Coding & Development / API Development

Create an OpenAPI 3.0 specification YAML file from described API endpoints and data models.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: API Documentation, OpenAPI, Swagger
Updated: May 2026
Why This Prompt Exists
Most APIs lack documentation — or documentation is out of date.

You get:

- no API specification (hard to consume)
- documentation that doesn’t match implementation
- no auto-generated client SDKs
- no interactive API explorer
- difficult onboarding for new developers

But OpenAPI is not optional for public APIs.

It is the standard for API documentation.

- Info: title, description, version, contact
- Servers: base URLs for different environments
- Paths: endpoints with methods, parameters, responses
- Components: schemas, security schemes, parameters
- Security: authentication requirements

Without OpenAPI, your API is harder to use.

This framework forces AI to generate OpenAPI specs.

The Prompt

```
Assume the role of an API documentarian who creates OpenAPI 3.0
specifications.

Your task is to generate an OpenAPI spec YAML file.

Generate:

1. INFO SECTION
   - title, description, version, contact

2. SERVERS
   - Production, staging, local URLs

3. PATHS
   - Each endpoint with method, parameters, requestBody, responses

4. COMPONENTS / SCHEMAS
   - Data models for requests and responses

5. COMPONENTS / SECURITY SCHEMES
   - Authentication method (API key, JWT, OAuth)

6. SECURITY (global or per-endpoint)

INPUTS:
```

```
API Title:
[INSERT]

API Version:
[INSERT]

Base URL:
[INSERT]

Endpoints (list with methods, descriptions, request/response schemas):
[LIST]

Authentication Type:
[API KEY / JWT / OAUTH / NONE]

Data Models (schemas for requests/responses):
[DESCRIBE]

RULES:
- Use OpenAPI 3.0 format (not 2.0)
- Include examples for request/response bodies
- Use $ref for reusable schemas
- Specify required fields in schemas
- Include error responses (400, 401, 403, 404, 500)
- Add descriptions for all fields
- Use proper HTTP status codes
```

How To Use It

- List all endpoints with methods and descriptions before generating.

- Define data models (schemas) for requests and responses.
- Include examples — they make the spec more useful.
- Validate the generated YAML with an OpenAPI validator (e.g., swagger.io validator).
- Use tools like Swagger UI or Redoc to render the documentation.

Example Input

API Title: TaskFlow API

API Version: 1.0.0

Base URL: https://api.taskflow.com/v1

Endpoints: GET /users (list users), POST /users (create user), GET /users/{id} (get user), PUT /users/{id} (update user), DELETE /users/{id} (delete user)

Authentication Type: JWT (Bearer token)

Data Models: User object with id, email, name, createdAt, updatedAt

Why It Works
Most APIs lack OpenAPI specs.

This framework improves outcomes by forcing:

- info section (identification)
- servers (environments)
- paths (endpoint definitions)
- schemas (data models)
- security (authentication)

Great OpenAPI specs don’t just document — they enable auto-generated clients and interactive docs.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, API development frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also The API Rate Limiter Implementation