WFAP – Wells Fargo Agent Protocol

Team- Credit Avengers

1. Executive Summary

System Overview:

A2A Consumer Banking System is a sophisticated agent-to-agent (A2A) financial network designed to automate corporate credit line processing using AI agents. The system enables financial institutions to streamline credit applications, risk assessment, and loan offer generation through secure inter-agent communication based on the Wells Fargo Agent Protocol (WFAP).

The architecture features a Consumer Agent (host) that coordinates with multiple competing Bank Agents. Each bank agent applies its own risk assessment policies, ESG (Environmental, Social, Governance) evaluation criteria, and interest rate calculations to generate personalized loan offers. ESG factors are integrated into lending decisions, providing interest rate discounts for companies with strong sustainability profiles.

The core business problem addressed is the automation and standardization of corporate credit applications across multiple financial institutions, ensuring regulatory compliance and modern ESG integration in lending decisions.

Technical Stack:

Frontend

- **HTML5**
- **CSS3**
- **JavaScript (ES6+)**

Backend

- **Python 3.10+**
- **Flask** (REST API server)

Agents & Protocol

- **Tachyon ADK Client**
- **A2A SDK**
- **Google ADK**
- **Google Generative AI**

Key Technical Achievements:

- Implementation of a custom financial protocol (WFAP) with JSON Schema validation
- Real-time multi-bank credit processing with parallel agent communication
- Advanced ESG integration with external report fetching and carbon footprint analysis
- Comprehensive risk assessment framework with industry-specific benchmarking
- Production-ready web interface with real-time offer comparison and negotiation capabilities

Technical Architecture:

The A2A Consumer Banking System follows a distributed microservices architecture pattern with the following core components:

Consumer Agent (Host):

- Web-based user interface for credit applications
- Agent discovery and connection management
- Offer aggregation and comparison capabilities
- Interactive negotiation facilitation
- Session management and audit trails

Bank Agents (5 Independent Services):

- CloudTrust Financial Agent (Port 10002) - Conservative ESG-focused lending
- Finovate Bank Agent (Port 10003) - Competitive rates with ESG integration
- Zentra Bank Agent (Port 10004) - Specialized lending solutions
- NexVault Bank Agent (Port 10005) - Digital-first banking approach

- Byte Bank Agent (Port 10006) - Technology sector focused lending

Communication Protocol:

- Wells Fargo Agent Protocol (WFAP) for standardized messaging
- JSON-RPC over HTTP for real-time communication
- Digital signature verification for message integrity
- Structured data exchange with JSON Schema validation

Host Agent Workflow:

User Interaction (Web UI)

- The user interacts with the web interface (index.html), which is styled by styles.css and powered by script.js.
- The user can:
    - Start a credit application via a guided form (startCreditApplication() in script.js).
    - Type a natural language request directly into the chat input.

---

2. Credit Application Submission

- When the user submits a request (form or chat), the message is sent to the backend via /api/chat (sendToAgent).
- The UI shows a typing indicator and updates the status timeline (e.g., "Processing request", "Contacting banks", etc.).

---

3. Backend Processing (Flask Server)

- The Flask server (web_server.py) receives the request.
- It creates or retrieves a session and passes the message to the ConsumerAgent.
- The agent parses the request and builds a structured WFAP Intent Packet, including company info and credit requirements.

---

4. Bank Agent Discovery & Broadcast

- The Consumer Agent discovers available bank agents (CloudTrust, Finovate, Zentra, Byte, NexVault).
- It sends the credit request to all bank agents in parallel using the WFAP protocol (host_tools.py).
- Each bank agent runs its own risk assessment, ESG evaluation, and offer generation workflow.

---

5. Offer Collection & Aggregation

- The Consumer Agent collects responses (loan offers) from all bank agents.
- Responses are validated and aggregated.
- The UI updates to "Receiving offers" and then "Analyzing offers".

---

6. Offer Analysis & Recommendation

- The agent analyzes offers using criteria like interest rate, ESG impact, repayment terms, etc.
- It may use a decision logic engine (see loan_offer_analyzer_tool.py) for ranking and recommendations.
- The best offer is selected and presented to the user.

