**📑 Trent Concept to Catwalk — PoC Documentation**

**1. Overview**

This PoC demonstrates an **agentic multi-agent AI pipeline** for trend discovery in fashion.  
The system ingests social/media fixture data, processes it through a sequence of agents, and outputs **trend briefs, clusters, and color palettes**.

The design ensures **compliance**, **scalability**, and **extensibility** — starting with fixture-based flows and progressively adding vision + clustering intelligence.

**2. High-Level Workflow**

flowchart TD

A[Seed Input: season, keywords] --> B[Orchestrator]

B --> C1[Agent 1: Discovery]

C1 -->|Fixture posts| C2[Agent 2: Parser]

C2 -->|Tokens + Hashtags| C3[Agent 3: Vision]

C3 -->|Palettes + Features| D[Cluster Builder]

D --> E[Output: Trend Brief + Clusters + Palettes]

E -->|JSON| F[Backend API (FastAPI)]

F --> G[Frontend Dashboard (React)]

**3. System Components**

**Backend (FastAPI + Agents)**

* **Orchestrator** – Coordinates the pipeline, calls each agent in sequence, aggregates results.
* **Agents**:
  + **DiscoveryAgent** → Collects posts (fixture JSON, later social APIs).
  + **ParserAgent** → Extracts tokens & normalizes hashtags.
  + **VisionAgent** → Extracts dominant color palettes from images.
  + **Cluster Builder** (inside Orchestrator for now) → Groups posts into trend clusters.

**Frontend (React + Tailwind)**

* Displays **Trend Briefs**, **Clusters**, and **Color Palettes** interactively.

**Storage**

* Fixture JSON + local images.
* Later extensible to Postgres + pgvector.

**4. Sprint Breakdown**

**Sprint 1 — Skeleton & Orchestrator (Fixture Based)**

* Setup project structure (frontend + backend).
* Implemented **BaseAgent** (lifecycle, logging, async).
* Added **DiscoveryAgent** reading from local seed\_posts.json.
* Orchestrator returns:
  + clusters: [] (empty placeholder)
  + brief\_md: "stub brief"
  + items: fixture posts

✅ Frontend showed **Brief only**, no clusters.

**Sprint 2 — ParserAgent**

* Added **ParserAgent** to tokenize captions and normalize hashtags.
* Orchestrator pipeline updated:
  + Discovery → Parser → fake clustering (group by first hashtag).
* Output:
  + Items enriched with tokens and normalized hashtags.
  + Clusters now contain **labels** and **post IDs**.

✅ Frontend displayed **Top Clusters** (hashtags as cluster labels).

**Sprint 3 — VisionAgent (Palette Extraction)**

* Integrated **Pillow + scikit-learn KMeans** for color palette extraction.
* VisionAgent attaches palette (dominant colors) to each post.
* Orchestrator enriches clusters with combined palettes.

✅ Frontend displayed **clusters with color swatches**.

**5. Current Output Example**

**Input**

{

"season": "SS25",

"keywords": ["olive", "pleated"]

}

**Output (simplified)**

{

"clusters": [

{

"label": "#olive",

"posts": ["p1"],

"palettes": ["#6d7f4b", "#c7b87e"]

},

{

"label": "#sunset",

"posts": ["p2"],

"palettes": ["#f68c3d", "#2e2d2c"]

}

],

"brief\_md": "## Trend Brief\n\nClusters now enriched with color palettes.",

"items": [

{

"id": "p1",

"caption": "Olive organza pleated skirt making waves!",

"tokens": ["olive", "organza", "pleated", "skirt", "making", "waves"],

"hashtags": ["#olive", "#organza", "#pleated"],

"palette": [...]

},

...

]

}

**6. Compliance & Ethical AI**

* **Fixture-first**: no personal data, only synthetic posts.
* **Modular agents**: easy to replace with official APIs (Instagram Graph API, YouTube API).
* **Transparency**: clusters + palettes derived from explicit parsing & vision steps.
* **Ethics**: no raw PII stored, only aggregated tokens and colors.

