Backend System Design for Exoplanet Habitability Prediction

This document provides a detailed description of the backend architecture designed for the Exoplanet Habitability Prediction project. The backend is responsible for deploying the trained machine learning model and making it accessible through a RESTful API. The design emphasizes simplicity, reliability, and scientific usability.

1. Backend Objective

The primary objective of the backend is to serve the finalized machine learning model and allow users or applications to obtain habitability predictions for exoplanets. It acts as a bridge between the trained ML model and external systems such as frontends or research tools.

2. Technology Stack

- **Flask:** A lightweight Python web framework used to create REST APIs. Flask is chosen because it is simple, flexible, and well-suited for ML-based applications.
- **Python:** The core programming language used for backend development and ML integration.
- **scikit-learn:** Used for training the machine learning models and for making predictions using the saved model.
- **joblib:** A utility used to serialize and deserialize trained ML models efficiently.
- **JSON:** The standard data format used for sending input features and receiving prediction outputs.

3. Machine Learning Model Integration

The trained Random Forest model is stored as a serialized file (.pkl). The backend loads this model once during application startup to avoid repeated loading overhead. This ensures faster inference and efficient memory usage during runtime.

4. API Design and Endpoints

The backend follows REST architectural principles. A POST endpoint is exposed to accept exoplanet features as input. The API processes the input, performs model inference, and returns predictions in a structured JSON format.

5. Input Data Handling

Input data is received in JSON format containing numerical planetary and stellar features. The backend converts this JSON data into a Pandas DataFrame, ensuring compatibility with the machine learning pipeline used during training.

6. Prediction Logic

The backend uses the predict_proba method of the trained model to generate probability-based predictions. A habitability score is computed as the probability of the planet being habitable. Based on a predefined threshold (typically 0.5), the final habitability class is determined.

7. Output Format

The API response includes both the habitability class and habitability score. This allows users to interpret not only the classification result but also the confidence of the prediction.

8. Error Handling and Reliability

Basic error handling is implemented to manage invalid inputs or runtime exceptions. Errors are returned as JSON responses, ensuring that the API remains robust and user-friendly.

9. Scalability and Extensibility

The backend design is modular, allowing easy integration of additional models, new endpoints, or frontend interfaces. The system can be deployed on cloud platforms for scalable usage.

In conclusion, the backend system provides a clean and efficient mechanism for deploying the exoplanet habitability prediction model. It ensures scientific usability, maintainability, and future scalability, making it suitable for real-world applications.