AI Astrologer Application: Project Summary

August 18, 2025

1 Project Overview

The AI Astrologer is a web application that collects user birth details (Name, Date, Time, Place) through a clean UI, generates astrological signs (Sun, Moon, Ascendant) and personality traits, and responds to free-text questions with astrology-based advice. The project meets requirements for a user-friendly interface, astrology-based output (rule-based and AI-driven), and question-answering functionality. Deliverables include a codebase (zip or GitHub) and a 2–5 minute demo video by EOD August 19, 2025.

2 Technical Components

2.1 Frontend (`index.html`)

- Built with HTML, CSS, and JavaScript.
- Features a form for birth details and a question input field.
- Uses AJAX (`fetch`) to communicate with backend routes (`/process`, `/process_message`
- Styled with responsive, inline CSS for simplicity.

2.2 Backend (`app.py`)

- **Framework**: Flask with CORS for frontend communication.
- **Astrology**: Uses `pyswisseph` to calculate Sun, Moon, and Ascendant positions based on Julian date.
- **Geolocation**: `geopy` (Nominatim, 5-second timeout) converts place names to coordinates; `timezonefinder` and `pytz` handle timezone conversions.
- **Traits**: PyTorch neural network (`TraitModel`) predicts five traits (confidence, luck, creativity, health, love) using dummy data.
- **Questions**: `nltk` tokenizes questions for rule-based responses based on keywords (e.g., "horoscope", "love").
- **Error Handling**: Logging and try-catch blocks handle geolocation, ephemeris, and model errors.
- **Storage**: Global `user_data` dictionary stores user details.

2.3 Dependencies (`requirements.txt`)

- Includes `flask`, `pyswisseph`, `geopy`, `torch`, `nltk`, `gunicorn`, etc.
- Requires Swiss Ephemeris files in `ephe/` folder.

3 Approach

3.1 Design

- Chose Flask for lightweight backend development.
- Used single-page HTML for a simple, responsive UI.
- Employed `pyswisseph` for accurate astrology calculations.
- Implemented a basic PyTorch model for traits and rule-based responses for questions.

3.2 Development

- Created two routes: `/process` for birth details and `/process_message` for questions.
- Integrated geolocation and timezone handling for accurate calculations.
- Added logging and timeouts to manage external service issues.
- Used global storage for simplicity, avoiding a database.

3.3 Challenges and Solutions

- **Syntax Error**: Fixed a malformed string in `horoscopes` dictionary.
- **Timeouts**: Added 5-second timeout to `geopy` and ensured ephemeris files were included.
- **Title-Only Display**: Resolved with robust error handling and logging.

4 Key Features

- **Input**: Clean UI for entering birth details and questions.
- **Output**: Displays Sun Sign, Moon Sign, Ascendant, and AI-predicted traits.
- **Questions**: Personalized responses based on keywords and user data.
- **Robustness**: Handles errors gracefully with clear feedback.

5 Setup Instructions

1. Place `app.py`, `requirements.txt`, `static/index.html`, and `ephe/` (with ephemeris files) in the project folder.
2. Activate virtual environment: `env\Scripts\activate`.
3. Install: `pip install -r requirements.txt`.
4. Run: `python app.py` and open `http://127.0.0.1:5000`.

6 Deliverables

- **Codebase**: Zip or GitHub repository with all files.
- **Demo Video**: 2–5 minute video showing input, results, and question handling, recorded with OBS Studio and uploaded to Google Drive.