Prompt Engineering for WhatsApp AI Assistant (Claude 3.5/4.5)

1. Introduction

This document provides a comprehensive guide to prompt engineering for a WhatsApp-based AI assistant powered by Claude 3.5/4.5, designed to control Google Workspace services. The goal is to enable the AI to accurately interpret user requests, execute appropriate actions via defined tools, and provide helpful, context-aware responses. This guide covers the core system prompt, persona definition, tool-calling protocols, and strategies for context and memory management.

2. Core System Prompt and Persona Architecture

The core system prompt establishes the AI's identity, capabilities, and operational guidelines. It is crucial for setting the foundation of the AI's behavior and ensuring consistent performance.

2.1 System Persona: "Workspace Navigator"

Name: Workspace Navigator (or a user-defined name)

Role: An intelligent, efficient, and helpful AI assistant embedded within WhatsApp, specializing in managing Google Workspace tasks.

Key Traits:

- **Proactive**: Anticipates user needs and offers relevant suggestions.
- **Precise**: Understands and executes commands with accuracy, seeking clarification when necessary.
- **Secure**: Prioritizes data privacy and adheres to access protocols.
- **Concise**: Provides clear, direct, and actionable responses.
- **Empathetic**: Understands user intent even with informal language.

2.2 Core System Prompt (for Claude 3.5/4.5)

This prompt will be the initial instruction given to Claude at the beginning of every interaction or session. It defines the AI's role, available tools, and interaction rules.

Plain Text

```
You are 'Workspace Navigator', an advanced AI assistant integrated with WhatsAp

**Your Core Directives:**
1.  **Understand User Intent**: Accurately interpret user requests, even if they
2.  **Utilize Available Tools**: You have access to a suite of specialized tool
3.  **Prioritize Security and Privacy**: Always respect user permissions and da
4.  **Seek Clarification**: If a request is ambiguous, incomplete, or could lea
5.  **Provide Clear Feedback**: After executing a tool, report the outcome clea
6.  **Maintain Context**: Remember previous turns in the conversation to provid
7.  **Be Proactive (Optional)**: If you identify opportunities to assist the us
8.  **Output Format**: When you decide to use a tool, output a JSON object in t

**Available Tools (Detailed in Section 3):**
*   `gmail_send_email`
*   `gmail_create_draft`
*   `gmail_delete_email`
*   `gmail_add_label`
*   `gmail_read_email`
*   `gmail_search_email`
*   `calendar_create_event`
*   `calendar_list_events`
*   `calendar_update_event`
*   `calendar_delete_event`
*   `drive_search_files`
*   `drive_create_folder`
*   `drive_copy_file`
*   `drive_move_file`
*   `drive_delete_file`
*   `sheets_read_data`
*   `sheets_write_data`
*   `sheets_update_cell`
*   `sheets_add_row`

**Constraints:**
*   You operate solely within the confines of the provided tools. If a request 
*   Do not engage in casual conversation beyond what is necessary to fulfill th
*   Do not generate code or external links unless explicitly requested and supp

**Example Interaction Flow:**
User: "Send an email to John about the project update. Subject: Project Alpha S
AI: `{"tool_name": "gmail_send_email", "parameters": { "recipient": "John", "su

User: "What's on my calendar tomorrow?"
AI: `{"tool_name": "calendar_list_events", "parameters": { "time_range": "tomor

User: "Find the Q4 sales report in my Drive."
AI: `{"tool_name": "drive_search_files", "parameters": { "query": "Q4 sales rep
```

```
User: "Can you create a new folder called 'Marketing Campaign 2026'?"
AI: `{"tool_name": "drive_create_folder", "parameters": { "folder_name": "Marke

User: "I need to update cell A1 in 'Budget Sheet' with 'Q1 Revenue'."
AI: `{"tool_name": "sheets_write_data", "parameters": { "spreadsheet_name": "Bu

User: "I want to book a flight."
AI: "I can help you manage your Google Workspace, but I cannot book flights. Is
```

This core prompt establishes the AI's identity, defines its operational boundaries, and provides clear examples of expected input and output formats, which is crucial for Claude to understand its role and how to interact with the external tools. The persona traits guide the AI's conversational style, making it more user-friendly and efficient. The explicit mention of available tools and constraints helps prevent hallucinations and ensures the AI stays within its designated capabilities.

3. Tool-Calling Protocols for Google Workspace APIs

This section details the specific tools available to the AI assistant for interacting with Google Workspace. For each tool, a description of its functionality and the expected JSON schema for its parameters are provided. Claude MUST adhere strictly to these schemas when generating tool calls.

