Technical Here are the carefully crafted prompts used in the scripts to generate ABAP technical documentation:

1. Class Component Documentation Prompts

Opening Definition Prompt:

```
python
"""As a senior SAP ABAP developer, you MUST create complete, technically precise documentation of this {ctype}'s definition. Document ONLY the core definition details.


CRITICAL REQUIREMENTS for documentation:
1. YOU MUST INCLUDE EVERY SECTION BELOW - NO EXCEPTIONS
2. EVERY TABLE MUST HAVE PROPER HEADERS WITH | SEPARATORS
3. MAINTAIN EXACT FORMATTING WITH PROPER MARKDOWN
4. NEVER USE "..." OR SIMILAR ABBREVIATIONS


## Overview
- MUST be one or two sentences about the technical purpose of this component
- NEVER repeat information that appears in other sections
- In general, if any abbreviations appear, do not expand them or guess their meaning-keep using them exactly as they appear.
- Focus ONLY on the high-level purpose


## Class Definition
| Aspect | Details |
|--------|---------|
| Scope | [PUBLIC/PRIVATE/PROTECTED] |
| Type | [FINAL or ABSTRACT] |
| Superclass | [Superclass name or "None"] |
| Create Permission | [PUBLIC/PROTECTED/PRIVATE] |
| Friends | [Friend classes/interfaces or "None"] |


(do not include this code in your response):
{chunk}
"""
```

Methods Definition Prompt:

```
python
"""As a senior SAP ABAP developer, document these {section_type} methods for {ctype}. Document ONLY the method definitions.

CRITICAL REQUIREMENTS for method documentation:
1. Document EVERY method in this {section_type} section
2. NEVER skip parameters
3. Use exact types and modifiers
4. MAINTAIN EXACT MARKDOWN FORMAT

Format each method as:

### Method name
| Importing | Exporting | Changing | Returning | Exceptions | Purpose |
|-----------|-----------|----------|-----------|------------|---------|
[For each method:
- Importing: List all IMPORTING parameters with exact types and modifiers (or "None")
- Exporting: List all EXPORTING parameters with exact types and modifiers (or "None")
- Changing: List all CHANGING parameters with exact types and modifiers (or "None")
- Returning: List all RETURNING parameters with exact types and modifiers (or "None")
- Exceptions: List all possible exceptions (or "None")
- Purpose: Brief description of method's purpose]

ABAP Methods to document:
{methods}

Batch {current_batch} of {total_batches}
"""
```

Implementation Overview Prompt:

```
python
"""As a senior SAP ABAP developer, create complete, technically precise documentation of this {ctype} implementation overview.
```

```
Important rules!
1. Keep any abbreviations exactly as they appear.
2. MAINTAIN EXACT MARKDOWN FORMAT
3. NEVER USE "..." OR SIMILAR ABBREVIATIONS

## Implementation Overview
Brief technical summary of the implementation approach and key patterns used.

## Method Dependencies
| Method | Calls | Called By |
|--------|-------|-----------|
[Document which methods call each other. Include external function calls]

## Redefined Methods
| Method | Source | Implementation Details |
|--------|--------|----------------------|
[Document methods redefined from superclass/interfaces. If none exist, write "No methods redefined"]

## Database Tables Used
| Table Name | Purpose | Key Fields |
|------------|---------|------------|
[Document ONLY direct database table access. If none used, write "No database tables used"]

## Critical Sections
| Section | Methods Involved | Purpose | Considerations |
|---------|-----------------|---------|---------------|
[Document areas requiring special attention]

Based on analyzing this code:
{chunk}
"""
```

2. Function Module Documentation Prompts

Main Function Module Prompt:

```
python
"""As a senior SAP ABAP developer, create comprehensive technical documentation for this ABAP function module with detailed explanations. You must document all modules with specific details about their processing logic and interactions.

# Technical Documentation: {self.program_name}

Component Type: module

Generated: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}

## Overview

Based on the provided ABAP code, document the purpose and key functionality of these modules. Identify:

- Main business process served
- User interface components handled
- Module interactions and flow
- Key data processing

## Dynapro Architecture

Document the screen flow and module architecture, including:

1. Screen sequence and navigation
2. Module organization (INPUT vs OUTPUT modules)
3. Pattern of control flow between modules
4. State management approach

## Screen Organization

Document the screens and their associated modules:

{screens_formatted}

## Module Details

For each module, provide a DETAILED description including:

1. What the module does and its exact purpose in the program
2. Whether it's an INPUT or OUTPUT module and what that means for its function
3. Any screen elements it interacts with
4. Any function modules or methods it calls
5. Any database interactions it performs
6. The complete processing steps in detail
```

```
7. How the module fits into the overall screen flow

{modules_formatted}

## Error Handling
Document the error handling approach and exception processing in detail:
1. How errors are identified
2. Error messages used
3. Exception handling patterns
4. Recovery mechanisms

Generate this documentation based on the following ABAP code:
```abap
{self.code_content}
```

Documentation Rules:

1. Focus on technical details rather than business logic
2. Document the ACTUAL functionality visible in the code, not theoretical usage
3. Be extremely detailed about modules and their processing - thoroughly explain each module's logic
4. Maintain precise technical language appropriate for SAP developers
5. Identify clear sections with headers
6. Explain processing sequences within and between modules
7. Note any unusual or non-standard ABAP patterns
8. Thoroughly explain the dynapro screen flow including how modules work together
9. For each module, explicitly note whether it's an INPUT or OUTPUT module and what this means for its function"""

3. **Interface Documentation Prompt**

```python
"""As a senior SAP ABAP developer, document ONLY the interfaces for this {ctype}.

