**AI Case Study: Social Support Application Workflow Automation**

**1. Background**

This role focuses on hands-on solution design to implementation acumen with embedded Data and AI thought leadership and influence Data and AI best practices for the public sector regionally and globally. The role involves rapid AI product prototyping and implementations to drive impact through operational efficiency and benefits to residents and citizens. This take-home assessment evaluates your practical Data and AI skills in designing and implementing an AI solution.

**2. Problem Statement**

You are developing an AI workflow for a government social security department, which provides financial and economic enablement support for needy applicants. The current application process takes 5 to 20 working days to assess and approve support for applicants and families.

The current process has several pain points:

* **Manual Data Gathering:** Manual entry of applicant details from scanned documents, physical document collection from government offices, and extraction of information from handwritten forms, leading to data entry errors and delays.
* **Semi-Automated Data Validations:** Basic form field validation with manual checks for inconsistencies, requiring significant manual effort to identify and correct errors.
* **Inconsistent Information:** Discrepancies in address information between application forms and credit bureau reports, variations in income reporting across different financial documents, and conflicting family member details across submitted documents.
* **Time-Consuming Reviews:** Multiple rounds of application reviews involving different departments and stakeholders, causing delays and bottlenecks in the approval process.
* **Subjective Decision-Making:** Assessment prone to human bias, leading to inconsistent decisions and potential unfairness in the allocation of support.

**3. Solution Scope**

The goal is to automate the application process, aiming for up to 100% automated social support applications decision-making within a few minutes with live interactions with GenAI Chatbot.

For this exercise, you can make assumptions of your choice and leverage as much synthetic or mockup data to complete the solution. The solution should:

* Ingest data from interactive application form and attachments such as bank statement, emirates ID, resume, assets/liabilities excel file and credit report.
* Assess applications based on criteria such as income level, employment history, family size, wealth assessment and demographic profile to determine eligibility for financial support and economic enablement support.
* Provide recommendations to approve (or soft decline) for economic social support.
* Provide recommendations for economic enablement support (e.g., upskilling and training opportunities, job matching, career counseling).

The solution must include:

* Locally hosted ML and LLM models.
* Multimodal (text, images, and tabular data) data processing and storage.
* Interactive Chat interaction.
* Agentic AI Orchestration.

**4. Technology Stack and Tools**

You are encouraged to use the recommended tools below under each category.

* **Programming Language:** Python.
* **Data Pipeline:** Pandas, LlamaIndex, ChromaDB, Postgres SQL.
* **AI Pipeline:**
  + Select and justify specific Scikit-learn algorithms for classification, explaining your choices based on data characteristics and problem statement.
  + Select and justify specific tools to handle specific data types (text, images, tabular data).
  + Design and develop GenAI Agents such as master orchestrator, data extraction, data validation, eligibility check and decision recommendation.
* **Agent Reasoning Framework:** Use one or more reasoning framework in your solution such as ReAct, Reflexion, PaS etc.
* **Agent Orchestration:** Use one or more Agentic orchestration tool such as LangGraph, LlamaIndex, Phidata, Crew.AI etc.
* **Model Hosting:** Use one or more tool to host LLM locally such as LMStudio, Ollama, OpenWebUI, LLMWare.
* **Agent Observability:** Use LangSmith or Langfuse for end-to-end AI Observability.
* **Model Serving:** Use one or more tool such as Flask, FastAPI, Ollama, LMStudio, OpenWebUI.
* **Front-End: Use** Streamlit or Gradio.
* **Version Control:** Use GitHub for hosting end to end solutions artifacts and detailed step by step documentation.

**5. Submission Guidelines**

* All source code files via private GitHub link with detailed documentation.
* A README.md file with clear instructions on how to run the application.
* A solution summary document (up to 10 pages) outlining your solution design, including:
  + A high-level architecture diagram of your solution, detailing data flow and component interactions.
  + Justify your tools choice specifically considering their suitability, scalability, maintainability, performance, and security.
  + A breakdown of the AI solution workflow into modular components.
  + Suggestions for future improvements and how this solution could be integrated into existing systems, including API design and data pipeline considerations.

**6. Evaluation Criteria**

Your submission will be evaluated based on the following criteria:

* **Functionality:** Does the prototype address all core requirements in sections 2 and 3, using the tools in section 4?
* **Code Quality:** Is the code clean, well-organized, and properly documented? Is it modular and easy to support and extend?
* **Solution Design:** Is the architecture well-thought-out, scalable, and relevant to the problem and solution scope? Does it show an understanding of AI/ML principles, software engineering best practices, and system design?
* **Integration:** How effectively are key components integrated, and are tools used to drive a scalable and robust solution? Are APIs and data pipelines designed effectively?
* **Demo UI:** Is the demo UI user-friendly?
* **Problem-Solving:** How well did you address the challenges met during development?
* **Communication:** Is your documentation clear, concise, and thorough?

**7. Timeframe**

The expected time to complete this take home prototype and documentation is up to 5 days, followed by half an hour in-person or online demo and presentation.