**Individual Documentation – API Component**

**Githmin Dulsara :** s16344

**1. Overview of My Contribution**

In this group project, my responsibility was to design and implement the API layer of our Loan Application Review Agent. The API acts as the bridge between the user and the agent system, allowing requests and responses to flow in a structured way.

The API ensures that loan officers can interact with the agent without needing to directly handle raw CSV data or manually run code.

**2. Purpose of the API**

* Provide a standardized entry point for loan application review requests.
* Connect the synthetic datasets (loan\_applications.csv and kyc\_watchlist.csv) with the agent’s decision-making engine.
* Return consistent, machine-readable outputs (JSON) that contain:
  + Final decision (APPROVE / FLAG / REJECT)
  + Risk score
  + Reasons behind the decision
  + LLM-generated human explanation

**3. API Design**

**Framework Used**

We implemented the API using **FastAPI** because it is:

* Lightweight and fast
* Automatically generates interactive API docs (Swagger UI)
* Easy to integrate with Pydantic for validation

**Endpoints**

**1. GET /review/{application\_id}**

* **Input**: Loan Application ID
* **Process**:
  1. Fetch applicant details from loan\_applications.csv.
  2. Cross-check against kyc\_watchlist.csv.
  3. Run the policy checks (KYC, AML, DTI, Delinquency, etc.).
  4. Generate a risk score and decision.
  5. Pass facts + reasons + decision to the LLM for explanation.
* **Output** (JSON):

{

"application\_id": "12345",

"decision": "FLAG",

"risk\_score": 0.72,

"reasons": [

"Debt-to-Income ratio above 45%",

"Credit score below 600"

],

"explanation": "The loan application was flagged because..."

}

**4. Workflow Example**

1. A user calls the API endpoint: http://localhost:8000/review/12345
2. The API fetches the loan record:
   * Name: John Silva
   * Age: 34
   * Income: 80,000
   * Requested Loan: 60,000
   * Credit Score: 590
3. Checks against rules:
   * Credit Score < 600 → Warning
   * DTI > 45% → Fail
4. Decision generated: FLAG
5. LLM creates a business-friendly explanation.
6. API returns the structured JSON response.