مسار | Masar

Every Code is a Step Forward

1. Introduction

1.1 Purpose

The purpose of this document is to specify the requirements for an **online learning platform specialized in programming education**. The platform provides structured **learning tracks (roadmaps)**, enabling learners to follow a step-by-step path toward mastering specific fields such as Web Development, Data Science, or Artificial Intelligence.

The system allows **students** to consume courses, track their progress, and validate their knowledge through quizzes and practical assignments. **Instructors** can create and manage courses, add multimedia content (videos, PDFs, articles), and design assessments. **Administrators** are responsible for managing users, courses, and the overall platform configuration

1.2 Scope

The system will be a **web-based learning management platform** focused on programming education. Its primary features include:

- **Courses & Content Management**: Upload and organize courses into modules and lessons.
- **Learning Tracks (Roadmaps)**: Provide predefined paths that guide learners step by step.
- **Quizzes & Assignments**: Assess learners' knowledge through auto-graded quizzes and coding assignments.
- **Gamification**: Engage learners using levels, badges, and a ranking system.
- **Certificates**: Issue verifiable certificates (PDF with QR/Link) upon completion of tracks or courses.
- **Progress Tracking**: Show learners their advancement across courses and tracks.

The platform aims to simplify programming education by combining structured paths, practice-based learning, and community-driven motivation, making it easier for students to stay engaged and measure their progress.

1.3 Definitions, Acronyms, and Abbreviations

- **LMS**: Learning Management System.
- **Track (Roadmap)**: A structured learning path consisting of a sequence of courses/modules designed for a specific goal.
- **Module**: A section of a course containing multiple lessons.
- **Lesson**: A unit of learning content (video, PDF, or article).
- **Assignment**: A practical task (e.g., coding exercise) designed to reinforce learning.
- **Quiz**: An assessment containing questions (MCQ, True/False, Fill-in-the-blank).

---

2. Overall Description

2.1 Product Perspective

The system is a **web-based Learning Management System (LMS)** focused on programming education.
It provides an integrated environment where:

- **Students** can enroll in tracks, consume learning materials, complete quizzes, and track their progress.
- **Instructors** can create and manage courses, upload content, and design quizzes/assignments.
- **Administrators** can manage the platform, users, and content at a global level.

The platform is standalone but can be extended in the future with integrations (e.g., payment gateways, live class tools, cloud-based video storage).

2.2 Product Functions

At a high level, the system provides the following functionality:

- **Course & Track Management**: Upload, organize, and maintain content.
- **Progress Tracking**: Track learner progress at module, course, and track levels.

- **Quizzes & Assignments**: Enable automated assessments and practical coding tasks.
- **Gamification**: Motivate learners through badges, levels, and ranking boards.
- **Certificates**: Provide verifiable certificates upon completion.
- **User Management**: Handle authentication, roles, and permissions.

2.3 User Classes and Characteristics

1. **Administrators**
   - Manage platform configuration, user accounts, and course approvals.
   - Require advanced technical knowledge.
2. **Instructors**
   - Create and manage courses, modules, lessons, quizzes, and assignments.
   - Require ease of content upload and reporting dashboards.
3. **Students**
   - Enroll in tracks and courses, consume lessons, solve quizzes, and track progress.
   - Require a user-friendly and intuitive interface.

2.4 Operating Environment

- **Platform**: Web-based application.
- **Backend**: ASP.NET Core (C#).
- **Frontend**: React or Angular (final choice to be decided).
- **Database**: SQL Server.
- **Authentication**: JWT / Identity.
- **Deployment**: Cloud-hosted (e.g., Azure, AWS) or on-premise server.

2.5 Design and Implementation Constraints

- Must be developed using **ASP.NET Core for backend**.
- Must support **role-based access control** (Admin, Instructor, Student).
- Initial release will focus only on **web** (no mobile apps).
- Video storage may require integration with cloud storage solutions.

2.6 Assumptions and Dependencies

- All content (videos, PDFs, articles) will be uploaded by instructors.
- Users are expected to have stable internet connections for streaming content.
- Payment functionality will not be active in the initial release (simulated only).
- Certificates will be generated automatically using a pre-defined template.
- The system depends on:
    - Database availability (SQL Server).
    - Cloud or server storage for video content.
    - Authentication/authorization services.

---

3. System Features

3.1 Course & Content Management

- **Description**: Instructors can upload and manage course materials including videos, PDFs, and articles. Courses are divided into modules and lessons.
- **Inputs**: Course title, description, content files (video, PDF, text).
- **Processing**: Store content in database/cloud storage, organize under modules & lessons.
- **Outputs**: Published course with structured content.
- **Priority**: High.

