

Project Report Template

Title: “AI-Assisted Loan Evaluation Dashboard Using Multi-Agent Financial Reasoning”

1. (5pts) Honor Code and LLM Usage for this Report.

We affirm that this final report adheres to the institutional honor code. All design decisions, coding, and system implementation were performed by us. Large Language Models (LLMs) were used ethically for support in debugging, refining documentation, generating design ideas, and improving explanations. No part of the codebase or analysis was copied from external proprietary systems. We remain fully responsible for the originality and integrity of our final product.

2. (10pts) Learning Objectives:

List the learning objectives from your proposal. In your own words explain whether you met those objectives and how (50-100 words each objective). Also describe if you learned something different than expected or anything additional.

Learning Objective 1 – Conceptual Understanding:

Original objective from proposal: “Understand how multi-agent systems can emulate decision-making in financial processes.”

We successfully met this objective by implementing a complete multi-agent pipeline consisting of a Data Agent, Risk Agent, Compliance Agent, Decision Agent, and an Explanation Agent. Each agent contributes a different layer of logic, creating a realistic simulation of how banks evaluate loan applications. We learned how rule-based logic differs from statistical credit models and how multi-agent architectures can create transparency by separating responsibilities. Additionally, we learned more than expected about designing explainable outputs and risk breakdowns that resemble real financial assessments.

Learning Objective 2 – Skill Development:

Original objective from proposal: “Build an interactive web application integrating Python, APIs, and a frontend UI.”

We fully achieved this objective by developing a complete web app built with HTML/CSS/JavaScript for the frontend and Python/FastAPI for the backend. We implemented API communication, data validation, and UI/UX design principles to create a professional banking-style interface. We also integrated multiple dynamic charts (using Chart.js) to generate visual financial analytics. Beyond our initial expectations, we improved our skills in front end responsiveness, data visualization,

API structuring, and implementing collapsible cards, traffic-light indicators, and a risk gauge for real-time analytics.

3. (10pts) Timeline:

Outline how you spent time on your project. Break down the time into specific tasks or milestones. Here is an adjustable schedule to get you started. Actual Details should be 50-100 words each and should compare or reflect on differences from your proposal.

| Time | Task | Expected Details from Proposal | Actual Details |
|----------|-------------------------------|--|--|
| Hour 1-2 | Research and gather resources | Research criteria used in real loan evaluations and select technologies. | We researched lending criteria such as debt-to-income ratio, credit score ranges, income stability, loan-to-income ratios, and common compliance checks used by banks. We also studied how rule-based systems and agent-based architectures function. We finalized our tech stack: HTML/CSS/Javascript for the frontend, FastAPI for the backend, and Chart.js for data visualizations. We additionally researched UI patterns from financial institutions to design premium banking experience. |

| | | | |
|----------|--|--|--|
| Hour 3-4 | Design the project structure and plan | Define architecture, agents, and division of responsibilities. | <p>We designed five-agent architecture—Data, Risk, Compliance, Decision, and Explanation Agents. We planned how data would flow from the frontend to the backend and back. We structured the project into clearly separated frontend and backend folders. Responsibilities were divided: one teammate focused more on backend logic while the other improved frontend UI and visual analytics. We also planned the enhanced dashboard, collapsible cards, and colors indicators.</p> |
| Hour 5-6 | Start coding the basic functionalities | Build backend and first agent, basic UI, and connect API. | <p>We built a fully operational FastAPI backend capable of receiving user data, validating it, and performing baseline computations like DTI and free income. We implemented the Data Agent and Risk Agent first. Simultaneously, we built an initial HTML layout and connected it to the backend using JavaScript fetch calls. The system was able to return a basic loan approval or rejection response by the end of this phase.</p> |

