QuizMaster Project Report

Author

Rajesh Parmar
23f1002993
23f1002993@ds.study.iitm.ac.in

I'm a dedicated student with a strong interest in web development and educational technology. I enjoy building applications that can help others learn more effectively and track their progress through interactive features.

Description

QuizMaster is a web-based quiz application that allows administrators to create and manage subjects, chapters, and quizzes, while enabling students to take quizzes and track their performance. The system provides a structured learning path with comprehensive performance analytics.

Technologies Used

- **Flask**: Lightweight web framework used for routing and request handling
- **Flask-Login**: Handles user authentication and session management
- **Flask-SQLAlchemy**: ORM for database interaction, simplifying CRUD operations
- **SQLite**: Database for storing application data in development
- **Werkzeug**: For security features including password hashing
- **Jinja2**: Templating engine for dynamic HTML generation
- **Python-dotenv**: Environment variable management for secure configuration
- **Bootstrap** (frontend): For responsive design and consistent UI components

These technologies were chosen for their simplicity, flexibility, and strong community support, making them ideal for educational applications requiring role-based access control.

DB Schema Design

The database is designed with the following models:

1. **User**:
    - Fields: id (PK), email (unique), password (hashed), full_name, qualification, dob, is_admin ○ Relationships: One-to-many with Score
    - Constraints: Unique email, non-nullable fields
2. **Subject**:
    - Fields: id (PK), name, description ○ Relationships: One-to-many with Chapter ○ Purpose: Top-level content organization

3. **Chapter**:
    - Fields: id (PK), name, description, subject_id (FK) ○ Relationships: Belongs to Subject, one-to-many with Quiz
    - Purpose: Intermediate content organization
4. **Quiz**:
    - Fields: id (PK), chapter_id (FK), date_of_quiz, time_duration, remarks ○ Relationships: Belongs to Chapter, one-to-many with Question and Score ○ Purpose: Assessment container
5. **Question**:
    - Fields: id (PK), quiz_id (FK), question_statement, options (1-4), correct_option ○ Relationships: Belongs to Quiz ○ Purpose: Individual assessment items
6. **Answer**:
    - Fields: id (PK), score_id (FK), question_id (FK), selected_option ○ Relationships: Belongs to Score and Question
    - Purpose: Records user responses
7. **Score**:
    - Fields: id (PK), quiz_id (FK), user_id (FK), timestamp, correct_answers, total_questions ○ Relationships: Belongs to Quiz and User, one-to-many with Answer ○ Purpose: Performance tracking

The hierarchical design (Subject→Chapter→Quiz→Question) provides a logical organization of educational content while enabling efficient querying and performance analysis.

Architecture and Features

Architecture

The project follows a modular blueprint-based architecture:

- **Routes**: Organized into auth, admin, and user blueprints
- **Models**: Database models in models.py with relationship definitions
- **Application Factory**: Centralized app creation in app.py
- **Config**: Environment-based configuration in config.py
- **Templates**: Jinja2 templates organized by blueprint

