AI Engineering Lab
Week 1

Haidar Chaito & Noureddine Kamzon

September 5, 2025

Model Pipeline — Main Steps

1. **Image loading**: resize longest side to 1024 (keep aspect).
2. **SAM init**: auto-download checkpoint; run on GPU if available.
3. **Mask generation**.
4. **Panoptic + classify**: `top_n`=10 segments; ResNet-50 $\rightarrow$ zero-shot; aggregate counts.

Model Pipeline — Segmentation & Results

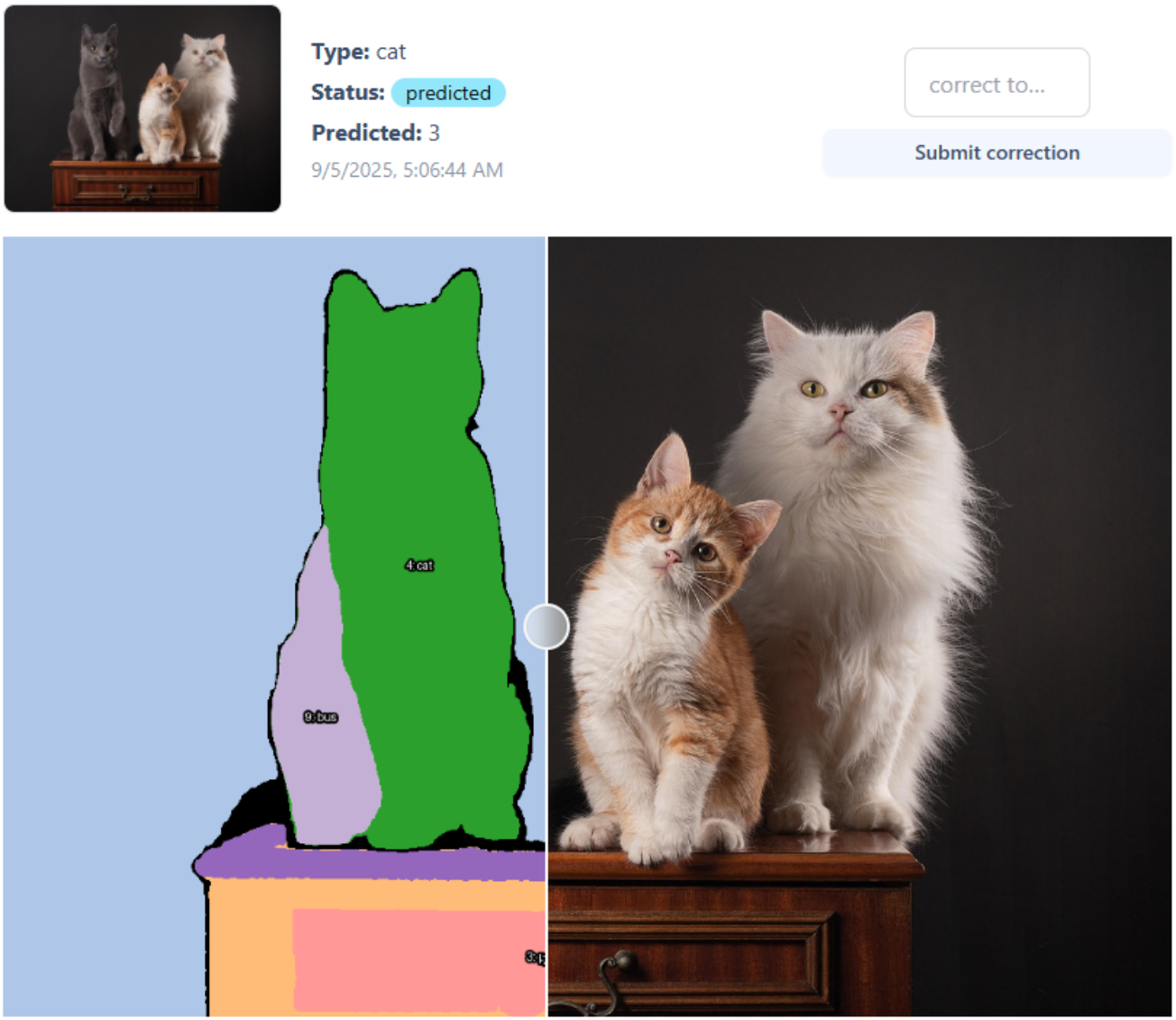
Segmentation

- Original + panoptic overlay.
- Distinct colors; IDs/labels on segments.
- Annotated image saved as artifact.

Classification & counts

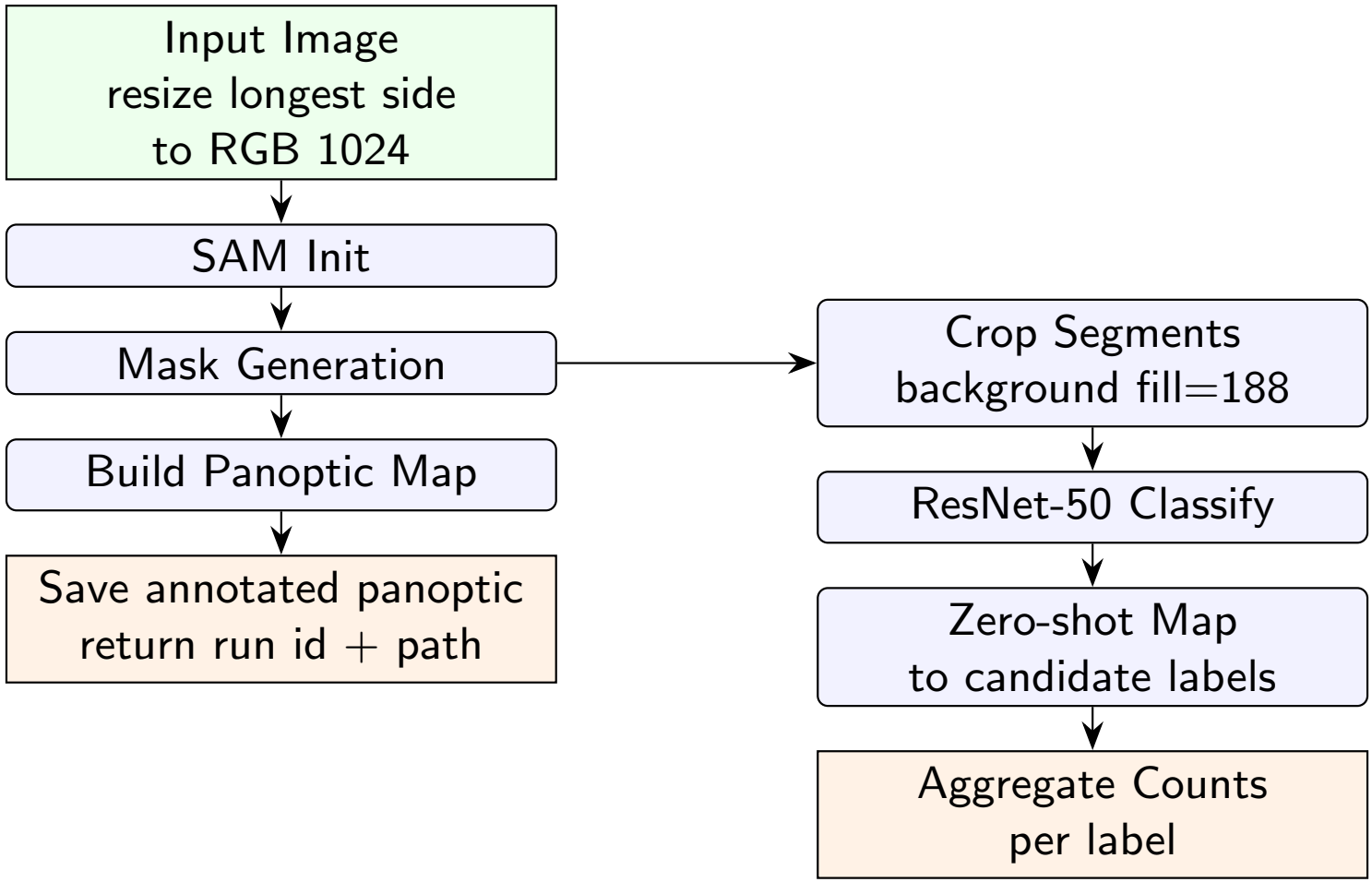
- ResNet-50 predicts classes.
- Zero-shot maps to candidate labels.
- Aggregate per-label counts; batch supported.

