Backend Assessment – Secure Quiz API with CRUD & Scoring

Technology: Python (FastAPI + SQLAlchemy + JWT) or Node.js (Express + Sequelize/Mongoose + JWT)

Objective

Build a secure backend service that supports login, full CRUD operations on quiz questions, and evaluates results based on user submissions.

Authentication

1. `POST /register`

- Registers a new user.

Input:
```
{
  "email": "user@example.com",
  "password": "securepassword"
}
```

2. `POST /login`

- Logs in an existing user and returns a **JWT token**.

Output:
```
{
  "accessToken": "<JWT_TOKEN>"
}
```

Protected Routes (Require JWT)

All routes below should require the JWT token (except `/register` and `/login`). Token should be sent in the `Authorization` header as:

```
Authorization: Bearer <token>
```

---

1. CRUD Operations for Questions (Admin-only)

a. `POST /questions`

b. `GET /questions`

c. `GET /questions/{id}`

d. `PUT /questions/{id}`

e. `DELETE /questions/{id}`

---

2. `GET /quiz`

- Returns 2 random quiz questions (without correct answer) to the authenticated user.

---

3. `POST /quiz/result`

- Accepts the answers submitted by the logged-in user.
- Calculates score and saves result in DB.
- Returns correct answers for UI display.

Guidelines

- Use in-memory DB like SQLite or MongoDB memory server for simplicity.
- Secure endpoints using JWT.
- Validate inputs, handle errors gracefully.
- Structure code modularly (routes, models, controllers, utils).
