# Frontend and Backend Documentation

# ExoHabitAI: Technical Documentation

## 1. Project Overview

**ExoHabitAI** is an advanced machine learning platform designed to assess the habitability of exoplanets. It goes beyond simple prediction by providing a **Dynamic 3D Simulation** of the target planet based on its physical properties.

The system features a **React Frontend** for real-time visualization, a **Three.js Engine** for 3D rendering, and a **Flask Backend** that serves the AI model.

---

## 2. Key Features

### A. Dynamic 3D Planet Simulation

Unlike static images, the 3D planet **mutates** based on user input:

- **Magma World:** If Temperature > 330K, the planet glows red with volcanic activity.
- **Ice World:** If Temperature < 200K, the surface becomes reflective white/cyan (frozen).
- **Gas Giant:** If Radius > 1.6 Earths, the planet turns into a thick, hazy Jovian world (purple/gas texture).
- **Terra Class:** If conditions are habitable, it renders as a blue, ocean-covered world with white clouds.

### B. Commander's Mission Log

- **Auto-Save:** Every scan is automatically saved to the browser's local storage.
- **Persistence:** The history remains even if the page is refreshed or the browser is closed.
- **Quick Recall:** Users can click on past logs to instantly reload that planet's data and 3D model.

## 3. Technology Stack

| Component | Technology | Purpose |
|---|---|---|
| **Frontend** | React 18 (Vite) | User Interface & State Management |
| **3D Engine** | Three.js / React-Three-Fiber | Real-time 3D rendering of planets |
| **Styling** | Tailwind CSS | Glassmorphism UI & Responsive Design |
| **Backend** | Python (Flask) | API Server & AI Logic |
| **AI Model** | Scikit-Learn | Random Forest Classifier |
| **Data Handling** | Pandas / NumPy | Data processing & Scientific calculations |

## 4. System Architecture

**Data Flow (The "Brain" of the App)**

1. **Input:** User enters Radius, Mass, Temperature in Analyzer.jsx.
2. **API Call:** React sends a POST request to Flask (/predict).
3. **AI Inference:** Flask loads the Random Forest model and predicts:
   - **Habitable (1):** High probability of liquid water.
   - **Non-Habitable (0):** Hostile conditions.
4. **3D Rendering Logic:**

    - React receives the prediction AND the raw input data.
    - Planet3D.jsx analyzes the data locally to decide the **texture** and **color** of the sphere (e.g., "Is it hot? Make it red.").
5. **History:** The result is appended to the localStorage array for the "Mission Log."

---

## 5. Frontend Documentation (/frontend)

### A. Analyzer.jsx (The Core Tool)

- **Role:** The main interactive dashboard.
- **State:** Manages formData (inputs), result (prediction), and history (saved logs).
- **Integration:**
    - Fetches data from Backend.
    - Passes data to <Planet3D /> for rendering.
    - Saves scans to localStorage.

### B. Planet3D.jsx (The Visualization Engine)

- **Role:** Renders the interactive 3D sphere.

**Logic (Procedural Generation):**

JavaScript

```
const biome = useMemo(() => {
  if (radius > 1.6) return { type: "GAS GIANT", color: "purple" };
  if (temp > 330)   return { type: "LAVA", color: "red", emissive: true };
  if (temp < 200)   return { type: "ICE", color: "white", roughness: 0.1 };
  return { type: "TERRA", color: "blue" }; // Earth-like
}, [radius, temp]);
```

- 
- **Effects:** Uses Sparkles (for stars/atmosphere) and MeshStandardMaterial for realistic lighting.

### C. Dashboard.jsx (The Overview)

- **Role:** Displays global statistics (Total Planets, Habitable Count).
- **Charts:** Uses Recharts to show Feature Importance (which variables the AI cares about most).

## 6. Backend Documentation (/backend)

**app.py (The API)**

- **Endpoints:**
  - POST /predict: Handles the core logic. Includes a **safety fallback** (if Period is empty, it assumes Earth-like 365 days).
  - GET /dashboard-data: Returns dataset statistics.
- **Error Handling:** Wraps predictions in try/except blocks to prevent 500 crashes on bad input.

## **Tech Stack:** Python, Flask, Pandas, Scikit-Learn

## System Architecture

The backend is designed as a lightweight **REST API** using the Flask framework. Due to the development environment (Google Colab), the API is instantiated and tested interactively within the notebook runtime using a `test_client` rather than a persistent web server.

