Author

Anubhav Agarwal

21F3001668

21f3001668@ds.study.iitm.ac.in

I am Anubhav Agarwal, currently in my last term of diploma level; I am doing my integrated master's from IIT Madras as well.

Description

We need to make a quiz-taking platform where the admin can create subjects, a subject can have multiple chapters, and each chapter can have multiple quizzes. The user will be able to take any quiz of his choice, and can later see his performance in the quiz. The user will also be able to see some chart summaries of his performance.

Technologies Used

1. Flask – For Backend logic
2. Vuejs – For Frontend templates
3. SQLite - For storing user data, subjects, chapters, quizzes and scores.
4. HTML & Bootstrap CSS – For frontend
5. Chart.js - For creating summary charts
6. SQLAlchemy – For CRUD operations related to Flask

DB Schema design

1. Subject Table
   Purpose: Stores subject information.
   Columns:
   1. id (PK, Integer, Auto-increment) → Unique identifier.
   2. name (String, Unique, Not Null) → Subject name.
   3. description (Text, Nullable) → Brief description of the subject.

   Constraints: name should be unique to avoid duplicates.
   Reasoning: Subjects form the top-level hierarchy and should have a unique identifier.

2. Chapter Table
   Purpose: Stores chapters under a specific subject.
   Columns:
   1. id (PK, Integer, Auto-increment) → Unique identifier.
   2. name (String, Not Null) → Chapter name.
   3. description (Text, Nullable) → Brief details about the chapter.
   4. subject_id (FK, Integer, Not Null) → References subject.id.

   Constraints: subject_id is a foreign key, ensuring each chapter belongs to a valid subject.
   Reasoning: Each chapter must belong to a subject, ensuring structured organization.

3. Quiz Table
   Purpose: Stores quizzes linked to chapters.
   Columns:
   1. id (PK, Integer, Auto-increment) → Unique identifier.
   2. chapter_id (FK, Integer, Not Null) → References chapter.id.
   3. date_of_quiz (Date, Not Null) → Date of the quiz.
   4. time_duration (String, Nullable) → Time duration in HH:MM format.
   5. remarks (Text, Nullable) → Additional remarks about the quiz.

   Constraints: chapter_id is a foreign key to ensure quizzes are linked to a valid chapter.
   Reasoning: This structure ensures quizzes are categorized under specific chapters.

4. Question Table
   Purpose: Stores questions for quizzes.
   Columns:
   1. id (PK, Integer, Auto-increment) → Unique identifier.
   2. quiz_id (FK, Integer, Not Null) → References quiz.id.
   3. title (String, Not Null) → Short question title.
   4. question_statement (Text, Not Null) → Full question text.
   5. option1 (String, Not Null) → First choice.
   6. option2 (String, Not Null) → Second choice.
   7. option3 (String, Nullable) → Third choice.
   8. option4 (String, Nullable) → Fourth choice.
   9. correct_option (Integer, Not Null) → Stores correct answer index (1-4).

   Constraints: quiz_id ensures a question is assigned to a valid quiz, correct_option is limited to 1-4.
   Reasoning: Using separate option columns ensures structured multiple-choice questions.

5. User Table
   Purpose: Stores user details.
   Columns:
   1. id (PK, Integer, Auto-increment) → Unique identifier.
   2. username (String, Unique, Not Null) → Login username.
   3. password (String, Not Null) → Hashed password.
   4. full_name (String, Nullable) → User's full name.
   5. qualification (String, Nullable) → Education details.
   6. dob (Date, Nullable) → Date of birth.
   7. is_admin (Boolean, Default: False) → Role management.

   Constraints: username should be unique to prevent duplicate accounts.
   Reasoning: The structure ensures secure authentication and supports user roles.

6. Score Table
   Purpose: Stores quiz scores of users.
   Columns:
   1. id (PK, Integer, Auto-increment) → Unique identifier.
   2. quiz_id (FK, Integer, Not Null) → References quiz.id.
   3. user_id (FK, Integer, Not Null) → References user.id.
   4. time_stamp_of_attempt (DateTime, Not Null) → Quiz attempt time.
   5. total_scored (Integer, Not Null) → Score obtained.

   Constraints: quiz_id and user_id ensure scores belong to valid users and quizzes.
   Reasoning: Keeps track of user performance over time.

