OWL PLATFORM

Implementation Plan

Priority Features — Full Codebase Analysis & Build Roadmap

Prepared: February 2026 **Features Analyzed:** 14 **Classification:** Confidential

Owl Platform — Implementation Plan

Priority Features: User Roadmap Analysis *Prepared: February 2026*

Executive Summary

14 priority features have been analyzed against the current codebase. They fall into four categories:

Category	Count	Description
🔴 **Missing**	6	Not built at all — needs full implementation
🟡 **Partial**	5	Infrastructure exists, UI or backend incomplete
🟢 **Near Complete**	2	Small fixes or wiring needed
⚙️ **Config/Design**	2	Requires prompt tuning or architectural decision

Recommended build order: Quick wins first (3 items under a day each), then high-impact medium features, then complex new systems.

Priority 1 — Quick Wins (1–3 days each)

P1.1 — Merge Tab: Document Opens in Background

Status: 🟢 Near Complete — z-index / portal fix **Effort:** 0.5 days **Also affects:** Map view (same issue reported there)

Root Cause: The document viewer and map popups open in the same DOM layer as the modal. When a modal (MergeEntitiesModal) is open with z-index ~50, the DocumentViewer renders inside or below it rather than above.

What Needs to Change:

File: `/frontend/src/components/MergeEntitiesModal.jsx` - Identify where `DocumentViewer` is invoked inside the modal - Wrap DocumentViewer in a React Portal (`ReactDOM.createPortal`) so it renders at the document body level, above all modals

File: `/frontend/src/components/DocumentViewer.jsx` - Ensure the component's overlay has the highest z-index in the app (z-index: 9999) - Add a close button that returns focus to the merge modal without closing it

File: `/frontend/src/components/MapView.jsx` - Apply same portal fix to any popups or detail panels opened from map markers

Steps: 1. Import `ReactDOM` in MergeEntitiesModal.jsx 2. Wrap `<DocumentViewer>` in `ReactDOM.createPortal(..., document.body)` 3. Set DocumentViewer overlay CSS to `z-index: 9999` 4. Test: Open merge modal → click document → verify document opens on top → close document → merge modal still open 5. Apply same fix to MapView popup panels

---

P1.2 — AI Creates Too Many Entities — Reduce Noise

Status: ⚙️ Config — Prompt + profile tuning **Effort:** 1 day

Root Cause: The extraction prompt in `llm_client.py` instructs the LLM to extract all entities it can find, including minor mentions, generic references, and individual rows from tables. This produces a noisy graph.

What Needs to Change:

File: `/backend/ingestion/scripts/llm_client.py` (extraction prompt ~line 477–516)

Update the system prompt to add:

```
ENTITY EXTRACTION RULES:
- Only extract entities that are SIGNIFICANT to the investigation (key people,
organizations, accounts, locations directly involved in the matter)
- Do NOT extract generic references, job titles without a named person, or
passing mentions
- For tables with many rows: summarize repeated transaction parties into a
single entity; do not create one entity per row
- Minimum importance threshold: an entity must appear in context of a specific
event, transaction, or relationship to be extracted
- Prefer fewer, higher-quality entities over exhaustive extraction
- If unsure whether an entity is significant, do NOT extract it
```

File: `/backend/ingestion/scripts/entity_resolution.py` - Increase fuzzy match threshold for entity resolution (reduce false "new entity" creation) - Current: likely 80–85% similarity → raise to 90%+ for names

File: `/profiles/*.json` (all profile configs) - Add `"max_entities_per_chunk": 25` to ingestion settings - Add `"min_entity_importance": "medium"` — filter out "low" importance extractions

File: `/backend/ingestion/scripts/llm_client.py` — post-processing - After extraction, add importance scoring step: ask LLM to rate each extracted entity 1–5 - Filter out entities scored 1–2 before saving to graph

Steps: 1. Update extraction system prompt with rules above 2. Add importance threshold parameter to profile configs 3. Test on a known case — compare entity count before/after 4. Tune threshold based on results (target: 30–50% reduction in entity count) 5. Document the new behavior in profile README