Model Pipeline — Example Output

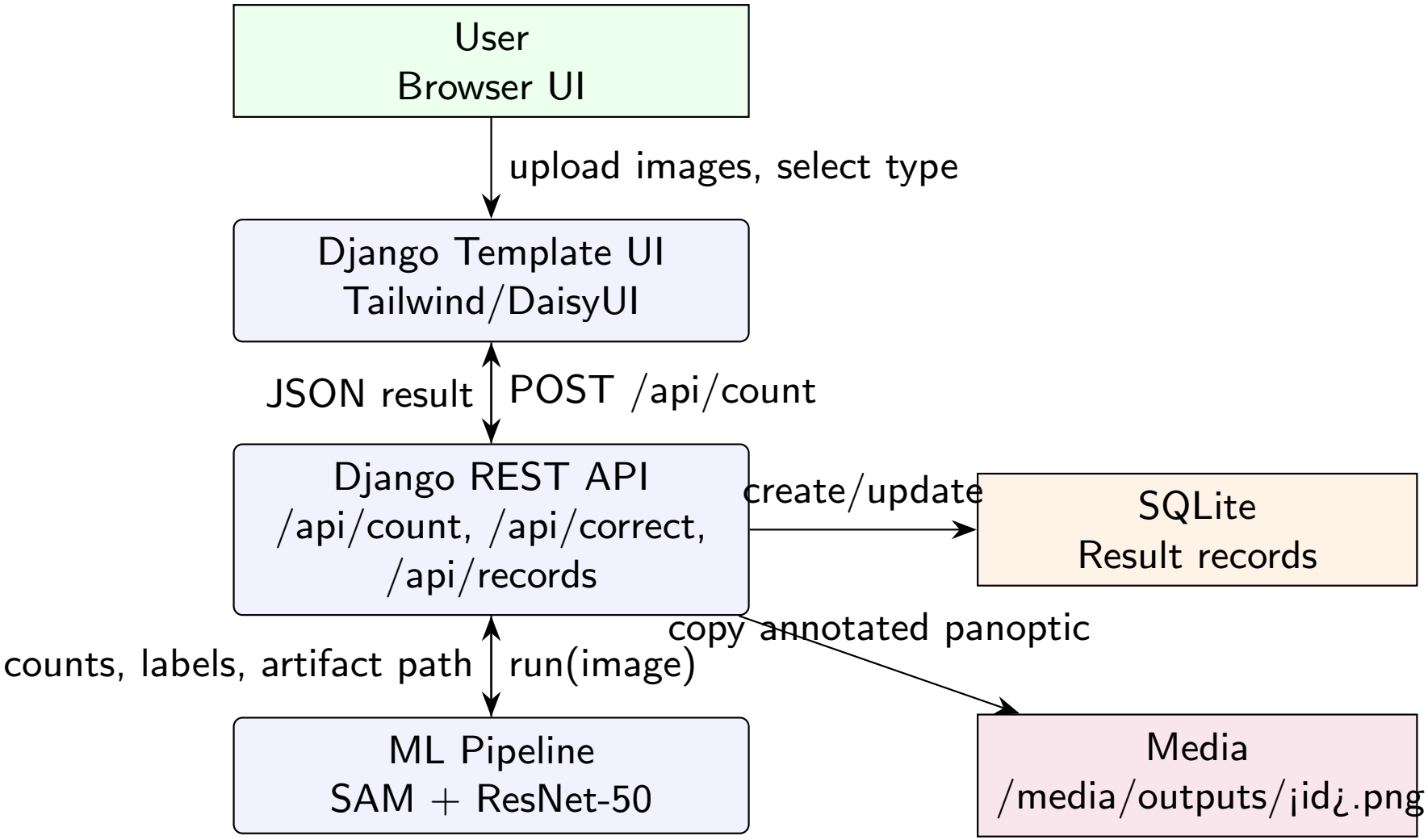


Saved annotated panoptic result generated by the pipeline.