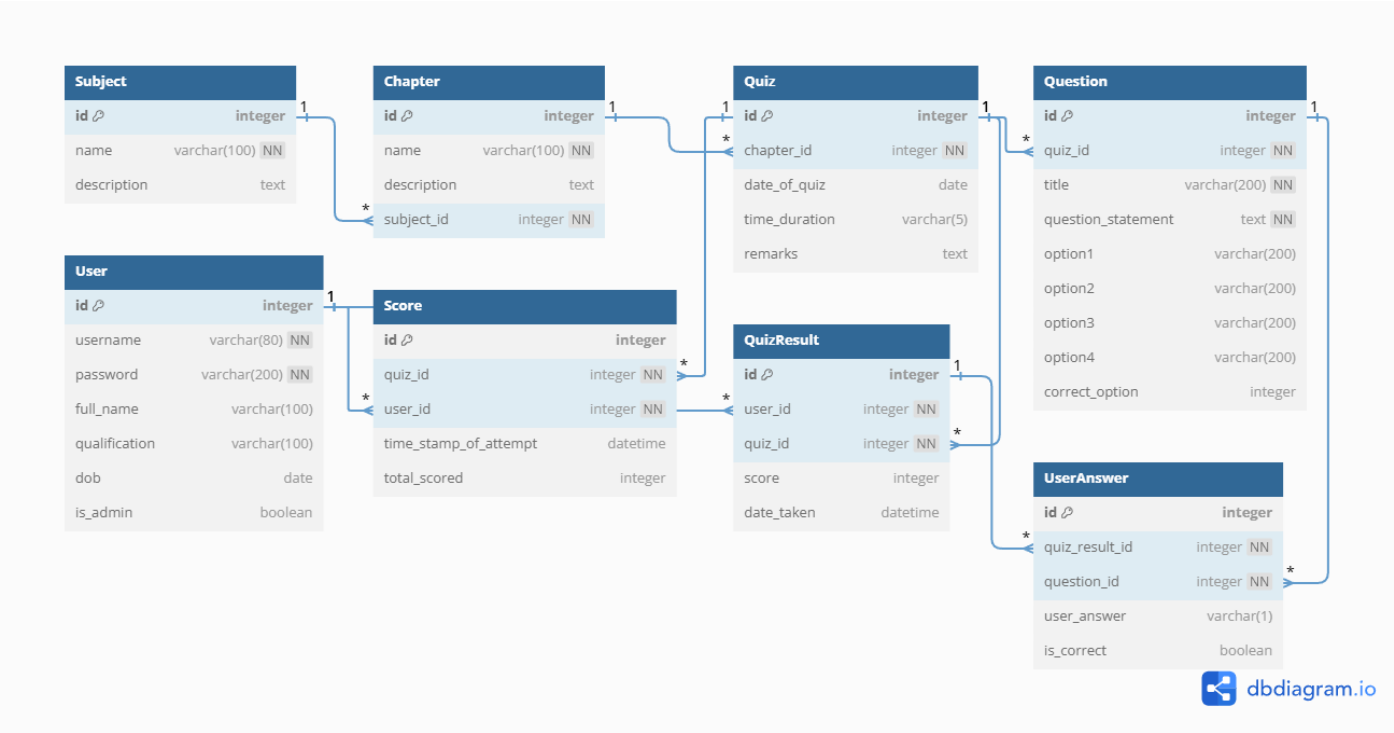
7. QuizResult Table
   Purpose: Stores quiz completion data.
   Columns:
   1. id (PK, Integer, Auto-increment) → Unique identifier.
   2. user_id (FK, Integer, Not Null) → References user.id.
   3. quiz_id (FK, Integer, Not Null) → References quiz.id.
   4. score (Integer, Not Null) → Final score.
   5. date_taken (DateTime, Default: Current Time) → Quiz completion timestamp.

   Constraints: user_id and quiz_id ensure valid relationships.
   Reasoning: This table records final scores and completion timestamps for users.

8. UserAnswer Table
   Purpose: Stores user responses to quiz questions.
   Columns:
   1. id (PK, Integer, Auto-increment) → Unique identifier.
   2. quiz_result_id (FK, Integer, Not Null) → References quiz_result.id.
   3. question_id (FK, Integer, Not Null) → References question.id.
   4. user_answer (String, Not Null) → Chosen answer (A, B, C, D).
   5. is_correct (Boolean, Not Null) → True if correct.

   Constraints: quiz_result_id ensures each answer is linked to a valid quiz attempt. question_id ensures the response belongs to a valid question.
   Reasoning: This structure ensures detailed tracking of user answers and accuracy.

ER Diagram



Architecture and Features

The project is situated in quiz master folder. Inside the folder:

1. /frontend which contains all the routers, index.js, and css files
2. model.py, which stores database models (e.g., `User,` `Quiz`) defining SQLite tables and relationships.
3. /backend, which has all backend related files such as app.py, auth.py, admin.py, user.py

This structure separates concerns (MVC-like pattern), making the app scalable and easy to debug. app.py helps in running and tackling the backend, models manage data, and templates render views, while static files ensure a polished UI.

The features of the app include a single login page for both normal users and admin; the route would check for the credentials and redirect the user to the respective page. Admin dashboard, where he can see the subjects and related chapters and the option to delete and add subjects. The chapters can be edited, deleted, and created. The admin can also see how many questions/chapters are available. The admin can create quizzes and add questions to the quiz, with 4 options and 1 correct option, for the chapters added, and it can also see the performance of the users' taking quizzes, like Subject-Wise User Attempts and Subject-Wise Top & Lowest Scores. The admin can see the user's info, like DOB, username, etc. The admin can also search for the user/subject/quiz and get the relevant information as output, such as the name of the user searched, subject name, date of the quiz, etc.

The user can log in using his credentials and will see his dashboard, where he can see all the quizzes, whether past, present, or future. If a quiz is taken once, it can't be retaken, and the button will get disabled. The user can click the particular quiz to get information about the quiz(subject, chapter name, etc). The user can also see his score, as well as charts related to his performance like Quizzes Attempted Per Subject and Month-Wise Quiz Attempts. The user can search for quizzes and subjects and can get the relevant output like marks, date of quiz, subject, etc.

Video link

https://drive.google.com/file/d/19njuJv4ql1JQqI-9JADgAR9Zcgzo44RH/view?usp=sharing