3.1 Gmail Tools

gmail_send_email

- **Description**: Sends an email to one or more recipients.
- **Parameters Schema**:

  JSON

  ```json
  {
    "type": "object",
    "properties": {
      "to": {"type": "string", "description": "Comma-separated list of recipie
      "subject": {"type": "string", "description": "The subject line of the em
      "body": {"type": "string", "description": "The main content of the email
      "cc": {"type": "string", "description": "Optional: Comma-separated list
      "bcc": {"type": "string", "description": "Optional: Comma-separated list
      "attachments": {"type": "array", "items": {"type": "string"}, "descripti
    },
  ```

```
  "required": ["to", "subject", "body"]
}
```

- **Example User Input**: "Send an email to alice@example.com and bob@example.com with the subject 'Meeting Agenda' and body 'Please review the attached agenda before our meeting.'"
- **Example Tool Call**: `{"tool_name": "gmail_send_email", "parameters": { "to": "alice@example.com,bob@example.com", "subject": "Meeting Agenda", "body": "Please review the attached agenda before our meeting." }}`

`gmail_read_email`

- **Description**: Reads the content of a specific email by its ID.
- **Parameters Schema**:

JSON

```json
{
  "type": "object",
  "properties": {
    "email_id": {"type": "string", "description": "The unique ID of the emai
  },
  "required": ["email_id"]
}
```

- **Example User Input**: "Read the email with ID '123abc456def'."
- **Example Tool Call**: `{"tool_name": "gmail_read_email", "parameters": { "email_id": "123abc456def" }}`

`gmail_search_email`

- **Description**: Searches for emails based on various criteria.
- **Parameters Schema**:

JSON

```json
{
  "type": "object",
  "properties": {
    "query": {"type": "string", "description": "The search query (e.g., 'fro
    "max_results": {"type": "integer", "description": "Optional: Maximum num
  },
  "required": ["query"]
}
```

- **Example User Input**: "Find emails from 'project_manager@example.com' about 'Q4 results'."
- **Example Tool Call**: `{"tool_name": "gmail_search_email", "parameters": { "query": "from:project_manager@example.com subject:\"Q4 results\"" }}`

3.2 Google Calendar Tools

`calendar_create_event`

- **Description**: Creates a new event on the user's Google Calendar.
- **Parameters Schema**:

JSON

```
{
  "type": "object",
  "properties": {
    "summary": {"type": "string", "description": "The title of the event."},
    "start_time": {"type": "string", "format": "date-time", "description": "
    "end_time": {"type": "string", "format": "date-time", "description": "En
    "description": {"type": "string", "description": "Optional: Description
    "attendees": {"type": "array", "items": {"type": "string"}, "description
    "location": {"type": "string", "description": "Optional: Location of the
  },
  "required": ["summary", "start_time", "end_time"]
}
```

- **Example User Input**: "Schedule a team sync meeting tomorrow at 10 AM for 1 hour. Invite alice@example.com and bob@example.com. Title it 'Team Sync'."
- **Example Tool Call**: `{"tool_name": "calendar_create_event", "parameters": { "summary": "Team Sync", "start_time": "2026-02-10T10:00:00-05:00", "end_time": "2026-02-10T11:00:00-05:00", "attendees": ["alice@example.com", "bob@example.com"] }}`

`calendar_list_events`

- **Description**: Lists events from the user's Google Calendar.
- **Parameters Schema**:

JSON

```
{
  "type": "object",
  "properties": {
    "time_min": {"type": "string", "format": "date-time", "description": "Op
    "time_max": {"type": "string", "format": "date-time", "description": "Op
    "max_results": {"type": "integer", "description": "Optional: Maximum num
    "query": {"type": "string", "description": "Optional: Text search agains
  }
}
```

- **Example User Input**: "What are my meetings for today?"
- **Example Tool Call**: `{"tool_name": "calendar_list_events", "parameters": { "time_min": "2026-02-09T00:00:00-05:00", "time_max": "2026-02-09T23:59:59-05:00" }}` (assuming current date is Feb 9, 2026)

3.3 Google Drive Tools

`drive_search_files`

- **Description**: Searches for files and folders in Google Drive.
- **Parameters Schema**:

```JSON
{
  "type": "object",
  "properties": {
    "query": {"type": "string", "description": "The search query (e.g., 'nam
    "file_type": {"type": "string", "description": "Optional: Filter by comm
    "max_results": {"type": "integer", "description": "Optional: Maximum num
  },
  "required": ["query"]
}
```

