Capstone Project: Loan Navigator Agent Suite for Fintech

## Abstract

In India’s fast-paced fintech space, BlueLoans4all is empowering micro-entrepreneurs by offering accessible, small-ticket loans. Their support centers, however, face a deluge of repetitive yet vital queries like EMI status, prepayment scenarios, and top-up eligibility. This project introduces a multi-agent AI system, built using

**LangGraph or Google ADK**, powers the Agentic App that acts as a smart "Loan Navigator". The solution will combine NLP-to-SQL capabilities, RAG-based policy lookups from a vector database, and a what-if simulation engine to provide accurate, secure, and compliant answers to customer queries. By automating these interactions, the system enhances the borrower experience, reduces operational load, and ensures regulatory adherence.

## Background

BlueLoans4all is a digital microlender serving thousands of customers across India. A significant portion of their operational cost and effort is spent on manually handling common loan-related queries at their support desk. Each response must be accurate, consistent, and compliant with RBI norms, which requires querying internal databases and interpreting dense policy documents.

**Key Challenges:**

* **High Call Volume:** A constant influx of repetitive queries strains support staff.
* **Delayed Responses:** Manual lookups lead to increased customer wait times.
* **Compliance Risks:** Inconsistent or incorrect manual responses pose a significant regulatory risk.
* **High Operational Costs:** The manual resolution process is resource-intensive and not scalable.

## Objective

To develop an autonomous AI agent suite that accurately resolves common loan queries, simulates financial scenarios, and ensures all responses are compliant and audit-ready. The goal is to automate the support process, providing 24x7 availability while significantly reducing manual overhead and improving key business metrics like query resolution time and prepayment uptake.

## Problem Statement

**Detailed Problem:** The core issue is the inefficiency and risk associated with manually resolving high-volume, low-complexity customer queries. Support staff spend their time fetching data from a loan database and cross-referencing policy documents to answer simple questions about EMIs, prepayments, and eligibility. This process is slow, error-prone, and prevents staff from focusing on more complex customer issues, creating a bottleneck that hinders the company's ability to scale efficiently.

**Proposed Solution:** We will build a multi-agent system that simulates an expert financial support team. This "Loan Navigator Agent Suite" will feature specialized agents:

* **SQL Analyst Agent:** Converts natural language questions into secure SQL queries to fetch data from the loan database.
* **Policy Guru Agent:** Uses RAG to retrieve and cite information from regulatory and internal policy documents stored in a vector database.
* **What-If Calculator Agent:** Runs financial simulations for scenarios like loan prepayments.
* **Supervisor Agent:** Analyzes user intent and orchestrates the other agents to generate a comprehensive, final response.

**Solution Implementation Requirements**

* The solution must be hosted on **Google Cloud Platform (GCP)**.
* AI models will be **Gemini models** accessed via **Vertex AI**.
* The application must be containerized and deployable on **Google Cloud Run**.
* The system requires robust logging and traceability using **Langfuse or MLflow** integrated with **Google Cloud's operations suite**.
* All sensitive data, including database URIs and API keys, will be managed by **Google Secret Manager**.

## Data Provided

* **SQLite Loan Database:** A database containing structured records for approximately 1000 loans, with fields like loan\_amount, tenure, interest\_rate, and topup\_eligible. This will be used by the SQL Analyst Agent.
* **Policy Documents (PDF):** A corpus of internal policies, risk guidelines, and regulatory mandates for the Policy Guru Agent.
* **Chroma Vector DB (Pre-Fed):** A pre-populated Chroma vector database containing embeddings of the policy documents, enabling fast semantic search.
* **Amortization Schedule & Documents:** Technical documents and schedules outlining EMI formulas and prepayment logic, to be used for developing and validating the What-If Calculator Agent.

