Software Requirements Specification (SRS)

Project Title: Quiz JEC – Online Examination System

Prepared by: [Your Name]

Date: [Insert Date]

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to define the requirements for Quiz JEC, an online examination system for colleges that provides a secure and user-friendly interface for conducting and attending exams.

## 1.2 Scope

Quiz JEC is a cross-platform desktop application that enables teachers to create, manage, and evaluate tests, while students can take exams in a restricted, tab-blocked environment powered by Electron.js. It uses a MERN (MongoDB, Express.js, React, Node.js) stack and ShadCN for UI components.

## 1.3 Definitions, Acronyms, and Abbreviations

MERN: MongoDB, Express, React, Node.js  
SRS: Software Requirements Specification  
CRUD: Create, Read, Update, Delete  
UI: User Interface  
ID: Identifier

## 1.4 References

IEEE 830 SRS Standard  
MongoDB Documentation  
Electron.js Documentation  
ShadCN UI Library Docs

## 1.5 Overview

This document outlines the functional and non-functional requirements of the system. It includes system features, design constraints, and software interfaces.

# 2. Overall Description

## 2.1 Product Perspective

Quiz JEC is an independent system using web technologies wrapped in an Electron shell for enhanced exam security. It is not part of a larger system.

## 2.2 Product Functions

- Teacher Dashboard: Manage tests, questions, and results.  
- Student Dashboard: View and attend available tests.  
- Exam Environment: Tab-switch prevention and full-screen enforcement (via Electron).  
- Authentication: Secure login and registration.  
- Result Management: Auto-evaluation of objective questions and result display.

## 2.3 User Classes and Characteristics

- Admin/Teacher: Educators who create and manage exams.  
- Student: Users who take the exams.

## 2.4 Operating Environment

Desktop: Windows, macOS, Linux (Electron app)  
Server: Node.js with Express, MongoDB

## 2.5 Design and Implementation Constraints

Application must function offline once downloaded (Electron-based).  
Must prevent tab-switching and cheating during exams.

## 2.6 User Documentation

User manual for teachers and students (to be developed).  
Setup guide for deployment.

## 2.7 Assumptions and Dependencies

Internet access is required for login and syncing test data.  
MongoDB must be running for backend functionality.

# 3. Specific Requirements

## 3.1 Functional Requirements

### 3.1.1 User Authentication

Users can register and log in securely.  
Passwords must be hashed using a secure algorithm.

### 3.1.2 Test Management (Teacher)

Create, edit, and delete tests.  
Add questions to a test (question IDs stored in Test model).  
Publish/unpublish tests.

### 3.1.3 Question Management

Add MCQs, subjective, and multiple-answer questions.  
Questions are stored separately in the Question model.

### 3.1.4 Exam Interface (Student)

Students can view available tests.  
Students can take tests in full-screen mode.  
Prevent switching tabs and accessing other applications.

### 3.1.5 Result Handling

Automatically evaluate objective-type questions.  
Display results post submission or after publishing.

## 3.2 Performance Requirements

The app must load all test and question data within 2 seconds.  
Results must be generated within 5 seconds of submission.

## 3.3 Design Constraints

Frontend must use React with Tailwind and ShadCN.  
Backend must be built using Node.js and Express.  
Desktop app must use Electron.js with tab-lock and fullscreen enforcement.

## 3.4 Software System Attributes

### 3.4.1 Reliability

System should recover gracefully from crashes with local autosave.

### 3.4.2 Availability

System should be available 99% of the time during exam windows.

### 3.4.3 Security

Must hash passwords.  
Prevent copying text from questions.  
Disable right-click and developer tools during exams.

### 3.4.4 Maintainability

Code should follow modular structure with reusable components and services.

### 3.4.5 Portability

Must run on Windows/macOS/Linux with minimal changes.

# 4. Appendices

Appendix A: Sample MongoDB Schemas  
Appendix B: Flowcharts for Teacher and Student Modules  
Appendix C: Screenshots of UI (optional)