**7. Next Steps (Future Sprints)**

* **TrendClusterer** → semantic embeddings (OpenAI/CLIP + HDBSCAN).
* **Engagement Metrics Agent** → normalize likes/views (where permitted).
* **BriefWriter Agent** → auto-generate markdown briefs for decks.
* **Phase 2** → product design (generative AI fashion sketches) + virtual showcase.

📌 **Summary**:  
The PoC already demonstrates an **end-to-end async agentic workflow** with discovery, parsing, vision enrichment, and clustering, exposed via API and visualized on a UI. It is modular, compliant, and extensible for multi-platform real-time trend discovery.

**📊 Slide-Deck Summary — Trent Concept to Catwalk PoC**

**Slide 1 — Title**

**Agentic AI in Fashion: From Concept to Catwalk**

* Hackathon PoC Submission
* Multi-Agent AI Workflow for Fashion Trend Discovery

**Slide 2 — Problem Statement**

* Fashion industry struggles to **spot & act on emerging trends quickly**
* Social media trends are **fragmented across TikTok, Instagram, YouTube**
* Need: **autonomous AI pipeline** → discover → interpret → design → showcase

**Slide 3 — Objectives**

* **Track** creators, hashtags, motifs, visual aesthetics
* **Aggregate** from social/web sources
* **Analyze** silhouettes, colors, palettes, engagement
* **Generate** trend briefs & visual outputs
* **Showcase** via interactive UI

**Slide 4 — Architecture (High-Level Flow)**

**Agents in Sequence**

1. DiscoveryAgent → Collect posts
2. ParserAgent → Extract tokens & normalize hashtags
3. VisionAgent → Extract color palettes
4. Cluster Builder → Group posts into trends
5. Orchestrator → Combine results → API → Frontend

Diagram (suggest to insert flowchart I gave you earlier).

**Slide 5 — Tech Stack**

* **Backend**: FastAPI, Async Agents, Python
* **Vision**: Pillow, Scikit-Learn (KMeans for palettes)
* **Frontend**: React + Tailwind Dashboard
* **Storage**: Fixture JSON + local assets (future: Postgres + pgvector)
* **Extensible**: APIs (IG Graph, TikTok, YouTube), CLIP embeddings

**Slide 6 — Sprint 1**

**Skeleton & Orchestrator**

* BaseAgent (lifecycle + logging)
* DiscoveryAgent (fixture JSON)
* Orchestrator returns posts + stub brief  
  ✅ Backend live, frontend shows **Brief**

**Slide 7 — Sprint 2**

**ParserAgent**

* Tokenized captions
* Normalized hashtags
* Fake clusters (group by first hashtag)  
  ✅ Frontend shows **Top Clusters**

**Slide 8 — Sprint 3**

**VisionAgent (Palette Extraction)**

* Extract dominant colors via KMeans
* Add palette to posts
* Enrich clusters with palettes  
  ✅ Frontend shows **clusters + colors**

**Slide 9 — Current Demo Output**

* **Trend Brief** (Markdown)
* **Clusters** (hashtags + colors)
* **Posts** (captions, tokens, palettes)

*(Insert screenshot of your Postman output + frontend cluster view)*

**Slide 10 — Compliance & Ethics**

* Fixture-first (synthetic data only)
* No PII stored
* Modular → switch to official APIs
* Transparent + explainable AI steps
* Bias mitigation roadmap

**Slide 11 — Roadmap**

**Next Sprints:**

* TrendClusterer → semantic embeddings + HDBSCAN
* Engagement metrics (normalized rates)
* BriefWriter → auto 1-pager insights  
  **Phase 2:**
* Generative fashion design (AI sketches)
* Virtual runway showcase

**Slide 12 — Key Outcomes**

* **Agentic Orchestration** in action
* **End-to-End Automation** (data → insight → visualization)
* **Business Potential**: trend spotting → faster design decisions
* **Ethical AI** with compliance-first approach

**Slide 13 — Thank You**

**Trent Concept to Catwalk — Hackathon PoC**  
Q&A

**🔹 Expanded Data Sources**

* **Social media** (IG, TikTok, Pinterest, YouTube) → we already planned.
* **Fashion launches & WGSN** → curated trend forecasts.
* **Reposts / virality signals** → which looks are being re-shared.
* **Sales data (synthetic for PoC)** → connect styles → sell-through.
* **Cinema & International** → cultural/seasonal references.
* **Marketplaces** (Myntra, ASOS, Zara, etc).
* **Google Trends** → keyword-level signals.

