Got it! I'll create a full-fledged PRD with the following details:

- **Tech Stack:** Python-based implementation
- **AI Integration:** Utilize prebuilt LLMs from your local Ollama setup
- **Deployment:** Strictly local, no cloud integration
- **Milestones:** A structured breakdown to guide a junior developer through implementation

I'll provide the PRD shortly.

Product Requirements Document (PRD)

Objective

Develop a **local AI system** that computes life path predictions using six methods: **Bazi, Feng Shui, Western Astrology, Chinese Zodiac, I Ching,** and **Vedic Astrology**. The system will:

1. Collect **method-specific data** via user input.
2. Compute results using **preloaded formulas**.
3. Deliver **individual and combined summaries** of findings.

1. Technical Specifications

1.1 Tech Stack

- **Programming Language:** Python
- **Frameworks/Libraries:** FastAPI (for API endpoints), SQLite (local data storage), Ollama (for LLM integration), Matplotlib (for data visualization)
- **Local AI Models:** Prebuilt LLMs hosted via Ollama (e.g., `gemma3:27b`, `deepseek-rag`, `qwen`)
- **Data Handling:** JSON files for formula storage, SQLite for input/output caching
- **Deployment:** Local-only execution (no cloud dependencies), with optional Docker containerization for portability

2. Functional Requirements

2.1 Data Collection

Each method requires different input types:

Method	Required Inputs	Example Format
Bazi	Birth time (UTC±X), solar time offset	`{ "birth_time": "1990-05-15T14:30+08:00" }`
Feng	Floor plan, compass	`{ "door_degree": 120, "floor_plan":`
Western Astro	Birth coordinates, house system	`{ "lat": 40.7128, "lon": -74.0060, "house_system": "Placidus" }`

2.2 Computation Engine

- **Bazi:** Compute **Heavenly Stems and Earthly Branches**, and determine Luck Pillars.
- **Feng Shui:** Use **Flying Star** charts to analyze home energy distribution.
- **Western Astrology:** Calculate **planetary placements** (using Swiss Ephemeris) for natal charts.
- **Chinese Zodiac:** Match the birth year to one of the **12 zodiac animals**.
- **I Ching:** Generate a hexagram and interpretation based on the **coin toss method**.
- **Vedic Astrology:** Compute **Nakshatra** (lunar mansion) positions and **Dasha** (planetary period) timelines.

3. System Components

3.1 Question Generator (LLM-Powered)

- Utilizes local LLMs (via Ollama) to generate user-friendly prompts for any missing inputs.
- **Example:** Python function to prompt for Bazi input: `def ask_bazi():`

```
•     return {
•         "question": "Enter birth time (YYYY-MM-DD HH:MM
  UTC±X):",
•         "validation": "regex: ^\d{4}-\d{2}-\d{2} \d{2}:
  \d{2} UTC[+-]\d{1,2}$",
•         "error_msg": "Invalid format. Example:
  1995-07-20 08:30 UTC+8"
•     }
•
```

3.2 Computation Modules

- Implements specialized calculation functions for each method using preloaded formulas or rulesets.
- **Example:** Flying Star Feng Shui calculator in Python: `def flying_star(compass_degree, construction_year):`

```
•     base_star = (180 - compass_degree) // 15 % 9 + 1
•     return f"Period 9 Star {base_star}"
•
```

4. Output & Reporting

- **Individual Reports:** Each method produces a personalized report. For example, *"Your Bazi chart suggests a strong Water element; avoid Fire-heavy careers."*
- **Combined Summary:** The system provides an aggregated overview. For example, *"5 out of 6 methods predict a major career shift in 2025."*
- **Action Plan Recommendations:** Generate practical advice based on findings from multiple methods. For example, *"Rearrange your workspace for career success, as indicated by both Feng Shui and Western Astrology."*

5. Error Handling & Data Validation

- **Invalid Input Handling:** If a user input does not match the expected format, the system will prompt the user to retry with the correct format.
- **Missing Data/Formula Errors:** If a required formula or data set is missing, the system alerts the user. For example: *"Vedic Astrology Nakshatra lookup missing. Please upload the required JSON rules."*

6. Milestone Plan

Phase	Task	Details
1	**Setup**	Install all dependencies and configure the Python environment.
2	**Data Collection**	Implement input validation and LLM-driven question prompts.
3	**Computation Engine**	Develop computation modules for all six prediction methods.
4	**Output Generation**	Build the reporting module for individual results and combined summary.
5	**UI & Interaction**	Create a CLI or lightweight UI to facilitate user interaction.
6	**Testing**	Validate all calculations and handle edge cases or errors.

7. Deliverables

1. **Python Codebase:** The complete source code, including the FastAPI backend and SQLite database schema.
2. **Ollama Integration:** Configuration and usage instructions for local LLM models to generate questions.
3. **User Interaction Interface:** A CLI tool or simple graphical UI that allows users to input data and receive predictions.
4. **Final Documentation:** An installation guide and API reference detailing how to set up and use the system.