#### Core Components

1. **`app` (Flask Instance):** Handles HTTP routing and error management.
2. **`utils.preprocess_input`:** A dedicated preprocessing module that mirrors the training phase. It transforms raw JSON data into the specific **16-feature vector** required by the model.
3. **`model` (Random Forest):** The pre-trained `.pkl` artifact loaded into memory for real-time inference.

#### 1. API Overview

The ExoHabitAI backend is built on **Flask**. It exposes the trained Random Forest model to perform real-time inference. It accepts raw planetary data, performs real-time feature engineering (matching the training pipeline), and returns a probabilistic classification.

## 2. Endpoints

| Endpoint | Method | Description |
|---|---|---|
| / | GET | Health check. Returns API status. |
| /predict | POST | Accepts exoplanet JSON, returns habitability prediction |
| /rank | GET | Returns the top 10 pre-ranked habitable candidates |

## 3. Request Format (/predict)

**JSON:**

{ "Name": "Earth 2.0", "Radius": 1.2, "Mass": 1.5, "Period": 200, "SemiMajorAxis": 0.8, "EqTemp": 270, "Density": 5.5, "StarTemp": 5000, "StarLum": 0.8, "StarMet": 0.0, "Insolation": 0.9, "StarType": "K" }

## 4. Response Format

**JSON:**

{
  "confidence_score": 0.87,
  "habitable_flag": 1,
  "input_planet": "Earth 2.0",
  "prediction": "Potentially Habitable"
}

## 5)Endpoint: Predict Habitability

- **URL:** `/predict`
- **Method:** `POST`

- **Description:** Accepts raw planetary parameters, applies scaling/feature engineering, and returns a binary classification with a confidence score.

**Request Format**

JSON:


```
{
  "Name": "Kepler-186 f",
  "Radius": 1.17,
  "Mass": 1.4,
  "Period": 129.9,
  "EqTemp": 250,
  "Insolation": 0.45,
  "StarTemp": 3755,
  "StarType": "M"
}
```


**Response Format**

JSON:


```
{
  "input_planet": "Kepler-186 f",
  "prediction": "Non-Habitable",
  "habitable_flag": 0,
  "confidence_score": 0.0
}
```


## 3. Data Processing Logic

To ensure the API predictions match the model's training performance, the backend implements the following pipeline on every request:

1. **Feature Engineering:**
    - Calculates `Habitability_Score` using the Earth Similarity Index (ESI) formula.

    - Applies `Log(Period + 1)` transformation to normalize orbital periods.
    - Performs One-Hot Encoding for `StarType` (G, K, M, Other).
2. **Vectorization:**
    - Aligns input data to exactly **16 features** (Radius, Mass, Density, etc.).
    - Fills missing physical values (e.g., `Density`) with training defaults (0) to prevent crashes.
3. **Scaling:**
    - Applies the saved `StandardScaler` (`scaler.pkl`) to normalize inputs to the standard normal distribution expected by the model.

## 4. Validation & Testing

- **Methodology:** Integration testing was performed using the Flask `test_client()` to simulate HTTP POST requests within the notebook environment.
- **Result:** The API successfully processed a request for candidate **Kepler-186 f**, returning a valid JSON structure without runtime errors or dimension mismatches.
- **Error Handling:** The system includes try-catch blocks to return `500 Internal Server Error` messages with debug details if the input JSON is malformed or the model fails to load.

---

## 7. Setup & Run Guide

**Step 1: Backend (The Brain)**

Bash

```
cd backend
# 1. Install Libraries
pip install flask flask-cors pandas numpy scikit-learn joblib
# 2. Build the AI Model (Run Once)
python setup.py
# 3. Start Server
python app.py
```

