Coding Challenge: Google Forms-Based Quiz System

Objective

You are required to build a **Quiz System** that dynamically creates Google Forms for quizzes from a structured text-based question list. Your system will consist of:

- A **FastAPI backend** that:
  - Accepts a text file or structured text input containing quiz questions and answers.
  - Interacts with the **Google Forms API** (or an equivalent Google Apps Script) to generate Google Forms as quizzes.
  - Implements a **review workflow** before sending the quiz link to selected users.
- A **ReactJS frontend** that:
  - Handles file uploads or manual entry of quiz questions.
  - Displays newly created quizzes for review.
  - Allows approval of quizzes (triggering an email notification with the quiz link).
- A **Python SDK** (generated using OpenAPI Generator CLI) for programmatic quiz management.
- **Automation scripts** (PowerShell, Shell, or any other script) to simplify setup and execution.

Your solution should demonstrate **best practices** in:

- **FastAPI integration** with external services (Google Forms API).
- **React UI flows** for quiz creation and review.
- **OpenAPI documentation** for SDK generation.
- **Automation scripts** to efficiently deploy the system.

Leverage **LLMs, open-source libraries, or Google documentation** where applicable. Creativity in adding extra features is encouraged!

📌 Tasks & Requirements

1. Backend Development (FastAPI + Google Forms API)

Create Quiz

- **POST** `/quizzes/`
- **Request Body:** A structured text blob containing quiz questions and multiple-choice answers.
- **Behavior:**
  - Parse and validate the quiz questions and answers.
  - Create a draft quiz entry (`new quiz, not yet reviewed`).
  - Use **Google Forms API** or **Apps Script** to generate a Google Form-based quiz.
  - Save metadata (Form URL, Form ID) in the system.
- **Response:** Return metadata (quiz ID, status, Form URL).

Review & Approve Quiz

- **POST** `/quizzes/{quiz_id}/approve`
- **Behavior:**
  - Mark the quiz as **approved**.
  - Send an email (via **Gmail API, SMTP, or another method**) with the quiz Form URL to selected participants.
- **Response:** Confirmation message/status update.

Retrieve Quizzes

- **GET** `/quizzes/` → List all created quizzes (title, status, Form URL).
- **GET** `/quizzes/{quiz_id}` → Get details for a specific quiz.

Delete Quiz (Optional)

- **DELETE** `/quizzes/{quiz_id}` → Mark the quiz as deleted.
- **Response:** HTTP 204 or JSON confirmation message.

Trick Logic: Status Transitions

- Quizzes **start as draft**.
- They can only move from **draft** → **approved**.
- **Invalid transitions (e.g., approved → draft) should return a 400 error**.

OpenAPI Docs

- Ensure **FastAPI** exposes an OpenAPI spec (`http://localhost:8000/openapi.json`).
- Document request/response schemas properly.

Unit Tests

- Tests for:
  - Creating a quiz from text.
  - Attempting **valid/invalid** status transitions.
  - Approving a quiz and triggering an email notification.

---

2. Frontend Client (ReactJS)

Develop a ReactJS Application that Communicates with FastAPI:

✅ **Upload or Enter Quiz Questions**

- A simple form to paste/write questions and answer options.
- On submit, call **POST /quizzes/**.

✅ **List & View Quizzes**

- Fetch and display quizzes (`GET /quizzes/`).
- Show quiz details (`title, status, Form URL`).

✅ **Review & Approve**

- Button/dialog to approve a quiz (`POST /quizzes/{quiz_id}/approve`).
- Show success message when the email is sent.

✅ **Optional: Deletion**

- Support removing a quiz via **DELETE /quizzes/{quiz_id}**.

✅ **UI/UX Considerations**

- Focus on **correct interactions & error handling**.
- If an **invalid status transition** is attempted, display the backend's error message.

---

3. Python SDK (OpenAPI Generator CLI)

Generate the SDK

- Use the OpenAPI spec (`http://localhost:8000/openapi.json`).
- Example command:

```
openapi-generator-cli generate -i http://localhost:8000/openapi.json -g python -o google_quiz_sdk

```

Validate & Use the SDK

- After generation, ensure it supports:
  - **create_quiz()** → Create a quiz.
  - **approve_quiz()** → Approve a quiz.
  - **list_quizzes() / get_quiz_by_id()** → Retrieve quizzes.
  - **delete_quiz()** → Remove a quiz.

Sample Script for SDK Usage

```
from google_quiz_sdk.api.quizzes_api import QuizzesApi
from google_quiz_sdk import ApiClient

client = ApiClient()
quizzes_api = QuizzesApi(client)

# Retrieve all quizzes
quizzes = quizzes_api.get_quizzes()
print(quizzes)

```

---

4. Automation Scripts

Setup Script (PowerShell, Bash, etc.)

✅ **Python Virtual Environment**

- Create & activate a virtual environment.

✅ **Install Python Dependencies**

```
pip install -r requirements.txt

```

(Should include `fastapi, uvicorn, google-api-python-client, oauthlib`, etc.)

✅ **Configure Google Credentials**

- Download/provide `credentials.json` or use environment variables.
- Document how to obtain and place them correctly.

✅ **Install React Dependencies**

```
npm install

```

Execution Script

✅ **Start FastAPI Backend**

```
uvicorn main:app --host 0.0.0.0 --port 8000

```

✅ **Start React Frontend**

```
npm start

```

🎯 Completion Criteria

✅ **Functional System:**

- Accepts quiz questions and answers → Creates a new **Google Form (draft status)**.
- Presents the newly created quiz for **review in React UI**.
- Allows approval, triggering an **email notification**.

✅ **Status Rules Enforced:**

- Draft → Approved only (invalid transitions return a 400 error).

✅ **React Frontend:**

- Quiz creation interface.
- List, view, and approval flow.

✅ **Python SDK:**

- **Generated via OpenAPI.**
- **Demonstrated with a sample script.**

✅ **Automation:**

- One script to set everything up.
- One script (or set of commands) to run the system.

✅ **Testing:**

- Backend tests covering status transitions & quiz creation logic.

🚀 Bonus Features (Optional)

✅ **Enhanced Email Workflow**

- Custom email body, multiple recipients, or invitation messages.

✅ **Preview Mode**

- UI preview of quiz before form creation.

✅ **Authentication**

- Restrict quiz creation/approval to logged-in users (OAuth, JWT, etc.).

✅ **Detailed Error Messages**

- Clear prompts for missing/invalid **Google API credentials**.

📦 Deliverables

- ✅ **Backend (FastAPI) source code**

- ✅ **ReactJS frontend code**
- ✅ **Python SDK (OpenAPI generated)**
- ✅ **Setup & Execution Scripts**
- ✅ **Unit tests for backend**
- ✅ **README with setup instructions**

Good luck! 🚀