Structuring the FastAPI Endpoints

- Define the routes for each feature (e.g., POST /tasks, GET /users).
- Use Pydantic models to validate request and response data.
- Error Handling: Ensure proper error handling and response codes for each endpoint.
- Authentication: Consider adding user authentication via JWT tokens.
- Scalability: Ensure your endpoints can scale and handle errors gracefully.

Example

```python
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

# Pydantic model to define the request body structure
class Item(BaseModel):
    name: str
    description: str

# POST endpoint to create an item
@app.post("/items/")
def create_item(item: Item):
    return {"message": "Item created successfully", "item": item}

# GET endpoint to retrieve all items
@app.get("/items/")
def get_items():
    return {"items": ["item1", "item2", "item3"]}

# GET endpoint for a specific item by ID
@app.get("/items/{item_id}")
def get_item(item_id: int):
    return {"item_id": item_id, "name": f"Item {item_id}"}
```