Integrating AI with Applications via OpenAI API

Unlock innovation with Large Language Models (LLMs). Learn strategic AI implementation and optimize your development process with powerful AI integration.

W **by Wojciech Żółtowski**

Agenda

- When to Use AI vs. Traditional Coding
- Case Study: AI for Content Generation
- Key Considerations for LLM Integration
- Best practices
- Examples of LLM Applications
- The Future of AI-Powered Applications
- Code or Not to Code?
- Code Examples

When to Use AI vs. Traditional Coding

AI Applications

- Unstructured data processing: Text, images, audio, etc.
- Natural language understanding or generation: Conversational agents, summarization, sentiment analysis
- Complex pattern recognition (data analysis): Detecting fraud, generating recommendations.

Example: Chatbots or virtual assistants, Image recognition or captioning, semantic search

Traditional Coding

- Well-defined rules
- Deterministic and predictable outcomes
- Performance-critical computations

Example: Processing financial transactions with fixed rules, input validation, usage of well-known mechanisms/algorithms, security critical systems.

Hybrid Approach

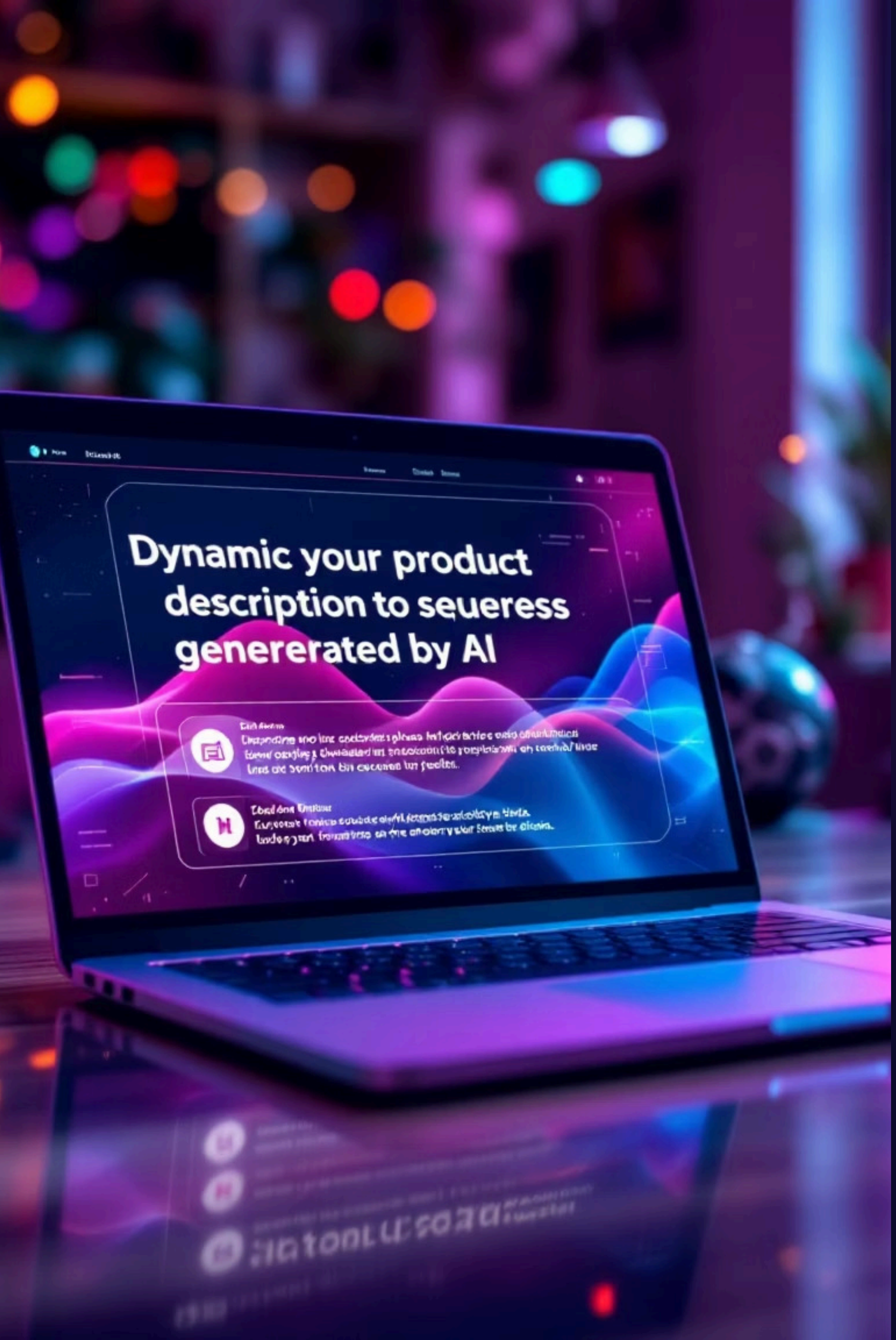
Use AI for:

- Handling user input ("What can I help you with today?")
- Extracting intent or entities from text
- Recommending actions or generating responses

Use Traditional Code for:

- Executing user commands
- Validating and sanitizing AI-generated input
- Enforcing business rules and data integrity

Example:
A support chatbot uses AI to understand user queries, but traditional code to pull account info or reset a password.


Dynamic your product description to seueress genererated by AI

Case Study: AI for Content Generation

Scenario

Dynamic product descriptions needed for an e-commerce platform.

AI Approach

Generates unique, scalable descriptions based on product attributes.

Traditional Approach

Manual template writing and detail filling for each product.

Trade-offs

AI offers scalability but may produce inaccuracies; requires prompt engineering.

Key Considerations for LLM Integration

Data Privacy

Ensure compliance with relevant law. What information may/mustn't be sent to external companies like Open AI.

Prompt Engineering

Crafting effective prompts to achieve accurate outputs.

Cost Management

Monitor API usage with rate limiting and caching strategies.

Security Best Practices

✓ Input Validation

Sanitize inputs to prevent prompt injection attacks.

→ Output Moderation

Filter responses to remove harmful or inappropriate content. Example: Use OpenAI's Moderation API for filtering.

Logging and Auditing

Log prompts and responses, track prompt changes over time to debug or understand output behavior.

Application Architecture with LLMs

Microservices

Isolate LLM functions into independent services for scalability.

Asynchronous Processing

Handle long-running LLM tasks in the background smoothly.

Caching

Store responses to reduce latency and API calls.

QA and Debugging LLM-Based Applications

1 Challenges

Non-deterministic outputs and subjective performance evaluations.

2 Frameworks/tools

Prompts tracing, testing, evaluation. The most popular languages: JS/Python.

3 Continuous monitoring

Also technological changes.

Examples of LLM Applications

Conversational Interfaces

Chatbots, virtual assistants, customer support

Text Understanding & Summarization

Summarization, classification, sentiment analysis

Retrieval-Augmented Generation (RAG)

Fact-based answers from your own data

Text Generation

Content writing, email drafting, product descriptions

Code Generation and Dev Tools

Autocomplete, refactoring, code explanation

Education and Tutoring

Interactive tutors, personalized learning

Reasoning & Decision Support

Financial modeling, legal reasoning, strategic planning

The Future of AI-Powered Applications

Strategic AI Integration

Unlock new functionalities and business opportunities.

Balanced Approach

Combine AI with traditional coding for best results.

Continuous QA & Enhancement

Maintain success through ongoing evaluation and improvement.

Embrace Innovation

Stay ahead by adopting AI technologies proactively.

Code or Not to Code?