🎨 Module 4: Domain-Specific Applications

> ***Weeks 10-12*** *| Real-world prompt applications*

Week 10: Creative and Content Generation

✍️ Writing Assistance and Editing

Prompt for Writing Improvement:

```
Act as a professional editor. Improve the following text:

<original>
The meeting was really long and we talked about many things
and some of them were important but some weren't.
</original>

Improve for:
- Clarity and conciseness
- Professional tone
- Active voice

Provide:
1. Edited version
2. List of changes made
3. Brief explanation of why each change improves the text
```

Prompt for Style Matching:

```
Analyze the writing style of this sample:

<sample>
[Sample text in desired style]
</sample>

Based on this style (tone, vocabulary, sentence structure),
rewrite the following text to match:

<rewrite_this>
[Text to be rewritten]
</rewrite_this>
```

Grammar and Proofreading:

```
Proofread this text for:
- Grammar errors
- Spelling mistakes
- Punctuation issues
- Word choice improvements

<text>
```

```
Their going to the store to buy there supplies for they're trip.
</text>

Format response as:
| Original | Correction | Rule/Reason |
|----------|------------|-------------|
```

📣 Marketing Copy and Advertising

Product Description Framework:

```
Create a product description using the PAS framework:

Product: [Product name and basic info]
Target audience: [Who is this for]

Structure:
**Problem:** What pain point does this solve?
**Agitation:** Why is this problem really frustrating?
**Solution:** How does our product solve it?

Include:
- Compelling headline
- 3 key benefits (not features)
- Call to action

Tone: [Professional/Casual/Luxurious/etc.]
Length: [word count]
```

Example:

```
Create a product description using the PAS framework:

Product: Noise-canceling wireless headphones ($199)
Target audience: Remote workers, 25-45 years old

Structure:
- Problem: Distractions during work calls
- Agitation: Impact on productivity and professionalism
- Solution: Crystal clear audio anywhere

Tone: Professional but friendly
Length: 150 words
```

Social Media Variations:

```
Create social media posts for this product launch:

Product: [Product details]

Create versions for:
1. Twitter/X (280 chars, punchy, with hashtags)
```

```
2. LinkedIn (professional, 100-150 words)
3. Instagram (visual-focused, with emoji, CTA)

Each should highlight a different benefit.
```

📖 Creative Fiction and Storytelling

Story Prompt Template:

```
Write a short story with these parameters:

Genre: [Mystery/Sci-Fi/Romance/etc.]
Setting: [Time and place]
Main Character: [Name, brief description]
Conflict: [Central problem]
Tone: [Dark/Humorous/Heartwarming/etc.]
Length: [Word count]

Requirements:
- Start with action or dialogue (no "Once upon a time")
- Include sensory details
- End with [open ending/twist/resolution]
```

Character Development:

```
Create a detailed character profile:

Basic:
- Name: [Name]
- Age: [Age]
- Occupation: [Job]

Develop:
- Background (3-4 sentences)
- Core motivation
- Greatest fear
- Speech pattern (with example dialogue)
- A secret they keep
- How they change under pressure
```

🎨 Style Matching and Tone Control

Tone Spectrum Prompt:

```
Rewrite this message in 4 different tones:

Original: "Please submit your report by Friday."

1. **Formal/Corporate:** [rewrite]
2. **Friendly/Casual:** [rewrite]
```

```
3. **Urgent/Pressing:** [rewrite]
4. **Encouraging/Supportive:** [rewrite]
```

Brand Voice Guide:

```
You are writing for [Brand Name] with this voice:

BRAND VOICE GUIDE:
- Tone: [Friendly but professional]
- Vocabulary: [Simple, no jargon]
- Values: [Innovation, customer-first]
- Avoid: [Corporate speak, negative language]
- Signature phrases: [e.g., "Let's build together"]

Example brand-appropriate text:
"[Example of ideal brand writing]"

Now write: [Content request]
```

Week 11: Data Analysis and Code

💻 Code Generation

Basic Code Generation:

```
Write a Python function that:
- Name: calculate_discount
- Parameters: original_price (float), discount_percent (float)
- Returns: discounted price rounded to 2 decimal places
- Include: input validation, docstring, type hints

Example usage:
calculate_discount(100.00, 20) → 80.00
```

Code with Context:

```
Language: Python 3.11
Framework: FastAPI
Style: Follow PEP 8

Create an API endpoint that:
- Route: GET /users/{user_id}
- Returns user data from database
- Handles: user not found (404)
- Uses: Pydantic models for response

Include:
- Type hints
- Error handling
- Brief docstrings
```

🐛 Debugging Assistance

Debug Prompt:

```
Debug this code. It should [expected behavior] but instead
[actual behavior].

<code language="python">
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n-1) + fibonacci(n-2)

# Problem: fibonacci(50) takes forever
</code>

Provide:
1. Root cause of the bug/issue
2. Explanation of why it happens
3. Fixed code with comments
4. Performance comparison if relevant
```

