**HSN Codes Validation Agent development using Google-ADK (python)**

**1. Agent Design & Architecture (High-Level)**

The HSN Validation Agent is designed as a specialized conversational AI system. Its core architecture leverages **Google's** **Agent Development Kit (ADK)** to build and manage the conversational flow, integrating with Google's **Gemini Generative AI model** for understanding user intent and executing custom functions (tools).

* **Role:** Its primary purpose is to receive user requests related to HSN codes, such as validating specific codes or finding codes based on descriptions, and provide accurate information.
* **Conversational Flow:** The agent maintains a conversation history (though currently impacted by a known ADK bug) to understand context.
* **Tool-Use Paradigm:** It operates on a tool-use paradigm, meaning the Gemini LLM isn't just generating text; it's empowered to "call" specific Python functions (the HSN validation and lookup tools) to perform tasks that require external data.
* **Framework:** The **ADK** provides the necessary structure for defining agents, managing sessions, and integrating with web frameworks (FastAPI).

**2. Agent Components**

The agent is built from several interacting components:

* + ***HSN\_Master\_Data.csv:*** The CSV dataset containing all valid HSN codes and their corresponding descriptions. (Should be kept locally within the project directory.)
  + ***HSN Master Data Loader* (**load\_hsn\_data **function):**  Reads the HSN\_Master\_Data.csv file into a Pandas DataFrame.
  + **Tool Functions:** These are standard Python functions exposed to the Gemini model via the ADK.
    - *validate\_hsn\_code(hsn\_code: str):* Takes a single HSN code string as input and performs validation checks (length, existence in master data) and returns a structured dictionary indicating status, the HSN code, and its description or an error message.
    - *validate\_multiple\_hsn\_codes(hsn\_codes: list[str]):* Takes a list of HSN code strings as input and iterates through the list, calling validate\_hsn\_code for each, and returns a list of individual validation results.
    - *lookup\_by\_description(search\_text: str)*: Takes a descriptive search term as input and searches the HSN master data for descriptions containing the search\_text (case-insensitive) and returns a list of matching "HSNCode: Description" strings.
  + ***HSNValidationAgent* Class:** The custom agent class, inheriting from google.adk.agent.Agent. It defines the agent's unique behaviour. It contains an instance of GeminiAgent, which is the core for interacting with the Gemini LLM.
  + ***\_handle\_message* Method:** This is an asynchronous method which is the entry point for processing incoming user messages (Event objects). It delegates the actual AI processing and tool execution to \_gemini\_agent.handle\_event().
  + ***GeminiAgent* (from *google.adk.agents*):** A pre-built ADK component that wraps the interaction with Google's Generative AI models. It handles the input conversions, JSON requests generation, interpreting Gemini’s responses and feeding the output.
  + ***AgentRouter (*from *google.adk.integrations.fastapi):*** Connects the main python “HSNValidationAgent” to the FastAPI web framework. It sets up the HTTP endpoints, handles incoming HTTP requests, extracts messages and manages session state via a SessionService.
  + ***InMemorySessionService* (from *google.adk.sessions*):** This component is intended to store and retrieve conversation history for each unique session ID. It keeps session data in memory.
    - **Known Issue:** In the current version of google agent development kit “google-adk==1.0.0”, this specific service has a bug causing the ValueError: Session not found error, preventing consistent session management.
  + **FastAPI (**app**) & Uvicorn:**
    - **FastAPI** is the web framework that provides the HTTP server capabilities. It receives incoming API calls (e.g., from curl) and then the *AgentRouter* handles routing these calls to the custom agent.
    - **Unicorn** is an ASGI (Asynchronous Server Gateway Interface) server used to run the FastAPI application.

