SVKM'S NMIMS Deemed to be UNIVERSITY

IMAP ASSIGNMENT

Quiz Hub

CHALLENGE YOUR MIND, ONE QUIZ AT A TIME!

MADE BY:

ANMOL SINGH - S007

RISHI PONDA - S046

SHIVANSH KHANDELWAL - S054

Software Requirements Specification (SRS)

1. Introduction

The project is a quiz website built with Django for the frontend and SqlLite3 for the database. The system will allow users to register, take quizzes, view their results, and admins to manage quiz and view analytics. This SRS document outlines the functional and non-functional requirements of the quiz website, which aims to offer an interactive, scalable, and secure platform for hosting quiz.

1.1 Purpose

The purpose of this document is to provide a detailed description of the system requirements for the quiz website, including the design and functionality needed to achieve user registration, quiz participation, result generation, and administrative controls.

1.2 Scope

This system will allow:

- **Users** can register, login, attempt quiz, and view their results.
- **Administrators** to manage quiz content, questions, and track user performance. The application will include features like multi-type questions (MCQs), leaderboard tracking, and quiz result analytics.

1.3 Objectives

- Provide an easy-to-use interface for users and admins.
- Ensure data integrity and security, especially for user details and quiz results.
- simultaneous quiz attempts efficiently.

2. Functional Requirements

2.1 User Registration and Login

- Users must be able to create an account by providing a valid email, username, and password.
- Implement a secure login system where registered users can log in to their accounts.
- Users should have the ability to reset their password in case of forgotten credentials.

2.2 Profile Management

- Each user will have a personal profile where they can update their username, email, and password.
- Users can track their quiz history and see past performance results.

2.3 Quiz Management (Admin)

- **Admin users** will have the ability to:
    - Create new quiz, including setting titles, descriptions, and quiz types.
    - Add, update, or delete questions from the quiz.
    - Specify question types (MCQs, True/False, Fill-in-the-blank).
    - Set time limits for quiz and assign difficulty levels.
    - View analytics on quiz performance, including user scores and participation rates.

2.4 Quiz Attempt (User)

- Users can browse available quiz and select one to attempt.
- Quiz may have multiple pages, depending on the number of questions.
- Users must be able to navigate between quiz pages before final submission.
- Upon completion, users receive immediate feedback on their scores.

2.5 Result Calculation

- After submission, the system calculates the score based on the user's answers.
- The result will show total questions answered, correct answers, and percentage score.
- Users can view a detailed breakdown of their performance after completing the quiz.

2.6 Question Types

- The system will support various types of questions, including:
    - Multiple-choice questions (with one or multiple correct answers).

3. Non-Functional Requirements

3.1 Performance

- The system should efficiently handle multiple concurrent users attempting quizzes.
- The backend should support fast query processing to display results in real-time.

3.2 Security

- All user passwords should be hashed using strong cryptographic algorithms (e.g., bcrypt).
- Secure HTTPS protocols will be enforced for data transmission.
- Users' quiz results and personal information will be stored securely in the database.

3.3 Scalability

- The system should be scalable to accommodate a growing number of users and quizzes.
- Database design should allow for easy addition of new features without major changes.

3.4 Reliability

- The system should ensure 99.9% uptime with proper error handling and failover mechanisms.
- Scheduled database backups should be maintained to avoid data loss.

3.5 Usability

- The user interface should be intuitive, with clear instructions on how to attempt quizzes.
- Mobile responsiveness should be supported, allowing users to access quizzes from smartphones and tablets.

4. System Design Overview

4.1 Technologies

- **Frontend**: Django for user interface, providing a smooth and responsive experience.
- **Backend**: Node.js with Express.js for handling API requests.
- **Database**: SqlLite for managing user data, quiz questions, and results.

4.2 External APIs

- If needed, third-party authentication services (e.g., Google OAuth) can be integrated for user login.

4.3 Reporting and Analytics

- The admin dashboard will include detailed reports on user participation, average scores, and popular quiz..

ERD Diagram (Entity Relationship Diagram)