| | | | |
|------------|---|---|--|
| Hour 7-8 | Test and debug the initial version | Debug API flow and UI issues. | We performed iterative testing by submitting different loan profiles to identify errors in the formulas and input validation. We resolved backend 422 errors, fixed JSON parsing issues, and improved error messages. We also stabilized the front-end-to-backend communication and added protective checks so the dashboard wouldn't break if fields were empty. This phase ensured that the system produced consistent, accurate results. |
| Hour 9-10 | Refine and add advanced features | Improve interface, add explanations and risk scoring. | We implemented all advanced features: collapsible cards, traffic-light indicators, six analytical charts, a professional risk gauge, executive summary bullets, and deep narrative explanations. We created a premium black-and-gold UI styled like a bank application. We significantly expanded the Explanation Agent to produce multi-paragraph analysis, recommendations, conclusions, and personalized interest-rate suggestions. This transformation elevated the project beyond expectations and resulted in a polished, near-professional decision-support tool. |
| Additional | We spent extra time refining UI aesthetics, adding golden animations, creating a centered and compact layout, simplifying number inputs, and designing a fully responsive frontend. These improvements were beyond the original scope but greatly improved the final user experience. | | |

4. (55pts) Final Product Description:

Include your proposed MVP, Target, and Reach versions.

i. Minimum Viable Product (MVP):

A functional rule-based loan evaluator that accepts user financial inputs and outputs a loan approval or rejection result determined by a single agent. Includes basic input handling, one endpoint, and a simple UI to display the decision.

ii. Target Product:

A multi-agent system consisting of Data, Risk, Compliance, and Decision Agents, each handling separate evaluation steps. Includes explanations, cleaner UI, API-driven logic, and structured decision-making that resembles real financial workflows.

iii. Reach Version:

A full financial dashboard with multi-agent reasoning, deep analysis, risk narratives, interest-rate suggestions, traffic-light indicators, six interactive charts, collapsible report sections, premium UI, and dynamic affordability and DTI simulations. This version exceeds the typical scope of a 10-hour academic project.

5. (20pts) Description of final product including target audience, user story, problem statement, key features, technical details and technologies used. (100 – 150 words).

The final product is a complete AI-assisted loan evaluation web application designed to simulate how financial institutions analyze creditworthiness. The system targets students, developers, and individuals interested in understanding banking decision models. It solves the problem of opaque loan decisions by using a transparent, rule-based multi-agent pipeline that explains every step. Users enter their financial data, and the system computes key metrics such as DTI, loan-to-income ratio, affordability, and risk level. It then displays the results through an interactive dashboard featuring charts, risk gauges, color indicators, analysis cards, and personal recommendations. The backend uses Python and FastAPI, while the front end is built with HTML, CSS, JavaScript, and Chart.js for data visualizations.

6. (20pts) Link to a video demo of product. This should be 1-2 minutes and narrated. Level of difficulty and detail of the project should be reasonable for 10 hours of work with LLM support. The project should not be something an LLM can solve without significant effort by the developer. (Be sure to have someone else test that your link is working).

Youtube Link: https://youtu.be/haFGxU_iGtw

7. (15pts) Any input files, coding files, test files should be uploaded. Provide a list here of file names and purposes, or any links to live sites or artifacts. Remember code should also be commented. A README file should be created and uploaded so that we have the option to follow your instructions to run your project.

Github Link: https://github.com/RazorJDCZ/Bank_AI

List of Files:

Backend (backend/)

- main.py — FastAPI backend that runs the multi-agent loan evaluation system.
- requirements.txt — Python dependencies for backend execution.

Frontend (frontend/)

- index.html — User input form for loan application.
- result.html — Dashboard displaying evaluation results.
- script.js — Sends data to backend, receives analysis, renders charts and UI.
- style.css — Styling for the frontend and dashboard components.

Documentation (docs/)

- architecture.md — System architecture diagram.
- SS.pdf — Supporting screenshot/documentation.
- .gitkeep — Maintains folder structure.

Data (data/)

- README.md — States that no external dataset is used; all data is user input.
- .gitkeep — Maintains folder structure.

Report:

- this pdf

Project Root

- README.md — Project overview, demo link, installation, setup, and how to run.

8. (10pts) Consultation and Use of LLMs:

Each student must create a unique project but is allowed to consult with other people and use Large Language Models (LLMs). Describe how you incorporated these resources into your project:

- **Consultation Description:**

Describe how you ended up seeking advice or feedback from peers, mentors, or online communities.

