# Google Ads KPI MCP Server Prototype - Implementation Plan (Enhanced for Artifacts)  
  
This document outlines the phased implementation plan for the Google Ads KPI MCP server prototype, with a specific focus on enabling business data visualizations within Claude Artifacts.  
  
## General Notes:  
  
\* This plan assumes a single developer working on the prototype.  
\* Each phase is estimated to take approximately 3-5 days, but this is subject to change based on complexity and unforeseen issues.  
\* Testing and documentation should be ongoing throughout the project.  
\* The goal is to deliver a functional prototype that demonstrates the core data retrieval and serving capabilities, with a clear path towards supporting visualizations within Claude Artifacts.  
  
## Phases:  
  
### Phase 1: Project Setup and Dependencies  
  
\* \*\*Objective:\*\* Set up the development environment and install necessary dependencies.  
\* \*\*Tasks:\*\*  
 \* Create a project directory and initialize a Git repository.  
 \* Set up a Python 3.10.x virtual environment (using `venv` or `conda`).  
 \* Install the Flask framework (version 3.0.x), the Google Ads Python Client Library SDK (version 27.0.0 or the latest stable version), and the Python MCP Server SDK (version 0.4.0 or the latest stable version) using `pip`.  
 \* Create a basic `requirements.txt` file to track dependencies.  
 \* \*\*Create a basic project structure with the following files/directories (as outlined in the Tech Stack Document, Section 5):\*\*  
 \* `src/`: Main source code directory.  
 \* `google\_ads\_client.py`: Module for interacting with the Google Ads API.  
 \* `mcp\_server.py`: Module for implementing the MCP server.  
 \* `utils.py`: (Optional) Utility functions.  
 \* `tests/`: Unit tests (initially empty).  
 \* `README.md`: Project documentation (initially minimal).  
\* \*\*Deliverables:\*\*  
 \* A project directory with a virtual environment and installed dependencies.  
 \* A basic project structure with the core files created.  
 \* A `requirements.txt` file.  
  
### Phase 2: Google Ads API Client Implementation  
  
\* \*\*Objective:\*\* Implement the core functionality for retrieving Google Ads data using the Google Ads API.  
\* \*\*Tasks:\*\*  
 \* Implement authentication with the Google Ads API using the provided Developer Token (store securely as an environment variable).  
 \* Create functions within `google\_ads\_client.py` to retrieve account-wide cost and conversion KPIs for a specified date range.  
 \* Implement basic error handling to catch potential Google Ads API exceptions.  
\* \*\*Deliverables:\*\*  
 \* A functional `google\_ads\_client.py` module capable of retrieving basic Google Ads KPI data.  
  
### Phase 3: Segmentation Logic  
  
\* \*\*Objective:\*\* Implement the logic for segmenting the retrieved Google Ads data by campaign type, name, and label.  
\* \*\*Tasks:\*\*  
 \* Extend the functions in `google\_ads\_client.py` to support segmentation of the KPI data by `campaign\_type`, `campaign\_name`, and `campaign\_label`.  
 \* Ensure proper handling of "AND" logic across dimensions and "OR" logic within dimensions, as specified in PRD Section 4.1.3.  
\* \*\*Deliverables:\*\*  
 \* Enhanced `google\_ads\_client.py` module that supports data segmentation.  
  
### Phase 4: Comparison Logic  
  
\* \*\*Objective:\*\* Implement the logic for comparing KPI data over two specified date ranges.  
\* \*\*Tasks:\*\*  
 \* Extend the functions in `google\_ads\_client.py` to allow for period-over-period comparison of the KPI data.  
 \* Implement the calculation of absolute and percentage change, as specified in PRD Section 4.4.1.  
 \* Handle cases where data might be missing for one or both periods, as well as invalid date ranges.  
\* \*\*Deliverables:\*\*  
 \* Enhanced `google\_ads\_client.py` module that supports data comparison.  
  
