Software Requirements Specification (SRS)

Quiz Management and Examination System (Streamlit Web Application) Version 1.0 Prepared by: omar sayed team **Date:** 21-Nov-2025

---

1. Introduction

1.1 Problem Definition

Traditional quiz creation and exam administration often rely on manual or semi-manual processes such as printed exams, simple Google Forms, or disconnected tools that do not provide real-time control, secure question management, or consistent evaluation.

These methods introduce several challenges:

- Difficulty creating and managing multiple exams and question banks.
- Lack of a unified system for teachers and students.
- No automated scoring with detailed analytics.
- Limited tracking of students' performance.
- No centralized storage or role-based access control.

As the need for digital examinations grows—especially in universities and training centers—a system is required to support digital quiz creation, timed examinations, automated scoring, and performance analytics through a modern and user-friendly interface.

The Quiz Management and Examination System addresses these issues by providing a centralized web platform built using Python + Streamlit for teachers and students. It enables easy exam creation, secure question storage, timed assessments, automated scoring, and student performance reports.

1.2 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define all functional and non-functional requirements for the Quiz Management and Examination System.

This document ensures that:

- Developers understand system modules and interactions.
- Testers can validate features and behaviors.
- Instructors/clients can confirm that the system fulfills the educational objectives.

The document describes:

- Quiz creation and management features.
- Student exam-taking workflow.
- Authentication, timing, scoring, and report generation.
- Streamlit-based user interface.
- Database design and dependencies.

1.3 Scope

The Quiz Management and Examination System is a web-based Streamlit application designed for teachers and students in academic environments.

User Roles

- **Teacher (Admin):** Creates and manages exams, questions, and results.
- **Student:** Takes exams, views scores, and tracks performance.
- **System Admin (optional future role):** Manages users and system configurations.

Core Features

- User registration and login.
- Exam creation, editing, and deletion.
- Question bank with multiple-choice questions.
- Timed examinations (per question or full exam timer).
- Automated scoring and incorrect answer breakdown.
- Student performance reports.
- Secure database storage (SQLite).
- Web interface built using Streamlit.

Not Included in This Version

- Payment systems.
- AI-based question generation.
- Multimedia question types.
- Mobile application.

1.4 Definitions, Acronyms, and Abbreviations

Term	Definition
SRS	Software Requirements Specification
UI	User Interface
DBMS	Database Management System
CRUD	Create, Read, Update, Delete
Exam	A structured set of questions assigned to students
MCQ	Multiple-Choice Question
Student	User who takes exams

Teacher	User who creates exams
Streamlit	Python framework used to build the web application

3. Functional Requirements

3.1 Authentication

- FR1: The system shall allow users (teachers and students) to log in using a username and password.
- FR2: The system shall enforce role-based access control.

3.2 Exam Management (Teacher)

- FR3: The system shall allow teachers to create exams.
- FR4: The system shall allow teachers to add questions to an exam.
- FR5: The system shall allow teachers to edit or delete existing exams.
- FR6: The system shall store exam details in the database.

3.3 Question Bank

- FR7: Teachers shall be able to create multiple-choice questions.
- FR8: The system shall store all questions securely in the database.

3.4 Exam Taking (Student)

- FR9: The system shall display available exams to students.
- FR10: The system shall start a countdown timer once the exam begins.
- FR11: Students shall be able to answer each question once.
- FR12: The system shall auto-submit the exam when the timer expires.

3.5 Scoring and Results

- FR13: The system shall automatically grade all exams.
- FR14: The system shall store the exam results.
- FR15: The system shall provide students with an immediate score and incorrect answer breakdown.

3.6 Reporting (Teacher)

- FR16: The system shall provide performance reports for each exam.

---

4. Non-Functional Requirements

4.1 Performance Requirements

- NFR1: All pages should load within 2 seconds.
- NFR2: The system should support up to 200 concurrent users.

4.2 Security Requirements

- NFR3: All passwords must be stored in hashed format.
- NFR4: Only teachers may create or manage exams and questions.

4.3 Usability Requirements

- NFR5: The interface must be simple and intuitive for non-technical users.
- NFR6: Students should be able to complete exams without assistance.

4.4 Reliability Requirements

- NFR7: The system must frequently save progress to prevent data loss.
- NFR8: The system should maintain 99% availability.

4.5 Maintainability Requirements

- NFR9: The code must follow a modular and extensible structure.
- NFR10: The database must support future feature expansion.

4.6 Portability Requirements

- NFR11: The system must operate on any device with a modern browser.

---

6. Use Case Specifications

UC01 – User Login

Actor: Teacher / Student
Precondition: User has an existing account.
Main Flow:

1. User enters username and password.
2. System validates credentials.

3. User is directed to their role-specific dashboard.
   Postcondition: User is authenticated. **Exception:** Invalid login credentials.

UC02 – Create Exam

Actor: Teacher
Precondition: Teacher is logged in.
Main Flow:

1. Teacher selects "Create Exam".
2. Teacher enters exam name, description, and timer settings.
3. Teacher selects questions from the question bank.
4. System saves the exam.
   Postcondition: Exam is stored and becomes available.

UC03 – Manage Question Bank

Actor: Teacher
Main Flow:

1. Teacher selects "Add Question".
2. Enters question text and multiple-choice options.
3. Marks the correct answer.
4. System stores the question.
   Postcondition: Question is added to the database.

UC04 – Take Exam

Actor: Student
Precondition: Exam is assigned/made available.
Main Flow:

1. Student selects the exam.
2. Timer starts.
3. Student answers questions.
4. Student submits exam.
   Postcondition: Exam submission recorded. **Exception:** Timer reaches zero → auto-submit.

UC05 – Auto-Score Exam

Actor: System
Main Flow:

1. System compares student answers with correct answers.
2. Score is calculated.
3. Incorrect answers are identified.
4. Results are saved.

UC06 – View Results

Actor: Student / Teacher
Main Flow:

1. User selects "Results".
2. System displays scores and detailed breakdown.

UC07 – Generate Performance Report

Actor: Teacher
Main Flow:

1. Teacher selects an exam.
2. System analyzes student performance.
3. Report is displayed with analytics.