Here's a polished, reviewer-friendly version of your architecture doc. I've kept the technical depth, but improved readability, flow, and framing so that someone evaluating the system's architecture and rationale can quickly grasp the design choices, motivations, and outcomes.

MontageAgent – Architecture & Rationale

Project: MontageAgent — AI Agent for Event Photo Curation & Instagram Publishing
Author: Prem Kondru (BTech Engineering Physics, IIT Guwahati; Photography Club: Montage)

Purpose & Problem

At IIT Guwahati's Photography Club (Montage), every event generates hundreds of photos. The manual workflow involves:

1. Removing duplicates
2. Grouping by theme/moment
3. Writing captions in a consistent style
4. Assembling Instagram carousel posts

This process is repetitive, error-prone, and consumes hours of human effort.

MontageAgent automates the workflow end-to-end:
Ingest → Dedupe → Categorize → Cluster → Caption → Export/Publish

How the Agent Works

Reasoning

- **Visual Understanding:** CLIP embeddings assess similarity, remove near-duplicates, and infer labels (zero-shot).
- **Captioning:** Two captioner modes:
  - **Template Mode** → Deterministic, rule-based captions.
  - **BLIP-2 (with LoRA)** → Generates per-image captions, then fuses them into one cluster caption based on common words.
- **Stylistic Consistency:** Retrieval-augmented input (RAG) over past captions maintains tone, avoids hashtags in body text, and ensures continuity across posts.

Planning

- A **Supervisor (Planner)** orchestrates a fixed pipeline:

Ingest → Embed → Dedupe → Categorize → Cluster → Caption → Export.

- The plan is deterministic, simplifying debugging and evaluation.

Execution

- **Executors (Workers):** Each step runs as an independent tool with clear inputs/outputs.
- **Human-in-the-loop UI:** Built with Streamlit, allowing users to include/exclude photos, adjust labels, and preview posts before export.

Components

- **Supervisor (Planner):** Loads config, applies UI overrides, sequences executors.
- **Ingestor:** Scans folders/uploads, normalizes paths & EXIF, builds image list.
- **Embedding Service (CLIP):** Provides `encode_paths` and `encode_texts` for similarity, clustering, and label assignment.
- **Deduper:** Removes near-duplicates via cosine similarity & NMS.
- **Categorizer/Labeler:** Zero-shot label assignment using CLIP and user-provided label sets.
- **Clusterer:** Groups images by visual similarity; enforces `max_images_per_post` with balanced sampling.
- **Captioner:**
  - **Template mode:** Deterministic phrasing.
  - **BLIP-2 mode:** Batch captioning + LoRA fine-tuning; fuses outputs into one abstract caption. Supports event-name injection policies (off | hint | only_proper_noun).
- **Exporter:** Builds JSON for Instagram carousel upload: `{caption, hashtags, images[]}`.
- **User Interface:** Streamlit app with IG-style previews, zoom, toggles, runtime configs, and JSON export.

Key Design Choices

- **Multi-agent setup:** Planner + Executors → modular, debuggable, human-auditable pipeline.
- **Fixed pipeline:** Single path improves reliability, reproducibility, and transparency.

- **CLIP for structural tasks (dedupe, clustering, labels):** Efficient, lightweight, and non-generative.
- **BLIP-2 + LoRA for style:** Strong caption quality with minimal compute/storage overhead.
- **Config-driven + UI overrides:** Enables repeatability, A/B testing, and flexible runtime control.
- **Guardrails:** Caption length limits, "event name only" injection, and exclusion of proper nouns ensure alignment with club style.

Interaction Flow

1. **User Input:** Upload images, set labels, choose captioner mode, define event name, adjust `max_images_per_post`.
2. **Pipeline Execution:** Ingest → Embed → Deduplicate → Categorize → Cluster → Caption → Export.
3. **Captioning:**
    - **BLIP-2 mode:** Generate per-image captions, extract common words, produce abstract caption.
    - **Template mode:** Deterministic, rule-based caption.
    - Hashtags pulled from base + labels + historical RAG hints.
4. **Preview & Export:** IG-style preview, per-image inclusion/exclusion, JSON export for carousel.

Models & Rationale

- **OpenCLIP (e.g., ViT-B/32, laion2b_s34b_b79k):**
    - Lightweight, efficient visual similarity.
    - Ideal for deduplication, clustering, zero-shot labels, CLIPScore.
- **RAG over Past Captions:**
    - Ensures consistency in tone/hashtags.
    - Avoids over-fitting to rigid templates.
- **BLIP-2 + Flan-T5 with LoRA:**
    - Generates abstract, mood-driven captions aligned with club style.
    - LoRA enables style specialization without full finetuning.
    - Per-image batching + common-word fusion prioritizes shared visual cues.

