[**2.** **Coding Standards & Naming Conventions** 12](#_Toc210945095)

[**3.** **Modular Code & Reusability** 12](#_Toc210945096)

[**4.** **Security & Error Handling** 12](#_Toc210945097)

[**.5.1Version Control & Collaboration** 12](#_Toc210945098)

[**1.** **Version Control Repository** 12](#_Toc210945099)

[**2.** **Branching Strategy** 12](#_Toc210945100)

[**3.** **Commit History & Documentation** 12](#_Toc210945101)

[**4.** **CI/CD Integration (if applicable)** 12](#_Toc210945102)

[**.5.3Deployment & Execution** 12](#_Toc210945103)

[**1.** **Installation steps:** 12](#_Toc210945104)

[**2.** **System requirements (Hardware/Software):** 13](#_Toc210945105)

[**3.** **Configuration instructions:** 13](#_Toc210945106)

[**4.** **Execution guide:** 13](#_Toc210945107)

[**5.** **API documentation (if applicable):** 13](#_Toc210945108)

[**6. Testing & Quality Assurance** 14](#_Toc210945109)

[**6.1 Test Cases & Test Plan** 14](#_Toc210945110)

[**6.2 Automated Testing (Planned)** 15](#_Toc210945111)