- **Example User Input**: "Find all PDF documents related to 'marketing campaign'."
- **Example Tool Call**: `{"tool_name": "drive_search_files", "parameters": { "query": "name contains \'marketing campaign\' and mimeType = \'application/pdf\'" }}`

`drive_create_folder`

- **Description**: Creates a new folder in Google Drive.
- **Parameters Schema**:

```JSON
{
  "type": "object",
  "properties": {
    "folder_name": {"type": "string", "description": "The name of the new fo
    "parent_id": {"type": "string", "description": "Optional: The ID of the 
  },
  "required": ["folder_name"]
}
```

- **Example User Input**: "Create a new folder called 'Client Projects' in my Drive."
- **Example Tool Call**: `{"tool_name": "drive_create_folder", "parameters": { "folder_name": "Client Projects" }}`

3.4 Google Sheets Tools

`sheets_read_data`

- **Description**: Reads data from a specified range in a Google Sheet.
- **Parameters Schema**:

```JSON
{
  "type": "object",
  "properties": {
    "spreadsheet_id": {"type": "string", "description": "The ID of the sprea
    "range": {"type": "string", "description": "The A1 notation or R1C1 nota
  },
  "required": ["spreadsheet_id", "range"]
}
```

- **Example User Input**: "Read data from 'Budget Tracker' spreadsheet, Sheet1, cells A1 to C5."
- **Example Tool Call**: `{"tool_name": "sheets_read_data", "parameters": { "spreadsheet_id": "<spreadsheet_id_here>", "range": "Sheet1!A1:C5" }}`

`sheets_write_data`

- **Description**: Writes data to a specified range in a Google Sheet.
- **Parameters Schema**:

```JSON
```

```
{
  "type": "object",
  "properties": {
    "spreadsheet_id": {"type": "string", "description": "The ID of the sprea
    "range": {"type": "string", "description": "The A1 notation or R1C1 nota
    "values": {"type": "array", "items": {"type": "array", "items": {"type":
    "value_input_option": {"type": "string", "enum": ["RAW", "USER_ENTERED"]
  },
  "required": ["spreadsheet_id", "range", "values"]
}
```

- **Example User Input**: "In 'Sales Report' spreadsheet, update cell B2 with '1500' and C2 with '200'."
- **Example Tool Call**: `{"tool_name": "sheets_write_data", "parameters": { "spreadsheet_id": "<spreadsheet_id_here>", "range": "B2", "values": [["1500"], ["200"]] }}`

These detailed tool definitions, including their schemas and examples, will guide Claude in generating precise and executable tool calls, forming the backbone of the AI assistant's functionality.

4. Context and Memory Management Strategies

Effective context and memory management are crucial for the AI assistant to provide coherent, personalized, and efficient interactions. This section outlines how Claude will handle different types of memory to maintain conversational flow and recall relevant information.

4.1 Conversational Context (Short-Term Memory)

Claude will maintain a short-term memory of the immediate conversation history. This is typically achieved by including a limited number of previous user turns and AI responses in the prompt for each new query.

- **Strategy**: For each new user message, append the last `N` turns of the conversation (e.g., 5-10 turns) to the prompt. This allows Claude to understand follow-up questions and references to previous statements.
- **Implementation**: The application layer will manage the conversation history and construct the prompt dynamically.
- **Example**: If a user asks "What's on my calendar today?" and then "How about tomorrow?", Claude should understand "tomorrow" in the context of listing calendar events.

4.2 Working Memory (Task-Specific Context)

Working memory is used to store information relevant to an ongoing task or a multi-step process. This prevents the need for users to repeat information and helps the AI complete complex requests.

- **Strategy**: When a multi-step task is initiated (e.g., creating a complex calendar event, drafting an email that requires multiple inputs), the AI will explicitly store relevant parameters and state information.
- **Implementation**: This can be managed within the application layer, potentially as a JSON object associated with the current conversation session. Claude can be prompted to output updates to this working memory.
- **Example**: If a user says "Draft an email," and then in subsequent messages provides the recipient, subject, and body, these pieces of information are held in working memory until the `gmail_send_email` tool can be fully constructed and called.

4.3 Long-Term Memory (User Preferences and Historical Data)

Long-term memory stores persistent information such as user preferences, frequently used contacts, common meeting locations, and historical interactions. This enables personalization and reduces repetitive inputs.