- Guardrails enforce reliability and stylistic alignment.

👉 This version presents **architecture, flow, and rationale in layers (problem → design → execution → choices → models)** so a reviewer can quickly grasp not just *what* was built, but *why* each decision was made.

Would you like me to also create a **1-page visual architecture diagram** (boxes + arrows + captions) to pair with this write-up for reviewers? That would make it easier to present in a deck or PDF.

You said:

from graphviz import Digraph # Create graph with top-to-bottom orientation for rows, left-to-right columns dot = Digraph("MontageAgent", format="png") dot.attr(rankdir="TB", size="11,8.5", nodesep="0.8", ranksep="1.0") dot.attr("node", shape="box", style="rounded,filled", fontname="Helvetica", fontsize="12", color="#333333", fillcolor="#ffffff", width="3.8", height="1.0") # Updated Title dot.attr(label="MontageAgent – Architecture Overview - Components", labelloc="t", fontsize="20", fontname="Helvetica-Bold") # Define labels ui_label = "🖥️\nUser Interface\n(Streamlit App)\n- Upload images\n- Runtime controls\n- IG-style preview\n- JSON export" sup_label = "🧩\nSupervisor (Planner)\n- Loads config & UI overrides\n- Orchestrates pipeline" ing_label = "📂\nIngestor\n- Folder scan / Upload\n- Normalize paths & EXIF" emb_label = "🧮\nEmbedding Service (CLIP)\n- encode_paths (images)\n- encode_texts (labels)" ded_label = "♻️\nDeduper\n- Remove near-duplicates\n- Cosine similarity + NMS" cat_label = "🏷️\nCategorizer / Labeler\n- Zero-shot with CLIP\n- User-provided label sets" clu_label = "🗃️\nClusterer\n- Group by visual similarity\n- Enforce max_images_per_post" cap_label = "✍️\nCaptioner\n- Template (deterministic)\n- BLIP-2 + LoRA\n- Common-word fusion\n- Event-name policy" exp_label = "📤\nExporter\n- JSON output for IG Carousel\n- Captions + Hashtags + Images" # Create columns (UI above Deduper, Supervisor above Categorizer, etc.) with dot.subgraph() as col1: col1.attr(rank="same") col1.node("UI", ui_label) col1.node("Deduper", ded_label) with dot.subgraph() as col2: col2.attr(rank="same") col2.node("Supervisor", sup_label) col2.node("Categorizer", cat_label) with dot.subgraph() as col3: col3.attr(rank="same") col3.node("Ingestor", ing_label) col3.node("Clusterer", clu_label) with dot.subgraph() as col4: col4.attr(rank="same") col4.node("Embedder", emb_label) col4.node("Captioner", cap_label) with dot.subgraph() as col5: col5.attr(rank="same") col5.node("Exporter", exp_label) # Row 1 horizontal edges dot.edge("UI", "Supervisor") dot.edge("Supervisor", "Ingestor") dot.edge("Ingestor", "Embedder") dot.edge("Embedder", "Exporter", style="invis") # keeps row1 aligned left to right # Row 2 horizontal edges dot.edge("Deduper", "Categorizer") dot.edge("Categorizer", "Clusterer") dot.edge("Clusterer", "Captioner") dot.edge("Captioner", "Exporter") # Vertical connector between rows (Embedder to Deduper) dot.edge("Embedder", "Deduper") # Render PNG file_path = "/mnt/data/montage_agent_architecture_grid_title" dot.render(file_path, format="png", cleanup=True) file_path + ".png"

ChatGPT said:

Analyzed

```
Always show details
```

```
from graphviz import Digraph

# Create graph with top-to-bottom orientation for rows,
left-to-right columns
dot = Digraph("MontageAgent", format="png")
dot.attr(rankdir="TB", size="11,8.5", nodesep="0.8",
ranksep="1.0")
dot.attr("node", shape="box", style="rounded,filled",
fontname="Helvetica", fontsize="12",
         color="#333333", fillcolor="#ffffff",
width="3.8", height="1.0")

# Updated Title
dot.attr(label="MontageAgent — Architecture Overview -
Components", labelloc="t", fontsize="20",
fontname="Helvetica-Bold")

# Define labels
ui_label = "🖥️\nUser Interface\n(Streamlit App)\n- Upload
images\n- Runtime controls\n- IG-style preview\n- JSON
export"
sup_label = "🧩\nSupervisor (Planner)\n- Loads config &
UI overrides\n- Orchestrates pipeline"
ing_label = "📂\nIngestor\n- Folder scan / Upload\n-
Normalize paths & EXIF"
emb_label = "🧱\nEmbedding Service (CLIP)\n- encode_paths
(images)\n- encode_texts (labels)"

ded_label = "♻️\nDeduper\n- Remove near-duplicates\n-
Cosine similarity + NMS"
cat_label = "🏷️\nCategorizer / Labeler\n- Zero-shot with
CLIP\n- User-provided label sets"
clu_label = "🗂️\nClusterer\n- Group by visual
similarity\n- Enforce max_images_per_post"
cap_label = "✍️\nCaptioner\n- Template (deterministic)\n-
BLIP-2 + LoRA\n- Common-word fusion\n- Event-name policy"
exp_label = "📤\nExporter\n- JSON output for IG
Carousel\n- Captions + Hashtags + Images"

# Create columns (UI above Deduper, Supervisor above
Categorizer, etc.)
with dot.subgraph() as col1:
    col1.attr(rank="same")
    col1.node("UI", ui_label)
```

```
    col1.node("Deduper", ded_label)

with dot.subgraph() as col2:
    col2.attr(rank="same")
    col2.node("Supervisor", sup_label)
    col2.node("Categorizer", cat_label)

with dot.subgraph() as col3:
    col3.attr(rank="same")
    col3.node("Ingestor", ing_label)
    col3.node("Clusterer", clu_label)

with dot.subgraph() as col4:
    col4.attr(rank="same")
    col4.node("Embedder", emb_label)
    col4.node("Captioner", cap_label)

with dot.subgraph() as col5:
    col5.attr(rank="same")
    col5.node("Exporter", exp_label)

# Row 1 horizontal edges
dot.edge("UI", "Supervisor")
dot.edge("Supervisor", "Ingestor")
dot.edge("Ingestor", "Embedder")
dot.edge("Embedder", "Exporter", style="invis")  # keeps
row1 aligned left to right

# Row 2 horizontal edges
dot.edge("Deduper", "Categorizer")
dot.edge("Categorizer", "Clusterer")
dot.edge("Clusterer", "Captioner")
dot.edge("Captioner", "Exporter")

# Vertical connector between rows (Embedder to Deduper)
dot.edge("Embedder", "Deduper")

# Render PNG
file_path = "/mnt/data/
montage_agent_architecture_grid_title"
dot.render(file_path, format="png", cleanup=True)

file_path + ".png"
```

Result

```
'/mnt/data/montage_agent_architecture_grid_title.png'
```
Done! Here's your generated diagram:

Download the PNG

Want me to:

- add a small **legend** (icons meaning, captioner modes),
- include **RAG side-input** and **LoRA** callouts, or
- export a crisp **SVG** version for slides?

You said:

add a small legend (icons meaning, captioner modes), include RAG side-input and LoRA callouts, export a crisp SVG version for slides?

ChatGPT said:

Analyzed

```
Always show details
from graphviz import Digraph

# Create graph with top-to-bottom orientation for rows,
left-to-right columns
dot = Digraph("MontageAgent", format="svg")
dot.attr(rankdir="TB", size="11,8.5", nodesep="0.8",
ranksep="1.0")
dot.attr("node", shape="box", style="rounded,filled",
fontname="Helvetica", fontsize="12",
         color="#333333", fillcolor="#ffffff",
width="3.8", height="1.0")

# Updated Title
dot.attr(label="MontageAgent — Architecture Overview -
Components", labelloc="t", fontsize="20",
fontname="Helvetica-Bold")

# Define labels
ui_label = "🖥️\nUser Interface\n(Streamlit App)\n- Upload
images\n- Runtime controls\n- IG-style preview\n- JSON
export"
sup_label = "🧩\nSupervisor (Planner)\n- Loads config &
UI overrides\n- Orchestrates pipeline"
ing_label = "📂\nIngestor\n- Folder scan / Upload\n-
```

```
Normalize paths & EXIF"
emb_label = "🧱\nEmbedding Service (CLIP)\n- encode_paths
(images)\n- encode_texts (labels)"

ded_label = "♻️\nDeduper\n- Remove near-duplicates\n-
Cosine similarity + NMS"
cat_label = "🏷️\nCategorizer / Labeler\n- Zero-shot with
CLIP\n- User-provided label sets"
clu_label = "🗂️\nClusterer\n- Group by visual
similarity\n- Enforce max_images_per_post"
cap_label = "✍️\nCaptioner\n- Template (deterministic)\n-
BLIP-2 + LoRA\n- Common-word fusion\n- Event-name policy"
exp_label = "📤\nExporter\n- JSON output for IG
Carousel\n- Captions + Hashtags + Images"

# Create columns (UI above Deduper, Supervisor above
Categorizer, etc.)
with dot.subgraph() as col1:
    col1.attr(rank="same")
    col1.node("UI", ui_label)
    col1.node("Deduper", ded_label)

with dot.subgraph() as col2:
    col2.attr(rank="same")
    col2.node("Supervisor", sup_label)
    col2.node("Categorizer", cat_label)

with dot.subgraph() as col3:
    col3.attr(rank="same")
    col3.node("Ingestor", ing_label)
    col3.node("Clusterer", clu_label)

with dot.subgraph() as col4:
    col4.attr(rank="same")
    col4.node("Embedder", emb_label)
    col4.node("Captioner", cap_label)

with dot.subgraph() as col5:
    col5.attr(rank="same")
    col5.node("Exporter", exp_label)

# Row 1 horizontal edges
dot.edge("UI", "Supervisor")
dot.edge("Supervisor", "Ingestor")
```

```
dot.edge("Ingestor", "Embedder")
dot.edge("Embedder", "Exporter", style="invis")  # keeps
row1 aligned left to right

# Row 2 horizontal edges
dot.edge("Deduper", "Categorizer")
dot.edge("Categorizer", "Clusterer")
dot.edge("Clusterer", "Captioner")
dot.edge("Captioner", "Exporter")

# Vertical connector between rows (Embedder to Deduper)
dot.edge("Embedder", "Deduper")

# --- Callouts for RAG and LoRA ---
dot.node("RAG", "📚\nRAG Side-Input\n- Past captions\n-
Consistent tone & hashtags",
         shape="note", color="#0055aa",
fillcolor="#e6f0ff", width="3.2")
dot.edge("RAG", "Captioner", style="dashed",
color="#0055aa", label="hint")

dot.node("LoRA", "🎨\nLoRA Adapters\n- Style alignment\n-
Lightweight tuning",
         shape="note", color="#aa5500",
fillcolor="#fff0e6", width="3.2")
dot.edge("LoRA", "Captioner", style="dashed",
color="#aa5500", label="style")

# --- Legend ---
with dot.subgraph(name="cluster_legend") as legend:
    legend.attr(label="Legend", fontsize="14",
fontname="Helvetica-Bold")
    legend.attr(style="dashed", color="#888888")
    legend.node("L1", "🖥️ UI: Human
interaction\n(Streamlit App)", shape="box",
style="rounded", width="3.5")
    legend.node("L2", "🧩 Supervisor: Orchestration",
shape="box", style="rounded", width="3.5")
    legend.node("L3", "📂 Ingestor / Data Loader",
shape="box", style="rounded", width="3.5")
    legend.node("L4", "🧱 CLIP Embedding Service",
shape="box", style="rounded", width="3.5")
    legend.node("L5", "♻️ Deduper | 🏷️ Categorizer | 📦
Clusterer", shape="box", style="rounded", width="3.5")
```

```
    legend.node("L6", "✍️ Captioner Modes:\n- Template\n-
BLIP-2 + LoRA\n- Event-name policy", shape="box",
style="rounded", width="3.5")
    legend.node("L7", "📤 Exporter (JSON for IG
Carousel)", shape="box", style="rounded", width="3.5")
    legend.node("L8", "📚 RAG Side Input (dashed edge =
hint)", shape="note", width="3.5")
    legend.node("L9", "🎨 LoRA Adapters (dashed edge =
style)", shape="note", width="3.5")

# Render SVG
file_path = "/mnt/data/
montage_agent_architecture_with_legend"
dot.render(file_path, format="svg", cleanup=True)

file_path + ".svg"
```

Result

```
'/mnt/data/montage_agent_architecture_with_legend.svg'
```

Here's your updated architecture diagram with:

✅ **Legend** (icons + captioner modes)
✅ **RAG side-input** and **LoRA callouts**
✅ **Exported as crisp SVG** (perfect for slides)

Download the SVG

Do you also want me to prepare a **clean PDF version** (with the same diagram + short description on one page) for easy reviewer sharing?

