Below is an **enhanced PRD** section (focused on the CLI workflow) reflecting your latest requirements. It outlines how a **single command** (`cli.py`) can gather all necessary data, run all six computations, and then generate comprehensive AI-driven analyses per method—storing or reusing user data as needed.

Enhanced CLI & Workflow Requirements

1. Single CLI Entry-Point

- File: `cli.py`
- **Command**: `python cli.py run`
-

  This single command will:
  1. **Collect all necessary inputs** (date of birth, birthplace, gender, etc.) via an interactive question-and-answer flow.
  2. **Run all six methods** (Bazi, Feng Shui, Western Astrology, Chinese Zodiac, I Ching, Vedic Astrology).
  3. **Display results** for each method in a structured, tabulated format.
  4. **Invoke AI agents** (one per method) to provide detailed multi-paragraph analyses.

2. Data Collection & "Question Table"

1. **Centralized Prompts**:
   - Instead of separate prompts for each method, create a single "question table" or structured dictionary of all required fields: `ALL_QUESTIONS = [`
   - `    {"field": "dob", "prompt": "Enter your date of birth (YYYY-MM-DD):", "validation": "..."},`
   - `    {"field": "birth_time", "prompt": "Enter your time of birth (HH:MM, UTC±X):", "validation": "..."},`
   - `    {"field": "gender", "prompt": "Enter your gender (M/F/Other):", "validation": "..."},`
   - `    {"field": "birth_place", "prompt": "Enter your birthplace (country, state, city):", "validation": "..."},`
   - `    # additional fields as needed for each method`
   - `]`
   -
2. **Data Storage & Reuse**:
   - After collecting input, store it in **SQLite** or **JSON** so users can reuse or modify data later without re-entering everything.
   - Example table (`user_data`): `CREATE TABLE user_data (`
   - `    user_id INTEGER PRIMARY KEY,`
   - `    dob TEXT,`
   - `    birth_time TEXT,`
   - `    gender TEXT,`
   - `    birth_place TEXT,`

-
  ```
      ...
  ```
- `);`
-
- If the user has previously run the program, we can **offer to reuse** saved data instead of prompting them again.

3. All-Methods Execution Flow

When the user runs `python cli.py run`:

1. **Check Existing User Data**:
   - If user data is found in the local database, ask if they want to reuse it.
   - Otherwise, prompt for all required fields and store them.
2. **Compute Each Method**:
   - **Bazi**:
     - Convert `dob` + `birth_time` → compute Heavenly Stems, Earthly Branches.
   - **Feng Shui**:
   - Use user's home data (if relevant) or fallback to partial data.
   - **Western Astrology**:
   - Convert date/time + birth_place → planetary positions, aspects.
   - **Chinese Zodiac**:
   - Use `dob` year → zodiac animal.
   - **I Ching**:
   - If a user's question is needed, prompt: "Enter a question for the I Ching."
   - **Vedic Astrology**:
   - Convert birth time/place → Nakshatra, Dasha system.
3. **Tabulate Results**:
   - After computations are done, display a summary table in the terminal using **Rich**:
     `from rich.table import Table`
   - `table = Table(title="Life Path Prediction Results")`
   - `table.add_column("Method", justify="left", style="cyan")`
   - `table.add_column("Key Findings", justify="left", style="magenta")`
   -
   - `table.add_row("Bazi", str(bazi_result))`
   - `table.add_row("Feng Shui", str(fengshui_result))`
   - `...`
   - `console.print(table)`
   -
4. **AI Agent Analyses**:
   - For each method, call the local LLM (Ollama) with an appropriate role prompt, e.g.:
     `def get_ai_analysis(method_name, method_result):`

   -
     ```
         prompt = f"Act like Master {method_name} practitioner. " \
     ```

-
  ```
                 f"Here is the user's computed result: {method_result}. " \
  ```

-
  ```
                 f"Provide a multi-paragraph analysis covering life, career, investment, health, relationships, worldview, focus, and execution."
  ```

