**Hackathon Problem Statement**

**🧠 "Smart Mortgage File Explorer: Cross-Document Semantic Search and Relationship Navigation for Home Loan Packages"**

**🏦 Business Context**

Mortgage operations in a bank involve reviewing **hundreds of pages per loan file**, containing documents like:

* Loan applications (1003)
* Credit reports
* Income verification (W-2s, tax returns, paystubs)
* Appraisals
* Disclosures and closing statements

Traditionally, users rely on:

* **Manual search** by file names (often inconsistent or missing)
* **Rigid tagging rules**, which don’t scale or adapt to unstructured data
* **Limited full-text search**, which doesn’t understand *what* you’re really asking

This leads to slow processes in:

* Underwriting and document validation
* Audit and compliance checks
* Post-close quality control
* Servicing and investor reviews

**🚨 The Challenge**

The bank’s IT team is tasked with **building smarter document intelligence tools** to help operations teams **quickly find answers buried across a loan package**, such as:

“Where is the borrower’s latest income verification document?”  
“What’s the declared income on the 1003, and is it supported by attached paystubs or W-2s?”  
“Which loans have inconsistencies between borrower names on different forms?”  
“Show me loans with mismatched property addresses across documents.”

Right now, there is **no way to ask these questions semantically** — and no easy way to **trace relationships across documents** in a unified way.

**💡 Hackathon Use Case**

Build a prototype system that enables **semantic search and relationship navigation** across mortgage loan documents.

**Core Capabilities:**

* Use an **LLM** to extract **entities** (borrowers, addresses, loan amounts, employers, dates) and **relationships** (e.g., “John Smith is borrower on loan #123”, “is employed by ACME Corp”).
* Store this in a **graph database** (like Neo4j) or a structured vector-based store.
* Enable **semantic search** over the documents:
  + Natural language queries like:
    - “Show income documents for the primary borrower”
    - “Find the appraisal report for 123 Main St.”
    - “Which documents mention co-borrower Jane Doe?”
    - “Are there any loans with missing employment verification?”
* Use **retrieval-augmented generation (RAG)** or vector search (e.g., FAISS or OpenAI embeddings) to map queries to the right document sections or nodes in the graph.

**🧑‍💻 Why This Is an IT Hackathon Challenge**

This use case cuts across:

* **NLP & AI** (LLM-based extraction + semantic understanding)
* **Search & IR** (embedding-based semantic retrieval)
* **Data architecture** (entity graph or index)
* **Document pipelines** (OCR, PDF parsing, etc.)
* **Risk & compliance** (validations and audit traceability)

It’s highly practical, IT-aligned, and impactful for core mortgage operations.

**✅ Success Criteria for the Hackathon Prototype**

* Load 10–20 example mortgage documents (can be anonymized or synthetic)
* Enable **natural language questions** that return relevant document snippets or answers
* Visualize the **relationships** across entities (optional: graph UI)
* Demonstrate 3–5 valuable real-world use cases (e.g., cross-check income, validate addresses, find missing docs)

**🔐 Real-World Payoff**

This solution supports:

* Faster underwriting & QC
* Fewer human errors in audits
* Stronger compliance with investor and CFPB guidelines
* A foundation for automated red-flag detection (e.g., fraud, data mismatches)

Let me know if you’d like:

* A synthetic mortgage document set
* Suggested entity schema or graph structure
* Example prompts or semantic queries
* Prebuilt pipeline suggestions (e.g., LlamaIndex, LangChain, OpenAI tools)

This would make for a **high-impact hackathon challenge** that also aligns with strategic tech modernization goals at a bank.

**How entity extraction helps?**

**Why Entity & Relationship Extraction Matters**

While document tagging or full-text search helps with *finding documents*, **entity and relationship extraction helps you understand and reason over the information inside them**.

By converting unstructured documents into structured knowledge — with **entities** (people, places, amounts, dates, etc.) and **relationships** (e.g., "is borrower on", "is employer of", "is subject property of") — you can answer complex questions that require **cross-document understanding**, **validation**, and **inference**.

**🔍 Examples of Questions You Can Answer**

Here’s a list of **high-value questions** that become answerable once entity and relationship extraction is in place — especially when backed by a graph or structured index.

**🧾 Borrower & Co-Borrower Validation**

* **“Who are the borrowers and co-borrowers on this loan?”**
* **“Are the names and SSNs consistent across all forms (1003, W-2, disclosures)?”**
* **“Does the co-borrower live at the same address as the borrower?”**

💡 *Helps detect data entry errors, fraud, or identity mismatches.*

**💼 Income Verification**

* **“What is the declared income on the loan application, and what supporting documents back it up?”**
* **“Is the income supported by multiple years of W-2s or paystubs?”**
* **“Does the stated employer match what’s on the paystub or VOE (Verification of Employment)?”**

💡 *Supports underwriting consistency and income fraud detection.*

**🏠 Property & Appraisal Cross-Checks**

* **“What is the subject property address across all documents?”**
* **“Are the addresses consistent between the appraisal, title report, and 1003 form?”**
* **“Which appraiser and appraisal company were used for this property?”**

💡 *Enables audit trails, detects discrepancies, and flags risk.*

**💵 Loan Terms & Disclosure Matching**

* **“What is the loan amount and interest rate on the 1003, and does it match the Closing Disclosure?”**
* **“Are all required disclosures present and consistent (e.g., LE, CD, GFE)?”**
* **“What fees were disclosed, and are they within acceptable tolerance?”**

💡 *Supports TRID compliance and investor eligibility.*

**📅 Timeline Consistency & Regulatory Compliance**

* **“Was the Closing Disclosure delivered at least 3 business days before closing?”**
* **“Are there date mismatches between documents (e.g., outdated W-2 or appraisal)?”**
* **“When were income documents generated vs. when the application was filed?”**

💡 *Helps meet CFPB and investor audit requirements.*

**🔄 Repeat Entity or Pattern Detection (Advanced)**

* **“Are any borrowers involved in multiple loans across packages?”**
* **“Is the same appraiser used across multiple questionable loans?”**
* **“Does this employer appear repeatedly across loans with overstated incomes?”**

💡 *Supports pattern detection, collusion risk, and internal fraud investigation.*

**🔗 Why It’s Useful — Beyond Just Search**

| **Benefit** | **Description** |
| --- | --- |
| 🔍 **Cross-document validation** | Connects related information across forms (e.g., borrower name + employer + income) |
| 🧠 **Data normalization** | Resolves variations like “John A. Smith” = “J. Smith” |
| 🚨 **Risk and anomaly detection** | Flags missing links, mismatches, or suspicious relationships |
| 📊 **Structured analytics** | Enables dashboards and metrics on loan quality, document completeness, etc. |
| ✅ **Audit and compliance** | Creates a traceable, queryable audit trail for regulators and investors |
| 📈 **Automation foundation** | Powers downstream automation like automated checklists, QC, fraud detection |

**💡 Real-World Analogy**

Think of this like going from:

“Here are 300 PDFs in a folder, good luck finding what you need…”

To:

“Here’s a structured graph of everything in this loan: borrowers, income, dates, property, disclosures, timelines — with alerts where things don’t match.”

That’s the power of entity and relationship extraction.

**Example output graph from real world mortgage documents**

Here's an example of what an **output knowledge graph** might look like based on information extracted from a **real mortgage loan package**. This graph connects key **entities** (people, places, documents, amounts) with **relationships** that reflect how they are linked across multiple documents in the loan file.

Since I can’t show visuals directly here without image generation, I’ll represent the graph structure textually, using nodes and edges.

**🧠 Example: Mortgage Knowledge Graph (Text Representation)**

css

CopyEdit

[Loan Application #12345]

├── hasBorrower → [John Smith]

├── hasCoBorrower → [Jane Smith]

├── hasSubjectProperty → [123 Main St, Springfield, IL]

├── hasLoanAmount → [$350,000]

├── hasLoanType → [Conventional Purchase]

├── declaredIncome → [$85,000]

[John Smith]

├── hasSSN → [XXX-XX-1234]

├── employedBy → [ACME Corp]

├── livesAt → [123 Main St, Springfield, IL]

├── appearsIn → [1003 Form], [W-2 2023], [Paystub Jan 2024]

[Jane Smith]

├── hasSSN → [XXX-XX-5678]

├── employedBy → [Global Health Inc.]

├── livesAt → [123 Main St, Springfield, IL]

├── appearsIn → [1003 Form], [W-2 2023]

[ACME Corp]

└── appearsIn → [VOE Letter, Paystub]

[Global Health Inc.]

└── appearsIn → [W-2, Paystub]

[123 Main St, Springfield, IL]

├── appraisedBy → [Premier Appraisal Inc.]

├── appraisedValue → [$360,000]

├── appearsIn → [Appraisal Report], [1003 Form], [Title Commitment]

[Closing Disclosure]

├── loanAmount → [$350,000]

├── interestRate → [6.25%]

├── closingDate → [Feb 5, 2024]

├── disclosedTo → [John Smith], [Jane Smith]

├── disclosedDate → [Feb 1, 2024]

[Appraisal Report]

├── appraiser → [Premier Appraisal Inc.]

├── date → [Jan 22, 2024]

├── appraisedValue → [$360,000]

├── property → [123 Main St, Springfield, IL]

[1003 Form]

├── borrower → [John Smith]

├── coBorrower → [Jane Smith]

├── statedIncome → [$85,000]

├── statedEmployment → [ACME Corp], [Global Health Inc.]