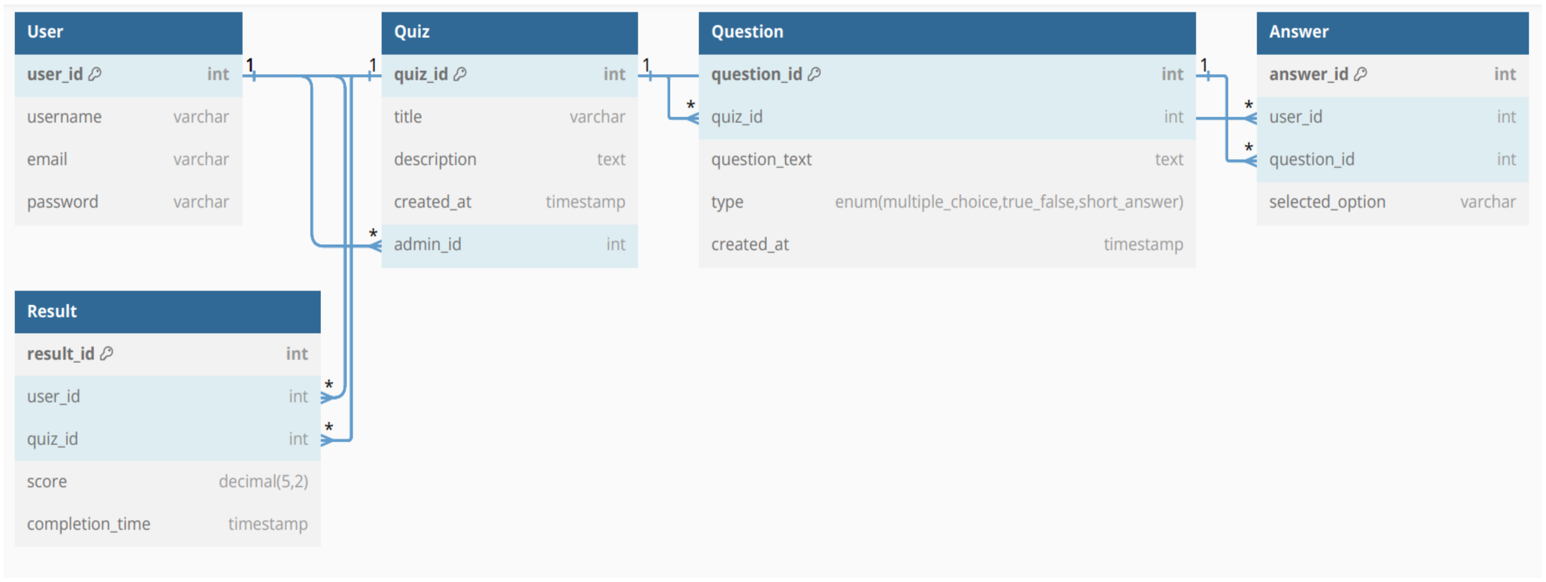
The key entities and their relationships:

1. **User**
    - `user_id` (Primary Key), `username`, `email`, `password`
2. **Quiz**
    - `quiz_id` (Primary Key), `title`, `description`, `created_at`, `admin_id` (Foreign Key references `User.user_id`)
3. **Question**
    - `question_id` (Primary Key), `quiz_id` (Foreign Key references `Quiz.quiz_id`), `question_text`, `type`, `created_at`
4. **Answer**
    - `answer_id` (Primary Key), `user_id` (Foreign Key references `User.user_id`), `question_id` (Foreign Key references `Question.question_id`), `selected_option`
5. **Result**
    - `result_id` (Primary Key), `user_id` (Foreign Key references `User.user_id`), `quiz_id` (Foreign Key references `Quiz.quiz_id`), `score`, `completion_time`

Database Relationships

- A **User** can take multiple quizzes.
- A **Quiz** contains multiple questions.
- A **Question** can have multiple answers from different users.
- **Results** are stored based on the user's participation in a specific quiz.

Relation Table:



User	
user_id	int
username	varchar
email	varchar
password	varchar

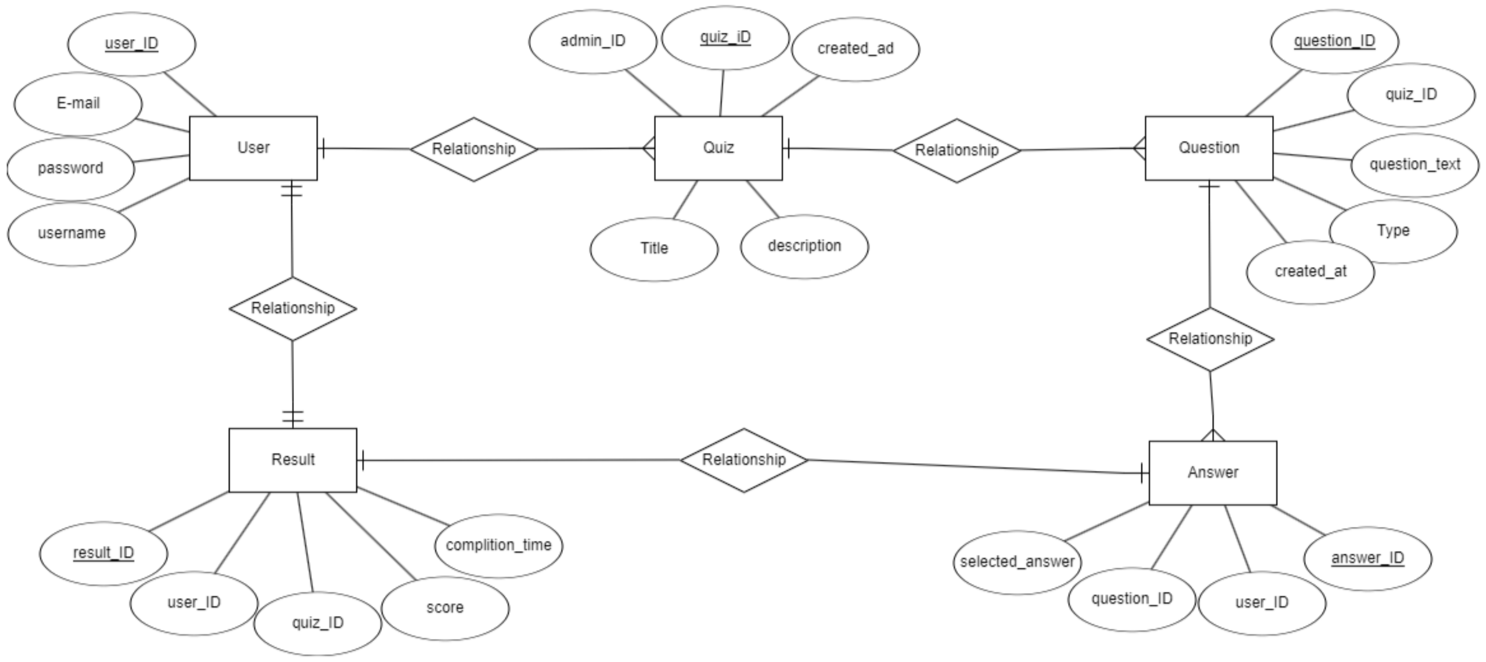
Quiz	
quiz_id	int
title	varchar
description	text
created_at	timestamp
admin_id	int

Question	
question_id	int
quiz_id	int
question_text	text
type	enum(multiple_choice,true_false,short_answer)
created_at	timestamp

Answer	
answer_id	int
user_id	int
question_id	int
selected_option	varchar

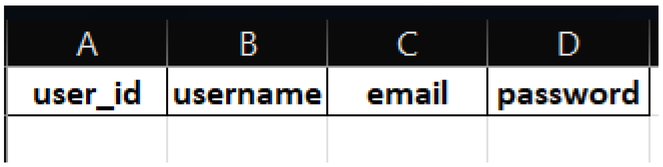
Result	
result_id	int
user_id	int
quiz_id	int
score	decimal(5,2)
completion_time	timestamp

ERD Diagram:



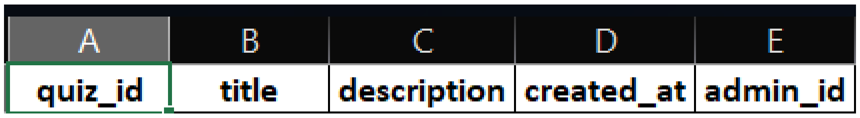
Database Table in Excel

- **User Table**



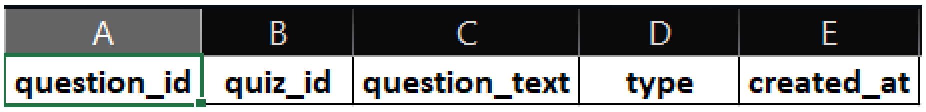
A	B	C	D
user_id	username	email	password

- **Quiz Table**



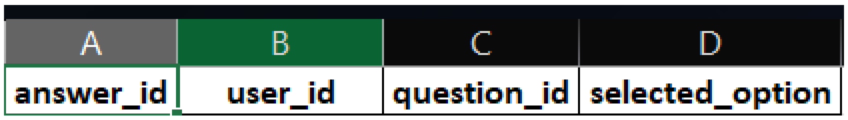
A	B	C	D	E
quiz_id	title	description	created_at	admin_id

- **Question Table**



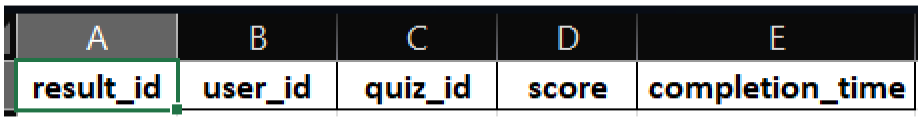
A	B	C	D	E
question_id	quiz_id	question_text	type	created_at

- **Answer Table**



A	B	C	D
answer_id	user_id	question_id	selected_option

- **Result Table**



A	B	C	D	E
result_id	user_id	quiz_id	score	completion_time