P1.3 — Save AI Chat Responses as Notes

Status: 🟢 Near Complete — frontend wiring only **Effort:** 0.5 days **Priority:** "Want not must" — but very fast to build

What Exists: - Full notes CRUD API already exists in `/backend/routers/workspace.py` - `POST /{case_id}/notes` — create note with title, content, tags - Chat messages rendered in `/frontend/src/components/ChatPanel.jsx`

What Needs to Change:

File: `/frontend/src/components/ChatPanel.jsx` - Add a "Save as Note" button (📌 icon) to each assistant message bubble - On click: open a small inline modal pre-filled with: - **Title:** First 60 chars of the user's question that prompted it - **Content:** Full AI response text - **Tags:** ["ai-chat", "auto"] - On confirm: call `POST /api/workspace/{caseId}/notes` - Show success toast: "Response saved to case notes"

File: `/frontend/src/services/api.js` - Verify `createNote(caseId, noteData)` function exists — if not, add it (single API call)

Steps: 1. Add save icon button to assistant message rendering in ChatPanel.jsx 2. Build SaveNoteModal (or inline form) — 20–30 lines of JSX 3. Wire to existing notes API 4. Add success/error toast 5. Test: chat → save → open notes panel → verify note appears

---

P1.4 — Financial Dashboard: Bulk Categorization

Status: 🟡 Partial — backend complete, frontend UI missing **Effort:** 1 day

What Exists: - `POST /api/financial/batch-categorize` endpoint fully working in `/backend/routers/financial.py` - `selectedKeys` state array already exists in `/frontend/src/components/financial/FinancialTable.jsx`

What Needs to Change:

File: `/frontend/src/components/financial/FinancialTable.jsx` - Add checkbox column to each table row (bind to selectedKeys state) - Add "Select All" checkbox in table header - Add batch action toolbar (appears when >1 row selected): - Shows: "X transactions selected" - Category dropdown: [Income | Expense | Transfer | Legal Fee | Asset | Other] - "Apply to Selected" button - "Clear Selection" button - On "Apply": call `POST /api/financial/batch-categorize` with `{node_keys: selectedKeys, category: selectedCategory}` - Refresh table after success

Steps: 1. Add checkbox column to FinancialTable rows 2. Add selection state management (already partially exists) 3. Build BatchActionToolbar component (appears when selectedKeys.length > 0) 4. Wire to existing batch-categorize API 5. Add visual feedback (highlight selected rows, success toast) 6. Test bulk selection and categorization

---

P1.5 — Map: Fix and Remove Locations

Status: 🔴 Missing — editing not implemented **Effort:** 1.5 days

What Exists: - `/frontend/src/components/MapView.jsx` renders location markers using Leaflet - `/backend/services/neo4j_service.py` `get_entities_with_locations()` retrieves location data

What Needs to Change:

File: `/backend/services/neo4j_service.py` - Add `update_entity_location(node_key, case_id, lat, lng, name)` method - SET node.latitude, node.longitude, node.location_name - Add `remove_entity_location(node_key, case_id)` method - REMOVE node.latitude, node.longitude (node remains, just no map pin)

File: `/backend/routers/graph.py` - Add `PUT /api/graph/node/{key}/location` endpoint (update coordinates + name) - Add `DELETE /api/graph/node/{key}/location` endpoint (remove location data only)

File: `/frontend/src/components/MapView.jsx` - Add right-click context menu on markers: "Edit Location" | "Remove from Map" - Build `LocationEditorModal`: - Fields: Location Name,

Latitude, Longitude (or drag pin on mini-map) - Save / Cancel buttons - On "Remove": call DELETE endpoint, remove marker from local state immediately - On "Edit": call PUT endpoint, update marker position

Steps: 1. Add two methods to neo4j_service.py 2. Add two endpoints to graph.py router 3. Add right-click handler to Leaflet markers in MapView.jsx 4. Build LocationEditorModal component 5. Wire save/delete actions to API 6. Test: right-click marker → edit → confirm coordinates change on map 7. Test: right-click → remove → marker disappears, node still exists in graph

---

Priority 2 — Medium Features (3–7 days each)

