# Social Support Application Workflow Automation

## 1. Problem Overview

The government social security department currently processes financial assistance applications manually, resulting in delays, inconsistencies, and inefficiencies. The goal is to automate the end-to-end process using GenAI and Agentic AI to enable decision-making within minutes.

## 2. High-Level Architecture Diagram

**Components:**

- **Frontend (Streamlit)**: Collects form inputs and file uploads.

- **GenAI Chatbot (Ollama + ReAct)**: Engages users to fill forms interactively.

- **Document Parsing Agent**: Uses LlamaIndex, Unstructured, OCR for extraction.

- **Reconciliation Agent**: Validates consistency between form inputs and supporting documents uploaded by user.

- **Validation Agents**:

1. Family Details Validator (Gov API)
2. Income Validator (Bank API + Statement Parsing)
3. Wealth Validator (Credit API + Excel Parser)

- **Eligibility Decision Agent**: Rule-based/XGBoost model for social support decisioning.

- **Recommendation Agent**: RAG-based LLM retrieval for economic enablement.

- **LangGraph Orchestration**: Manages agent flow.

- **ChromaDB + PostgreSQL**: Stores vectors and structured records.

- **LangSmith**: Agent observability.

High Level Architecture Diagram for Agentic AI Application


*High Level Architecture Dig.*

## 3. Tool Justification

| Category | Tool | Justification |
| --- | --- | --- |
| Programming | Python | Rich ecosystem for AI/ML, integration, and APIs |
| UI | Streamlit | Easy to build interactive chat and form UIs |
| LLM Host | Ollama | Local hosting for data security and performance |
| LLM Framework | ReAct | Supports explainable, dynamic agent reasoning |
| Agent Orchestration | LangGraph | Modular, scalable agent workflows |
| Document Parsing | LlamaIndex, Unstructured, Tesseract | Handles PDFs, images, tables |
| Storage | PostgreSQL, ChromaDB | Structured + vector storage for hybrid search |
| Model | XGBoost, SHAP | Fast, explainable decision model |
| Validation | FastAPI | Secure, scalable API interface |
| Observability | LangSmith | Trace agent decisions and outputs |
| Versioning | GitHub | CI/CD and collaboration |

## 4. Modular AI Workflow Components

### 1. **Chatbot Form Collector (GenAI Chatbot Agent)**

* Uses Ollama-hosted LLM with ReAct prompting.
* Collects critical user inputs such as Emirates ID, address, income, employment status, asset/liability, resume and family size.
* Interactively engages the user and maintains session context.
* Captures responses in structured JSON and stores them in PostgreSQL.

**Flow:** User → Chatbot (LLM) → Extracted Form Data → Stored in DB

### 2. **Document Parser Agent**

* Uses LlamaIndex + Unstructured + Tesseract OCR to process uploaded files.
* Identifies and extracts data from Emirates ID images, bank statements (PDF), resumes, and Excel sheets.
* Converts tabular data to structured formats using Pandas and PDF parsers (e.g., Camelot).
* **Flow:** File Upload → File Type Classifier → Relevant Parser → Extracted Entities (EmiratesId , Name, Income, Employer, Address, Family Members, Assets/Liabilities)

### 3. **Reconciliation Agent**

* Compares form-collected data with parsed document values.
* Evaluates for mismatches across key attributes (e.g., income, family size).
* Uses LangGraph to make reconciliation decisions and LangSmith to log discrepancies.
* If mismatches are found, a ReAct loop prompts user for clarification.