We consulted classmates for UI feedback and logic clarity, and we interacted with online developer communities to better understand FastAPI edge cases and charting techniques. Peer feedback helped us refine the user experience and simplify the input process. Online discussions also clarified best practices for organizing multi-agent systems and implementing data validation. All external consultations served only advisory purposes; the implementation remained our responsibility.

- **Use of LLMs:**

Explain how you ended up utilizing LLMs to assist with coding, debugging, learning technologies and concepts, or generating ideas.

LLMs were used for brainstorming architecture ideas, debugging issues, improving explanations, and refining our UI design. The models helped clarify questions about FastAPI, chart rendering, and CSS optimization. However, all core logic—including agent design, decision rules, risk scoring, dashboard analytics, and multi-agent reasoning—was designed and implemented by us. We used LLMs responsibly to support learning and productivity but not to generate unoriginal or copied solutions.

9. (10 pts) Ethical Considerations:

Address ethical considerations by documenting *data provenance and consent* (1): specify what data you used, ownership, permissions to use/share, presence of personal data, consent obtained, anonymization/pseudonymization, and retention policy. Explain *privacy and security risks* (2): which sensitive attributes might be exposed and how you protect data and artifacts (access controls, secret management, redaction). Assess *fairness and bias* (3): identify groups that could be disadvantaged and describe checks and mitigations (sampling reviews, baselines, group metrics, error analysis). Evaluate *misuse and safety* (4): outline potential dual-use harms (surveillance, harassment, fraud) and safeguards (restrictions, rate limits, filters, warnings, scope limits). Provide *transparency and accountability* (5): what users can know (sources, limitations, failure modes), include a Model Card and

Data Datasheet, and name maintenance responsibility. Consider *cross-cultural and accessibility* (6) factors across the U.S. and Ecuador (language, bandwidth, norms, laws), and inclusive design. Clarify *intellectual property, licensing, compliance, and attributions* (7). Optionally, address compute and sustainability: model size, costs, and impacts.

- i. Completeness & Specificity (4 pts): Addresses at least four areas with project-specific details (not generic statements).

Our project uses only user-provided financial inputs (income, expenses, debt, credit score, employment status) that are entered voluntarily through the interface. No external datasets, personal records, or persistent storage are used, so ownership remains fully with the user and no consent beyond voluntary entry is required. Sensitive inputs are processed locally in real time and never saved, transmitted, or logged. The system separates responsibilities across agents (Risk, Compliance, Decision, Explanation) to ensure clear reasoning and avoid hidden automated behaviors. Bias is minimized by limiting the evaluated features to financial metrics only and by providing interpretability through detailed, transparent explanations. The system includes clear disclosures of its educational purpose to avoid misinterpretation as a real credit-scoring tool.

- ii. Risk Analysis & Mitigation (4 pts): Identifies realistic risks and documents at least one implemented mitigation with evidence.

The most relevant risks involve potential exposure of sensitive financial attributes and the possibility of biased or unfair lending decisions. To mitigate privacy risks, the system intentionally avoids data retention and does not use databases, logs, cloud storage, or external APIs. All computations remain in local memory and disappear after evaluation, reducing the possibility of unauthorized access.

Regarding fairness, risk classifications—such as debt-to-income ratio or affordability—are checked using transparent formulas, and the system prevents automatic rejection by offering conditional approvals and personal recommendations. This ensures that financial weaknesses are contextualized rather than used as rigid determinants. The Evidence of mitigation lies in the implemented multi-agent structure and the dashboard explanations showing exactly how each metric influenced the final assessment.

- iii. Cross-Cultural Insight (2 pts): Explicitly compares implications in U.S. vs. Ecuador and notes accessibility choices.

The tool was designed to function in both the U.S. and Ecuador, recognizing differences in financial practices, credit-scoring systems, and levels of digital infrastructure. Instead of using country-specific metrics (e.g., FICO thresholds or U.S. tax documentation), the system relies on universal ratios—such as DTI and income stability—that apply to diverse markets. To support accessibility, the

interface is lightweight, avoids heavy assets and API dependencies, and can run smoothly in low-bandwidth environments common across Latin America. Language, visual design, and instructions were kept neutral and easily translatable to ensure usability for both English-speaking U.S. users and Spanish-speaking Ecuadorian users.