Quiz Master Project Report

Author

Name: Vashisht D. Brahmbhatt
Roll No: 23F3000948
Email : 23f3000948@ds.study.iitm.ac.in
I am a passionate Data Science aspirant with experience in Python frameworks. I am currently pursuing a dual degree in B.Tech in Computer Science from GLS University and BS in Data Science from IIT Madras . I love exploring generative and agentic AI and using it efficiently.

Description

This project involves developing a multi-user web application that acts as an exam preparation platform for multiple courses. The application allows administrators to create and manage subjects, chapters, quizzes, and questions, while users can browse and attempt quizzes to assess their knowledge in different subjects.

Technologies Used

- **Flask**: Used as the main web framework for building the backend of the application
- **Flask-SQLAlchemy**: ORM for database operations and model definitions
- **Flask-Login**: Handles user authentication and session management
- **Flask-Bcrypt**: For secure password hashing and verification
- **Jinja2**: Templating engine for rendering dynamic HTML content
- **SQLite**: Lightweight database for storing application data
- **Bootstrap**: Frontend framework for responsive and modern UI design
- **Chart.js**: Used for visualizations in summary dashboards
- **JSON**: For data exchange in API endpoints

These technologies were selected to create a scalable, secure, and user-friendly application with minimal setup requirements, ensuring the application can run on local machines without additional dependencies.

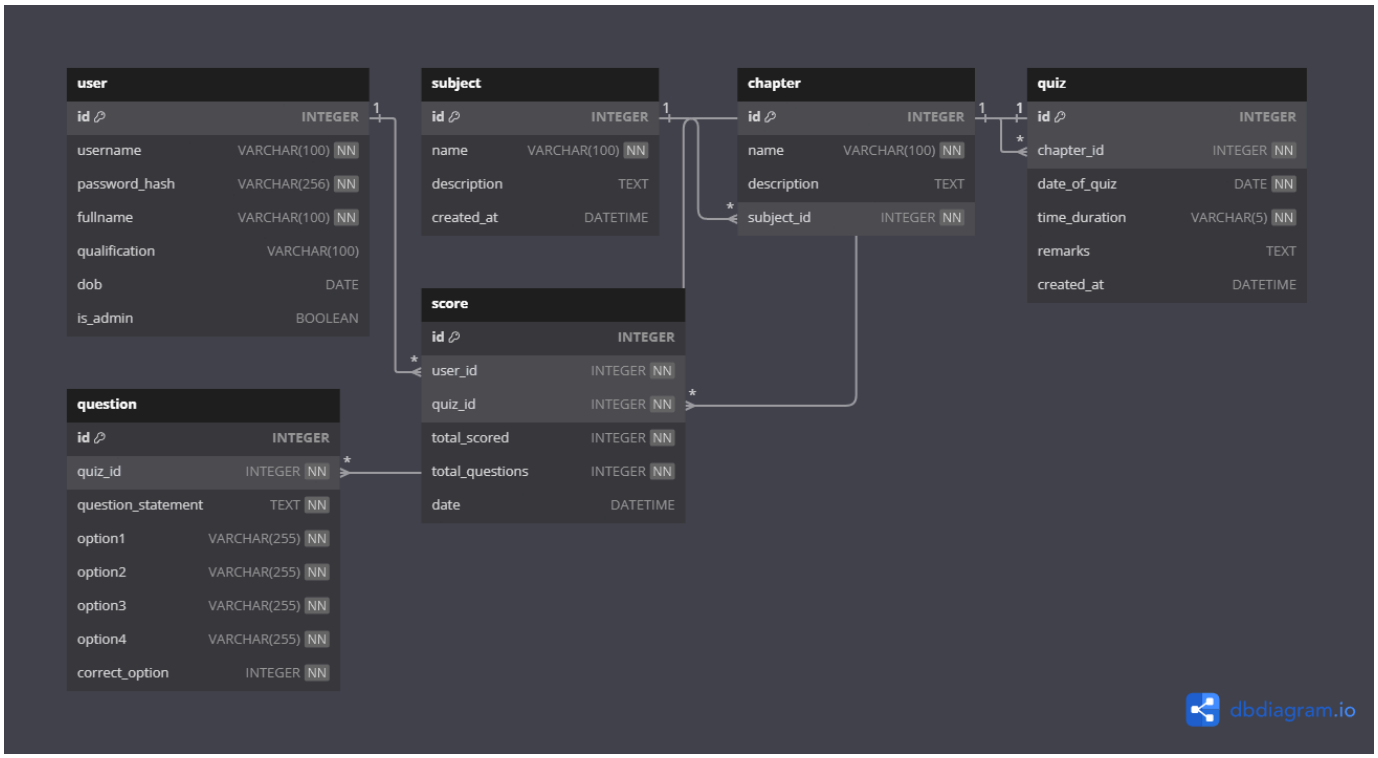
DB Schema Design

The database schema includes six main models:

1. **User**:
    - `id`: Primary key
    - `username`: Unique email address
    - `password_hash`: Securely stored password
    - `fullname`: User's full name
    - `qualification`: Educational qualification
    - `dob`: Date of birth
    - `is_admin`: Boolean flag for admin privileges
    - Relationships: One-to-many with Score
2. **Subject**:
    - `id`: Primary key
    - `name`: Subject name
    - `description`: Subject description
    - `created_at`: Timestamp
    - Relationships: One-to-many with Chapter (cascade delete)
3. **Chapter**:
    - `id`: Primary key
    - `name`: Chapter name
    - `description`: Chapter description
    - `subject_id`: Foreign key to Subject
    - Relationships: One-to-many with Quiz (cascade delete)
4. **Quiz**:
    - `id`: Primary key
    - `chapter_id`: Foreign key to Chapter
    - `date_of_quiz`: Scheduled date
    - `time_duration`: Time allowed for quiz (HH:MM)
    - `remarks`: Additional notes
    - `created_at`: Timestamp
    - Relationships: One-to-many with Question and Score (cascade delete)
5. **Question**:
    - `id`: Primary key
    - `quiz_id`: Foreign key to Quiz
    - `question_statement`: Question text
    - `option1` through `option4`: Multiple choice options
    - `correct_option`: Integer (1-4) indicating correct answer

6. **Score**:
    - `id`: Primary key
    - `user_id`: Foreign key to User
    - `quiz_id`: Foreign key to Quiz
    - `total_scored`: Number of correct answers
    - `total_questions`: Total questions in quiz
    - `date`: Timestamp of quiz attempt

The database design uses cascade delete relationships to maintain referential integrity, ensuring that when a parent record is deleted, all associated child records are automatically removed. This design supports the hierarchical organization of educational content (subject → chapter → quiz → question) while maintaining separation of user data.



API Design

The application includes several API endpoints for data retrieval and search functionality:

1. **Subject Search API** (`/api/search/subjects`):
    - Returns filtered subjects based on search query
    - Used for implementing dynamic search functionality in the frontend

2. **Top Scores Data API** (`/admin/api/top-scores-data`):
    - Returns highest scores for each quiz
    - Used for generating performance charts in admin summary
3. **User Attempts Data API** (`/admin/api/user-attempts-data`):
    - Provides analytics on quiz attempt counts by subject
    - Supports administrative dashboard visualizations
4. **User Quiz Status API** (implementation started but not fully exposed):
    - Checks quiz completion status for a user
    - Used to display taken/untaken quiz status

The API implementation follows RESTful principles, using JSON for data exchange and implementing proper authorization checks to ensure only authorized users can access specific endpoints.

Architecture and Features

The project follows the Model-View-Controller (MVC) architecture pattern:

- **Models** (`models.py`): Contains database entity definitions using SQLAlchemy ORM
- **Controllers** (`controllers.py`): Implements business logic and route handlers
- **Views** (templates directory): Jinja2 templates for rendering HTML pages
- **Routes** (`app.py`): Maps URL endpoints to controller functions

The application includes both core and recommended functionalities:

Core Features:

- **User Authentication**: Registration, login and role-based access control
- **Admin Dashboard**: Comprehensive management of subjects, chapters, quizzes, and questions
- **Quiz Management**: Creating, editing, and deleting quizzes with multiple-choice questions
- **User Dashboard**: Displaying available quizzes and quiz history
- **Quiz Taking**: Interface for users to attempt quizzes and view results

Additional Features:

- **Data Visualization**: Charts showing quiz participation and performance metrics
- **Search Functionality**: Dynamic search for subjects, users, and quizzes
- **User Management**: Admin capabilities to manage user accounts

- **API Endpoints**: JSON endpoints for data retrieval and search
- **Summary Dashboards**: Analytical views for both users and administrators

The application handles user sessions securely, implements proper authorization checks at both the route and controller levels, and follows database best practices with proper relationships and constraints.

Video

Project Demo Video