**Flow:** Form Data + Parsed Data → Reconciliation Logic → Match/Discrepancy → User Prompt (if needed)

A diagram of a company

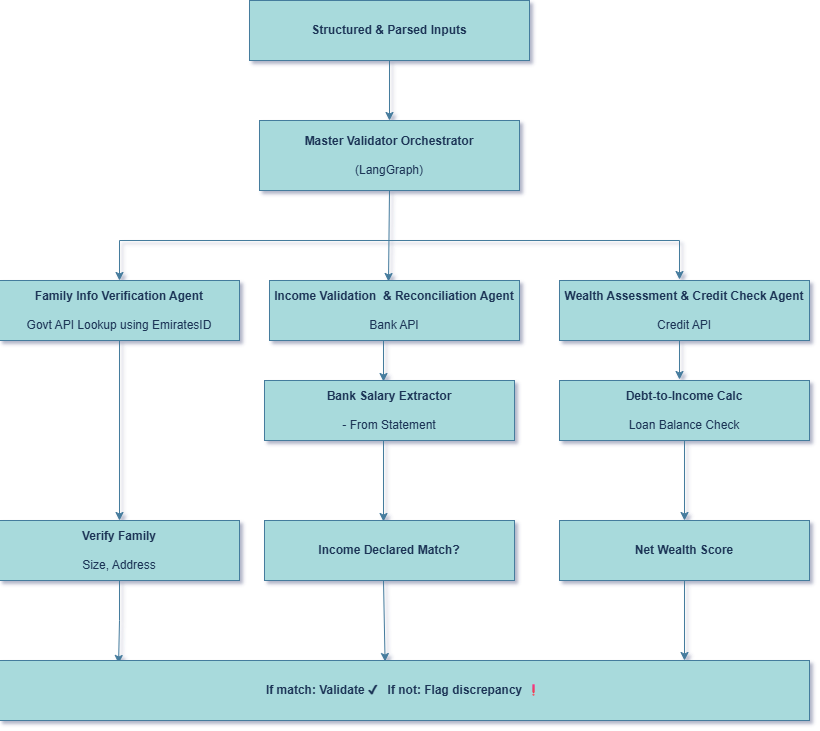
AI-generated content may be incorrect.

*Doc Parser Agent + Reconciliation Agent Dig.*

### 4. **Validation Agents (Family, Income, Wealth)**

* **Family Validator**:
  + Calls Government API with Emirates ID to retrieve official family size and dependents.
  + Cross-verifies user-submitted values.
* **Income Validator**:
  + Parses salary deposits from bank statement.
  + Calls Bank API to confirm employer and income range.
  + Reconciles with declared income.
* **Wealth Validator**:
  + Extracts assets/liabilities from Excel and Credit Reports.
  + Calls Credit API for current liabilities and credit score.

**Flow:** User ID → API Call → Response → Validation Result (✔ / ❌) → Stored in DB



**5. Eligibility Decision Agent**

* Central agent that uses validated data to decide social support eligibility.
* Works in two modes:
  + **Rule-based logic**: Direct criteria like income thresholds.
  + **ML-based scoring**: XGBoost model trained on historical applications.
* Outputs: Eligibility decision, confidence score, and reasoning (via SHAP).

**Flow:** Validated Data → Rule/Model → Decision + Explanation → Stored + Forwarded to Chatbot

### 6. **Recommendation Agent (Economic Enablement)**

* Suggests career counseling, upskilling, or job-matching programs.
* Uses a retrieval-augmented generation (RAG) approach.
  + Vectorizes applicant profile (skills, education, region).
  + Retrieves best-matching programs from ChromaDB.
  + Generates summary using LLM.

**Flow:** Applicant Profile → Vector Query → Top Program Matches → Summarized Output

### 7. **Chatbot Feedback Loop (Optional)**

* Reflexion Agent reviews agent outputs and user feedback.
* If discrepancies are unresolved or user objects to decision:
  + Triggers new collection loop.
  + Logs feedback for training and auditing.

**Flow:** Error/Dispute → Reflexion → Re-Ask/Reprocess → Re-run Pipeline

### 8. **Storage and Logging Layer**

* **PostgreSQL**: Stores structured user data, validation results, decision metadata.
* **ChromaDB**: Embedding store for upskilling/job programs.
* **LangSmith**: Logs agent actions, prompts, decisions, feedback for observability.

**Flow:** Every step → Logs + Persistence → Observability Dashboard

### 9**. Deployment Strategy & Goals**

To ensure that the social support automation system is scalable, modular, cloud-agnostic, and easy to maintain, we propose a container-based microservice architecture deployed on Kubernetes.

### **Containerized Modular Architecture Using Docker & Kubernetes**

Our deployment approach focuses on the following key benefits:

1. **Modularization**:  
   Every component (like the chatbot, document parser, or eligibility model) runs independently as its own service. This allows teams to build, test, and deploy each module separately.
2. **Cloud Agnostic:**  
   The solution can run on any cloud provider (AWS, Azure, GCP) or even on-premise, as long as Kubernetes is available.
3. **Scalability:**  
   Kubernetes automatically scales up or down the number of service instances (pods) based on usage. For example, more chatbot agents can be launched during peak hours.
4. **Security:**  
   Sensitive services (like document parsing or APIs connected to government systems) are isolated in separate containers for controlled access.
5. **Observability:**  
   Logs, metrics, and traces are collected at every step to monitor system health, track user interactions, and debug issues quickly.