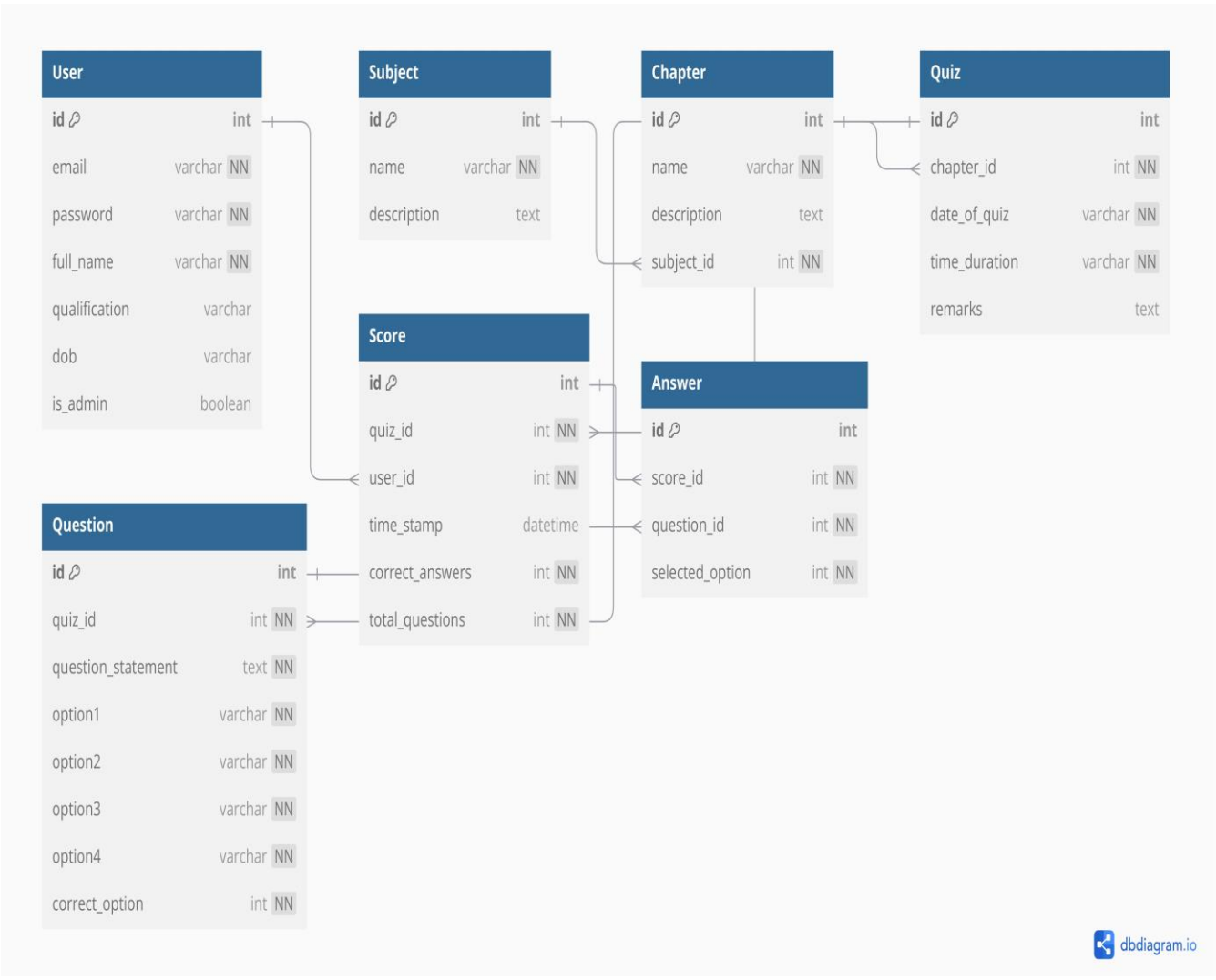
Features

1. **Authentication System**:
    - User registration and login ○ Role-based access control (admin/student) ○ Password hashing for security ○ Remember-me functionality
2. **Admin Features**:
    - Subject and chapter management ○ Quiz creation with multiple-choice questions ○ Student performance monitoring ○ Default admin account creation on first run
3. **Student Features**:
    - Taking quizzes with automatic scoring ○ Performance tracking with grade calculation (A-F) ○ Historical quiz review ○ Average score calculation across all quizzes
4. **Additional Features**:
    - Automatic grade assignment based on score percentage ○ Status tracking (Passed/Failed) for quizzes ○ Quiz availability status (Open/Taken)
    - Subject and chapter statistics (total quizzes, total students)

The application uses helper methods in models (like get_percentage(), get_grade(), get_status()) to encapsulate business logic and promote code reusability.

Video

https://drive.google.com/file/d/1J7anlexn-C2Xd6wHoq3eZefYPDXFCY9h/view?usp=sharing


User
id int
email varchar NN
password varchar NN
full_name varchar NN
qualification varchar
dob varchar
is_admin boolean
Subject
id int
name varchar NN
description text
Chapter
id int
name varchar NN
description text
subject_id int NN
Quiz
id int
chapter_id int NN
date_of_quiz varchar NN
time_duration varchar NN
remarks text
Score
id int
quiz_id int NN
user_id int NN
time_stamp datetime
correct_answers int NN
total_questions int NN
Answer
id int
score_id int NN
question_id int NN
selected_option int NN
Question
id int
quiz_id int NN
question_statement text NN
option1 varchar NN
option2 varchar NN
option3 varchar NN
option4 varchar NN
correct_option int NN
dbdiagram.io