You said:

remove the legend

ChatGPT said:

Analyzed

```
Always show details
from graphviz import Digraph

# Create graph with top-to-bottom orientation for rows,
left-to-right columns
dot = Digraph("MontageAgent", format="svg")
dot.attr(rankdir="TB", size="11,8.5", nodesep="0.8",
ranksep="1.0")
dot.attr("node", shape="box", style="rounded,filled",
fontname="Helvetica", fontsize="12",
         color="#333333", fillcolor="#ffffff",
```

```
width="3.8", height="1.0")

# Updated Title
dot.attr(label="MontageAgent — Architecture Overview -
Components", labelloc="t", fontsize="20",
fontname="Helvetica-Bold")

# Define labels
ui_label = "🖥️\nUser Interface\n(Streamlit App)\n- Upload
images\n- Runtime controls\n- IG-style preview\n- JSON
export"
sup_label = "🧩\nSupervisor (Planner)\n- Loads config &
UI overrides\n- Orchestrates pipeline"
ing_label = "📂\nIngestor\n- Folder scan / Upload\n-
Normalize paths & EXIF"
emb_label = "🧱\nEmbedding Service (CLIP)\n- encode_paths
(images)\n- encode_texts (labels)"

ded_label = "♻️\nDeduper\n- Remove near-duplicates\n-
Cosine similarity + NMS"
cat_label = "🏷️\nCategorizer / Labeler\n- Zero-shot with
CLIP\n- User-provided label sets"
clu_label = "🗂️\nClusterer\n- Group by visual
similarity\n- Enforce max_images_per_post"
cap_label = "✍️\nCaptioner\n- Template (deterministic)\n-
BLIP-2 + LoRA\n- Common-word fusion\n- Event-name policy"
exp_label = "📤\nExporter\n- JSON output for IG
Carousel\n- Captions + Hashtags + Images"

# Create columns (UI above Deduper, Supervisor above
Categorizer, etc.)
with dot.subgraph() as col1:
    col1.attr(rank="same")
    col1.node("UI", ui_label)
    col1.node("Deduper", ded_label)

with dot.subgraph() as col2:
    col2.attr(rank="same")
    col2.node("Supervisor", sup_label)
    col2.node("Categorizer", cat_label)

with dot.subgraph() as col3:
    col3.attr(rank="same")
    col3.node("Ingestor", ing_label)
```

```
    col3.node("Clusterer", clu_label)

with dot.subgraph() as col4:
    col4.attr(rank="same")
    col4.node("Embedder", emb_label)
    col4.node("Captioner", cap_label)

with dot.subgraph() as col5:
    col5.attr(rank="same")
    col5.node("Exporter", exp_label)

# Row 1 horizontal edges
dot.edge("UI", "Supervisor")
dot.edge("Supervisor", "Ingestor")
dot.edge("Ingestor", "Embedder")
dot.edge("Embedder", "Exporter", style="invis")  # keeps
row1 aligned left to right

# Row 2 horizontal edges
dot.edge("Deduper", "Categorizer")
dot.edge("Categorizer", "Clusterer")
dot.edge("Clusterer", "Captioner")
dot.edge("Captioner", "Exporter")

# Vertical connector between rows (Embedder to Deduper)
dot.edge("Embedder", "Deduper")

# --- Callouts for RAG and LoRA ---
dot.node("RAG", "📚\nRAG Side-Input\n- Past captions\n-
Consistent tone & hashtags",
         shape="note", color="#0055aa",
fillcolor="#e6f0ff", width="3.2")
dot.edge("RAG", "Captioner", style="dashed",
color="#0055aa", label="hint")

dot.node("LoRA", "🎨\nLoRA Adapters\n- Style alignment\n-
Lightweight tuning",
         shape="note", color="#aa5500",
fillcolor="#fff0e6", width="3.2")
dot.edge("LoRA", "Captioner", style="dashed",
color="#aa5500", label="style")

# Render SVG without legend
file_path = "/mnt/data/
```

```
montage_agent_architecture_no_legend"
dot.render(file_path, format="svg", cleanup=True)

file_path + ".svg"
```

Result

```
'/mnt/data/montage_agent_architecture_no_legend.svg'
```

Here's the updated diagram without the legend, but still showing the **RAG** and **LoRA** callouts:

Download the SVG

Do you also want me to make a **minimalist version** (just the main pipeline without callouts), for use in slides where space is tight?