## 5. Future Enhancements

### 🔧 System Improvements:

* Integrate Kafka for real-time streaming of document parsing.
* Add a workflow engine (e.g., Temporal) for retry/error handling.
* Expand to **multilingual LLM agents** using fine-tuned Arabic/Urdu support (e.g., Meta's LLaMA, Mistral models)
* Integrate **auto-document summarization** for long resumes, PDFs, or credit reports to reduce token cost.

### 🔌 Integration Ideas:

* Direct integration into government portals (Emirates ID, Tamkeen APIs).
* REST APIs for third-party service providers (training centers, banks).
* Real-time dashboards for social workers with audit logs.
* Integrate **MCP to standardize the context-sharing** between agents and models across the pipeline. Enables **agent interoperability**, traceability, and consistent grounding for LLM prompts.

### 📊 ML Model Roadmap:

* Continuous learning pipeline with human feedback.
* **Implement MCP across agents** to create a **shared memory layer**, improving reasoning flow and decision audit trails.
* Experiment with tabular+text multi-modal decision models (e.g., TabTransformer).

## Functional Example: Social Support Automation for Ahmed

**Applicant: Jamal Khan**

* Age: 30
* Marital Status: Married
* Dependents: 5
* Location: Sharjah
* Employment: Unemployed (laid off 3 months ago)
* Monthly income: AED 12,000
* Assets: AED 8,000
* Liabilities: AED 5,000
* Credit Score: 670
* Applied via: Government Social Support Portal

**Step-by-Step Workflow with Functional Data**

**1. Chatbot Form Input Collection (LLM Agent)**

* **Tool**: Ollama + ReAct + Streamlit
* **Captured Inputs** (via conversation):

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**2. Document Parsing Agent**

* **Tool**: LlamaIndex + Tesseract + Camelot + Unstructured
* **Extracted Data**:
  + **Emirates ID (OCR)**: Name: Ahmed Al Mansoori, DOB: 1987-03-10
  + **Resume**: Last employer – Dubai Mall, Sales Associate (2018–2023)
  + **Bank Statement**: Last salary deposit AED 3,500 (March 2024)
  + **Assets Excel**:
  + **Credit Report**:

**3. Reconciliation Agent (Cross-check Form vs Docs)**

* **Tool**: LangGraph + ReAct
* **Logic**:

A screenshot of a document

AI-generated content may be incorrect.

**4. Validation Agents (via APIs + Parsing)**

* **Family Validator**
  + ✅ Emirates ID → Government API: Dependents = 5
* **Income Validator**
  + ✅ Bank API confirms last income = AED 12,000
  + Employer: Dubai Mall, Status: Job Ended Mar 2024
* **Wealth Validator**
  + ✅ Assets = AED 8,000; Liabilities = AED 5,000
  + ✅ Credit Score = 670

Validation Output Stored

**5. Eligibility Decision Agent (ML + Rules)**

* **Tool**: XGBoost + SHAP
* **Input Features**:

*{*

*"income": 12000,*

*"dependents": 5,*

*"employment\_status": "Unemployed",*

*"net\_assets": 8000,*

*"credit\_score": 670*

*}*

* **ML Output**:
  + Eligibility = Yes
  + Confidence = 92%
  + SHAP Explanation:
    - **Main drivers: Unemployed (↑), Low assets (↑), 5 dependents (↑), borderline credit (↓)**

**6. Recommendation Agent (Economic Enablement)**

* **Tool**: RAG + ChromaDB + LLM
* **Profile Vectorization**:

*{*

*"skills": [ "Office Administration", "Customer Service", "Time Management",*

*"MS Excel", "CRM", "Problem Solving"*

*],*

*"education": "Bachelor of Business Administration (BBA)",*

*"experience": "Admin Executive at ABC Logistics, Customer Service at Al Noor Telecom",*

*"languages": ["English", "Urdu", "Arabic (Basic)"],*

*"location": "Sharjah"*

*}*

* **Top Matches Retrieved**:
  1. Government-Backed Admin & Coordination Certification – Online
  2. Customer Success Career Track – Funded Program – Dubai
  3. Arabic-English Bilingual Call Center Training – Abu Dhabi

**7. Final Chatbot Response**

* **Message**:

A screenshot of a computer

AI-generated content may be incorrect.

**8. Logging and Persistence**

* **PostgreSQL**:
  + Stores all structured records: form data, validation, eligibility decision
* **ChromaDB**:
  + Stores applicant profile embedding
* **LangSmith**:
  + Logs every LLM prompt, decision, SHAP score, and user interaction

**✅ Result**

* **End-to-End Time**: 1 minutes 57 seconds
* **Decision**: Declined
* **Recommendations**: 3 upskilling programs
* **Audit Trail**: All decisions logged and explainable