

Milestone 3 : ExoHabitAI – Backend & Frontend Documentation

1. Introduction

The backend of ExoHabitAI is responsible for exposing the trained Machine Learning model through RESTful APIs. It allows users or frontend applications to send exoplanet parameters and receive:

- Habitability prediction (Habitable / Non-Habitable)
- Confidence score (probability)
- Ranking of multiple exoplanets based on habitability

The backend is implemented using Python Flask and follows a clean, modular structure.

ExoHabitAI is a Machine Learning–based system designed to analyze exoplanet data and predict their potential habitability.

2. Technologies Used

- **Python 3.x** – Core programming language
- **Flask** – Web framework for building REST APIs
- **NumPy** – Numerical computations and array handling
- **Joblib** – Loading trained ML model
- **Scikit-learn (Pipeline)** – Preprocessing and prediction (used inside the model)

3. Requirements.txt

The requirements.txt file lists all dependencies required to run the backend.

File: requirements.txt

```
flask
numpy
joblib
scikit-learn
pandas
```

Purpose:

- Ensures consistent environment setup
- Simplifies installation using:

```
python -m pip install -r requirements.txt
```

4. utils.py

Purpose:

utils.py contains reusable helper functions that:

- Validate incoming API data
- Perform feature engineering
- Encode categorical features (star types)
- Prepare model-ready input
- Format prediction output

This separation improves readability, maintainability, and follows industry best practices.

4.1 Feature Order

The model was trained on **23 features**, and the same order must be preserved during prediction.

```
FEATURE_ORDER = [  
    'pl_rade', 'pl_bmasse', 'pl_orbper', 'pl_orbsmax', 'pl_eqt',  
    'st_teff', 'st_lum', 'st_met', 'pl_insol',  
    'Habitability_Score', 'Stellar_Compatibility', 'Orbital_Stability_Score',  
    'Star_Type_A', 'Star_Type_B', 'Star_Type_D', 'Star_Type_F',  
    'Star_Type_G', 'Star_Type_K', 'Star_Type_L', 'Star_Type_M',  
    'Star_Type_T', 'Star_Type_W', 'Stellar_Compatibility'  
]
```

4.2 validate_input()

Validates input JSON and converts it into a NumPy array suitable for the ML pipeline.

Functions performed:

- Checks for missing required fields
- Converts values to numeric form
- Computes engineered features
- Performs one-hot encoding for star types

4.3 Feature Engineering

Derived features such as:

- Habitability Score
- Orbital Stability Score
- Stellar Compatibility

are recomputed in the backend to match the training pipeline.

4.4 format_prediction()

Formats the model output into a clean JSON response.

Output: Prediction label & Confidence score

5. app.py

Purpose:

app.py is the main entry point of the backend application. It:

- Initializes the Flask app
- Loads the trained ML model
- Defines REST API endpoints
- Handles request-response lifecycle

6. API Endpoints

6.1 Home Endpoint

URL: /

Method: GET

Purpose: Check if backend is running

Response:

```
{  
  "message": "ExoHabitAI Backend API is running",  
  "status": "success"  
}
```

6.2 Predict Endpoint

URL: /predict

Method: POST

Input (JSON):

```
{  
  "pl_rade": 1.1,  
  "pl_bmasse": 1.2,  
  "pl_orbper": 365,  
  "pl_orbsmax": 1.0,  
  "pl_eqt": 288,  
  "st_teff": 5778,  
  "st_lum": 1.0,  
  "st_met": 0.0,  
  "pl_insol": 1.0,  
  "star_type": "G"}  

```

Output:{

```
  "status": "success",  
  "result": {  
    "prediction": "Habitable",  
    "confidence_score": 0.82}}
```