---

7. Negotiation (Optional)

- The user can negotiate terms with specific banks.
- The agent relays negotiation messages to the chosen bank agent(s).
- Updated offers are collected and re-analyzed.

---

8. Final Selection & Acceptance

- The user selects and accepts the best offer.

- The agent sends an acceptance message to the chosen bank agent.
- The final agreed offer is returned and displayed to the user.

---

9. Session Tracking & Feedback

- The UI tracks session info, message count, and request status.
- The user can view company profile data sent with applications.
- Help, sample requests, and API info are available via sidebar links.

Key Files

- index.html: UI layout
- script.js: UI logic, chat handling, form submission
- web_server.py: Flask backend, API endpoints
- agent.py: ConsumerAgent logic and workflow
- host_tools.py: Bank agent communication utilities

Summary:
The agent workflow is:
User submits → ConsumerAgent builds request → Broadcast to banks → Collect offers → Analyze/recommend → Negotiate (optional) → Accept offer → Display results.

Bank Agents Workflow:

Here is the step-by-step workflow for the **Bank Agent** (CloudTrust Financial) as implemented in your codebase:

---

1. Receive Credit Request

- The agent receives a structured credit request (Intent) from the consumer agent via the A2A protocol.
- The request includes company info, financials, ESG requirements, and desired loan terms.

---

2. Initial Risk Assessment (STEP 1)

- The agent **always** starts by calling perform_initial_risk_assessment.
    - Checks industry eligibility, loan amount limits, jurisdiction, minimum revenue, debt ratios, and basic financial health.
- If the result is "PASS": proceed to next steps.
- If the result is "FAIL": **stop** and return a detailed rejection explanation (no further tools called).

---

3. Loan Offer Calculation (STEP 2A)

If risk assessment is "PASS", the agent executes these tools in order:

1. **Interest Rate Calculation**
    - Call calculate_interest_rate_and_offer with financials, ESG, collateral, etc.
    - Computes base rate, risk premium, ESG discount, and final interest rate.
2. **Approved Amount Calculation**
    - Call calculate_final_approved_amount.
    - Determines max approved amount based on revenue, risk, ESG, and policy limits.
3. **Repayment Duration Calculation**
    - Call calculate_approved_repayment_duration.
    - Decides final approved duration based on policy, risk, and requested terms.
4. **ESG Impact Summary Generation**
    - Internally generate a concise, positive ESG summary (never shown to user directly).
    - Extract ESG data from context and translate to descriptive terms.
5. **Loan Offer JSON Generation**

    - Call generate_loan_offer_json with results from previous steps and ESG summary.
    - Returns a complete, valid JSON offer object.

---

4. Negotiation Handling (STEP 3)

- If the user sends a negotiation message:
    - Analyze the negotiation (e.g., promises to invest in cleaner machinery).
    - Generate a negotiation score (1-10).
    - Call negotiate_loan with negotiation score, preferred interest rate, and current offer rates.
    - Return a new offer JSON with a reduced interest rate if applicable.

---

5. Offer Acceptance (STEP 4)

- If the user accepts the offer:
    - Respond with a final agreed-upon offer JSON, including all terms and ESG summary.
    - Add a status message: "Thank you for accepting our offer. Our bank representatives will come back shortly for further steps."

---

6. Offer Rejection (STEP 5)

- If the user rejects the offer:
    - Respond with a simple text: "We are sorry that we could not meet your line of credit requirements."

---

7. Execution Rules

- **Strict order:** Never skip risk assessment; never call offer tools if risk assessment fails.

- **Retry mechanism:** If a tool fails due to invalid arguments, retry up to 2 times before reporting error.
- **Response format:** Always return only the final JSON from generate_loan_offer_json (no extra text, no markdown).

Summary:
The bank agent workflow is:
Receive request → Risk assessment → (If PASS) Calculate rate, amount, duration → Generate ESG summary → Build offer JSON → Handle negotiation → Accept/reject → Respond.