API Error Handling & Best Practices

Building Robust LLM Applications

⚠ 401 Unauthorized

Cause: Invalid or missing API key

Solution: Check API key format and permissions

Prevent: Secure key storage, regular key rotation

⚠ 429 Too Many Requests

Cause: Rate limit exceeded

Solution: Implement exponential backoff

Prevent: Track request rates, use queuing

⚠ 400 Bad Request

Cause: Invalid parameters or format

Solution: Validate inputs before sending

Prevent: Input validation, parameter checking

⚠ 500 Internal Server Error

Cause: Provider-side issues

Solution: Retry with exponential backoff

Prevent: Implement robust retry logic

Understanding Limits

- **RPM:** Requests per minute (e.g., 60 RPM)
- **TPM:** Tokens per minute (e.g., 90,000 TPM)
- **Daily quotas:** Total usage per day
- **Concurrent requests:** Simultaneous request limits

Mitigation Strategies

- **Request queuing:** Queue requests to stay under limits
- **Token estimation:** Count tokens before API calls
- **Batch processing:** Combine multiple requests when possible
- **Graceful degradation:** Fallback options for limit hits

Best Practices Code Example

```python
import time
import random

def call_llm_with_retry(prompt, max_retries=3):
    for attempt in range(max_retries):
        try:
            response = client.chat.completions.create(
                model="gpt-3.5-turbo",
                messages=[{"role": "user", "content":
prompt}],
                timeout=30
            )
            return response
        except RateLimitError:
            if attempt < max_retries - 1:
                wait_time = (2 ** attempt) + random.uniform(0,
1)
                time.sleep(wait_time)
            else:
                raise
        except APIError as e:
            print(f"API Error: {e}")
            return None
```

Production Readiness Checklist

✓	**API Key Security:**	Environment variables, secure storage
✓	**Error Handling:**	Comprehensive try-catch blocks
✓	**Retry Logic:**	Exponential backoff with jitter
✓	**Rate Limiting:**	Request queuing and throttling
✓	**Monitoring:**	Log errors and track usage
✓	**Fallback Options:**	Alternative responses for failures
✓	**Timeout Handling:**	Set appropriate timeout values
✓	**Input Validation:**	Sanitize and validate all inputs