6.3 Rank Endpoint

URL: /rank

Method: POST

Purpose: Rank multiple exoplanets based on habitability score

Input: List of exoplanet objects

Output:{

"status": "success",

"ranked_exoplanets": [

{

"planet_name": "Planet-A",

"prediction": "Habitable",

"habitability_score": 0.82,

"rank": 1},

{

"planet_name": "Planet-B",

"prediction": "Non-Habitable",

"habitability_score": 0.54,

"rank": 2

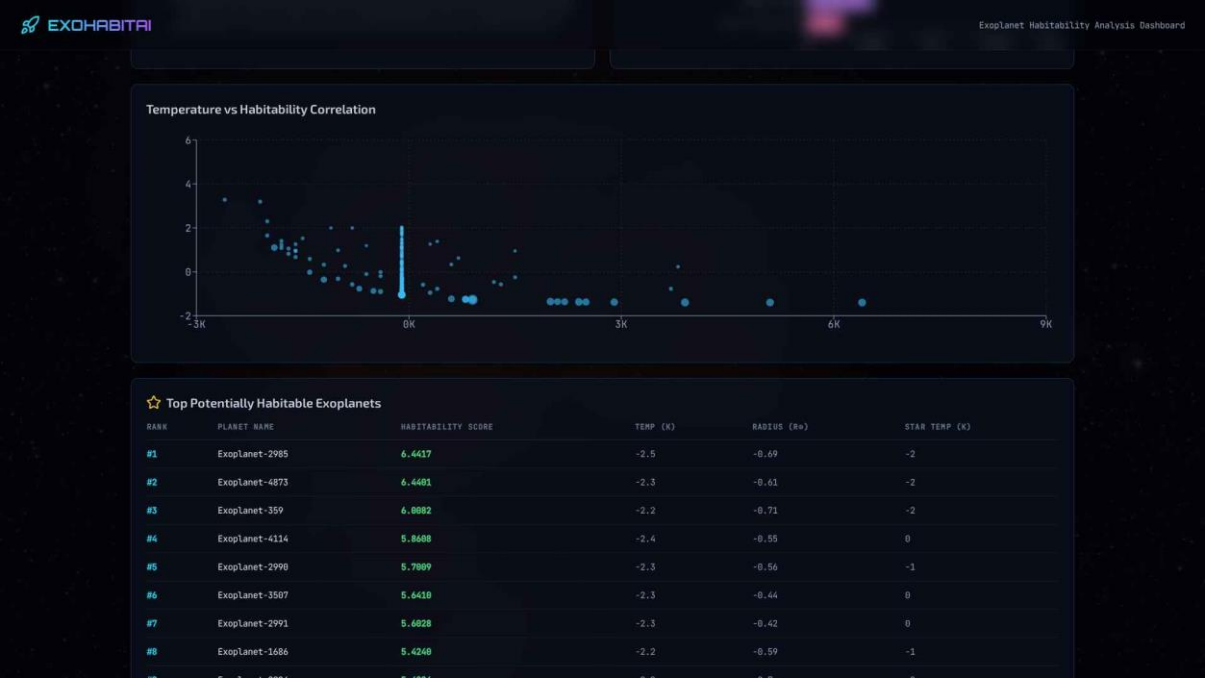
}}}

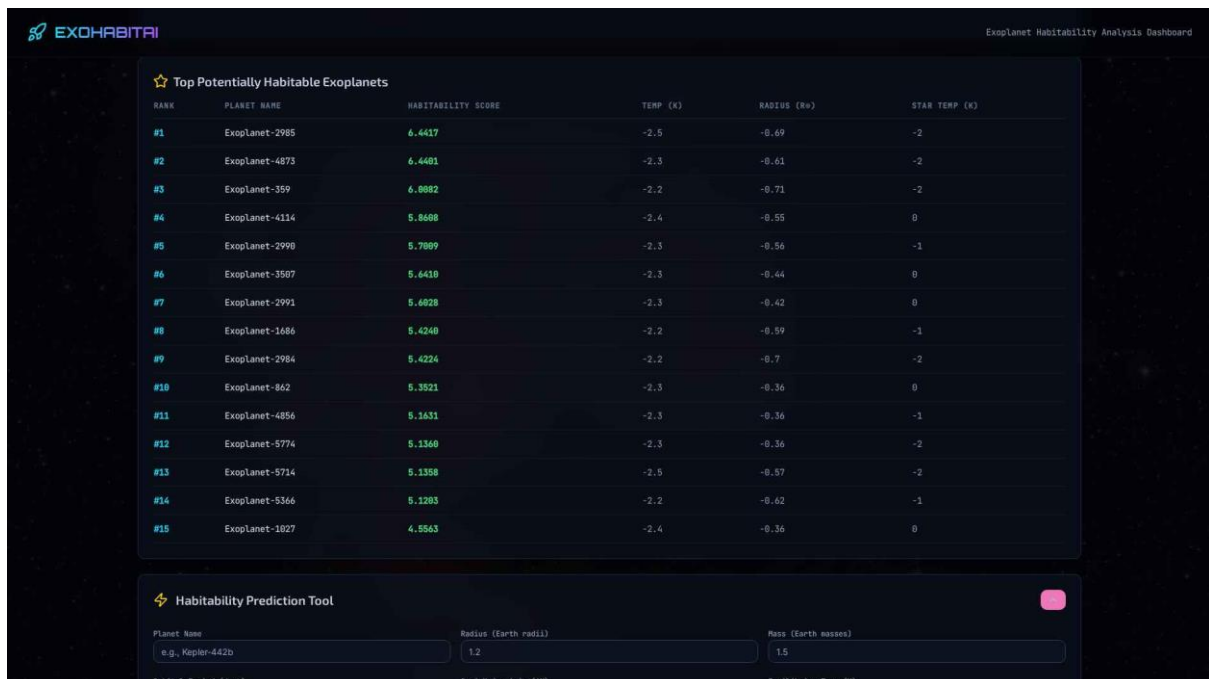
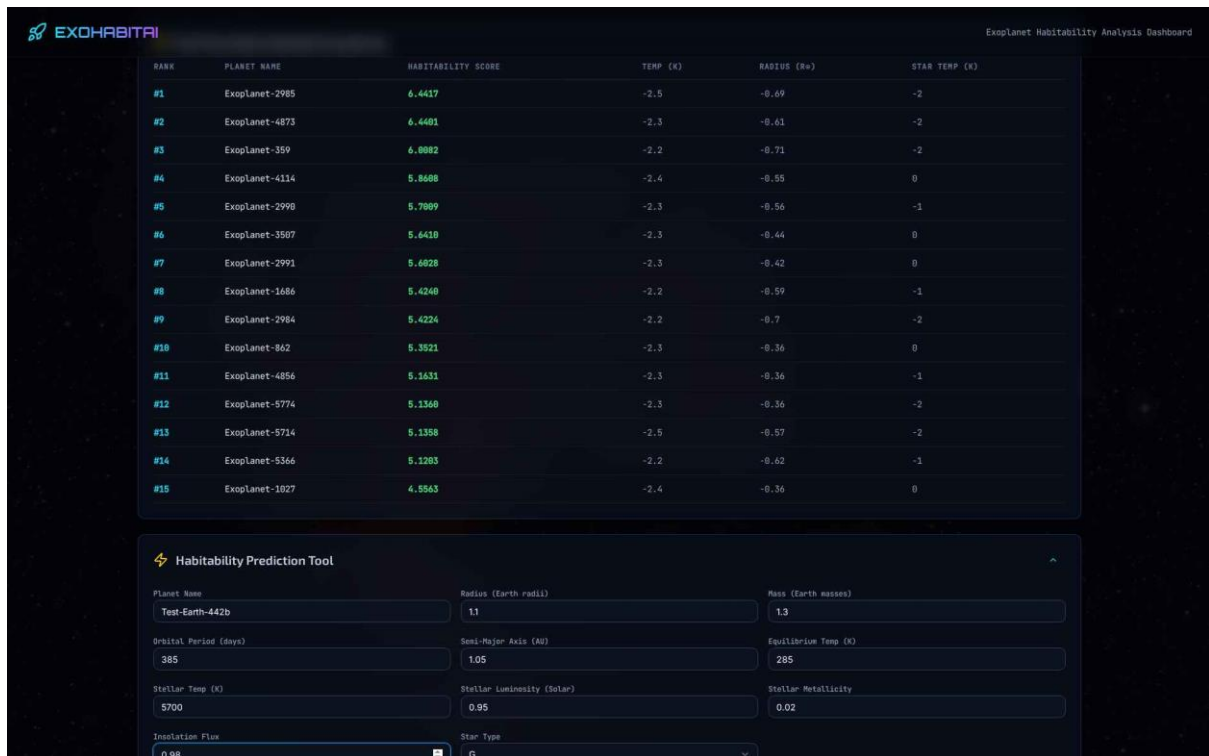
7. Frontend

7.1 Frontend Files

- **index.html**
- **index.css**
- **GlassCars.js**
- **MetricCard.js**
- **HabitabilityDistribution.js**
- **Featureimportance.js**
- **Scatterplot.js**
- **TopOPlanetstable.js**
- **PredictionPanel.js**
- **App.js**
- **App.css**

7.2 Frontend ScreenShots





8. Conclusion

The ExoHabitAI Backend and Brontend successfully provides an intuitive, informative, and interactive interface for exoplanet habitability analysis.

It enhances the usability of the machine learning model and enables users to understand both predictions and underlying data patterns effectively.

