Software Requirements Specification (SRS)

Project Title: Inquizitive Web Application

1. Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for the Inquizitive Web Application, an AI-powered quiz platform. The document serves as a reference for developers, project managers, testers, and stakeholders to ensure alignment on the system's objectives, scope, and features.

1.2 Scope

The Inquizitive application will:
- Provide AI-based automatic quiz generation.
- Enable secure quiz distribution and instant grading.
- Generate detailed performance analytics for educators.
- Support multiple user roles (teachers and students).
- Deliver a web-based solution with secure data storage.

Exclusions:
- No integration with external LMS.
- No multimedia (audio/video) support in quizzes.
- No offline quiz attempts.
- No real-time chat or doubt-solving system.

1.3 Definitions, Acronyms, and Abbreviations

- AI – Artificial Intelligence
- LMS – Learning Management System
- UAT – User Acceptance Testing

1.4 References

- Project Plan: Inquizitive Application (2025)
- SRS Template

1.5 Overview

The application will deliver automated quiz services, with AI-generated questions, instant grading, analytics dashboards, and secure student logins. The system will be hosted online and deployed for real classroom use.

2. Overall Description

2.1 Product Perspective

The application is a standalone web-based system, built with a backend API and a frontend user interface. It will integrate with AI models (OpenAI API or Hugging Face) for quiz question generation.

2.2 Product Functions

- Teacher dashboard for quiz creation, class management, and analytics.
- Student portal for secure quiz attempts.
- AI-based quiz generation and instant grading.
- Performance reports for both students and teachers.
- Secure storage of results and logs.

2.3 User Classes and Characteristics

- Teachers: Create and distribute quizzes, view analytics.
- Students: Attempt quizzes, view scores.
- Admin (optional): Manage users and system-level settings.

2.4 Operating Environment

- Web-based application accessible via browsers.
- Backend server with hosting environment.
- Database for storage of quiz data and results.

2.5 Design and Implementation Constraints

- Budget limited to PKR 20K.
- Timeline capped at 4 months.
- Hosting and deployment limited to standard web servers.

2.6 User Documentation

- Teacher Guide / User Manual.
- Online help section (FAQs, documentation).

3. Specific Requirements

3.1 Functional Requirements

1. System shall allow teachers to create and distribute quizzes.
2. System shall generate quiz questions using AI models.
3. System shall automatically grade student submissions.
4. System shall provide performance analytics for teachers.

5. System shall support secure login for teachers and students.
6. System shall save all quizzes and results in a secure database.
7. System shall provide randomized quiz questions per student to prevent cheating.

3.2 Non-Functional Requirements

1. The system shall ensure quiz grading accuracy ≥ 90%.
2. The system shall handle at least 500 concurrent users.
3. The system shall ensure response time ≤ 3 seconds for quiz submissions.
4. The system shall provide high security with encrypted data storage.

3.3 Interface Requirements

- User Interface: Web-based UI for both teacher and student portals.
- Hardware Interface: Standard computing devices with internet access.
- Software Interface: AI module integration via APIs.
- Communications Interface: HTTPS for secure communication.

4. System Models

System Models

- Use Case Diagram: Teacher creates quiz → AI generates questions → Student attempts quiz → System grades and reports analytics.
- Entity Relationship Diagram: Entities include Teacher, Student, Quiz, Question, Result, Report.

AI Module Integration (Expanded Section)

AI Module Integration

The AI module will be the core component responsible for quiz generation.

Options:
- OpenAI API (e.g., GPT models)
- Hugging Face models (transformer-based models like BERT, GPT-2, or domain-specific models)

Evaluation Plan:
- During development, both APIs will be tested for:
  - Question accuracy
  - Relevance to academic content
  - Response time

- Cost-effectiveness
- The final choice will depend on performance during pilot testing.

Implementation:
- Standardized API integration layer will be developed.
- The chosen model will be integrated into backend quiz generation workflow.
- Manual override option for teachers to validate/edit AI-generated questions.

5. Appendices

Glossary

- Quiz Analytics: Statistical data on student performance.
- AI Question Generation: Automatic creation of quiz questions using NLP models.

Supporting Information

- Project budget breakdown.
- Project timeline milestones.