- **Strategy**: Implement a persistent storage mechanism (e.g., a simple database or JSON file) to store user profiles. Claude can be prompted to retrieve and update this information.
- **Implementation**: The application layer will query this storage based on user ID. Claude can be given a `retrieve_user_preference` or `update_user_preference` tool, or the relevant information can be injected into the prompt.
- **Example**: If a user frequently schedules meetings with "Alice" at "Conference Room A", the AI can learn these preferences. When the user says "Schedule a meeting with Alice," the AI might proactively suggest "at Conference Room A."

4.4 Retrieval-Augmented Generation (RAG) for Context

RAG can be employed to dynamically retrieve relevant information from both working and long-term memory to augment Claude's understanding and response generation, especially for complex queries or when external knowledge is required.

- **Strategy**: Before invoking Claude, relevant snippets from long-term memory (e.g., user preferences, past successful tool calls for similar requests) or working memory (for ongoing tasks) are retrieved and inserted into the prompt.

- **Implementation**: This involves a retrieval component that searches the memory stores based on the current user query and conversation history. The retrieved information is then prepended or inserted into a specific section of the prompt.
- **Example**: If a user asks "What was the subject of the last email I sent to Bob about Project X?", the RAG system would search long-term memory for past `gmail_send_email` tool calls involving "Bob" and "Project X" and provide that context to Claude, allowing it to formulate an accurate answer without directly calling the `gmail_search_email` tool if the information is already known.

By systematically managing these layers of memory, the WhatsApp AI assistant can offer a highly intelligent, efficient, and personalized experience for controlling Google Workspace.

5. References

[1] OpenClaw GitHub Repository:
[2] OpenClaw WhatsApp Integration Guide:
[3] Google Developers: Using OAuth 2.0 to Access Google APIs -
[4] Google Cloud Console:

3.1.1 Advanced Gmail Tools

`gmail_create_draft`

- **Description**: Creates a draft email.
- **Parameters Schema**:

JSON

```json
{
  "type": "object",
  "properties": {
    "to": {"type": "string", "description": "Comma-separated list of recipie
    "subject": {"type": "string", "description": "The subject line of the em
    "body": {"type": "string", "description": "The main content of the email
    "cc": {"type": "string", "description": "Optional: Comma-separated list 
    "bcc": {"type": "string", "description": "Optional: Comma-separated list
  },
  "required": ["to", "subject", "body"]
}
```

- **Example User Input**: "Draft an email to charlie@example.com with subject 'Project X Update' and body 'The project is progressing well.'"

- **Example Tool Call**: `{"tool_name": "gmail_create_draft", "parameters": { "to": "charlie@example.com", "subject": "Project X Update", "body": "The project is progressing well." }}`

`gmail_delete_email`

- **Description**: Deletes an email by its ID.
- **Parameters Schema**:

  JSON
  ```
  {
    "type": "object",
    "properties": {
      "email_id": {"type": "string", "description": "The unique ID of the emai
    },
    "required": ["email_id"]
  }
  ```
- **Example User Input**: "Delete the email with ID '789ghi012jkl'."
- **Example Tool Call**: `{"tool_name": "gmail_delete_email", "parameters": { "email_id": "789ghi012jkl" }}`

`gmail_add_label`

- **Description**: Adds a label to an email.
- **Parameters Schema**:

  JSON
  ```
  {
    "type": "object",
    "properties": {
      "email_id": {"type": "string", "description": "The unique ID of the emai
      "label_name": {"type": "string", "description": "The name of the label t
    },
    "required": ["email_id", "label_name"]
  }
  ```
- **Example User Input**: "Add the label 'Important' to email ID '345mno678pqr'."
- **Example Tool Call**: `{"tool_name": "gmail_add_label", "parameters": { "email_id": "345mno678pqr", "label_name": "Important" }}`

3.2.1 Advanced Google Calendar Tools

`calendar_update_event`

- **Description**: Updates an existing event on the user's Google Calendar.
- **Parameters Schema**:

JSON

```json
{
  "type": "object",
  "properties": {
    "event_id": {"type": "string", "description": "The unique ID of the even
    "summary": {"type": "string", "description": "Optional: The new title of
    "start_time": {"type": "string", "format": "date-time", "description": "
    "end_time": {"type": "string", "format": "date-time", "description": "Op
    "description": {"type": "string", "description": "Optional: New descript
    "attendees": {"type": "array", "items": {"type": "string"}, "description
    "location": {"type": "string", "description": "Optional: New location of
  },
  "required": ["event_id"]
}
```

- **Example User Input**: "Change the 'Team Sync' meeting (ID: xyz123) to start at 11 AM instead of 10 AM."
- **Example Tool Call**: `{"tool_name": "calendar_update_event", "parameters": { "event_id": "xyz123", "start_time": "2026-02-10T11:00:00-05:00" }}`

`calendar_delete_event`

- **Description**: Deletes an event from the user's Google Calendar.
- **Parameters Schema**:

JSON

```json
{
  "type": "object",
  "properties": {
    "event_id": {"type": "string", "description": "The unique ID of the even
  },
  "required": ["event_id"]
}
```

- **Example User Input**: "Cancel the meeting with ID 'abc456'."
- **Example Tool Call**: `{"tool_name": "calendar_delete_event", "parameters": { "event_id": "abc456" }}`

3.3.1 Advanced Google Drive Tools

drive_copy_file

- **Description**: Copies an existing file in Google Drive.
- **Parameters Schema**:

```json
{
  "type": "object",
  "properties": {
    "file_id": {"type": "string", "description": "The ID of the file to copy
    "new_name": {"type": "string", "description": "Optional: The new name fo
    "parent_id": {"type": "string", "description": "Optional: The ID of the
  },
  "required": ["file_id"]
}
```

- **Example User Input**: "Make a copy of the document with ID 'file123' and name it 'Project Proposal - Draft 2'."
- **Example Tool Call**: `{"tool_name": "drive_copy_file", "parameters": { "file_id": "file123", "new_name": "Project Proposal - Draft 2" }}`

drive_move_file

- **Description**: Moves a file to a different folder in Google Drive.
- **Parameters Schema**:

```json
{
  "type": "object",
  "properties": {
    "file_id": {"type": "string", "description": "The ID of the file to move
    "destination_folder_id": {"type": "string", "description": "The ID of th
  },
  "required": ["file_id", "destination_folder_id"]
}
```

- **Example User Input**: "Move the file 'report_final.docx' (ID: file456) to the 'Archived Reports' folder (ID: folder789)."

- **Example Tool Call**: `{"tool_name": "drive_move_file", "parameters": { "file_id": "file456", "destination_folder_id": "folder789" }}`

`drive_delete_file`

- **Description**: Deletes a file or folder from Google Drive.
- **Parameters Schema**:

JSON

```
{
  "type": "object",
  "properties": {
    "file_id": {"type": "string", "description": "The ID of the file or fold
  },
  "required": ["file_id"]
}
```

- **Example User Input**: "Delete the old presentation file with ID 'file000'."
- **Example Tool Call**: `{"tool_name": "drive_delete_file", "parameters": { "file_id": "file000" }}`

3.4.1 Advanced Google Sheets Tools

`sheets_update_cell`

- **Description**: Updates a single cell in a Google Sheet.
- **Parameters Schema**:

JSON

```
{
  "type": "object",
  "properties": {
    "spreadsheet_id": {"type": "string", "description": "The ID of the sprea
    "sheet_name": {"type": "string", "description": "The name of the sheet (
    "cell_address": {"type": "string", "description": "The A1 notation of th
    "value": {"type": "string", "description": "The value to write to the ce
  },
  "required": ["spreadsheet_id", "sheet_name", "cell_address", "value"]
}
```

- **Example User Input**: "In 'Monthly Budget' spreadsheet, update cell B5 on 'Expenses' sheet to '500'."

- **Example Tool Call**: `{"tool_name": "sheets_update_cell", "parameters": { "spreadsheet_id": "<spreadsheet_id_here>", "sheet_name": "Expenses", "cell_address": "B5", "value": "500" }}`

`sheets_add_row`

- **Description**: Adds a new row of data to a Google Sheet.
- **Parameters Schema**:

  JSON

  ```
  {
    "type": "object",
    "properties": {
      "spreadsheet_id": {"type": "string", "description": "The ID of the sprea
      "sheet_name": {"type": "string", "description": "The name of the sheet (
      "values": {"type": "array", "items": {"type": "string"}, "description":
    },
    "required": ["spreadsheet_id", "sheet_name", "values"]
  }
  ```

- **Example User Input**: "Add a new row to 'Sales Data' sheet with values 'Product A', '100', '2026-02-09'."
- **Example Tool Call**: `{"tool_name": "sheets_add_row", "parameters": { "spreadsheet_id": "<spreadsheet_id_here>", "sheet_name": "Sales Data", "values": ["Product A", "100", "2026-02-09"] }}`