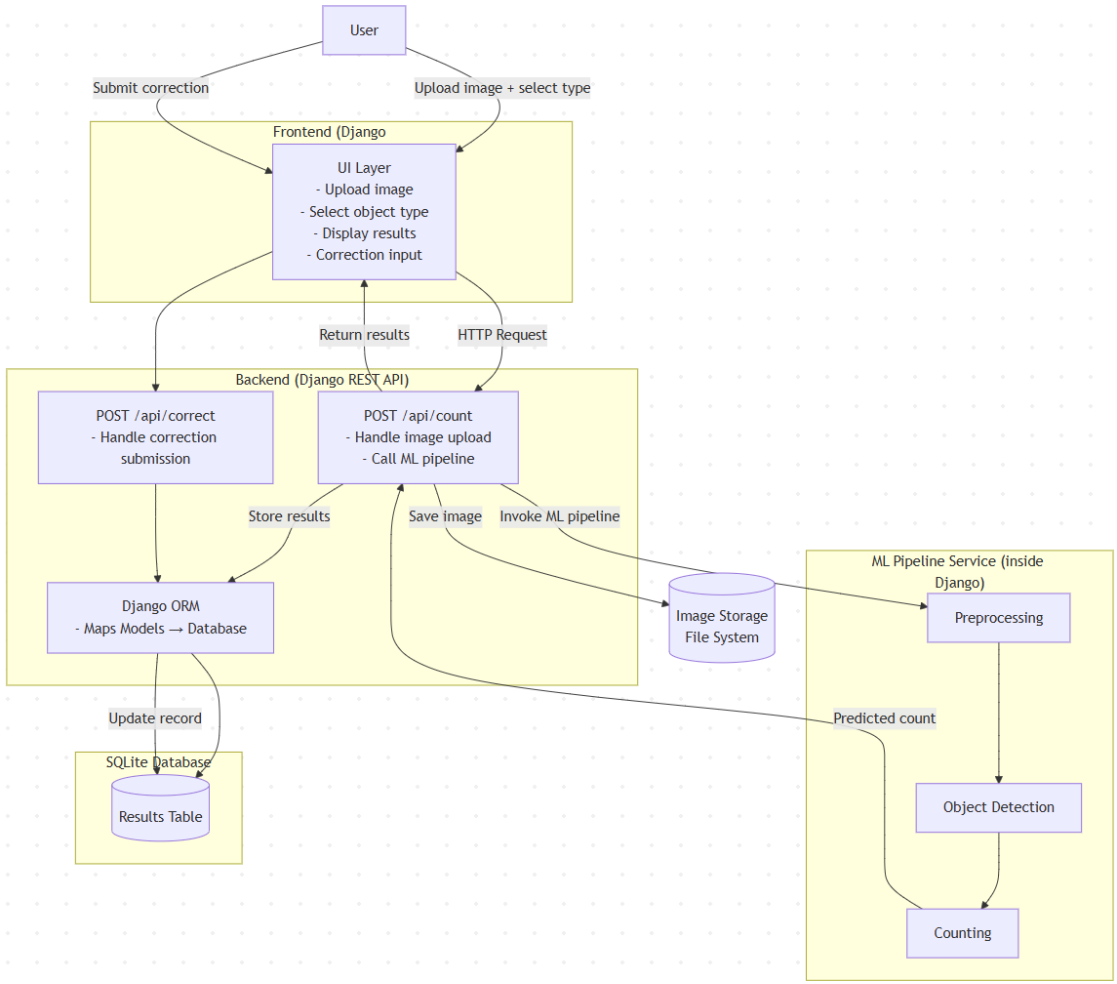
Model Pipeline — Flow Diagram



End-to-End System — Architecture Flow



Architecture Diagram



Data flows from user input through the frontend, backend, ML pipeline, and database.

Model Pipeline: Strengths & Limitations

- **Where It Works Well**
  - Clear, high-resolution images with distinct boundaries.
  - Segments and counts objects (e.g., cats, cars, people) in uncluttered scenes.
- **Where It Struggles**
  - Overlapping objects, complex backgrounds, low contrast.
  - Small/ambiguous objects.
- **Insights from Testing**
  - Parameter tuning improves results but can't fix poor input quality.
  - Visual inspection remains essential.
- **Recommendations**
  - Use high-quality, well-lit images.
  - Consider model fine-tuning or post-processing for complex scenes.

API Endpoints

- **POST /api/count/**
  - Multipart: `image` (single) *or* repeated `images` (multiple), and `object_type`.
  - Returns created `Result`(s) with `predicted_count`, `status`, and `meta.panoptic_url`.
- **POST /api/correct/**
  - JSON or form: `result_id`, `corrected_count`.
  - Updates result to `status=corrected`.
- **GET /api/records/**
  - CRUD for stored results (read-only in UI).
- **Docs**
  - OpenAPI schema: `/api/schema` Interactive docs (ReDoc): `/api/docs`

Data Model — Result

- **Fields**
  - `id`, `image`, `object_type`, `predicted_count`, `corrected_count`
  - `status` (`pending|processing|predicted|corrected|failed`), `meta`, timestamps
- **Notes**
  - `OBJECT_TYPES` come from a configurable candidate label list.