Error Analysis:

```
I'm getting this error:

<error>
TypeError: cannot unpack non-iterable NoneType object
  at line 15 in process_data.py
</error>

<code>
[Relevant code snippet]
</code>

What's causing this and how do I fix it?
```

📊 Data Extraction and Transformation

Data Extraction:

```
Extract structured data from this unstructured text:

<text>
John Smith called on March 15, 2024 at 2:30 PM about a
billing issue. His account number is AC-789456. He was
charged $150 instead of $100. Issue resolved by crediting $50.
</text>

Extract into JSON:
{
  "customer_name": "",
```

```
  "date": "",
  "time": "",
  "issue_type": "",
  "account_number": "",
  "amounts_mentioned": [],
  "resolution": ""
}
```

Data Transformation:

```
Convert this CSV data to a different format:

<csv_data>
name,age,city
John,30,NYC
Jane,25,LA
Bob,35,Chicago
</csv_data>

Convert to:
1. JSON array
2. Python dictionary
3. SQL INSERT statements (table: users)
```

📝 Technical Documentation

Documentation Generator:

```
Generate documentation for this function:

<code>
def process_order(order_id, items, customer_info,
                  apply_discount=False, priority="normal"):
    # ... implementation
</code>

Include:
1. Description (what it does)
2. Parameters table (name, type, description, required/optional)
3. Return value
4. Raises (potential exceptions)
5. Example usage
6. Notes/warnings

Format: Markdown
```

API Documentation:

```
Document this REST API endpoint:

Endpoint: POST /api/v1/orders
Request body: (provide example)
```

```
Response: (provide example for success and error)

Generate documentation including:
- Description
- Request format
- Response format
- Status codes
- Example curl command
- Common errors
```

Week 12: Customer Service and Conversational AI

🤖 Building Chatbots and Assistants

System Prompt for Customer Service Bot:

```
You are a customer service representative for [Company Name].

IDENTITY:
- Name: Alex
- Role: Customer Support Specialist
- Personality: Friendly, patient, helpful

CAPABILITIES:
- Answer questions about products
- Help with orders and shipping
- Process returns and refunds
- Troubleshoot common issues

LIMITATIONS:
- Cannot process payments
- Cannot access personal accounts (direct to secure portal)
- Cannot make promises about future products

CONVERSATION RULES:
1. Greet warmly on first message
2. Show empathy for frustrations
3. Always confirm understanding before solving
4. Offer next steps proactively
5. End with "Is there anything else I can help with?"

ESCALATION:
If issue requires human: "I'll connect you with a specialist
who can help further. Please hold while I transfer you."
```

💬 Handling Customer Queries

Query Classification:

```
Classify this customer message and suggest a response approach:
```

```
<message>
I ordered a blue shirt but received a red one. This is the
second time this happened! I want a refund immediately.
</message>

Classification:
1. Issue Type: [Order error/Billing/Technical/General inquiry]
2. Urgency: [High/Medium/Low]
3. Sentiment: [Positive/Neutral/Negative/Angry]
4. Customer History: Repeat issue (mentioned "second time")

Response Approach:
- Acknowledge: [What to acknowledge]
- Apologize: [Specific apology]
- Action: [Immediate steps]
- Prevention: [What to offer to prevent recurrence]
```

Response Template:

```
Generate a customer service response:

Customer Issue: [Issue description]
Customer Tone: [Frustrated/Calm/Confused]
Relevant Policy: [Policy information]

Response Structure:
1. Empathy statement (acknowledge feelings)
2. Apologize specifically (not generally)
3. Solution/Next steps
4. Goodwill gesture (if appropriate)
5. Invitation for further help

Tone: Professional but warm
Length: 3-4 sentences
```

Maintaining Brand Voice

Brand Consistency Checker:

```
Review this customer service response for brand alignment:

BRAND VOICE GUIDELINES:
- Tone: Warm, not corporate
- Use: "Happy to help" not "We apologize for any inconvenience"
- Avoid: Jargon, passive voice, blame
- Always: Use customer's name, offer specific solutions

<response>
[Draft response to check]
</response>

Provide:
```

```
1. Brand compliance score (1-10)
2. Issues found
3. Revised version
```

🔀 Escalation and Handoff

Escalation Detection:

```
Analyze if this conversation needs human escalation:

<conversation>
[Conversation history]
</conversation>

Escalation Triggers:
- Legal threats
- Requests for supervisor
- Complex multi-issue problems
- Emotional distress
- Account security concerns
- Repeat contact for same issue

Response format:
{
  "needs_escalation": true/false,
  "trigger_reason": "[reason]",
  "urgency": "high/medium/low",
  "suggested_department": "[department]",
  "handoff_summary": "[brief summary for human agent]"
}
```

💡 Key Takeaways

1. **Match tone** to audience and purpose
2. **Use frameworks** (PAS, AIDA) for marketing
3. **Provide context** for code generation
4. **Show examples** for data extraction
5. **Define personality and limitations** for chatbots

Next: Module 5 - Evaluation and Optimization →