Software Requirements Specification (SRS) for Quiz System

1. Introduction

1.1 Purpose

The purpose of this SRS document is to provide a detailed description of the requirements for the Quiz System. It includes functional and non-functional requirements, system design, and other critical aspects necessary for the development and implementation of the system.

1.2 Scope

The Quiz System will be an online platform that allows instructors to create and manage quizzes for their courses. Students will be able to take quizzes, and instructors will be able to evaluate student performance.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification
- **API**: Application Programming Interface
- **EF Core**: Entity Framework Core

1.4 References

- Entity Framework Core Documentation
- ASP.NET Core Documentation

1.5 Overview

This document outlines the requirements and specifications for the Quiz System, including user requirements, system architecture, data models, and interface requirements.

2. Overall Description

2.1 Product Perspective

The Quiz System will be a standalone application, accessible via web browsers. It will leverage .NET Core for the backend API, EF Core for database management, and a relational database for data storage.

2.2 Product Functions

- User management (Instructors, Students)
- Course management
- Quiz creation and management
- Question and choice management
- Quiz participation for students

- Result evaluation and feedback

2.3 User Classes and Characteristics

- **Instructors**: Create and manage courses, quizzes, questions, and choices.
- **Students**: Enroll in courses, participate in quizzes, and view their results.

2.4 Operating Environment

- Server: Windows or Linux with .NET Core runtime
- Database: SQL Server or another supported relational database
- Client: Modern web browser

2.5 Design and Implementation Constraints

- The system should use EF Core for database interactions.
- The backend should be developed using ASP.NET Core.
- The system should follow RESTful principles.

2.6 User Documentation

User guides and online help will be provided for both instructors and students.

2.7 Assumptions and Dependencies

- Users have access to the internet and a modern web browser.
- The development team is familiar with .NET Core and EF Core.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Management

- **Instructor Registration/Login**: Instructors can register and log in to the system.
- **Student Registration/Login**: Students can register and log in to the system.

3.1.2 Course Management

- **Create Course**: Instructors can create new courses.
- **Edit Course**: Instructors can edit course details.
- **Delete Course**: Instructors can delete courses.
- **Enroll Student**: Students can enroll in courses.

3.1.3 Quiz Management

- **Create Quiz**: Instructors can create quizzes for their courses.
- **Edit Quiz**: Instructors can edit quiz details.

- **Delete Quiz**: Instructors can delete quizzes.

3.1.4 Question Management

- **Add Question**: Instructors can add questions to quizzes.
- **Edit Question**: Instructors can edit questions.
- **Delete Question**: Instructors can delete questions.

3.1.5 Choice Management

- **Add Choice**: Instructors can add choices to questions.
- **Edit Choice**: Instructors can edit choices.
- **Delete Choice**: Instructors can delete choices.

3.1.6 Quiz Participation

- **Take Quiz**: Students can take quizzes assigned to their courses.
- **Submit Quiz**: Students can submit completed quizzes.

3.1.7 Result Evaluation

- **Evaluate Quiz**: System evaluates the submitted quizzes and provides results.
- **View Results**: Students and instructors can view quiz results.

3.2 Non-Functional Requirements

3.2.1 Performance

- The system should handle up to 100 concurrent users.

3.2.2 Usability

- The user interface should be intuitive and easy to use.

3.2.3 Reliability

- The system should have an uptime of 99.9%.

3.2.4 Security

- All data should be encrypted during transmission.
- Users should be authenticated and authorized.

3.2.5 Maintainability

- The code should follow best practices for clean code and be well-documented.

4. Additional Requirements

Please keep the following:

- Each instructor can view courses, questions and exams he created only
- When creating a question, instructor can determine its level (simple, medium, hard)
- Instructor can use the same question in multiple exams
- Exam has two types (Quiz / Final)
- Instructor should assign a student to a course first, then he can assign exams to him in that course
- Student can take exams for the courses and exams he assigned to only
- Student can take many quiz exams, but he can take only one final exam
- When creating an exam, instructor can determine the number of questions
- When creating an exam, instructor may choose to manually assign questions to the exam or the system assign them automatically
- If the exam is automatically created, the system should balance number of questions with the their level (simple, medium, hard)
- Student can view his result upon completion of the exam
- Instructor can view results for all students in the exam