👉 For PoC:  
We’ll simulate a few of these with synthetic JSON feeds or scraped open data (e.g. Google Trends API, fashion news RSS). We don’t need to integrate all real APIs right now — just stubs.

**🔹 Attributes to Track**

* **Style** (streetwear, feminine, oversized, etc.)
* **Print** (floral, check, abstract, etc.)
* **Color** (palettes from posts, Pantone mapping)
* **Length** (midi, maxi, cropped, etc.)
* **Time** (seasonal, trending week/month)
* **Trent Life** (brand-specific seasonal direction)

👉 These map neatly into our vision\_features schema. We just need to expand featurization beyond palette to style/print detection.

**🔹 3D / PLM Integration**

* **Outcome**: feed clusters + insights into a **3D Product Lifecycle Management (PLM)** pipeline.
* **3D outputs**: garments, models (gender/age/ethnicity), variability controls, and consumability (photos, video, p3d).
* **Why**: allow designers/buyers to not only *see* a trend but also *simulate products*.

👉 PoC approach:

* Use a **generative model** (e.g. Stable Diffusion + ControlNet / IP-Adapter) to create 2D apparel designs.
* Wrap with a lightweight 3D try-on pipeline (e.g. TripoSR / Kaedim or Three.js + mesh template) to simulate garments on avatars.
* Provide **parameter knobs** in frontend: color, print, length → regenerate render.

**🔹 Output Requirements**

1. **For Designers**: intermediate reports — cluster briefs, palette boards, print variations.
2. **For Buyers**: 3D models + placement in store → "If we buy into this trend, here’s how it looks in Trent Life store layout."
3. **Consumer-Facing** (eventually): variability, personalization, videos, AR try-ons.

**🔹 How It Fits Into Our Roadmap**

We already structured **Sprint 1 (agent skeleton)** and **Sprint 2 (clustering)**.  
This new requirement layers on:

* **Sprint 3 — Trend-to-Design Translation**
  + Agent that takes clusters → generates design prompts (style, print, palette).
  + Diffusion-based generative outputs (2D mood boards → garments).
  + Store synthetic metadata for 3D PLM compatibility.
* **Sprint 4 — Virtual Showcase**
  + Simple digital runway (React + Three.js).
  + Control panel for designers/buyers (choose trend cluster, tweak color/print).
  + Generate 3D avatars (different genders, ages, ethnicities).

⚖️ **PoC Balance**  
We can’t fully build PLM or CAD-grade outputs in the hackathon timeframe — but we can show:  
✅ Data in → clusters → AI garment renders → 3D avatars (basic) → export.  
This checks the “agentic AI, end-to-end pipeline” box **and** addresses designer/buyer needs.

Now for **Sprint 3** we shift gears to **Agent 2: Product Design**. That means:

* **2.1 Style-to-Design Translator**: take trend clusters (attributes, captions, colors) and extract structured design specs (silhouette, print, palette, length, season).
* **2.2 Generative Design Agent**: use those specs to generate mock product designs (initially 2D sketches with Stable Diffusion / ControlNet, later 3D-ready).
* **2.3 Virtual Model Agent**: render designs onto models (gender, age, ethnicity variability).
* **2.4 Digital Runway Orchestrator**: combine designs + models into short videos or carousels.

For Sprint 3.1, I suggest we start with a **DesignSpecAgent** (turns cluster features into structured design parameters). This keeps it lightweight and prepares for generative image models in 3.2.