**Individual Documentation – Streamlit UI**

**Vinuk Thismalpola: s16370**

**1. Introduction**

My individual contribution in this project was to design and implement the **user interface (UI)** using **Streamlit**. The UI is the entry point for users to interact with the Loan Application Review Agent. It allows users to upload, view, and test synthetic loan application data and get immediate feedback on approval, rejection, or flagging decisions.

**2. Objectives**

* Provide a **simple, interactive, and user-friendly interface**.
* Connect the UI with the **backend agent logic** through API calls.
* Display results in both **technical (JSON, risk score)** and **business-friendly (explanation, metrics)** formats.
* Enable users to explore loan applications, run checks, and visualize decisions without writing code.

**3. Tools & Frameworks Used**

* **Streamlit**: For creating the web-based UI.
* **Pandas**: For loading and displaying loan application data.
* **Pathlib**: For handling file paths.
* **Agent Core Functions**: Integrated with backend functions like review\_application and LLMClient.

**4. Implementation Details**

**4.1 Layout**

The UI has **two main panels**:

* **Left Panel (col1):** Displays loan applications in a table and allows selection of an application ID.
* **Right Panel (col2):** Runs the decision process and shows outputs (decision, risk score, policy checks, explanations).

**4.2 Features**

* **Data Preview:** Users can see all loan applications in a dataframe.
* **Application Selection:** User selects an application ID from dropdown.
* **Run Review Button:** Executes the agent decision-making process.
* **Decision Metrics:** Shows recommended action (Approve/Reject/Flag) and risk score.
* **Policy Checks:** Displays results of KYC, AML, DTI, etc.
* **Explanations:**
  + Rule-based reasons (from policy checks).
  + LLM-generated explanation (business-friendly).

**4.3 Code Snippet Example**

st.title("🏦 Loan Application Review Agent")

st.caption("Powered by LLM + Policy Rules")

col1, col2 = st.columns([1, 2], gap="large")

with col1:

st.subheader("Applications")

st.dataframe(apps, height=400)

app\_id = st.selectbox("Select Application ID", apps["application\_id"].tolist())

model = st.text\_input("Ollama model", value="tinyllama")

with col2:

st.subheader("Decision")

if st.button("Run Review"):

result = review\_application(row, watch, llm, use\_retrieval=True, policy\_path="policy.md")

st.metric("Recommended Action", result["recommended\_action"])

st.metric("Risk Score", f"{result['risk\_score']\*100:.1f}%")

**5. Outputs Produced by UI**

* **Visual metrics:** Risk Score and Recommended Action.
* **Policy Check JSON:** Shows pass/warn/fail for each rule.
* **Readable Explanations:** LLM explanation in styled box.
* **Data Interaction:** Table and dropdown for exploration.

**6. Challenges & Solutions**

* **Challenge:** Handling layout neatly in limited space.  
  **Solution:** Used Streamlit columns() for side-by-side display.
* **Challenge:** Showing technical + human-friendly results.  
  **Solution:** Added separate sections (JSON vs explanation).
* **Challenge:** Connecting with backend properly.  
  **Solution:** Imported agent core functions and used review\_application().

**7. Business Value of the UI**

* Provides **real-time decisions** in a simple dashboard.
* Helps **loan officers** quickly understand risk without reading raw data.
* Ensures **clarity** by separating rules and AI explanations.
* Makes the system **demo-friendly** for both technical and non-technical stakeholders.

**8. Conclusion**

The Streamlit UI plays a critical role by making the loan decision system **accessible and interactive**. My contribution ensures that the backend logic and AI models are easily usable by end-users through a professional and intuitive interface.