AI Astrologer Application: Code Explanation and Approach

August 18, 2025

1 Problem Statement

The objective is to develop an AI Astrologer web application that:

- Collects user birth details (Name, Date, Time, Place) via a clean UI.
- Generates astrology-based output (Sun Sign, Moon Sign, Ascendant, personality traits) using rule-based or AI-driven methods.
- Allows users to ask a free-text question and receive an astrology-based response.
- Includes deliverables: a 2–5 minute demo video and a codebase with setup instructions (GitHub or zip file) by EOD August 19, 2025.

2 Initial Approach

2.1 Design

- **Frontend**: A simple HTML form (`index.html`) with JavaScript to collect birth details and questions, sending data to the backend via POST requests to `/process` and `/process_message`.
- **Backend**: A Flask application (`app.py`) to:
  - Calculate astrological signs using `pyswisseph`.
  - Determine coordinates and timezone using `geopy` and `timezonefinder`.
  - Predict personality traits with a PyTorch neural network.
  - Generate rule-based responses to questions using `nltk`.
- **Storage**: Store user data in a global dictionary for simplicity.
- **Deployment**: Use Flask's development server for testing, with `gunicorn` for production.

2.2 Problems Encountered

- **Syntax Error**: An incomplete string in the `horoscopes` dictionary (`'Aquarius': 'Innovmeaningful comment on code`) caused a `SyntaxError`, preventing the server from starting.
- **Service Timed Out**: Likely due to:
  - Slow or failing geopy geolocation requests.
  - Missing Swiss Ephemeris files in the ephe folder.
  - Resource constraints in Flask's debug mode.
- **Title-Only Display**: If the backend failed (e.g., due to syntax errors or timeouts), the frontend loaded but showed only the title.

2.3 Solutions

- Fixed the `horoscopes` dictionary syntax.
- Added a 5-second timeout to geopy's Nominatim and robust error handling.
- Included logging to diagnose issues.
- Provided instructions to download ephemeris files.
- Suggested `gunicorn` for production to handle timeouts.

3 Code Explanation: `app.py`

3.1 Imports and Setup

- **Flask, CORS**: Handle HTTP requests and enable cross-origin requests.
- **swisseph**: Calculate planetary positions.
- **geopy, timezonefinder, pytz**: Handle geolocation and timezone conversions.
- **torch, nn**: PyTorch for personality trait prediction.
- **nltk**: Tokenize user questions for rule-based responses.
- **logging**: Log debug and error messages.

3.2 Ephemeris Setup

- Sets the path to Swiss Ephemeris files in ephe.
- Raises an error if files are missing, logged for debugging.

3.3 PyTorch Model

- `TraitModel`: A neural network with 4 inputs (day, month, year, hour), a 10-neuron hidden layer (ReLU), and 5 outputs (confidence, luck, creativity, health, love; sigmoid activation).
- `train_model`: Trains the model with dummy data (10 random inputs) using Adam optimizer and MSE loss for 100 epochs.

3.4 Astrological Data

- `signs`: List of zodiac signs.
- `horoscopes`: Dictionary of horoscope messages for each sign.
- `user_data`: Stores user's birth details globally.

3.5 Routes

- `/@route('/')` Serves `index.html`.
- `/@route('/process', methods=['POST'])`:
  - Parses birth details from JSON.
  - Uses geopy to get coordinates (with timeout).
  - Determines timezone with `timezonefinder`.
  - Converts local time to UTC.
  - Calculates Julian date and planetary positions (Sun, Moon, Ascendant) using `swisseph`.
  - Maps longitudes to zodiac signs.
  - Predicts traits using `TraitModel`.
  - Returns JSON with signs and traits.
- `/@route('/process_message', methods=['POST'])`:
  - Tokenizes user questions with `nltk`.
  - Matches keywords (e.g., "horoscope", "love") to generate responses using `user_data`.
  - Returns JSON with the response.

4 Setup Instructions

1. Ensure `app.py`, `requirements.txt`, `static/index.html`, and `ephe/` (with `sepl_18.se1`, etc.) are in the project folder.
2. Activate virtual environment: `env\Scripts\activate` (Windows).

3. Install dependencies: `pip install -r requirements.txt`.
4. Run: `python app.py` and open `http://127.0.0.1:5000`.