3.2 Learning Tracks (Roadmaps)

- **Description**: Courses are grouped into structured tracks (e.g., Web Development, Data Science). Students can enroll in tracks to follow a guided learning path.
- **Inputs**: Track name, description, list of courses.
- **Processing**: Map courses to track sequence, update progress as student completes.
- **Outputs**: Progress tracking per track, recommended next course.
- **Priority**: High.

3.3 Quizzes

- **Description**: Quizzes assess student knowledge after each module.
- **Inputs**: Questions (MCQ, True/False, Fill-in-the-blank), student answers.
- **Processing**: Auto-grading based on predefined correct answers.
- **Outputs**: Score, correct/incorrect feedback, progress update.
- **Priority**: High.

3.4 Practical Assignments

- **Description**: Assignments give students practical tasks (e.g., coding exercises).
- **Inputs**: Assignment instructions, student submissions (code/text).
- **Processing**: Store submissions; auto-grade if rules defined (optional manual review).
- **Outputs**: Feedback, grading (score or pass/fail).
- **Priority**: High.

3.5 Progress Tracking

- **Description**: Track progress at lesson, module, course, and track level.
- **Inputs**: Student activity (completed lessons, quizzes, assignments).
- **Processing**: Update student profile with progress percentage.
- **Outputs**: Visual tracker (progress bar, completion %).
- **Priority**: High.

3.6 Gamification (Levels & Badges)

- **Description**: Students earn badges and levels based on progress and performance.
- **Inputs**: Completion events (e.g., finished course, scored >80% in quiz).
- **Processing**: Check rules, assign badge/level.
- **Outputs**: Badge/level displayed in student profile.
- **Priority**: Medium.

3.7 Ranking System

- **Description**: Leaderboard showing top students based on quiz scores and completion.
- **Inputs**: Student scores and activity logs.
- **Processing**: Calculate ranking by points/achievements.

- **Outputs**: Leaderboard with top 10 students.
- **Priority**: Medium.

3.8 Certificates

- **Description**: Generate certificates for students upon course/track completion.
- **Inputs**: Student name, course/track name, completion date.
- **Processing**: Generate PDF certificate with QR/link for online verification.
- **Outputs**: Downloadable certificate (PDF), verification link.
- **Priority**: High.

3.9 User Management

- **Description**: Manage accounts, roles, and authentication.
- **Inputs**: Registration info, login credentials, role assignments.
- **Processing**: Verify credentials, enforce role-based access.
- **Outputs**: Access granted/denied, user dashboard.
- **Priority**: High.

---

4. External Interface Requirements

4.1 User Interfaces (UI)

- **Web Interface**:
  - Responsive design (works on desktop, tablet, mobile).
  - Student Dashboard: shows enrolled courses, progress, quizzes, certificates.
  - Instructor Dashboard: allows uploading content, creating quizzes, tracking student performance.
  - Admin Dashboard: manage users, courses, tracks, and system analytics.
- **Navigation**:
  - Clear menu (Home, Tracks, Courses, My Learning, Certificates).
  - Search and filter for courses/tracks.

4.2 Hardware Interfaces

- **Server Side**:
  - Standard web server capable of running ASP.NET Core applications.
  - Cloud storage (for video and large files).
- **Client Side**:
  - Any device with a modern web browser (desktop, laptop, tablet, mobile).

4.3 Software Interfaces

- **Backend**: ASP.NET Core (C#).
- **Database**: SQL Server (or any relational DB).
- **Frontend**: To be decided (React / Angular / Vue).
- **Authentication**: Identity system (JWT tokens, OAuth2).
- **Storage**: Cloud storage (Azure Blob / AWS S3 / Google Cloud Storage).
- **Certificate Generation**: PDF library (e.g., iTextSharp).

4.4 Communication Interfaces

- **Protocols**: HTTPS for secure communication.
- **API**: RESTful APIs for client-server communication.
- **Notifications**: Email notifications (SMTP), future extension for push notifications.

---

5. Non-Functional Requirements

5.1 Performance

- The system should support at least 500 concurrent users.
- Response time < 2 seconds for most requests.

5.2 Security

- Encrypted passwords (hashing & salting).
- Role-based access control (Admin, Instructor, Student).
- Secure certificate verification with unique QR code.

5.3 Reliability

- System uptime target: 99%.
- Data backup every 24 hours.

5.4 Usability

- User-friendly UI, easy navigation for non-technical users.
- Multilingual support (future scope).

5.5 Maintainability

- Modular code structure with layered architecture.
- Clear documentation for developers.

5.6 Scalability

- System should allow easy extension to add new tracks/courses.
- Future-ready for cloud scaling.