**Project Overview**

The main goal is to build a full-fledged **customer service agent** using Google's Agent Development Kit (ADK). This agent will serve as a base for demonstrating various features, with plans to later migrate the same use case to other frameworks like CrewAI and LangChain.

**Agent Configuration & Prompting**

* **Primary Instruction Prompt:** A detailed prompt defines the agent's **persona**, its core capabilities, the tools it has access to, and constraints on its behavior.
* **Global Instruction:** A separate global\_instruction is used to provide context that applies to **all agents** (the root agent and any sub-agents). This is useful for setting a common personality or injecting session-specific data.
* **Use Case for Global Instruction:** The customer's profile data (as a JSON object) is loaded into the global\_instruction. This ensures every agent in the hierarchy has immediate access to the current customer's information without needing to fetch it repeatedly.
  + global\_instruction = f"The profile of the current customer is {customer.get\_customer('123').to\_json()}"

**Data Modeling with Pydantic**

* **Entities:** The data model for the application domain (e.g., Customer, Address, Product, Purchase) is defined using **Pydantic** models. These classes act as the schema for all data.
* **Structure:** The models are organized in an entities directory. Complex relationships are established by nesting Pydantic models, such as a Customer model containing a list of Purchase models.
* **Test Data:** For development and testing, a single customer object is **hard-coded** with sample data. This avoids the complexity of setting up a database (like SQLite) initially, allowing the focus to remain on the agent's logic.

**Agent Architecture: Root and Sub-Agents**

* **Hierarchical Structure:** The architecture supports a tree-like structure with a **root agent** and multiple **sub-agents**.
* **Delegation of Tasks:** The root agent acts as a manager or **supervisor**. It receives the user's goal and delegates specific tasks to specialized sub-agents (e.g., a product manager agent, a GCP expert agent). This is an effective way to break down complex problems.
* **Shared Context:** The global\_instruction defined at the root level is automatically inherited by all sub-agents, providing a shared context for the entire team.

**Tool Development**

* **Purpose:** Tools are standard Python functions that allow the agent to interact with external systems or perform specific, deterministic actions.
* **Organization:** Tools are organized in a dedicated tools package for better code structure.
* **LLM Interaction:** The agent's LLM uses the **function signature and its docstring** to understand what a tool does, what parameters it requires, and when to use it. Therefore, writing clear and descriptive docstrings is critical.
* **Implemented Tools:**

The agent is equipped with a diverse set of tools, allowing it to handle a wide range of customer needs from initial contact to post-purchase support. These tools are Python functions that connect the agent to external systems and enable it to perform specific actions. They can be grouped by functionality:

**🛒 Sales & Discount Management**

These tools empower the agent to handle sales-related queries and promotions.

* **approve\_discount**: Approves discounts directly if they fall within pre-defined limits.
* **sync\_ask\_for\_approval**: Escalates a discount request to a human manager for approval when it exceeds the agent's authority.
* **generate\_qr\_code**: Creates a unique QR code for a customer to redeem a discount.

**🛍️ Cart & Product Information**

This set of tools allows the agent to manage the customer's shopping experience directly.

* **access\_cart\_information**: Retrieves and reviews the contents of the customer's current shopping cart.
* **modify\_cart**: Adds or removes items from the cart. The agent knows to check the cart's contents using access\_cart\_information *before* making changes.
* **get\_product\_recommendations**: Provides personalized product suggestions based on a customer's interests (e.g., suggesting specific fertilizers for petunias).
* **check\_product\_availability**: Checks the current stock level for a specific product.

**🤝 Customer Interaction & Support**

These tools facilitate direct communication and provide value-added information to the customer.

* **send\_call\_companion\_link**: Initiates a live video stream with the customer, perfect for visual diagnostics of plant issues or garden planning.
* **send\_care\_instructions**: Sends detailed plant care information to the customer.

**📅 Service Scheduling**

The agent can book and manage value-added services for the customer.

* **get\_available\_planting\_times**: Fetches a list of open time slots for the planting service.
* **schedule\_planting\_service**: Books a planting service appointment for the customer.

**⚙️ Backend System Integration**

This is a critical tool for maintaining data integrity with other business systems.

* **update\_salesforce\_crm**: Updates the customer's record in the Salesforce CRM system, ensuring all interactions and purchases are logged correctly.

--------------------------------------------------------------------------------------------------------------------------------------

**Purpose of Multiple Sample Purchases 🧐**

* **Question:** Why create multiple sample Purchase objects instead of testing with just one?
* **Answer:** Using a richer dataset with multiple purchases allows for more realistic and comprehensive testing. It enables developers to test a wider variety of prompts and scenarios, such as asking the agent to list all items from a user's purchase history, which would be trivial with only one item.

**Root Agent and Sub-Agent Relationship 👨‍👩‍👧‍👦**

* **Question:** How are sub-agents related to the root agent? Is it a form of class inheritance?
* **Answer:** The relationship is not class inheritance but is best understood as a **supervisor/worker pattern**.
  + The **root agent** acts as a manager or team lead that receives the primary goal from the user.
  + It then **delegates** specific, smaller tasks to specialized **sub-agents** (the workers). This creates a team of agents where each one is an expert in a specific function, allowing the system to solve complex problems in a modular way. The global\_instruction serves as a briefing that is shared with the entire team.

**Discount Approval Process ✅**

* **Question:** Is the discount approval process automatic or does it require a human?
* **Answer:** The system is designed for **both**.
  + The approve\_discount tool is fully **automatic**. It contains business logic to approve or deny requests based on predefined limits (e.g., a discount amount less than or equal to 10).
  + For requests that fall outside these automatic limits, the sync\_ask\_for\_approval tool is used to **escalate the request to a human manager**, creating a human-in-the-loop workflow.