Low Level Design Document

Introduction

This Low Level Design (LLD) document details the architecture and implementation plan for **ChurnGuard - Customer Churn Prediction API**. The project delivers a FastAPI-based backend for predicting customer churn using an XGBoost model, providing endpoints for prediction, model explanation (SHAP), and API documentation. The system is containerized with Docker and deployable on Hugging Face Spaces.

---

1. System Components

Component	Description	Key Responsibilities
FastAPI App	REST API backend	Expose endpoints, request/response handling
XGBoost Model	Pre-trained ML model	Predict churn
SHAP Explainer	Model interpretability tool	Generate feature attributions
Docker Container	Deployment unit	Encapsulate app and dependencies
Hugging Face Space	Hosting environment	Public deployment

---

2. Class/Interface Overview

Class/Module	Description	Key Methods/Attributes
`ChurnModel`	Model wrapper	`predict(input_data)`, `explain(input_data)`
`PredictionRequest`	Pydantic schema	Customer feature fields
`PredictionResponse`	Pydantic schema	`churn_probability`, `prediction`
`FastAPI`	API framework	`@app.post`, `@app.get` endpoints

Relationships:

- `FastAPI` endpoints use `ChurnModel` for predictions and explanations.
- Request/response schemas validate and serialize data.

---

3. Data Structure Overview

Model	Fields / Structure
`PredictionRequest`	Customer features (e.g., age, tenure, usage, etc.)
`PredictionResponse`	`churn_probability: float`, `prediction: bool`
SHAP Output	`feature_importances: dict[str, float]`

4. Algorithms / Logic

Prediction Flow:

```
@app.post("/predict")
def predict(request: PredictionRequest):
    input_data = request.dict()
    probability = model.predict(input_data)
    prediction = probability > THRESHOLD
    return PredictionResponse(churn_probability=probability, prediction=predicti
```

Explanation Flow:

```
@app.post("/explain")
def explain(request: PredictionRequest):
    input_data = request.dict()
    shap_values = model.explain(input_data)
    return {"feature_importances": shap_values}
```

5. Error Handling

Scenario	Handling Approach
Invalid input schema	Return 422 Unprocessable Entity (FastAPI)
Model not loaded	Return 500 Internal Server Error
Prediction/explanation failure	Return 500 with error message
Unexpected exceptions	Log error, return generic 500 response

End of Document