## Solution Design

**Phase 1: Foundation & Data Infrastructure**

* **GCP Setup:** Provision a GCP Project, enable the Vertex AI API, and set up Artifact Registry.
* **Database & Storage:** Host the SQLite loan database file on Google Cloud Storage and configure access. Host the pre-fed Chroma vector database on a Google Cloud Run instance with a persistent volume.
* **Security:** Store all credentials, database URIs, and API keys securely in **Google Secret Manager**.
* **Agent Framework Scaffolding:** Define the multi-agent graph structure using **LangGraph or CrewAI**, outlining the state and nodes for the Supervisor, SQL Analyst, Policy Guru, and Calculator agents.

**Phase 2: Core Agent Development & Logic**

* **SQL Analyst Agent:** Develop the agent to perform natural language to SQL conversion using a **Gemini model** via Vertex AI. Implement robust security with whitelisted, parameterized queries. Define a fallback mechanism where failed or empty-result queries are flagged to the Supervisor for user clarification.
* **Policy Guru Agent:** Implement the RAG pipeline to perform semantic search against the Chroma vector DB. Use a **Gemini model** to synthesize answers from retrieved chunks and include citations. Set a confidence score threshold for retrievals (e.g., < 0.75) to trigger a fallback, where the Supervisor retries with more context before providing a generic answer.
* **What-If Calculator Agent:** Build the agent as a stateless Python function to perform amortization simulations. Implement input validation to catch errors (e.g., prepayment exceeding balance) and return structured error messages to the Supervisor for clarification prompts.

**Phase 3: Multi-Agent Orchestration**

* **Supervisor Agent:** Implement the central orchestrator using **LangGraph or CrewAI**. Develop logic for intent classification based on user queries to route tasks to the appropriate sub-agent(s).
* **Response Synthesis:** The Supervisor will be responsible for merging responses from different agents, ensuring a consistent tone, and managing the clarification and fallback chains when sub-agents report issues.
* **Logging:** The Supervisor will log all interactions, agent decisions, and final outcomes for monitoring and feedback.

**Phase 4: API & UI Development and Deployment**

* **API Wrapper:** Create a FastAPI wrapper around the Supervisor Agent to expose its functionality via a secure REST endpoint.
* **Containerization & Deployment:** Write a Dockerfile to package the application. Push the container image to **Google Artifact Registry** and deploy it as a serverless application on **Google Cloud Run** with autoscaling enabled.
* **Authentication:** Secure the API endpoint using Google Cloud IAM and Identity-Aware Proxy (IAP) or standard OAuth 2.0 flows.

**Phase 5: Observability, Testing & Governance**

* **Monitoring & Tracing:** Integrate the application with **Google Cloud's operations suite** (Monitoring, Logging, Trace) to track API-level metrics. Implement **Langfuse or MLflow** for detailed LLM-specific tracing of agent performance, token usage, and fallback rates.
* **Testing:** Conduct unit tests for each agent's specific function (SQL, RAG, calculation) and integration tests for end-to-end user journeys. Perform User Acceptance Testing (UAT) with business stakeholders to validate response accuracy and tone.
* **CI/CD & Feedback:** Establish a CI/CD pipeline using **Google Cloud Build** or GitHub Actions for automated deployment. Implement a feedback loop where agent performance data is analyzed to retrain prompts and models monthly.

## Expected Deliverables

* **Deployment Artifacts:**
  + Containerized services for all four agents deployed on

**Google Cloud Run** via **Artifact Registry**.

* + Secure FastAPI endpoints for production use.
* **Documentation:**
  + OpenAPI specification for all APIs, agent interaction diagrams, prompt templates, and a setup runbook.
* **Monitoring & Observability:**
  + A pre-configured dashboard in **Google Cloud Monitoring** and **Langfuse/MLflow** for full traceability of latency, fallback events, and token usage.

## Tech Stack

* **Cloud Platform:** Google Cloud Platform (GCP)
* **AI Service:** Google Vertex AI (for Gemini Models)
* **Orchestration Framework:** LangGraph or CrewAI
* **Vector Database:** Pinecone / Weaviate
* **Deployment:** Docker, Google Cloud Run, Google Artifact Registry
* **Observability:** GoogleCloud's operations suite, Langfuse or MLflow
* **Security & Secrets:** Google Secret Manager, Google Cloud IAM
* **API:** FastAPI