Quiz Application - Project Proposal

Author: Getoar Krasniqi

Mentor: Dr. Sc. Ermira Daka

Email: gk74706@ubt-uni.net

Github: https://github.com/Geti23/quiz-app

Project Scope

The Quiz Application is a Python-based library for creating, managing, and administering multiple-choice quizzes with comprehensive scoring and analytics. The project delivers a flexible, reusable quiz engine suitable for educational platforms, training systems, assessment tools, and self-study applications.

In Scope	Out of Scope
Multiple-choice question management with validation	Graphical user interface (CLI/GUI)
Quiz creation and configuration (titles, time limits)	Database integration and persistence
Answer submission and automated scoring	User authentication and authorization
Difficulty levels (easy, medium, hard) and category organization	Web API or REST endpoints
Timed quiz functionality with automatic tracking	Multi-user/concurrent quiz sessions
Detailed results and performance analytics	Question types beyond multiple-choice
Answer review and feedback mechanisms	

User Stories

As an educator, I want to create quizzes with categorized questions so that I can assess student knowledge across different topics.

As a student, I want to take timed quizzes so that I can practice for real exam conditions.

As a learner, I want to review my incorrect answers with explanations so that I can understand my mistakes and improve.

As a quiz administrator, I want to organize questions by difficulty level so that I can create assessments appropriate for different skill levels.

As a developer, I want a well-tested, documented library so that I can integrate quiz functionality into my educational application.

As a self-learner, I want randomized question order so that I can retake quizzes without memorizing sequences.

Tech Stack

Component	**Technology**	**Justification**
Language	Python 3.9+	Readable syntax, excellent testing ecosystem, wide adoption in education
Testing Framework	pytest	Industry-standard, powerful fixtures, extensive plugin ecosystem
Code Coverage	pytest-cov	Integrated coverage reporting with pytest
Code Formatting	black	Opinionated formatter, eliminates style debates
Linting	flake8	Catches common errors and enforces PEP 8 standards
Type Checking	mypy	Static type analysis for improved code quality
CI/CD	GitHub Actions	Free for public repos, excellent Python support
Package Management	setuptools/pip	Standard Python packaging tools
Documentation	README.md	Clear, maintainable documentation format

Testing Strategy

Unit Tests (70%):	**Integration Tests** (20%):	**Edge Case Tests** (10%):
Individual Question validation and behavior	Complete quiz workflows (creation → answering → scoring)	Empty quizzes, invalid inputs
QuizResult calculations and thresholds	Multi-question scenarios	Duplicate question handling
Quiz management operations	Category and difficulty filtering	Time expiration boundary conditions

Testing Tools & Practices

- **Automated Testing**: All tests run on every push via GitHub Actions
- **Matrix Testing**: Tests across Python 3.9-3.12 and multiple OS platforms
- **Coverage Reports**: HTML reports generated for detailed analysis
- **Continuous Integration**: Automated linting, type checking, and test execution
- **Test Isolation**: Each test is independent with no shared state

Success Criteria

- All tests passing across all Python versions
- ≥95% code coverage maintained
- Zero critical linting errors
- Type hints validated with mypy
- CI/CD pipeline green on all platforms