High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components of **BoostHouse - House Price Prediction Platform**. The platform provides accurate house price predictions using XGBoost, exposes a REST API via FastAPI, offers an interactive frontend with Streamlit, and supports model deployment through Docker. Key features include feature importance visualization and hyperparameter tuning.

---

1. System Architecture Overview

Architecture Description:
BoostHouse is a modular, containerized solution comprising a machine learning model backend, a REST API, and an interactive frontend. Components communicate over HTTP within a Dockerized environment.

Module	Description
ML Model Service	Trains and serves XGBoost model; exposes prediction and tuning endpoints.
REST API (FastAPI)	Handles client requests, routes to ML service, manages input/output.
Frontend (Streamlit)	User interface for predictions, feature visualization, and tuning.
Docker Environment	Containerizes all components for deployment and scalability.

---

2. Component Interactions

Sequence Step	Interaction Description
1	User interacts with Streamlit frontend (inputs features, requests predictions/tuning).
2	Frontend sends HTTP requests to FastAPI REST API.
3	REST API processes requests, invokes ML Model Service for predictions/tuning.
4	ML Model Service returns results (predictions, feature importances, tuning outcomes).
5	REST API relays results to frontend for user display.

---

3. Data Flow Overview

Data Flow	Source	Destination	Description
User Input	Frontend	REST API	House features, tuning parameters.
Prediction Request	REST API	ML Model Service	Feature data for price prediction.
Prediction Response	ML Model Service	REST API	Predicted price, feature importances.

Visualization Data	REST API	Frontend	Feature importance, tuning results.

---

4. Technology Stack

Layer/Component	Technology/Framework
ML Model	XGBoost, scikit-learn
API	FastAPI (Python)
Frontend	Streamlit (Python)
Containerization	Docker
Data Processing	pandas, numpy

---

5. Scalability & Reliability

- **Scalability:**
  - Docker enables horizontal scaling of API and model services.
  - Stateless REST API allows load balancing.
- **Reliability:**
  - Containerization ensures consistent deployment.
  - Health checks and logging for monitoring.
- **Security:**
  - API input validation via FastAPI.
  - Network isolation between containers.

End of Document