**3. Input Validation Logic**

The agent employs a two-tiered approach to input validation:

1. **Implicit LLM-Driven Validation (by Gemini):**
   * When a user sends a message (e.g., "Validate HSN 123"), Gemini first processes this natural language input.
   * For example, if the user asks "What is the weather?", Gemini will realize no tool matches and generate a text response stating it's an HSN agent. If the user says "Validate HSN code ABC", Gemini might attempt to call validate\_hsn\_code and pass "ABC" as the hsn\_code argument. This should gives responses that it is an invalid HSN Code.
2. **Explicit Tool-Level Validation (within the Python functions):**
   * ***validate\_hsn\_code's* Logic:**
     + **Data Availability Check:** if HSN\_MASTER\_DATA.empty: ensures the HSN data was loaded correctly.
     + **Type & Format Check:** hsn\_code = str(hsn\_code).strip() ensures the input is treated as a string and whitespace is removed.
     + **Length Validation:** if not 2 <= len(hsn\_code) <= 8: is a crucial business rule. It checks if the HSN code has a valid number of digits (between 2 and 8). If not, it returns an "invalid" status with a clear message.
     + **Existence Validation:** if hsn\_code in HSN\_MASTER\_DATA.index: checks if the provided HSN code actually exists in the loaded master dataset. If not found, it returns an "invalid" status with a clear message.
   * ***validate\_multiple\_hsn\_codes*:** Delegates to validate\_hsn\_code, applying the same explicit validation to each code in the list.
   * ***lookup\_by\_description*:** Primarily performs text searching and checks for data availability. If exists, then code and description are returned.

**4. Validation Output**

The process of providing output is multi-step:

1. **Tool Execution and Return:** When the Gemini LLM decides to call a tool (e.g., validate\_hsn\_code), the Python function executes and returns a structured output (a Python dictionary -- {"status": "valid", "hsn\_code": "01011010", "description": "Live horses, asses, mules and hinnies"}).
2. **Tool Output to LLM:** The google-adk framework (GeminiAgent) takes this Python dictionary, converts it into a ToolOutput object and sends it back to the Gemini LLM.
3. **LLM's Interpretation and Response Generation:** Gemini receives the ToolOutput and "reads" it. It then uses its natural language generation capabilities to interpret this structured data and formulate a human-readable response that explains the validation result. For example, if the tool returned {"status": "valid", ...}, Gemini might respond: "HSN code 01011010 is valid and corresponds to 'Live horses, asses, mules and hinnies'." If it returned {"status": "invalid", "message": "HSN code not found..."}, Gemini would craft a response like "HSN code 99999999 is invalid because it was not found in the master data."
4. **Agent Output to Client:** The GeminiAgent yields AgentOutput events containing this final natural language response from the LLM. The AgentRouter then captures these events and sends them back as the HTTP response to the.

**5. Deployment Steps**

Please follow below steps to deploy this agent in the local development environments:

**1) Copy the code structure from GitHub:**

Link <<tbd>>

**2) Ensure Project Setup:** The project directory should contain the main python file “HSN\_ValidationAgent.py”, the master data “HSN\_Master\_Data.csv” and the requirements.txt to install the dependencies.

3) **Get API Key (Important):**

\*\* Subscribe and login, click ‘Get API Key’ -- <https://aistudio.google.com/app/apikey>

\*\* API Key should be linked to google cloud project (with billing enabled)

3) **Environment Setup:**

Go to the respective project folder in the command prompt/terminal.

* + **Create the virtual environment:** *python -m venv .venv*
  + **Activate it:** .venv\Scripts\activate (on Windows).
  + **Set API Key:** set GEMINI\_API\_KEY=User’s ACTUAL\_GEMINI\_API\_KEY



* + **Install Dependencies:** pip install -r requirements.txt
  + **Run the FastAPI Server:**
    1. In the activated terminal, execute the python file: python HSNValidatonAgent.py
    2. The Uvicorn should be starting the server (e.g., Uvicorn running on http://0.0.0.0:8080).

**4) Testing the Agent:**

* + Open a **new terminal window**.
  + Navigate to the respective project directory.
  + **Activate the same virtual environment** in this new terminal. (.venv\Scripts\activate)
  + **Set the** GEMINI\_API\_KEY **environment variable again** in this new terminal (as it's a new session).
  + Send requests using curl:

*curl -X POST -H "Content-Type: application/json" -d "{ \"text\": \"Validate HSN code 01011010\"}" http://127.0.0.1:8080/v1/projects/default/locations/default/agents/hsn\_validation\_agent/sessions/my\_test\_session/messages*

The generated output & response can be seen in both the curl terminal and the server terminal.

***Challenge Faced:***

The immediate and biggest challenge I face is the reported **ValueError: Session not found bug in google-adk==1.0.0's session service**, which impacts both standalone and FastAPI-based execution. This bug is already reported in github: <https://github.com/google/adk-samples/issues/117>

**Screenshot**:







