High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **ChurnGuard - Customer Churn Prediction API**. ChurnGuard is a FastAPI-based backend service that predicts customer churn using an XGBoost model, provides model explanations via SHAP, and exposes endpoints for integration and testing. The solution is containerized with Docker and deployable on Hugging Face Spaces.

1. System Architecture Overview

Architecture Description:
ChurnGuard consists of a RESTful API backend (FastAPI) serving prediction and explanation endpoints. The API loads a pre-trained XGBoost model and SHAP explainer at startup. The service is containerized for deployment and exposes a Swagger UI for interactive testing.

Main System Components:

Component	Role/Responsibility
FastAPI Server	Hosts REST API endpoints for prediction & explanation
XGBoost Model	Performs customer churn prediction
SHAP Explainer	Provides model interpretability (feature attributions)
Docker Container	Encapsulates the application for deployment
Swagger UI	Interactive API documentation & testing interface
Hugging Face Spaces	Deployment platform for public access

2. Component Interactions

Sequence Step	Interaction Description
1	Client sends request (prediction/explanation) to FastAPI endpoint
2	FastAPI validates and preprocesses input data
3	FastAPI invokes XGBoost model for prediction
4	(If explanation requested) FastAPI uses SHAP to generate feature attributions
5	FastAPI returns prediction and/or explanation to client
6	Swagger UI allows users to test endpoints interactively

3. Data Flow Overview

Data Flow Step	Source	Destination	Description
Input Data Submission	Client	FastAPI	Customer data sent via REST API
Model Inference	FastAPI	XGBoost Model	Input data passed for churn prediction
Explanation Request	FastAPI	SHAP Explainer	Generates feature importance/explanation
Response Delivery	FastAPI	Client	Returns prediction and/or explanation

---

4. Technology Stack

Technology	Purpose
Python	Core programming language
FastAPI	REST API framework
XGBoost	Machine learning model for churn
SHAP	Model interpretability/explanation
Docker	Containerization and deployment
Swagger UI	API documentation/testing
Hugging Face Spaces	Cloud deployment platform

---

5. Scalability & Reliability

- **Stateless API:** Each request is independent, enabling horizontal scaling via container orchestration.
- **Containerization:** Docker ensures consistent deployment across environments.
- **Model Loading:** Models are loaded at startup to minimize inference latency.
- **Security:** Input validation and CORS configuration recommended; sensitive data should be protected.
- **Reliability:** Health check endpoints and logging should be implemented for monitoring.

End of Document