  - Panoptic artifacts are copied to `/media/outputs/<id>.png` for display.

Frontend UX

- **Upload Count**
  - Multiple file selection supported; progress and status badges shown.
- **Visualization**
  - Diff slider to compare original and annotated panoptic output.
- **Corrections**
  - Inline correction form posts to `/api/correct/`; UI updates instantly.
- **History**
  - On-demand load from `/history/` to review past results.

Quality Assurance Measures

- **Unit Testing**
  - API tests cover corrections and validation paths; pipeline mocked for speed.
  - Edge cases and error handling validated (e.g., missing fields).
- **Automated Test Execution**
  - Run with Django framework: `python manage.py test`.
  - Ensures code changes don't break existing features.
- **API Documentation**
  - OpenAPI schema with drf-spectacular; interactive ReDoc at `/api/docs`.
- **Test Coverage**
  - Covers main workflows and critical logic.
  - Updated as new features are added.

References

- **Segment Anything Model (SAM)**
  https://github.com/facebookresearch/segment-anything
- **PyTorch** https://pytorch.org/
- **Transformers Library**
  https://github.com/huggingface/transformers
- **Django Framework** https://www.djangoproject.com/
- **Django REST Framework**
  https://www.django-rest-framework.org/
- **drf-spectacular (OpenAPI)**
  https://drf-spectacular.readthedocs.io/
- **Matplotlib** https://matplotlib.org/
- **Pillow** https://python-pillow.org/
- **Tailwind CSS / DaisyUI** https://tailwindcss.com/
  https://daisyui.com/