Excellent — this is a **high-value question** because it blends *practical engineering*, *AI flair*, and *core system design*. Let’s craft a **Taylor Stitch–style web app roadmap** that’s not just an e-commerce clone, but a **portfolio centerpiece** showing you understand *AI systems*, *front-end craft*, and *low-level back-end performance*.

**🧭 GOAL**

Build a **“TaylorStitch-inspired experimental web platform”** that feels real but clearly labeled as *an educational/experimental project*.  
It should highlight:

* Front-end polish (React/Next.js)
* AI augmentation (recommendations, personalization)
* Systems/low-level depth (NodeJS + optional C++ module)
* DevOps deployment (AWS, scalability)

**🧱 STAGE 1 — Core Replica (Weeks 1–3)**

**Front-End**

* **Stack:** React + Next.js + TailwindCSS + TypeScript
* **Key Features:**
  + Product grid with filters, sorting, and image carousels
  + Product detail pages with smooth transitions (Framer Motion)
  + Responsive design and accessibility baked in
* **Wow Factor:** Lightning-fast transitions, smooth animations, clean design system

**Back-End**

* **Stack:** Node.js (Express or Fastify) + PostgreSQL (via Prisma ORM)
* **Features:**
  + Products, inventory, users, orders
  + JWT-based authentication
  + REST + GraphQL API hybrid
* **Wow Factor:** GraphQL introspection, live API playground

**Deployment**

* AWS EC2 or ECS + RDS + S3 (for assets)
* CloudFront CDN, CI/CD pipeline with GitHub Actions
* Clear README showing the architecture diagram

**⚙️ STAGE 2 — Low-Level / Systems Layer (Weeks 4–6)**

**1️⃣ Native Add-On (C++ Module)**

* **Use:** NodeJS Addons (via N-API or node-addon-api)
* **Purpose:**
  + Implement a **C++ recommendation engine** or **text similarity engine** for product tags/descriptions.
  + Example: Compute cosine similarity between user-viewed products and catalog items (fast, low latency).
* **Wow Factor:**
  + Show measurable speedup vs pure JS implementation.
  + Include benchmark logs in README.

**2️⃣ Event Logging / Multithreaded Simulation**

* **Purpose:** Demonstrate deterministic multithreading simulation for backend event logs (tying into your low-level systems interest).
* **Wow Factor:** Visualize thread events in an admin dashboard → shows systems comprehension.

**🤖 STAGE 3 — AI / ML Layer (Weeks 7–9)**

**AI Features**

1. **Personalized Recommendations**
   * **Model:** Classical collaborative filtering (Matrix Factorization / KNN) or a lightweight ML model (Scikit-learn, running server-side)
   * **Data:** User activity logs, product tags, purchase history (synthetic)
   * **Deployment:** Model served through a simple REST microservice
2. **Visual Similarity Search**
   * Use a pretrained **CLIP or MobileNet model** to extract embeddings from product images (run offline, store embeddings in DB).
   * Add a “Find Similar Items” button that retrieves visually similar products.
3. **AI Assistant (optional)**
   * Integrate an **AI shopping assistant** powered by a small model or external API (e.g., OpenAI or Gemini).
   * Handles natural language queries like “Show me rugged shirts under $100.”

**Wow Factor:** AI-driven personalization, image search, and natural language filtering — all integrated cleanly.

**💻 STAGE 4 — Front-End “Wow Layer” (Weeks 10–12)**

**Features**

* **AI-powered search bar** → as-you-type suggestions using the AI model
* **Product comparison dashboard** with animated diffing (React + Recharts)
* **Story section** → “How it was made” page visualizing your architecture (interactive flow chart with animations)
* **Performance metrics overlay** → show real-time server response latency, model inference time, etc.

**Wow Factor**

Employers can *see* and *feel* the sophistication of your stack, not just read about it.

**☁️ STAGE 5 — Polishing for Employer Presentation (Final 2 Weeks)**

**1️⃣ Technical Documentation**

* Clean README with:
  + System diagram (frontend ↔ backend ↔ AI microservice ↔ C++ module)
  + Benchmark table (C++ vs JS)
  + Architecture explanation (why Node + C++)
  + Screenshots of AI and UI features

**2️⃣ Demo Video (2–3 minutes)**

* Narrate the flow:
  + Load page → smooth UI
  + Search → AI suggestion
  + Product → recommendation panel
  + Admin dashboard → thread visualization
  + Closing slide → architecture overview

**3️⃣ Deployment**

* Host live demo on AWS + public GitHub repo
* Add clear disclaimers: “Experimental Project – Non-commercial Replica for Learning”

**🧠 EMPLOYER VALUE HIGHLIGHTS**

| **Skill** | **How You Show It** |
| --- | --- |
| **Front-end polish** | React, animation, UX quality |
| **Full-stack architecture** | Node, GraphQL, AWS, CI/CD |
| **Systems engineering** | C++ addon, thread simulation, benchmarks |
| **AI/ML understanding** | Recommendation + visual similarity models |
| **Documentation & DevOps** | AWS, architecture diagrams, video demo |

This hits *every modern engineering dimension* — front-end, AI, and low-level — in a single unified project that’s visually impressive **and technically serious**.

Would you like me to **turn this into a structured 12-week roadmap** (timeline + goals + deliverables per week)?  
That would give you a precise, followable plan to execute it like a professional product.