### Phase 5: MCP Server Implementation  
  
\* \*\*Objective:\*\* Implement the MCP server interface using the Python MCP SDK to serve the retrieved Google Ads data.  
\* \*\*Tasks:\*\*  
 \* Implement the MCP server using Flask in `mcp\_server.py`.  
 \* Receive MCP requests from Claude Desktop.  
 \* Call the appropriate functions in `google\_ads\_client.py` to retrieve the requested data.  
 \* Format the data into the MCP response payload according to the examples provided in PRD Section 4.4.1.  
 \* \*\*Crucially, begin to structure the data in a way that is amenable to visualization. While the initial prototype might not directly render visualizations, consider the data structures that would be needed by common charting libraries (e.g., arrays of objects with key-value pairs for labels and values).\*\*  
\* \*\*Deliverables:\*\*  
 \* A functional `mcp\_server.py` module that implements the MCP server and serves Google Ads KPI data.  
 \* \*\*Data formatted in a structure that is suitable for potential visualization (even if not directly visualized in this phase).\*\*  
  
### Phase 6: Testing  
  
\* \*\*Objective:\*\* Implement basic unit tests and perform manual testing to ensure the functionality works as expected.  
\* \*\*Tasks:\*\*  
 \* Implement basic unit tests in `/tests/` to verify the core functionality of the `google\_ads\_client.py` and `mcp\_server.py` modules.  
 \* Send example MCP requests (as per PRD Section 4.1.3 and 4.4.1) to the running server to validate the responses.  
 \* \*\*Specifically, test the structure and content of the data being served to ensure it is suitable for potential visualization. This might involve manually inspecting the JSON output and comparing it to the expected input formats of common charting libraries.\*\*  
\* \*\*Deliverables:\*\*  
 \* A set of basic unit tests.  
 \* Validated MCP server responses.  
 \* \*\*Confirmation that the data being served is structured in a way that could be used for visualization.\*\*  
  
### Phase 7: Documentation  
  
\* \*\*Objective:\*\* Create basic documentation for the prototype.  
\* \*\*Tasks:\*\*  
 \* Update the `README.md` file with instructions on setting up and running the server.  
 \* Add comments to the code where necessary to explain complex logic or important implementation details.  
 \* \*\*Include a section in the `README.md` describing the data format being served by the MCP server and how it is intended to support visualization. Provide examples of the JSON structure and how it could be used with common charting libraries.\*\*  
\* \*\*Deliverables:\*\*  
 \* A documented codebase.  
 \* A `README.md` file with setup and usage instructions, \*\*including details on the data format for visualization.\*\*  
  
### Phase 8: Initial Integration with Claude (Limited)  
  
\* \*\*Objective:\*\* Attempt a limited integration with Claude to demonstrate the potential for visualization. This phase might involve manual steps or require coordination with the Claude team.  
\* \*\*Tasks:\*\*  
 \* \*\*If possible, work with the Claude team to test the data being served by the MCP server with the Claude Artifacts framework. This might involve providing example data and working with the Claude team to render basic charts.\*\*  
 \* \*\*If direct integration is not possible, focus on validating that the data format being served is compatible with the expected input formats of common charting libraries (e.g., Recharts, Chart.js) within Claude Artifacts. This might involve creating mock visualizations using these libraries based on the data served by the MCP server.\*\*  
\* \*\*Deliverables:\*\*  
 \* \*\*Ideally, a demonstration of basic visualization rendering within Claude Artifacts using the data served by the prototype.\*\*  
 \* \*\*At a minimum, validation that the data format being served is compatible with common charting libraries within Claude Artifacts.\*\*  
  
### Future Phases (Beyond Prototype):  
  
\* \*\*Enhance Data for Visualization:\*\* Refine the data format to be directly compatible with specific visualization libraries within Claude Artifacts. Include metadata about data types and suggested chart types.  
\* \*\*Support User-Driven Visualization Configuration:\*\* Enable the server to interpret natural language commands from Claude related to visualization preferences.  
\* \*\*Expand KPI Support:\*\* Increase the range of available Google Ads KPIs.  
\* \*\*Implement Robust Security and Scalability Measures:\*\* Prepare the server for production deployment.  
\* \*\*Implement Comprehensive Monitoring and Alerting:\*\* Ensure the health and performance of the production system.  
  
This enhanced Implementation Plan provides a roadmap for the Google Ads KPI MCP server prototype, explicitly incorporating the goal of enabling business data visualizations within Claude Artifacts. It outlines the necessary steps for the initial prototype while keeping the future vision of interactive and user-driven data visualization in mind.