CRITICAL REQUIREMENTS:
1. Document EVERY interface
2. Keep interface names EXACTLY as they appear
3. MAINTAIN EXACT MARKDOWN FORMAT
```
```

```
4. List ALL interfaces, even if purpose is unclear

### Interfaces
| Interface | Purpose |
|-----------|---------|
[List ALL implemented interfaces. If none, write "No interfaces implemented"]

List the interfaces in this code:
{chunk}
"""
```

4. Report/Forms Documentation Prompts

Main Report Prompt:

```
python
"""As a senior SAP ABAP developer, create comprehensive technical documentation for this ABAP report with detailed explanations. You must document all forms with specific details about their parameters and processing logic.

# Technical Documentation: {self.program_name}
Component Type: program
Generated: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}

## Overview
Based on the provided ABAP code, document the purpose and key functionality of this report. Identify:
- Main business process served
- Primary user audience
- System integration points
- Key data flows

## Program Flow
Document the execution flow of the program, including:
1. INITIALIZATION section purpose and actions
2. AT SELECTION-SCREEN processing
3. START-OF-SELECTION processing and its conditional logic
```

4. END-OF-SELECTION actions
5. Any other event blocks and their purpose

Data Structures

{data_structures_formatted}

Selection Screen

Document all parameters, select-options, and their purpose in the user interface.

Main Processing Logic

Analyze and document the core processing logic of the report:

1. Program initialization sequence
2. Data selection approach
3. Main processing algorithm
4. Output generation method
5. Error handling mechanisms

Subroutines (Forms)

For each form, provide a DETAILED description including:

1. What the form does and its exact purpose in the program
2. Each input and output parameter with its specific purpose in the form's execution
3. The complete processing steps in detail
4. Any specific business rules implemented
5. How the form interacts with other forms or functions
6. The different execution paths based on conditions
7. Error handling within the form

{forms_formatted}

Database Interaction

Tables Used

{tables_formatted}

Explain exactly how each table is used (read, update, etc.) and the key data being accessed.
Specify which forms and sections access each table.

```
## Function Module Usage
{function_calls_formatted}

For each function module, explain its specific purpose in the program context and the exact parameters being passed.

## Authorization and Security
Identify authorization checks and security considerations

## Performance Considerations
Note any batch processing, parallelization, or optimization techniques

## Integration Points
Document RFC connections, external system calls, and other integration methods

## Error Handling
Document the error handling approach and exception processing in detail:
1. How errors are identified
2. Error messages used
3. Exception handling patterns
4. Recovery mechanisms

Generate this documentation based on the following ABAP code:
```abap
{self.code_content}
```

Documentation Rules:

1. Focus on technical details rather than business logic
2. Document the ACTUAL functionality visible in the code, not theoretical usage
3. Be extremely detailed about forms and their processing - thoroughly explain each form's parameters and logic
4. Maintain precise technical language appropriate for SAP developers
5. Identify clear sections with headers
6. Explain data flow and processing sequences
7. Note any unusual or non-standard ABAP patterns
8. Document error handling and exception processing
```

9. Thoroughly explain the program execution flow including START-OF-SELECTION logic"""

Focused Forms Documentation Prompt:

```python
"""As a senior SAP ABAP developer, create detailed technical documentation ONLY for the following forms in the ABAP report "{self.program_name}":

{forms_to_document}

For each form listed above, provide a DETAILED description including:
1. A clear descriptive header with the form name (e.g., ### formname)
2. What the form does and its exact purpose in the program based on the code
3. Each input and output parameter with its specific purpose in the form's execution
4. The complete processing steps in detail - describe what the code is actually doing
5. Any specific business rules implemented that are visible in the code
6. How the form interacts with other forms or functions based on the code
7. The different execution paths based on conditions found in the code
8. Error handling within the form as implemented in the code

Generate this documentation based on the following ABAP code:
```abap
{self.code_content}
```

IMPORTANT INSTRUCTIONS:

- Start each form documentation with a level-3 header (###) showing the form name
- Focus strictly on the forms listed above - do not document other forms
- Be extremely detailed in the parameter descriptions and processing logic
- Document ONLY the ACTUAL functionality visible in the code, do not hallucinate or invent functionality
- ONLY describe what is provably present in the code - avoid assumptions
- For each parameter, explain exactly how it's used in the form
- For each operation or function call, explain its specific purpose based on the code context
- If you're not 100% certain about something, qualify it with phrases like "appears to" or "likely"
- Always describe the EXACT processing sequence in the form, step-by-step
- Do not omit any significant operations in the form
- Trace the full data flow through the form
```

- Make sure your documentation is complete - do not end mid-sentence or with partial information"""

Key Prompt Design Principles:

1. **Role Setting**: Always starts with "As a senior SAP ABAP developer"
2. **Strict Requirements**: Uses "CRITICAL REQUIREMENTS" and "MUST" for emphasis
3. **Structured Output**: Specifies exact markdown formatting with tables
4. **No Hallucination**: Repeatedly emphasizes documenting "ONLY what is visible in the code"
5. **Completeness**: Instructions to avoid abbreviations like "..." and ensure full documentation
6. **Technical Precision**: Maintains SAP-specific terminology and technical accuracy
7. **Detailed Instructions**: Step-by-step requirements for each section
8. **Error Prevention**: Multiple warnings against inventing functionality

Documentation Prompts used by claude