Building a Simple Joke API: Task Solutions

Your Name

November 22, 2024

Task 1: Create a GET Endpoint

The first task is to create a GET endpoint to retrieve a random joke. This endpoint allows users to request a joke from the server. Here is the solution:

```python
@app.get("/api/joke", response_model=Joke)
async def get_joke():
    if not jokes:
        raise HTTPException(status_code=404, detail="No jokes available")
    joke = random.choice(jokes)
    return joke
```

Explanation:

- We define a function `get_joke` using the `@app.get` decorator to specify that it's a GET endpoint.
- The function checks if there are any jokes available. If not, it raises an error with a message "No jokes available."
- If jokes are available, it randomly selects one and returns it.

Task 2: Create a POST Endpoint

The second task is to create a POST endpoint to add new jokes. This allows users to submit new jokes to the server. Here's the solution:

```python
@app.post("/api/add-joke", response_model=Joke)
async def add_joke(joke: str):
    joke_id = len(jokes) + 1  # Simple ID generation
    new_joke = Joke(content=joke, id=joke_id)
    jokes.append(new_joke)
    return new_joke
```

Explanation:

- The function `add_joke` is defined with the `@app.post` decorator to indicate it's a POST endpoint.
- A new joke ID is generated by adding 1 to the current number of jokes.
- The new joke is added to the list of jokes and returned to confirm it was added successfully.

Task 3: Create a DELETE Endpoint

The third task is to create a DELETE endpoint to remove jokes by ID. This lets users delete specific jokes. Here is the solution:

```
@app.delete("/api/delete-joke/{joke_id}", response_model=Joke)
async def delete_joke(joke_id: int):
    joke = next((j for j in jokes if j.id == joke_id), None)
    if joke is None:
        raise HTTPException(status_code=404, detail="Joke not found")
    jokes.remove(joke)
    return joke
```

Explanation:

- We use the `@app.delete` decorator to define the `delete_joke` function as a DELETE endpoint.
- The function searches for a joke with the given ID. If not found, it raises an error "Joke not found."
- If the joke is found, it is removed from the list.

Task 4: Add Error Handling

The final task is to add error handling to manage situations like no jokes being available or trying to delete a non-existent joke. This ensures the API behaves predictably even in error conditions.

Explanation:

- Error handling is implemented using `HTTPException`, which provides a way to return meaningful error messages and status codes to the user.
- In the GET endpoint, if no jokes are available, an error message is returned to the user.
- In the DELETE endpoint, if a joke with the specified ID does not exist, an error message is returned.