**Intelligent Complaint Analysis for Financial Services**

**Interim Submission Report**

**Date:** July 6, 2025  
**Name:** Bisrat Haile

**📌 1. Project Overview and Understanding**

This project aims to build an internal AI assistant for **CrediTrust Financial**, enabling teams to analyze customer complaints quickly and effectively. CrediTrust receives thousands of complaints across five key financial products:

* Credit Cards
* Personal Loans
* Buy Now, Pay Later (BNPL)
* Savings Accounts
* Money Transfers

The goal is to create a **Retrieval-Augmented Generation (RAG)** chatbot that allows stakeholders to ask plain-language questions (e.g., “Why are people unhappy with BNPL?”), and get insightful, evidence-based answers grounded in real customer narratives.

The system will:

* Embed complaint narratives into vector space
* Use semantic search to retrieve relevant complaints
* Use an LLM to synthesize answers from retrieved complaints

**⚙️ 2. Methodology – Process, Progress So Far, and Results**

**✅ Task 1: EDA and Preprocessing – *Completed***

* Loaded CFPB complaint dataset (~1M rows).
* Filtered to the five required financial products.
* Removed rows without complaint narratives.
* Cleaned text (lowercasing, special character removal, boilerplate stripping).
* Visualized word count distributions to assess text length.
* Saved processed data to:  
  data/filtered\_complaints.csv

📊 *Key Observations:*

* BNPL and Personal Loan complaints are increasing in volume.
* Complaint lengths vary widely: some under 20 words, others over 200.
* Many narratives use indirect or emotional language, which supports using semantic embedding.

**✅ Task 2: Chunking, Embedding, and Vector Store Indexing – *Completed***

* Split each cleaned narrative into overlapping chunks using RecursiveCharacterTextSplitter.
  + Chunk size: 300 characters
  + Overlap: 50 characters
* Chose embedding model: **all-MiniLM-L6-v2** from SentenceTransformers for its balance of speed and semantic accuracy.
* Used **ChromaDB** as the vector store.
  + Stored embeddings + metadata (product, complaint\_id) in batches to avoid max size errors.
* Saved embeddings to:  
  vector\_store/

🚀 *Current Status:*  
Over 400,000 complaint chunks have been embedded and indexed successfully.

## ⚠️ 3. Challenges & Solutions (Pointed)

**Challenge 1: ChromaDB batch size limit**

**Problem:** Tried to insert 400,000+ chunks at once — received an error: *“Batch size greater than max batch size of 5461.”*  
**Solution:** Implemented batching (5,000 chunks per batch) with a loop and tqdm for progress tracking.

**Challenge 2: Deprecated ChromaDB client setup**

**Problem:** Used old Client(Settings(...)), which caused InternalError.  
**Solution:** Switched to the new API using PersistentClient(path=...) as per updated docs.

**Challenge 3: Noisy complaint narratives**

**Problem:** Many narratives had boilerplate text like “I am writing to file a complaint…” and special characters.  
**Solution:** Cleaned text by lowercasing, removing boilerplate, and stripping special characters.

**Challenge 4: Missing narratives in dataset**

**Problem:** Some rows had no complaint narrative at all.  
**Solution:** Filtered dataset to drop rows with null or empty consumer\_complaint\_narrative.

**Challenge 5: Inconsistent narrative length**

**Problem:** Some narratives were too short, others too long — causing poor embedding consistency.  
**Solution:** Used LangChain’s RecursiveCharacterTextSplitter to chunk long text into consistent, overlapping segments.

**Challenge 6: Embedding model trade-off**

**Problem:** Large models were slow; small ones missed context.  
**Solution:** Chose all-MiniLM-L6-v2 — fast, accurate, and compatible with ChromaDB.

**Challenge 7: Memory bottlenecks during embedding**

**Problem:** Embedding large datasets at once caused slowdowns and high memory use.  
**Solution:** Batched embedding + avoided storing intermediate results in memory-heavy formats.

**📅 4. Future Plan – What’s Left and How I Plan to Finish**

**🔜 Task 3: Build RAG Core Logic and Evaluation**

* Write a retriever module to take in user queries → embed → retrieve top-k complaint chunks.
* Design a custom LLM prompt using retrieved chunks as context.
* Use Mistral or Hugging Face pipeline to generate responses.
* Create an evaluation framework:
  + Prepare 5–10 benchmark questions.
  + Score answers on quality, faithfulness, relevance.

**🔜 Task 4: Build Interactive Chat Interface**

* Use **Gradio** or **Streamlit** to create a user-facing app.
* Must support:
  + Text input
  + AI answer display
  + Retrieved sources display
  + Clear/reset functionality

**✅ 5. Conclusion – Summary of Progress and Confidence**

* I’ve completed all foundational tasks: preprocessing, chunking, embeddings, and vector indexing.
* Major technical challenges have been solved, and the project is well-structured with CI-ready code and a consistent project layout.
* I’m confident in building the RAG core and UI within the next two days.
* My understanding of vector databases, LLM context design, and ML workflows has deepened significantly during this challenge.

**🔄 6. Moving Forward**

* Begin building the retriever and generator modules tomorrow.
* Use dummy prompts and test queries to validate early output quality.
* Set up the interactive app interface by Monday night.
* Polish documentation, prepare screenshots, and submit by the final deadline: **July 8, 2025**.