PROJECT REPORT

QuizzMaster

Author

Name: Navneet Kumar Yadav
Roll no.: 23f20020292
Email: 23f2002092@ds.study.iitm.ac.in

Description

QuizzMaster is built using Vue.js (frontend), Flask (backend), SQLite (database), and Redis (for caching and job handling). It features two roles: Admin and User. The Admin can create and manage subjects, chapters, quizzes, and questions. Users can attempt quizzes and view their scores. The app ensures a smooth experience with real-time updates and efficient data handling.

Technologies Used

- Frontend: Vue.js, JavaScript, Prettier (code formatting), various JS packages for UI/UX.
- Backend: Flask (Python), Flask-CORS (CORS handling), Flask-JWT-Extended (authentication).
- Database: SQLite (lightweight storage for user & quiz data).
- Caching & Tasks: Redis (caching, Celery task queue).
- Email Service: Flask-Mail (Yandex SMTP for notifications).
- Development Tools: Linting (code quality), Prettier (formatting).

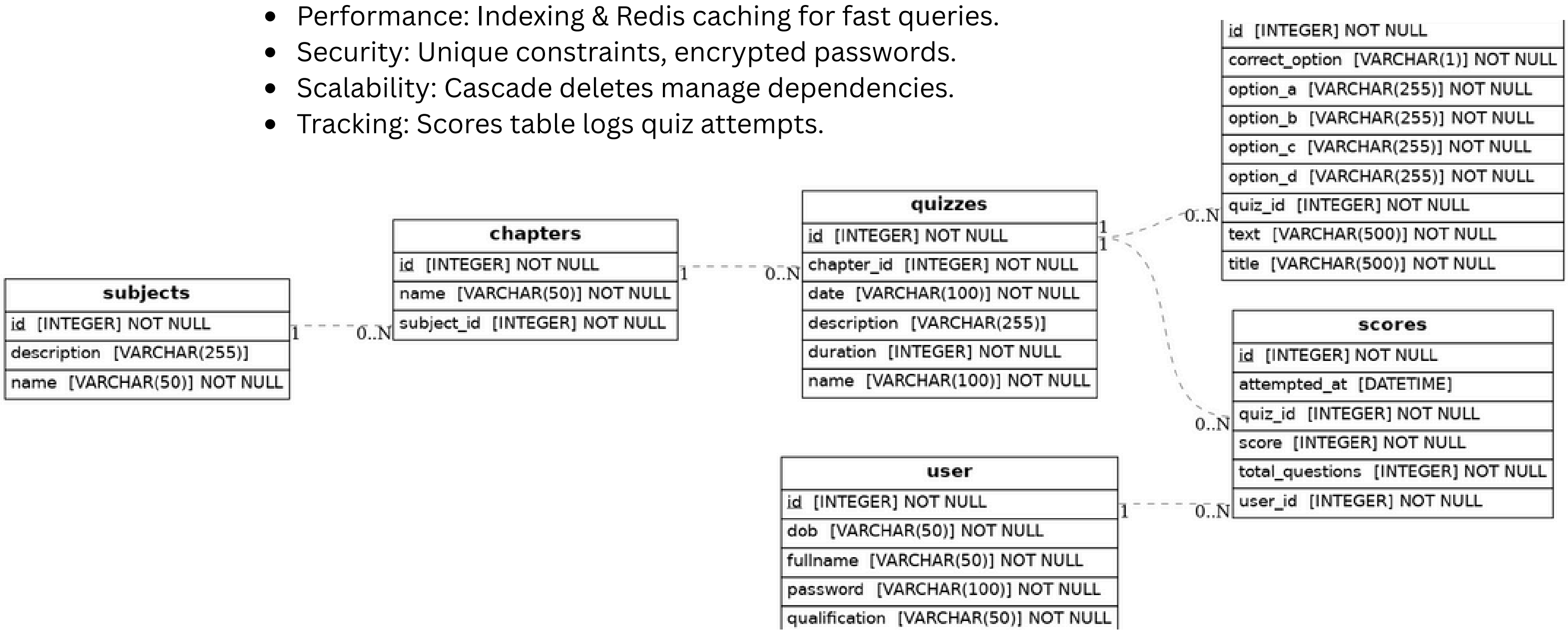
DB Schema Design

Tables & Columns

1. User (id, username*, password, fullname, qualification, dob) – Stores user details.
2. Subjects (id, name*, description) – Manages subjects.
3. Chapters (id, name, subject_id FK) – Chapters linked to subjects.
4. Quiz (id, name, description, chapter_id FK, date, duration) – Quiz metadata.
5. Question (id, title, text, quiz_id FK, option_a-d, correct_option) – Quiz questions.
6. Scores (id, user_id FK, quiz_id FK, score, total_questions, attempted_at) – Tracks user scores.

Design Justification

- Relational Integrity: Foreign Keys ensure structured data.
- Performance: Indexing & Redis caching for fast queries.
- Security: Unique constraints, encrypted passwords.
- Scalability: Cascade deletes manage dependencies.
- Tracking: Scores table logs quiz attempts.


id [INTEGER] NOT NULL
correct_option [VARCHAR(1)] NOT NULL
option_a [VARCHAR(255)] NOT NULL
option_b [VARCHAR(255)] NOT NULL
option_c [VARCHAR(255)] NOT NULL
option_d [VARCHAR(255)] NOT NULL
quiz_id [INTEGER] NOT NULL
text [VARCHAR(500)] NOT NULL
title [VARCHAR(500)] NOT NULL
quizzes
id [INTEGER] NOT NULL
chapter_id [INTEGER] NOT NULL
date [VARCHAR(100)] NOT NULL
