AI Personal Loan Assistant (Agentic Demo)

This repository contains a small demo of an Agentic conversational personal-loan assistant inspired by the Tata Capital challenge. The code demonstrates a Master Agent that orchestrates simple Worker Agents (Sales, Verification, Underwriting, Sanction) to complete an end-to-end, rule-driven loan flow and generate a sanction letter (PDF) on approval.

Quick Summary

- Purpose: simulate a conversational loan sales & underwriting flow in a web chat UI.
- Output: on approval the app produces `sanction_letter.pdf` in the project folder.
- Tech: Python, FastAPI, ReportLab (for PDF), simple frontend in `static/index.html`.

Project Structure

- `main.py` — FastAPI app and `/chat` endpoint. Maintains in-memory `SESSIONS` keyed by `session_id`.
- `agents.py` — Master Agent (orchestrator). Implements conversational steps and delegates to worker agents. Supports a `guest` onboarding flow and stores `last_reason` for explaining rejections.
- `sales_agent.py` — Sales/UX helper prompts (greetings, ask loan amount/tenure, guest prompts).
- `verification_agent.py` — Simple verification helpers (looks up `CUSTOMERS` by phone).
- `underwriting_agent.py` — Wraps `rules.evaluate_loan` to produce `(decision, reason)`.
- `rules.py` — Eligibility rules. Returns `(decision, reason)` with human-readable explanations.
- `sanction.py` — Uses ReportLab to generate a basic `sanction_letter.pdf` on approval.
- `data.py` — Mock customer data (pre-approved customers). See phone keys such as `9876543210` and `9999999999`.
- `static/index.html` — Frontend chat UI. Requests server greeting on load and keeps a session id for conversation continuity.
- `requirements.txt` — `fastapi`, `uvicorn`, `reportlab`.

How it works (conversation flow)

1. Client loads `/` (serves `static/index.html`), the page requests an initial greeting from `/chat` and receives a `session_id`.
2. Master Agent steps: `START` → `PHONE` → (either registered customer or `guest` onboarding) → `LOAN_AMOUNT` → `TENURE` → Underwriting → `END`.
3. Underwriting returns `(decision, reason)`. The `reason` is stored in the session and included in rejection messages. Typing `why` (or `explain`) will return the stored reason.
4. On `APPROVED`, `sanction.py` writes `sanction_letter.pdf` to the working folder.

Example chat API usage

- Request server greeting / start session (frontend does this automatically):

```cmd
curl -X POST http://127.0.0.1:8000/chat -H "Content-Type: application/json" -d "{\"message\": \"\"}"
```

- Send a message (with `session_id` returned by the server):

```cmd
curl -X POST http://127.0.0.1:8000/chat -H "Content-Type: application/json" -d "{\"message\": \"9876543210\", \"session_id\": \"__default__\"}"
```

Notes on the frontend:

- The UI will ask for a phone number or `guest` to start guest onboarding.
- If you type `guest`, the bot will collect name → salary → credit score → loan amount → tenure, then run underwriting.

How to run locally (Windows / cmd.exe)

1. Install dependencies (recommend a virtualenv):

```cmd
python -m pip install -r requirements.txt
```

2. Start the app:

```cmd
python -m uvicorn main:app --reload --port 8000
```

3. Open the UI in your browser:

```
http://127.0.0.1:8000/
```

Testing notes

- Pre-seeded customers exist in `data.py`. Use phone `9876543210` for a customer with good credit.
- Guest data is stored only in the in-memory session for the demo. Restarting the server clears sessions.

Limitations & next steps

- Sessions are in-memory — swap to Redis or a DB for production and multi-worker deployments.
- EMI calculation is simplified (loan_amount / 12). Replace with a proper EMI formula including interest rates.
- The Sales Agent is rule-based prompts. For natural, empathetic dialog integrate an LLM (OpenAI/Hugging Face) for message generation and intent parsing.
- Persist guest profiles to `guests.json` or a simple DB if you want guest accounts to be re-usable.
- Improve UI/UX: show sanction PDF link after approval, add file download, progress indicators, and conversational tone.

License & attribution

- This is a small demo created as an exercise. No license file is included; add one if needed.

If you'd like, I can also produce a polished PDF version of this README, persist guest profiles to a JSON file, or add a more detailed EMI breakdown when the user asks "why". Which would you prefer next?