Low Level Design Document

This Low Level Design (LLD) document outlines the core implementation details for **BoostHouse - House Price Prediction Platform**. The platform provides house price predictions using XGBoost, exposes a REST API (FastAPI), offers an interactive frontend (Streamlit), and supports model deployment via Docker. Key features include feature importance visualization and hyperparameter tuning.

---

1. System Components

Component	Technology	Key Responsibilities
ML Model Service	XGBoost, scikit-learn	Train, predict, tune, and persist models
REST API	FastAPI	Expose endpoints for prediction, training, tuning
Frontend UI	Streamlit	User interaction, data input, visualization
Containerization	Docker	Deployment, environment consistency

---

2. Class/Interface Overview

Class/Module	Responsibility	Key Methods/Attributes
`HousePriceModel`	Model training, prediction, tuning	`train()`, `predict()`, `tune()`, `save()`, `load()`
`APIController`	REST endpoint logic	`predict_endpoint()`, `train_endpoint()`, `tune_endpoint()`
`FeatureImportance`	Compute/visualize feature importances	`get_importance()`, `plot_importance()`
`StreamlitApp`	UI logic	`main()`, `display_form()`, `show_results()`

Relationships:

- `APIController` uses `HousePriceModel` and `FeatureImportance`.
- `StreamlitApp` interacts with `APIController` via HTTP.

---

3. Data Structure Overview

Model/Schema	Fields
HouseFeatures	Numeric/categorical fields (e.g., area, rooms, location)
PredictionRequest	`features: HouseFeatures`
PredictionResponse	`predicted_price: float`, `feature_importance: dict`

HyperparamConfig	`params: dict` (e.g., `max_depth`, `learning_rate`, etc.)

---

4. Algorithms/Logic

Prediction Flow (Pseudocode):

```
def predict_endpoint(request):
    features = parse_features(request)
    model = HousePriceModel.load()
    price = model.predict(features)
    importance = FeatureImportance.get_importance(model)
    return PredictionResponse(price, importance)
```

Hyperparameter Tuning (Summary):

- Accept hyperparameter config via API/UI.
- Use cross-validation to select best parameters.
- Retrain and persist the model.

---

5. Error Handling

Scenario	Handling Approach
Invalid input data	Return 400 Bad Request with error message
Model not trained/loaded	Return 503 Service Unavailable, log error
Prediction failure	Return 500 Internal Server Error, log details
Hyperparameter tuning error	Return 422 Unprocessable Entity, log error
Frontend/API connection failure	Show user-friendly error in UI, retry option

---

End of Document