description [VARCHAR(255)]
duration [INTEGER] NOT NULL
name [VARCHAR(100)] NOT NULL
chapters
id [INTEGER] NOT NULL
name [VARCHAR(50)] NOT NULL
subject_id [INTEGER] NOT NULL
subjects
id [INTEGER] NOT NULL
description [VARCHAR(255)]
name [VARCHAR(50)] NOT NULL
scores
id [INTEGER] NOT NULL
attempted_at [DATETIME]
quiz_id [INTEGER] NOT NULL
score [INTEGER] NOT NULL
total_questions [INTEGER] NOT NULL
user_id [INTEGER] NOT NULL
user
id [INTEGER] NOT NULL
dob [VARCHAR(50)] NOT NULL
fullname [VARCHAR(50)] NOT NULL
password [VARCHAR(100)] NOT NULL
qualification [VARCHAR(50)] NOT NULL
1
0..N

API Design

This Flask-based REST API manages authentication, subjects, chapters, quizzes, questions, and scores for a quiz app.
Key Endpoints

1. Auth → POST /register, POST /login (JWT-based authentication)
2. Subjects & Chapters → GET, POST /subjects, GET, POST, DELETE /subjects/<id>/chapters
3. Quizzes & Questions → GET, POST, DELETE /Quizzs, GET, POST, DELETE /questions
4. Scores → GET /scores, POST /scores
5. Email & Background Tasks → Uses Flask-Mail (SMTP) & Celery with Redis

Project Organization && Features

The project follows a modular Flask architecture with separate concerns:

- Backend (Flask)
    - app.py – Main entry point for the Flask API.
    - models.py – Defines database schema using SQLAlchemy.
    - routes/ – Contains controllers for different API endpoints (subjects.py, quizzes.py, etc.).
    - celery_config.py & tasks.py – Manages background tasks using Celery.
    - schedule.py – Handles periodic tasks.
    - instance/users.db – SQLite database file.
- Frontend (Vue.js)
    - public/ & src/ – Vue components structured for basic UI.

Features

1. User Authentication 🔑 – Implemented with JWT for secure login & registration.
2. Quiz Management 📝 – CRUD operations for subjects, chapters, quizzes, and questions.
3. Score Tracking 📊 – Users can submit & view scores.
4. Email Notifications 📧 – Configured with Flask-Mail for updates.
5. Background Tasks ⚡ – Celery handles asynchronous tasks efficiently.
6. API Security 🔐 – CORS enabled for frontend-backend communication.
7. Scheduler ⏳ – Uses Celery Beat for task automation (e.g., sending reports).

Project Video