*Server is live at: http://127.0.0.1:5000*

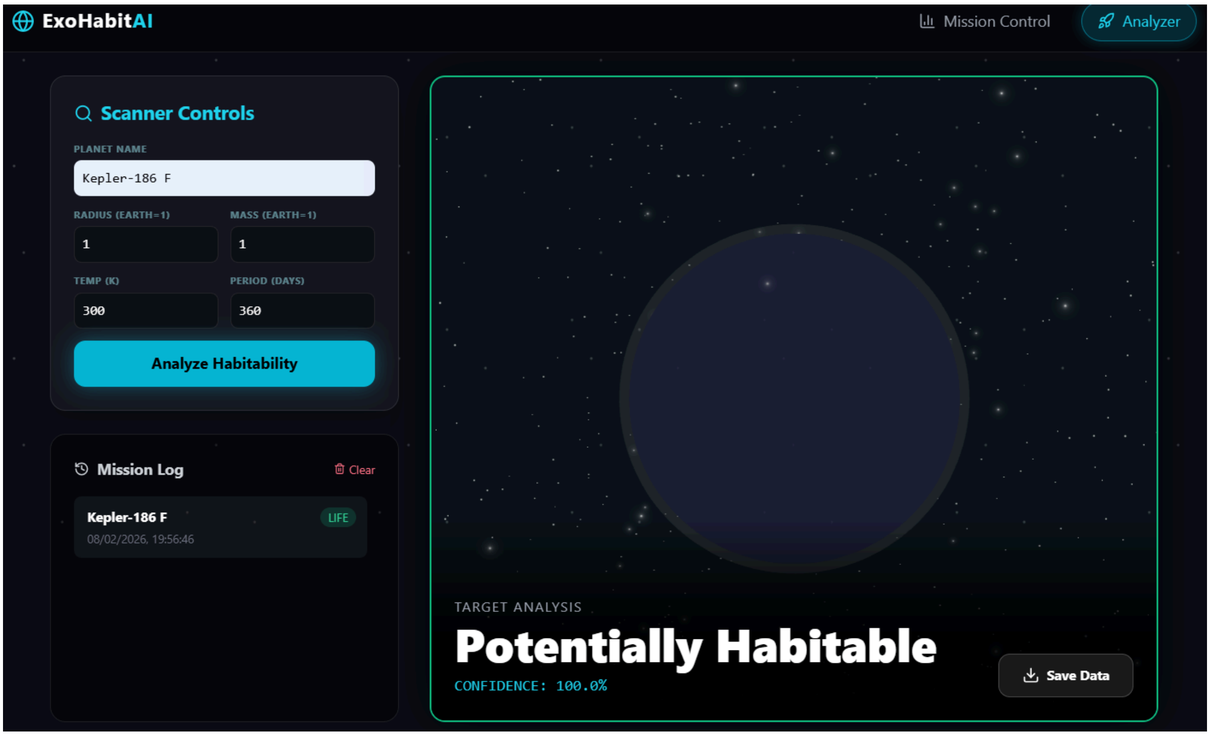
**Step 2: Frontend (The Interface)**

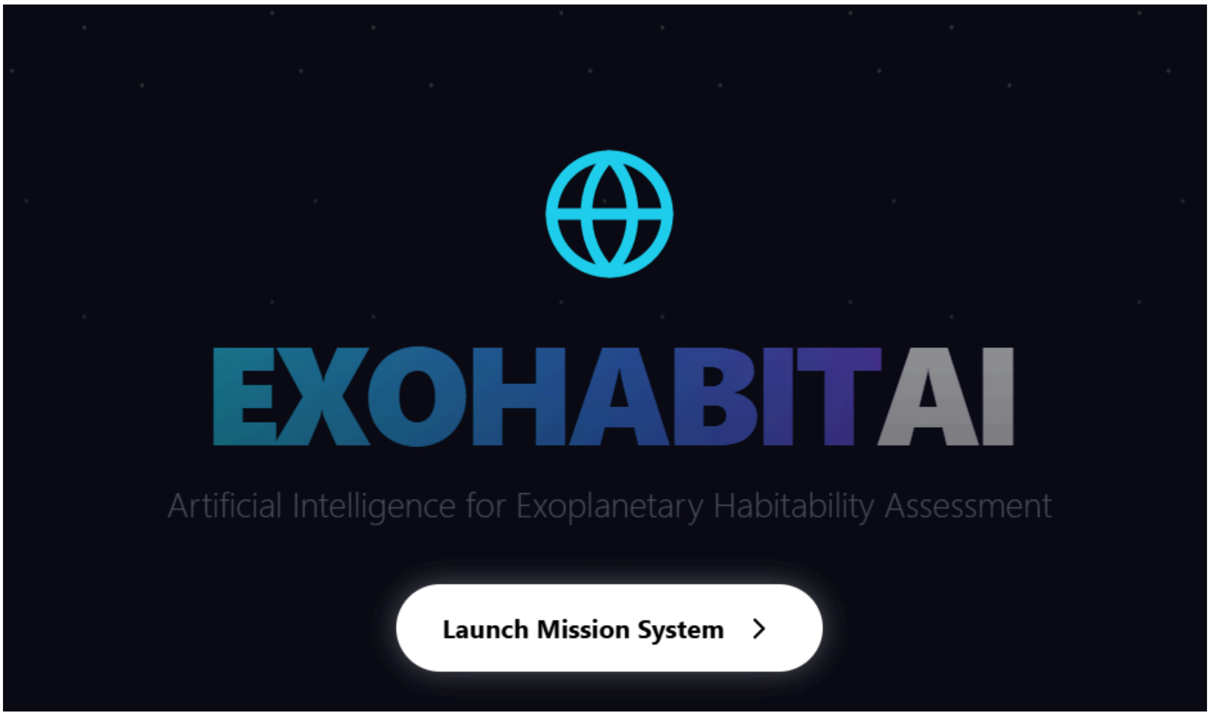
Open a **new terminal**:

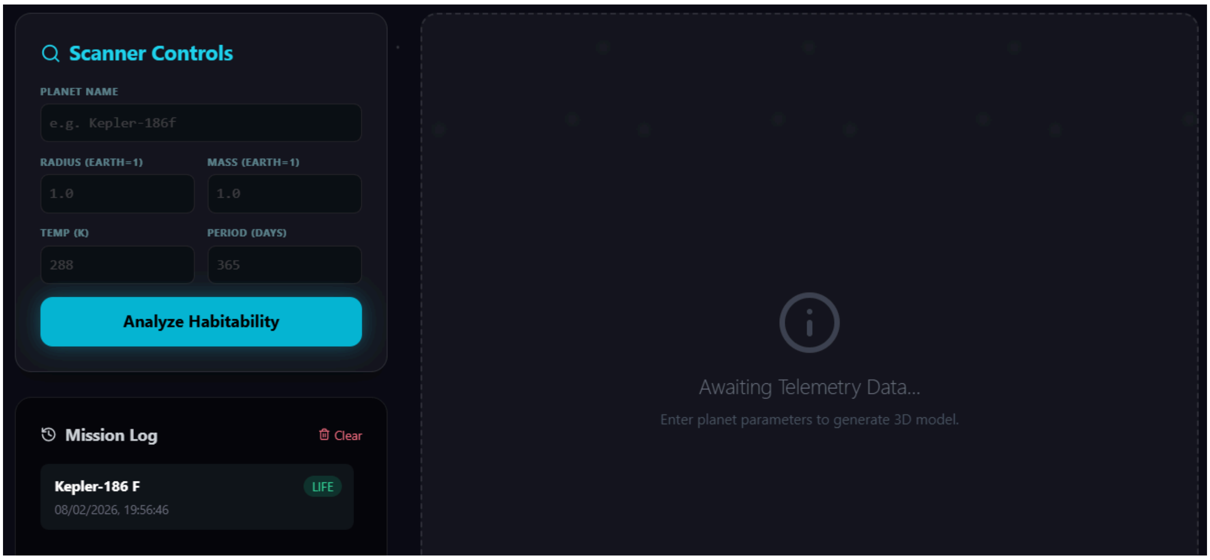
```
Bash
cd frontend
# 1. Install Libraries (including 3D tools)
npm install
npm install three @react-three/fiber@8.16.8 @react-three/drei
--legacy-peer-deps
# 2. Launch UI
npm run dev
```

*App is live at: http://localhost:5173*

**ScreenShorts:**


ExoHabitAI
Mission Control
Analyzer
Scanner Controls
PLANET NAME
Kepler-186 F
RADIUS (EARTH=1)
MASS (EARTH=1)
1
1
TEMP (K)
PERIOD (DAYS)
300
360
Analyze Habitability
Mission Log
Clear
Kepler-186 F
LIFE
08/02/2026, 19:56:46
TARGET ANALYSIS
Potentially Habitable
CONFIDENCE: 100.0%
Save Data


EXOHABITAI
Artificial Intelligence for Exoplanetary Habitability Assessment
Launch Mission System


Scanner Controls
PLANET NAME
e.g. Kepler-186f
RADIUS (EARTH=1)
MASS (EARTH=1)
1.0
1.0
TEMP (K)
PERIOD (DAYS)
288
365
Analyze Habitability
Mission Log
Clear
Kepler-186 F
LIFE
08/02/2026, 19:56:46
Awaiting Telemetry Data...
Enter planet parameters to generate 3D model.