

Project Proposal

Project: Merging Agentic AI and E-Commerce DB for Jewelry Outfit Styling and Sales Uplift

1. What is the project idea/focus?

We want to build a system that pairs agentic AI with a traditional e-commerce database to (1) help customers style jewelry with their outfits and (2) improve jewelry sales conversion. The agent takes a user's photo of clothing and preference inputs (style, budget, event, size, etc.), consults the catalog/inventory DB, and recommends only items that are actually in stock.

2. Why did you select this project idea?

We would like to improve the buying experience and the decision-making process for customers, which should also help the jewelry seller reduce choice overload and increase conversion on in-stock items. This project aims for a win-win situation where the customers save time and money on unnecessary bad-decision making and for companies not only to sell their products in an efficient way but also collect more metrics on customer satisfaction and behavior for further product improvement.

3. How will you design and implement it?

Goal: We intend to pair a simple Agentic AI front with a clean e-commerce database so customers can describe or upload an outfit and get jewelry recommendations that are actually in stock, thereby improving their styling confidence and our sales.

System overview (two parts in one pipeline)

- **Part A – Agentic AI layer:** We plan to design a single web page where the user can upload a clothing photo and just type a few preferences (such as style keywords, budget range, event, etc.). The “agent” here is a small orchestrator: it collects inputs, optionally extracts a couple of simple tags from the photo (such as color, formality), and packages everything as a session request.
- **Part B – Database-backed recommendation:** The request above will go to a rule-based recommender that queries our e-commerce database. It filters by stock and budget, scores items by style-tag overlap and price fit, and returns top

picks with product images pulled from the database.

Data we will use:

- **Inventory Data:** This can be generated via Mockaroo.
- **Images Data:** Product images will be stored as URLs in the inventory.
- **Descriptions Data:** Short descriptive text, initially found a dataset with 80 sample size.

Schema framework:

Relational (MySQL/RDS)

- JewelryItem(item_id PK, name, category, material, color, style_tags, price, image_url)
- Inventory(item_id PK/FK, stock_qty, status)
- RecommendationLog(rec_id PK, session_id, item_id FK, rank, created_at)

Document (MongoDB Atlas)

- UserProfile: { user_id, preferences: { styles: [...], budget_min, budget_max, metal_pref, size }, created_at }
- OutfitSession: { session_id, user_id, clothing_features: { colors: [...], style_tags: [...] }, user_inputs: { event, budget, styles }, created_at }

Recommendation logic:

Hard filters: The item it recommends must be active, in stock, and within the user's budget. If a preferred material is provided, this also should require a match.

Score and rank:

- **Style fit:** count the overlap between the user's selected tags and the item's tags.
- **Price fit:** search within the prices near the middle of the user's budget range.
- **Final score:** use this equation to score each potential item: $\text{overlap} + 0.5 \times \text{price_fit}$.

Return top K (e.g., 6–12) items: provide image, name, price, and a short explanation (e.g., "This item matches 'minimalist' and within your budget").

Implementation plan

- **Step 1: Create database**
 - 1) Create tables and constraints in **MySQL or RDS**.
 - 2) Generate data with Mockaroo and a small, consistent tag vocabulary.
 - 3) Validate basic inventory queries: in-stock and under budget.
- **Step 2: Add recommender query**
 - 1) Implement the SQL for filter, score, and rank.
 - 2) Add RecommendationLog inserts.
- **Step 3: Build UI**
 - 1) A single page form: budget, multi-select style tags, and optional material.
 - 2) Display result cards with image URLs from the DB.
- **Step 4: Match image with tags**
 - 1) Turn the clothing photo into a couple of tags (e.g., dominant color, formal/casual).
- **Step 5: demo**
 - 1) Seed a few realistic sessions, show logs, and walk through the end-to-end flow.

4. Who is the solution for?

- **Primary:** A jewelry company (merchandising/commerce team).
- **End users:** Jewelry buyers using the storefront.

This solution has the potential to open the door of utilizing agentic AI for other type of fashion e-commerce products in the future.

5. When is it used? (use cases)

The Customers:

- A customer wants to buy jewelry and accessories to **match a specific outfit** that they have.
- A customer wants to plan **multiple outfits** (wardrobe planning) and see matching jewelry options within a budget.
- A customer has an upcoming **specific event** (e.g., a wedding, a job interview, a holiday party) and needs jewelry that fits the dress code, formality, and occasion.

- A customer is looking for jewelry that fits strict price and material requirements. The agent acts as a smart, inventory-aware filter.
- A customer is **exploring a new style or trend** and relies on the agent's style-tag filtering to discover suitable options quickly.
- A customer is replacing a basic staple piece that got lost or damaged and wants to find something **similar to their past preferences** (can be stored in the MongoDB profile) but only sees what's *currently* available in the inventory.

The Merchandising/Commerce Team (Internal Use Case) :

- The merchandising team uses the recommendation log and outfit session data to identify **unmet demand**. For example, many searches for "boho silver earrings" yield no results. They will also be able to explore which combinations of clothing/event tags lead to the highest conversion, informing future inventory buying decisions.
- The company is **identifying emerging fashion trends** before they translate into actual jewelry sales, allowing the company to proactively design or order new jewelry items that match the soon-to-be-popular clothing styles.
- The company wants to implement a **dynamic pricing strategy** where, for example, a versatile item frequently matched with high-value "formal event" outfits could have its price maintained, while an item only matched with low-budget "casual" inputs could be targeted for a strategic promotion to clear stock

6. Where is it used?

- The system can be embedded into a jewelry e-commerce website (and optionally mobile site).
- It is accessible globally (or anywhere the e-commerce retailer operates) as it is available through the public web store and mobile app.