├── property → [123 Main St, Springfield, IL]

**🔎 What You Can Do with This Graph**

**🔍 Run Smart Queries**

* **"Show all documents that mention John Smith."**  
  → Traverse John Smith → appearsIn
* **"Is the appraised value greater than the loan amount?"**  
  → Compare appraisedValue vs. loanAmount
* **"Find inconsistencies between stated income and income on documents."**  
  → Compare declaredIncome in 1003 with income parsed from W-2 and paystub
* **"Did Jane Smith receive the required disclosures?"**  
  → Check disclosedTo relationship on Closing Disclosure
* **"Which loans have mismatched borrower addresses?"**  
  → Compare livesAt across documents

**🧠 Enable Automated Checks**

* ✅ Employment consistency between 1003, paystubs, VOE
* ✅ Address consistency across application, title, appraisal
* ✅ Check if closing disclosure was issued 3+ days before closing
* ✅ Confirm borrower name and SSN are consistent across all forms

**🔐 Real-World Implementation Notes**

* You can use **Neo4j** to model this graph (nodes = entities; edges = relationships).
* Use an **LLM** (e.g., GPT-4) to extract structured facts from raw document text.
* Store document references as edge properties: source: W-2.pdf, confidence: 0.95
* Enhance with **document embeddings** to enable fallback **semantic search**.

**Sample visual image**

A network of dots and lines

AI-generated content may be incorrect.

Here is a **visual knowledge graph** representing the entities and relationships extracted from a mortgage loan package. Each node type (borrower, document, organization, amount, property, etc.) is color-coded, and edges describe relationships like hasBorrower, appearsIn, loanAmount, etc.

This structure enables:

* Semantic queries over the loan file
* Cross-document consistency checks
* Clear audit trails and visualization of how data points connect

**What semantic search alone can do?**

understanding what **semantic search alone** can do (without building a full knowledge graph) is key for evaluating tradeoffs in a system's complexity and functionality.

**🔍 What Can Be Answered with Just Semantic Search?**

Semantic search uses **vector embeddings** of document content to match **natural language queries** to relevant **passages** (sentences or paragraphs) — even if the exact words aren't present.

It allows you to ask **open-ended or fuzzy questions** over unstructured text and retrieve **relevant snippets** or documents.

**✅ Examples of Questions Answered by Semantic Search (No Graph Needed)**

**📄 Document-Level Questions**

* “Where is the borrower’s income stated in the loan file?”
* “What is the interest rate for this loan?”
* “Is there a W-2 for John Smith in this package?”
* “Which document contains the employment details of Jane Smith?”
* “Where is the property address mentioned?”

**📑 Form Presence and Completeness**

* “Does this file contain a 1003 form?”
* “Is there a closing disclosure in this loan package?”
* “Is the appraisal report available?”
* “What disclosures are included in this loan file?”

**🔍 Factual Retrieval**

* “What is the loan amount?”
* “Who is the co-borrower?”
* “What company does the borrower work for?”
* “What is the appraised value of the property?”
* “What are the key dates mentioned in the closing disclosure?”

**📅 Timeline and Compliance Checks (basic)**

* “When was the closing disclosure issued?”
* “Is the document dated before the closing date?”
* “What is the closing date for this loan?”

**🧾 Supporting Document Retrieval**

* “Show me the income documents for the borrower.”
* “Find supporting documents for borrower’s declared income.”
* “What documents list the borrower’s employer?”

**🗂️ Search Across Variants or Ambiguities**

* “Find any mention of John A. Smith or J. Smith.”
* “Locate documents that discuss employment verification.”
* “Is there any mention of the borrower’s address in this loan file?”

**❗️What Semantic Search Can’t Do Well (Alone)**

Without structuring the data (like in a graph), semantic search **struggles with**:

| **Use Case** | **Why It Struggles** |
| --- | --- |
| **Cross-document consistency checks** | Needs linking info across multiple docs (e.g., compare stated income vs. actual income in W-2) |
| **Data validation rules** | Can't infer logic like “is income within ±10% of paystub?” |
| **Deduplication & entity resolution** | Hard to know if “John A. Smith” and “J. Smith” are same person without structured entity mapping |
| **Anomaly detection** | Cannot find structural gaps (e.g., “no appraisal linked to property X”) |
| **Chaining multi-hop relationships** | Hard to trace “Borrower → Employer → Company in 3 loans” relationships |

**🔁 Ideal Use of Semantic Search in a Hackathon**

In a hackathon, semantic search is great for:

* Rapid Q&A over loan packages
* Extracting and summarizing document chunks
* Building a chat-based interface (“Ask your loan file anything”)
* Combining with RAG (retrieval-augmented generation) for LLMs

**🧠 Bonus: How to Implement It**

* Embed all documents or paragraphs using **OpenAI**, **Cohere**, or **SentenceTransformers**
* Use **FAISS**, **Weaviate**, or **Pinecone** for vector storage
* Accept natural language queries → return top relevant chunks
* Optionally plug into **LLM (RAG)** for answer synthesis