P2.1 — Financial Dashboard: Correct the Numbers

Status: 🔴 Missing — amount is currently read-only **Effort:** 2 days

What Exists: - `update_transaction_details()` in neo4j_service.py saves purpose, notes, counterparty — but NOT amount - FinancialTable.jsx shows amount but has no edit control for it

What Needs to Change:

File: `/backend/services/neo4j_service.py` - Add `update_transaction_amount(node_key, case_id, new_amount, original_amount, correction_reason)` method - Store both original and corrected amount for audit trail - Add `amount_corrected: true`, `original_amount: X`, `correction_reason: "manual correction"` properties - SET node.amount = new_amount

File: `/backend/routers/financial.py` - Add `PUT /api/financial/transactions/{node_key}/amount` endpoint - Request body: `{new_amount: float, correction_reason: str}` - Validate: amount must be positive number - Returns: updated transaction

File: `/frontend/src/components/financial/FinancialTable.jsx` - Make amount cell clickable/editable (inline edit or pencil icon) - On click: show inline input field with current value - On save: call PUT endpoint - Show visual indicator (⚠️ or italic) on corrected amounts to distinguish from original - Tooltip: show original value and correction reason on hover

Audit Trail Consideration: Store `original_amount` and `correction_note` so investigators can see what was changed and why — important for legal defensibility.

Steps: 1. Add service method with audit fields 2. Add PUT endpoint with validation 3. Add inline amount editor to FinancialTable rows 4. Add visual indicator for corrected amounts 5. Add tooltip showing original value 6. Test: edit amount → verify graph updated → verify original preserved

---

P2.2 — Entity List with Summaries on Case Dashboard

Status: 🔴 Missing — needs full build **Effort:** 3 days

What Exists: - `CaseOverviewView.jsx` is the case dashboard component - Individual entity retrieval exists in graph.py and neo4j_service.py - Entity types in schema: Person, Company, Organisation, Bank, BankAccount, Location

What Needs to Change:

File: `/backend/services/neo4j_service.py` - Add `get_case_entity_summary(case_id)` method: `cypher MATCH (n {case_id: $case_id}) WHERE n:Person OR n:Company OR n:Organisation OR n:Bank OR n:BankAccount RETURN n.key, n.name, labels(n)[0] as type, n.summary, n.verified_facts, n.ai_insights, size(n.verified_facts) as facts_count, size(n.ai_insights) as insights_count ORDER BY type, n.name`

File: `/backend/routers/graph.py` - Add `GET /api/graph/cases/{case_id}/entity-summary` endpoint - Returns: list of entities with type, name, summary snippet, fact count, insight count

File: `/frontend/src/components/workspace/` — *new file:* `EntitySummarySection.jsx` - Tabbed or filtered list: All | People | Companies | Banks | Accounts | Organisations - Columns: Type (icon) | Name | Summary (first 100 chars) | Facts | Insights | Actions - Click row → open entity detail side panel (reuse existing EntityDetailPanel if exists) - Sort by: Type, Name, Facts Count - Search/filter bar at top - Count badges per tab: "People (12) | Companies (5) | Banks (3)"

File: `/frontend/src/components/workspace/CaseOverviewView.jsx` - Add EntitySummarySection below (or alongside) existing sections - Pass caseId as prop

Steps: 1. Build Neo4j query and service method 2. Add API endpoint 3. Build EntitySummarySection.jsx with tabs and table 4. Wire to API with loading states 5. Add click-through to entity detail 6. Integrate into CaseOverviewView

---

P2.3 — AI Chat: Analyze More Than 10 Documents

Status: 🟡 Partial — limit location needs verification, then config change **Effort:** 1–3 days (depending on where limit lives)

What Exists: - RAG service in `/backend/services/rag_service.py` - Vector DB service in `/backend/services/vector_db_service.py` - Document selection in ChatPanel.jsx

Investigation Needed First: Search `rag_service.py` and `vector_db_service.py` for: - `limit=10`, `max_docs`, `n_results=10`, `[:10]`, `top_k=10` - The token budget variable `CONTEXT_TOKEN_BUDGET` controls how much context fits in the LLM prompt

Likely Fix:

File: `/backend/services/rag_service.py` - Find where document/chunk retrieval is capped - Replace hardcoded `10` with configurable parameter - Add `max_documents: int = 50` parameter to `answer_question()` function - The real constraint is token budget — with GPT-4's 128K context window, 50+ documents are feasible if chunks are sized appropriately

File: `/backend/services/vector_db_service.py` - Check `n_results` parameter on ChromaDB queries — increase default

File: `/frontend/src/components/ChatPanel.jsx` - If document selector has a UI cap (checkbox list capped at 10), remove it - Allow selecting all documents or setting "Analyze All Documents" toggle

Token Budget Strategy: - With 50 docs × avg 2000 chars/doc = 100K chars ≈ 25K tokens - GPT-4 128K context: easily handles this - Local Ollama models (32K context): may need chunking strategy — retrieve fewer, more relevant chunks

Steps: 1. Search for limit in rag_service.py and vector_db_service.py 2. Make limit configurable via request parameter 3. Update frontend selector to allow more docs 4. Add "Analyze All Documents" toggle 5. Test with 20, 30, 50 docs — measure response quality and speed

---

P2.4 — Dashboard: Fix Case Files vs All Evidence Confusion

Status: ⚙️ Design + Implementation **Effort:** 3 days

Root Cause: The terms "Case Files" and "All Evidence" are used inconsistently. Users don't understand the distinction. From the codebase, both `DocumentsSection.jsx` and `AllEvidenceSection.jsx` exist but their relationship is unclear.

Proposed Architecture:

Redefine clearly: - **Case Documents** = files the user has uploaded and ingested (the working set) - **Evidence Items** = extracted entities, relationships, and facts derived from documents (the graph)

What Needs to Change:

Rename and clarify in UI: - "Case Files" → **"Uploaded Documents"** (the files you've added to the case) - "All Evidence" → **"Extracted Evidence"** or remove the section and replace with the Entity Summary (P2.2)

File: `/frontend/src/components/workspace/DocumentsSection.jsx` - Clear header: "Case Documents — Files uploaded and processed for this case" - Show: filename, upload date, processing status, page count - Actions: View | Re-process | Remove from case

File: `/frontend/src/components/workspace/AllEvidenceSection.jsx` - Either: Rename to "Extracted Evidence" and clarify it shows what was found *inside* the documents - Or: Merge into the Entity Summary section (P2.2) which is clearer - Add explanatory subtitle: "Entities and relationships extracted from your uploaded documents"

File: `/frontend/src/components/workspace/CaseOverviewView.jsx` - Reorder sections to create clear flow: 1. Case Summary 2. Uploaded Documents (what you've added) 3. Key Entities (what was found — new P2.2 section) 4. Recent Activity / Notes

Steps: 1. Audit exactly what each section shows (read both components carefully) 2. Rename labels in the UI (no backend changes required initially) 3. Add explanatory subtitles to each section 4. Reorder sections in CaseOverviewView for logical flow 5. User test: show to a non-technical attorney — do they now understand the distinction?

P2.5 — Export Transactions to PDF

Status: 🔴 Missing — new feature **Effort:** 2–3 days

What Exists: - `generate_docx_report.py` exists in ingestion/scripts (Word format, case-level) - `CaseExportModal.jsx` exists for case-level export - WeasyPrint is now installed (used for the market analysis PDF)

What Needs to Change:

File: `/backend/services/` *— new file:* `financial_export_service.py` - `generate_financial_pdf(case_id, transaction_keys=None, filters=None)` - Fetch transactions from Neo4j (same query as financial view) - Build HTML template with: - Header: Case name, export date, filter description - Summary: Total transactions, total value, date range, categories breakdown - Detail table: Date | From | To | Amount | Category | Purpose | Notes | Corrections - Flag corrected amounts with original value footnote - Sub-transaction groups shown with indented rows - Convert HTML → PDF using WeasyPrint - Return PDF bytes

File: `/backend/routers/financial.py` - Add `GET /api/financial/export/pdf` endpoint - Query params: `case_id`, `category` (optional), `entity_key` (optional), `date_from`, `date_to` - Returns: `application/pdf` file response with filename `transactions_{case_name}_{date}.pdf`

File: `/frontend/src/components/financial/FinancialView.jsx` - Add "Export PDF" button in the toolbar (near existing filters) - Pass current active filters to the export endpoint so the PDF matches what's on screen - Trigger file download

PDF Contents:

```
OWL INVESTIGATION PLATFORM
Financial Transaction Report
Case: [Case Name]
Generated: [Date]
Filters Applied: [Category / Entity / Date range]
──────────────────────────────────────────────
SUMMARY
Total Transactions: 47        Total Value: $4,231,500
Date Range: Jan 2022 – Dec 2024
Categories: Transfers (23) | Income (12) | Expenses (8) | Other (4)
──────────────────────────────────────────────
TRANSACTIONS
Date       | From           | To             | Amount    | Category | Notes
2024-01-15 | JOHN SMITH     | ACME LLC       | $150,000  | Transfer | Loan
payment
 └ [sub]   | ACME LLC       | BANK OF AMER   | $90,000   | Transfer | Principal
 └ [sub]   | ACME LLC       | FIRST NATL     | $60,000   | Transfer | Interest
...
──────────────────────────────────────────────
* Amounts marked † have been manually corrected. Original values available on
request.
```

Steps: 1. Create financial_export_service.py with HTML template 2. Add PDF export endpoint to financial.py router 3. Add Export button to FinancialView.jsx 4. Wire download (use blob URL approach in frontend) 5. Test with various filter combinations 6. Test sub-transaction grouping shows correctly in export

---

P2.6 — Insights System: Generate, Review, Accept/Reject

Status: 🟡 Partial — storage and verify endpoints exist, generation UI missing **Effort:** 4 days

What Exists: - `ai_insights` JSON array stored on entity nodes - `verify_insight()` endpoint in graph.py (converts insight → verified fact) - Merge modal handles insights selection

What Needs to Change:

File: `/backend/services/` *— add to* `llm_service.py` *or new* `insights_service.py` - `generate_entity_insights(entity_key, case_id)`: - Fetch entity data, verified facts, relationships, linked documents - Call LLM with prompt: "Based on this entity's data and relationships, generate 3–5 investigative insights. For each insight, provide: the insight text, your confidence (high/medium/low), and your reasoning." - Parse and return structured list - Store in entity's `ai_insights` array (with `status: "pending"`) - `generate_case_insights(case_id)`: - Run entity insights across all key entities - Also generate cross-entity insights (e.g., "Entity A and Entity B have 3 common financial connections")

File: `/backend/routers/graph.py` - Add `POST /api/graph/node/{key}/generate-insights` endpoint - Add `POST /api/graph/cases/{case_id}/generate-insights` (case-wide) - Add `DELETE /api/graph/node/{key}/insights/{insight_index}` (reject/delete insight) - Existing `verify_insight` endpoint handles acceptance

File: `/frontend/src/components/workspace/` *— new file:* `InsightsPanel.jsx` - Shows all pending insights across the case, grouped by entity - Each insight card shows: - Entity name and type - Insight text - Confidence badge (High / Medium / Low) - Reasoning (expandable) - ✅ Accept | ❌ Reject buttons - Bulk actions: "Accept All High Confidence" | "Reject All Low Confidence" - "Generate New Insights" button (runs case-wide generation)

File: `/frontend/src/components/workspace/CaseOverviewView.jsx` - Add "Insights" tab or section - Show count badge: "Insights (7 pending)"

Steps: 1. Build insight generation service method 2. Add generate and reject endpoints 3. Build InsightsPanel.jsx component 4. Add "Generate Insights" button on entity detail modal too (per-entity generation) 5. Integrate into CaseOverviewView 6. Test: generate → review → accept → verify stored as verified fact 7. Test: reject → verify removed from pending

---

Priority 3 — Complex Features (1–2 weeks each)

P3.1 — Sub-Transactions / Transaction Grouping

Status: 🔴 Missing — new data model + UI **Effort:** 5–7 days

What Needs to Change:

Neo4j Schema Change: - Add `PART_OF` relationship: `(childTx)-[:PART_OF]->(parentTx)` - Add `parent_transaction_key` property on child transaction nodes (for fast lookup) - Add `is_parent: true` property on parent transactions that have children

File: `/backend/services/neo4j_service.py` - Add `link_sub_transaction(parent_key, child_key, case_id)`: - Creates `PART_OF` relationship - Sets `parent_transaction_key` on child - Sets `is_parent: true` on parent - Add `unlink_sub_transaction(child_key, case_id)`: - Removes `PART_OF` relationship - Removes `parent_transaction_key` from child - Add `get_transaction_with_children(parent_key, case_id)`: - Returns parent + all children in one query

File: `/backend/routers/financial.py` - Add `POST /api/financial/transactions/{parent_key}/sub-transactions/{child_key}` — link - Add `DELETE /api/financial/transactions/{child_key}/parent` — unlink - Update `get_financial_transactions()` to return parent/child structure

File: `/frontend/src/components/financial/FinancialTable.jsx` - Parent transactions show as expandable rows with ▶ toggle - Children shown indented below parent when expanded - Parent row shows total amount, child rows show component amounts - Total should equal parent (show warning if components don't add up)

File: `/frontend/src/components/financial/` — *new:* `SubTransactionModal.jsx` - "Group Transactions" button in transaction row actions - Modal: search and select transactions to group under this parent - Shows: parent transaction at top, list of candidate children with checkboxes - Warns if child amounts don't sum to parent amount - Save creates all PART_OF relationships

Example Display:

```
▼ House Purchase — JOHN SMITH → SELLER        $1,300,000  [Transfer]
  └ Bank Loan — FIRST NATIONAL → SELLER         $900,000  [Loan]
  └ Gift Funds — PARENT → JOHN SMITH            $200,000  [Gift]
  └ Legal Fees — JOHN SMITH → LAW FIRM LLP      $200,000  [Legal Fee]
                                                ──────────
                                                $1,300,000  ✓
```

Steps: 1. Add PART_OF relationship to Neo4j schema documentation 2. Add 3 service methods (link, unlink, get-with-children) 3. Add 3 API endpoints 4. Update financial transactions query to return hierarchy 5. Add expandable rows to FinancialTable 6. Build SubTransactionModal 7. Add validation (amounts should sum to parent) 8. Test grouping, ungrouping, display, and PDF export with sub-transactions

---

P3.2 — Table View: Speed and Bulk Merge/Edit

Status: 🟡 Partial — merge exists, performance and bulk edit missing **Effort:** 5–7 days

Performance Issues: `GraphTableView.jsx` is 97KB — it likely renders all entities at once without virtualization.

File: `/frontend/src/components/GraphTableView.jsx`

Step 1 — Virtualized Rendering: - Install `react-window` or `react-virtual`: renders only visible rows - Replace `data.map(row => <TableRow>)` with `<FixedSizeList>` from react-window - Expected improvement: 10x faster render for 1000+ entity tables

Step 2 — Server-Side Pagination: - Add `page` and `page_size` params to graph entity endpoints - Load 100 rows at a time, fetch next page as user scrolls - Add total count to response for pagination display

Step 3 — Bulk Edit: *File:* `/backend/services/neo4j_service.py` - Add `batch_update_entities(updates: list[{key, properties}], case_id)`: - Single Cypher query using UNWIND for efficiency - Updates name, summary, type, and custom properties

File: `/backend/routers/graph.py` - Add `PUT /api/graph/batch-update` endpoint

File: `/frontend/src/components/GraphTableView.jsx` - Add checkbox column for row selection - Add batch action toolbar (when rows selected): - "Merge Selected" (already exists partially — wire to merge endpoint) - "Edit Property" — set same property on all selected entities - "Delete Selected" (with confirmation) - "Export Selected"

Step 4 — Optimized Queries: - Verify Neo4j indexes exist on `case_id` and `key` fields - Add composite index: `CREATE INDEX entity_case_key FOR (n) ON (n.case_id, n.key)` if missing

Steps: 1. Install react-window 2. Implement virtualized list in GraphTableView 3. Add pagination to backend entity queries 4. Add batch_update_entities service + endpoint 5. Add checkbox selection to table rows 6. Build batch action toolbar 7. Performance test: measure render time before/after with 500, 1000, 5000 entities

---

P3.3 — Filter by Entity in Financial Dashboard

Status: 🟢 Near Complete — client-side works, optimize server-side **Effort:** 0.5 days (frontend already works) or 2 days (add server-side filtering)

What Exists: - `entityFilter` state in FinancialView.jsx - FinancialFilterPanel.jsx with entity selector - Client-side filtering already works — transactions filtered by selected entity

What May Need Improving: - If the case has thousands of transactions, fetching all then filtering client-side is slow - Add `entity_key` query param to `GET /api/financial` endpoint for server-side filtering

File: `/backend/routers/financial.py` - Add `entity_key: Optional[str] = None` query param to `get_financial_transactions()` - Pass to neo4j_service

File: `/backend/services/neo4j_service.py` - Add WHERE clause: `AND (n.from_key = $entity_key OR n.to_key = $entity_key)`

This is low risk since the frontend filter already works — backend filter is an optimization.

Steps: 1. Verify FinancialFilterPanel is visible and functional in current UI 2. If UI works: done (this is already built) 3. If performance is an issue: add server-side entity_key filter 4. Test with entity selected — verify inflow/outflow totals update correctly

---

Implementation Roadmap

Sprint 1 — Quick Wins (Week 1–2)

Item	Effort	Assignee
P1.1 Merge tab document z-index fix	0.5d	Frontend
P1.3 Save chat as notes	0.5d	Frontend
P1.4 Bulk categorization UI	1d	Frontend
P1.2 Reduce entity noise (prompt tuning)	1d	Backend/AI
P1.5 Map location edit/remove	1.5d	Full stack
P3.3 Entity filter (verify/optimize)	0.5d	Full stack

Sprint 1 Total: ~5 days

Sprint 2 — High Impact Medium Features (Week 3–5)

Item	Effort	Assignee
P2.1 Correct transaction amounts	2d	Full stack
P2.4 Fix case files vs evidence confusion	3d	Full stack
P2.3 AI chat more than 10 docs	1–3d	Backend
P2.5 Export transactions to PDF	2–3d	Full stack

Sprint 2 Total: ~10 days

Sprint 3 — New Systems (Week 6–9)

Item	Effort	Assignee
P2.2 Entity summary on case dashboard	3d	Full stack
P2.6 Insights system	4d	Full stack
P3.1 Sub-transactions grouping	5–7d	Full stack
P3.2 Table performance + bulk edit	5–7d	Full stack

Sprint 3 Total: ~18 days

Risk Register

Risk	Likelihood	Impact	Mitigation
Sub-transaction schema change breaks existing graph queries	Medium	High	Run migration script; test all financial queries before/after
Reducing entity noise changes existing case graphs	High	Medium	Apply to new ingestions only; add "re-process with new settings" option
AI chat with 50 docs exceeds Ollama context window	High	Medium	Add smart truncation; use GPT-4 for large-context queries
Table virtualization breaks existing sort/filter behavior	Medium	Medium	Test comprehensively; keep fallback to current implementation
PDF export legal admissibility concerns	Low	High	Add "Generated by Owl" watermark + timestamp; preserve original values

---

Open Questions for Product Owner

1. **Sub-transactions:** Should linking transactions be permanent or easily reversible? (affects how aggressively we allow editing)
2. **Amount corrections:** Should corrected amounts be visible to the other side in a matter? (implications for evidence integrity)
3. **Entity noise reduction:** Should we re-process existing cases with new extraction settings, or only apply to future ingestions?
4. **Insights:** Should AI-generated insights be automatically generated on document upload, or only on-demand?

5. **Table bulk delete:** Should bulk delete of entities be allowed, or only merge? (risk of accidental data loss)

This plan was generated from a full codebase analysis of the Owl platform. File paths, function names, and line numbers are based on the current state of the repository.