Quiz Master Application (Knowlympics**)**

Author

Abhay Bairagi
22f1000829
22f1000829@ds.study.iitm.ac.in

A passionate programmer with a strong foundation in web development and Data Science. Currently pursuing B.S. in Data Science and Applications at IIT Madras.

Description

Quiz Master (Knowlympics) is a comprehensive quiz management system that allows users to take timed quizzes across various subjects and chapters. The application features user authentication, quiz scheduling, real-time scoring, and detailed performance analytics. Administrators can create and manage subjects, chapters, quizzes, and questions through an intuitive interface. User get daily updates and monthly report of performance in their mail.

Technologies Used

- **Backend**:
  - Flask: Python web framework for building the RESTful API
  - SQLAlchemy: ORM for database management
  - Flask-JWT-Extended: For secure authentication
  - Celery: For handling background tasks and scheduled jobs
  - Redis: For caching and message brokering
  - Flask-Mail: For sending email notifications
  - Flask-Limiter: For API rate limiting
- **Frontend**:
  - Vue.js: Progressive JavaScript framework
  - Vuex: State management
  - Vue Router: Client-side routing
  - Bootstrap: UI components and styling
  - Bootstrap Icons: For iconography

DB Schema Design

The database is designed with the following key entities:

1. **User**
   - Attributes: id, username, email, password_hash, name, created_at, updated_at
   - Relationships: Many-to-Many with Role through user_roles

2. **Role**
    - Attributes: id, name
    - Used for access control (admin/user)
3. **Subject**
    - Attributes: id, name, description
    - Relationships: One-to-Many with Chapter
4. **Chapter**
    - Attributes: id, name, description, subject_id
    - Relationships: One-to-Many with Quiz
5. **Quiz**
    - Attributes: id, name, description, difficulty, duration, chapter_id, start_time, end_time
    - Relationships: One-to-Many with Question
6. **Question**
    - Attributes: id, question, option1-4, correct_option, quiz_id
    - Constraints: correct_option must be between 1 and 4
7. **Attempt**
    - Attributes: id, user_id, quiz_id, score, timestamp, timespent, submitted_answers
    - Tracks user quiz attempts and scores

Architecture and Features

Project Structure

```
Quiz Master/
├── static/           # Frontend assets
│   └── js/
│       ├── admin/    # Admin dashboard components
│       ├── pages/    # Vue.js page components
│       └── components/ # Reusable components
├── templates/        # HTML templates
├── models.py        # Database models
├── main.py         # Flask application and routes
├── auth.py         # Authentication logic
├── tasks.py        # Celery background tasks
└── config.py       # Application configuration
```

Features

1. **User Management**
    - User registration and authentication

    - Role-based access control
    - Profile management
2. **Quiz Management**
    - Create/Edit/Delete quizzes
    - Schedule quizzes with start/end times
    - Multiple difficulty levels
    - Multiple choice questions
3. **Quiz Taking**
    - Timed quiz sessions
    - Real-time progress tracking
    - Immediate scoring and feedback
    - Result history
4. **Analytics**
    - Performance tracking
    - Monthly reports
    - Subject-wise analysis
    - Quiz attempt history
5. **Background Tasks**
    - Daily quiz reminders
    - Monthly performance reports
    - Export quiz history to CSV

Setup Instructions

1. **Prerequisites**

```
# Install Redis Server
# Windows: Download from https://github.com/microsoftarchive/redis/releases
# Linux: sudo apt-get install redis-server
```

5. **Installation**

```
# Create virtual environment
python -m venv venv
source venv/bin/activate  # Linux/Mac
venv\Scripts\activate     # Windows

 # Install dependencies
 pip install -r requirements.txt
```

13. **Configuration**
    - Update `config.py` with your email settings
    - Configure Redis connection if needed
    - Set appropriate JWT secrets
14. **Database Setup**

```
15.  # Initialize database
16.  flask db init
17.  flask db migrate
18.  flask db upgrade
```

19. **Running the Application**

```
20.  # Start Redis Server
21.  # Start Celery Worker
22.  celery -A workers.celery worker --loglevel=info
23.
24.  # Start Celery Beat (for scheduled tasks)
25.  celery -A workers.celery beat --loglevel=info
26.
27.  # Start Flask Application
28.  python main.py
```

API Documentation

The API documentation is available in the accompanying YAML file. Key endpoints include:

- Authentication: `/login`, `/register`
- Subjects: `/subjects`, `/create_subject`, etc.
- Quizzes: `/quizzes/<chapter_id>`, `/create_quiz`, etc.
- Questions: `/get_ques/<quiz_id>`, `/add_que/<quiz_id>`, etc.
- Attempts: `/submit_quiz`, `/get_attempts`
- Analytics: `/user/summary`

Video Demo

https://drive.google.com/file/d/1BB1_WPR-NVhqao7vdWXHgFnMR-VqmV9k/view?usp=drive_link