Technical Implementation Draft: Agentic FinTech Ecosystem (Lending & Wealth)

1. Project Overview

The **Agentic FinTech Ecosystem** is an end-to-end financial intelligence platform designed to automate wealth management and credit risk assessment. It bridges the gap between raw, unstructured financial data and actionable investment strategies using local LLMs, explainable AI (XAI), and mathematical optimization.

Core Tech Stack:

- **Languages:** Python, SQL
- **Data Engineering:** Apache Airflow, Delta Lake/Parquet
- **Generative AI:** Llama 3.2 (via Ollama), LangChain, CrewAI
- **Machine Learning:** XGBoost, Scikit-learn, SHAP, Prophet
- **Vector Database:** ChromaDB
- **Optimization:** Dynamic Programming (Knapsack variation), Monte Carlo Simulation
- **Deployment:** Docker, FastAPI, Streamlit
- **Hardware Target:** RTX 3050 6GB (Local LLM Inference), 16GB RAM

---

2. Secure Financial ETL & Data Lake (Stage 0)

2.1 Data Governance & PII Masking

Given the sensitivity of financial data, a security-first ingestion layer is implemented using **Apache Airflow**.

- **PII Masking Pipeline**: A Python-based pre-processor uses Regex and Named Entity Recognition (NER) to identify and encrypt Account_Numbers, Phone_Numbers, and Names before storage.
- **Multi-Source Ingestion**:
    - **Transaction Stream**: Mock bank JSON data.
    - **Credit Documents**: Unstructured PDF reports (processed via PyMuPDF).
    - **Market Data**: Real-time stock/bond prices via external APIs.

2.2 Storage Optimization (Delta Lake)

Data is stored in **Parquet** format, partitioned by User_ID and Month. This architecture reduces the storage footprint by 80% compared to CSV and enables high-speed predicate pushdown for ML training.

---

3. Expense Intelligence & LLM Parsing (Stage 1)

3.1 Local LLM Categorization (NLP)

To handle messy transaction strings (e.g., "ZOMATO*123-NEW-DELHI"), the system utilizes **Llama 3.2** running locally on the **RTX 3050 GPU**.

- **Prompt Engineering**: A structured prompt instructs the LLM to output JSON: {"Merchant": "Zomato", "Category": "Food", "Location": "New Delhi"}.
- **Inference Engine**: Ollama handles the model quantization to ensure it fits within the 6GB VRAM constraint.

3.2 Predictive Budgeting

- **Time-Series Forecasting**: **Prophet** is used to model seasonality in spending. It predicts the next month's total expenditure based on the previous 12 months.
- **User Clustering**: **K-Means Clustering** groups users into personas (e.g., "The Frugal Saver") to provide peer-benchmarked financial advice.

---

4. Explainable Credit Risk & Fraud Engine (Stage 2)

4.1 Supervised ML (XGBoost)

A binary classifier is trained to predict the **Probability of Default (PD)**.

- **Imbalanced Learning**: Since loan defaults are a minority class, **SMOTE (Synthetic Minority Over-sampling Technique)** is applied to balance the training set.
- **Feature Set**: Includes Debt-to-Income ratio, Payment History, and Categorized Spending from Stage 1.

4.2 Explainable AI (SHAP)

To ensure regulatory compliance and transparency, **SHAP (SHapley Additive exPlanations)** is integrated.

- **Local Explanation**: For every rejected loan, the system generates a human-readable reason (e.g., "High Credit Utilization contributed +0.40 to the risk score").
- **Global Feature Importance**: Visualizes which variables (e.g., spending on "Destroyers") most influence the model's decisions.

---

5. Wealth Optimization & Portfolio Balancing (Stage 3)

5.1 Monte Carlo Simulation

The system runs 1,000 simulations of market returns to estimate the probability of reaching a financial goal (e.g., 100,000 USD). $P(Goal) = \frac{\sum[Final\_Value \geq Target]}{Total\_Simulations}$

5.2 Portfolio Optimization (DSA Integration)

The asset allocation problem is modeled as a **0/1 Knapsack Problem** solved via **Dynamic Programming (DP)**.

- **Constraint**: Total Capital + Risk Tolerance.
- **Value**: Expected Return per Asset.
- **Algorithm**: The DP table finds the optimal weights for Stocks, Bonds, and Crypto to maximize returns while strictly staying under the user's defined risk cap.

---

6. The Autonomous Wealth Agent (Stage 4)

6.1 Multi-Agent Orchestration (CrewAI)

A "Council of Financial Agents" coordinates to provide holistic advice:

1. **The Auditor Agent**: Analyzes spending patterns from Stage 1.
2. **The Risk Officer Agent**: Evaluates the XGBoost/SHAP output from Stage 2.
3. **The Strategist Agent**: Interprets the DP-optimized portfolio from Stage 3.

6.2 Retrieval-Augmented Generation (RAG)

- **Vector DB**: **ChromaDB** stores indexed tax laws and investment news.
- **Contextual Retrieval**: When a user asks about tax-saving investments, the agent retrieves relevant sections of the tax code and cross-references them with the user's current income bracket.

---

7. Production Deployment & HUD (Stage 5)

7.1 Streamlit Financial HUD

- **Spending HUD**: Interactive Plotly charts showing month-over-month expense trends.
- **Loan Simulator**: A "What-If" tool where users adjust loan amounts, and the XGBoost model recalculates approval odds in real-time.
- **Agent Chat**: A conversational interface for the Autonomous Wealth Agent.

7.2 MLOps & Containerization

- **Dockerization**: The system is partitioned into containers for the PostgreSQL DB, the FastAPI backend, and the Streamlit frontend.
- **Resource Allocation**: Logic is pinned to the CPU, while LLM inference is routed to the RTX 3050 via NVIDIA Container Toolkit.

---

8. Portfolio Summary

"I built an **Agentic FinTech Ecosystem** that manages the entire wealth lifecycle. The system uses an Airflow-based pipeline to securely ingest and mask financial data. It leverages local LLMs (Llama 3.2) to parse messy bank statements and XGBoost with SHAP to provide explainable credit risk assessments. The core optimization engine uses Dynamic Programming to balance investment portfolios. The project culminates in an Autonomous AI Agent that synthesizes these insights to provide natural language financial coaching, bridging the gap between complex data and user action."