-
  ```
      return OllamaAgent().generate_response(prompt)
  ```
-
- Print each analysis below the table or in a separate section: `console.print("[bold green]AI Analysis - Bazi[/bold green]")`
- `console.print(bazi_analysis)`
- `console.print("[bold green]AI Analysis - Feng Shui[/bold green]")`
- `console.print(fengshui_analysis)`
- `...`
-

4. Example CLI Flow

Sample Terminal Session

1. **User runs**: `python cli.py run`
2.
3. **System checks** if user data is in the DB:
   - If yes: `Found existing user profile. Reuse? (Y/N):`
   -
   - If no or user chooses no, prompts: `Enter your date of birth (YYYY-MM-DD): 1990-05-15`
   - `Enter your time of birth (HH:MM UTC±X): 14:30 UTC+8`
   - `Enter your gender (M/F/Other): M`
   - `Enter your birthplace (country, state, city): USA, CA, San Francisco`
   - `...`
   -
4. **System computes** all methods automatically:
   - Bazi -> Output stored in `bazi_result`
   - Feng Shui -> `fengshui_result`
   - Western Astrology -> `western_result`
   - Chinese Zodiac -> `chinese_zodiac_result`
   - I Ching -> `iching_result` (prompts for question if needed)
   - Vedic Astrology -> `vedic_result`
5. **System displays** a summary table in the terminal:
6.
```
┌──────────────────────────────┬──────────────────────────────────────────────
──────────────────────────────┐
│ Life Path Prediction Results
│
```

```
7.  ├──────────────────────┼──────────────────────────────────
    ────────────────────┤
8.  │ Method               │ Key Findings
9.  ├──────────────────────┼──────────────────────────────────
    ────────────────────┤
10. │ Bazi                 │ Element: Water, Luck
    Pillars: ...                  │
11. │ Feng Shui            │ House orientation: NW, Base
    Star: 9, ...
12. │ Western Astrology    │ Sun in Taurus, Moon in Leo, ...
13. │ Chinese Zodiac       │ Horse
14. │ I Ching              │ Hexagram 23: Splitting
    Apart, ...                  │
15. │ Vedic Astrology      │ Nakshatra: Ashwini, Dasha:
    Ketu ...              │
16. └──────────────────────┴──────────────────────────────────
    ────────────────────┘
17.
```

18. **System invokes** AI agents per method, then prints multi-paragraph analyses: `AI Analysis - Bazi`

```
19. "As a Master Bazi practitioner, I see that your strong
    Water element...
20. (detailed multi-paragraph explanation)..."
21.
22. AI Analysis - Feng Shui
23. "Acting as a Master Feng Shui practitioner, I
    recommend...
24. (detailed multi-paragraph explanation)..."
25.
26. ...
27.
```

28. **User sees** all methods' raw outputs **and** expert-level explanations in one terminal session.

5. Data Persistence & Reusability

- **User Table**:
  - `user_id (PK)`, `dob`, `birth_time`, `gender`, `birth_place`, etc.
- **Session Table** (optional):
  - `session_id (PK)`, `user_id (FK)`, `timestamp`, `bazi_result`, `fengshui_result`, etc.
- **Reusability**:
  - Next time the user runs `python cli.py run`, they can choose to load an existing session or user data.

6. Implementation Tips

1. **Validation**: For each input, use either regex or custom logic (e.g., valid birth years = 1900–2025).
2. **Modular Code**:
   - `cli.py` orchestrates user input and triggers computations.
   - Each method's computation logic lives in its own module (`computations/bazi.py`, `computations/feng_shui.py`, etc.).
3. **Rich**:
   - Use `rich.prompt` for interactive prompts if you prefer theming, or keep to standard Python `input()` for simplicity.
4. **Error Handling**:
   - If data is incomplete (e.g., no floor plan for Feng Shui), skip or provide partial results with a warning.
5. **Agent Calls**:
   - The prompt structure for Ollama can be standardized in a utility function (e.g., `ollama_agent.generate(method_name, method_result)`).
6. **Testing**:
   - Create unit tests ensuring each method's output is correct for known inputs (e.g., a known Bazi chart for 1990-05-15 14:30 UTC+8).

Updated Deliverables

1. **Single CLI Tool** (`cli.py`):
   - A single `run` command that:
     - Prompts for user data (or loads existing data).
     - Computes all six methods in sequence.
     - Displays tabulated results.
     - Fetches AI agent analyses per method and prints them.
2. **User Data Persistence**:
   - An SQLite database with tables for user profiles and (optionally) session results.
3. **Comprehensive Output**:
   - Each method's raw computed results + an AI-driven analysis.
4. **README Updates**:
   - Clear instructions on:
     - **Installing** dependencies.
     - **Running** `python cli.py run`.
     - **Sample inputs** & **expected sample outputs**.
     - **Data reuse** (how to load an existing user).
     - **Troubleshooting** (common errors, e.g., incomplete data, invalid birth time format).

Conclusion

By consolidating all six methods under **one CLI run** flow, you achieve a **holistic** user experience. The user provides data once, the system executes every calculation, and the user sees both **numerical** (or textual) **results** and **expert-level analyses** in a single terminal session. This meets the requirement of a more **user-friendly, integrated** system with advanced AI insights.