🧠 Custom xG & Betting Intelligence System

Technical Specification v2.0 - Complete Implementation Guide

📋 Document Control

Field	Value
Document Title	xG & Betting Intelligence System - Technical Specification
Version	2.0
Owner	Pamui Afrika
Classification	Internal - Confidential
Status	Production Ready
Last Updated	June 17, 2025
Next Review	September 17, 2025
Stakeholders	Engineering, Data Science, DevOps, Product

🎯 Executive Summary

The Custom xG & Betting Intelligence System v2.0 is a comprehensive AI-powered platform designed to:

1. **Generate superior football predictions** through multi-modal data fusion and contextual modeling
2. **Identify high-value betting opportunities** using advanced market analysis and value detection
3. **Execute intelligent betting strategies** via autonomous AI agents with integrated risk management
4. **Continuously learn and adapt** through feedback loops and performance optimization

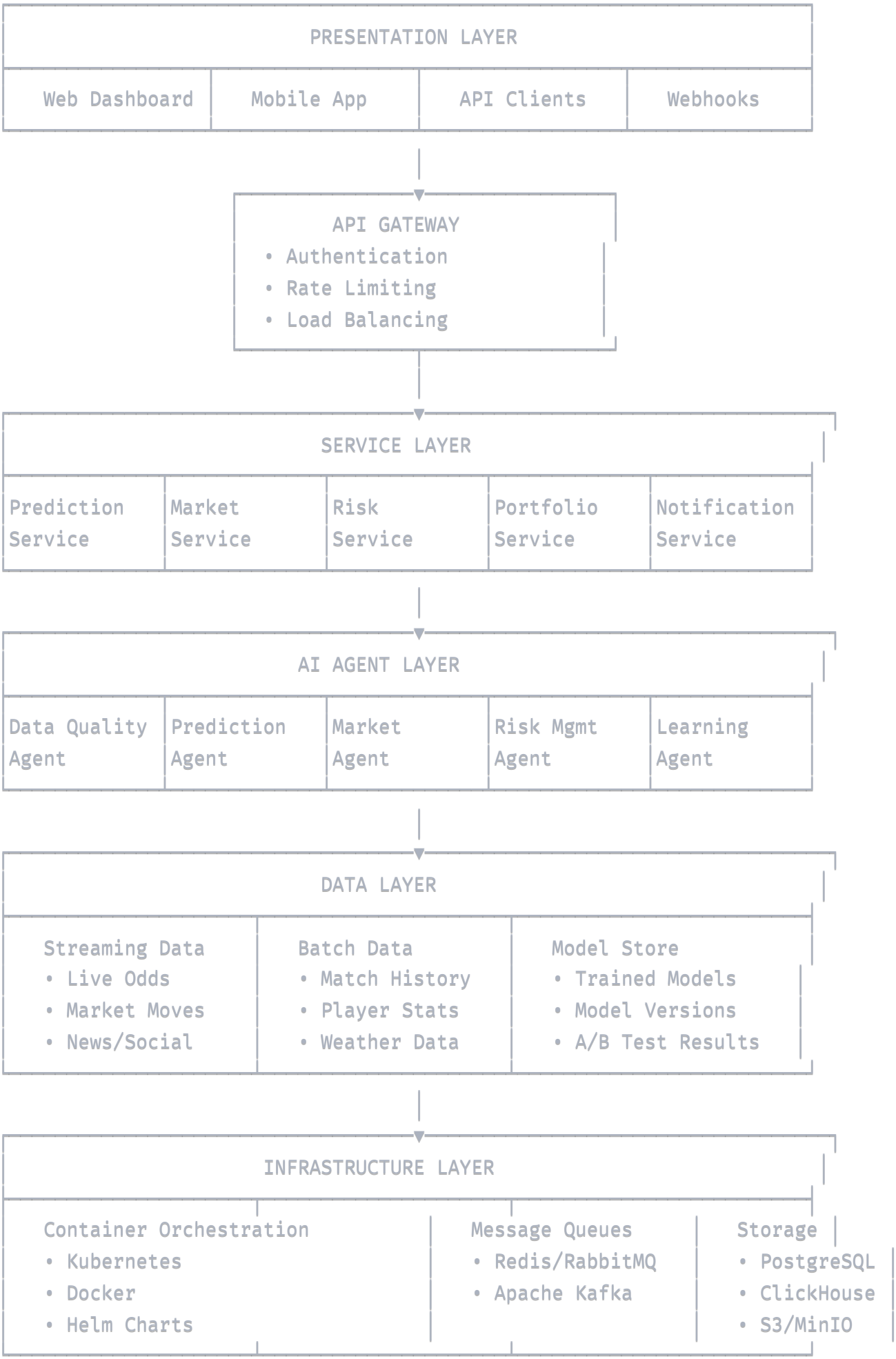
Key Performance Targets:

- **Prediction Accuracy**: >55% on match outcomes, >0.85 xG correlation
- **Financial Performance**: >15% annual ROI with <10% maximum drawdown
- **System Performance**: <100ms prediction latency, >99.5% uptime
- **Data Quality**: <1% missing data, real-time anomaly detection

---

🏗️ System Architecture

1. High-Level Architecture Diagram


PRESENTATION LAYER
Web Dashboard
Mobile App
API Clients
Webhooks
API GATEWAY
• Authentication
• Rate Limiting
• Load Balancing
SERVICE LAYER
Prediction Service
Market Service
Risk Service
Portfolio Service
Notification Service
AI AGENT LAYER
Data Quality Agent
Prediction Agent
Market Agent
Risk Mgmt Agent
Learning Agent
DATA LAYER
Streaming Data
• Live Odds
• Market Moves
• News/Social
Batch Data
• Match History
• Player Stats
• Weather Data
Model Store
• Trained Models
• Model Versions
• A/B Test Results
INFRASTRUCTURE LAYER
Container Orchestration
• Kubernetes
• Docker
• Helm Charts
Message Queues
• Redis/RabbitMQ
• Apache Kafka
Storage
• PostgreSQL
• ClickHouse
• S3/MinIO

2. Technology Stack

Layer	Technology	Purpose	Justification
Frontend	React + TypeScript	Web Dashboard	Modern, type-safe UI development
API Gateway	Kong/Traefik	Request routing	High-performance API management
Backend Services	FastAPI + Python 3.11	Core business logic	Async performance, auto-documentation
AI/ML Framework	PyTorch + Scikit-learn + XGBoost	Model training/inference	Flexibility + production stability
Streaming	Apache Kafka + Redis	Real-time data	High-throughput message processing
Databases	PostgreSQL + ClickHouse + Redis	Data storage	OLTP + Analytics + Caching
Model Serving	MLflow + Ray Serve	Model deployment	Version control + scalable inference
Orchestration	Apache Airflow	Data pipelines	Robust workflow management
Monitoring	Prometheus + Grafana	System monitoring	Industry-standard observability
Container	Docker + Kubernetes	Deployment	Scalable, cloud-native deployment
CI/CD	GitHub Actions	Automation	Integrated development workflow

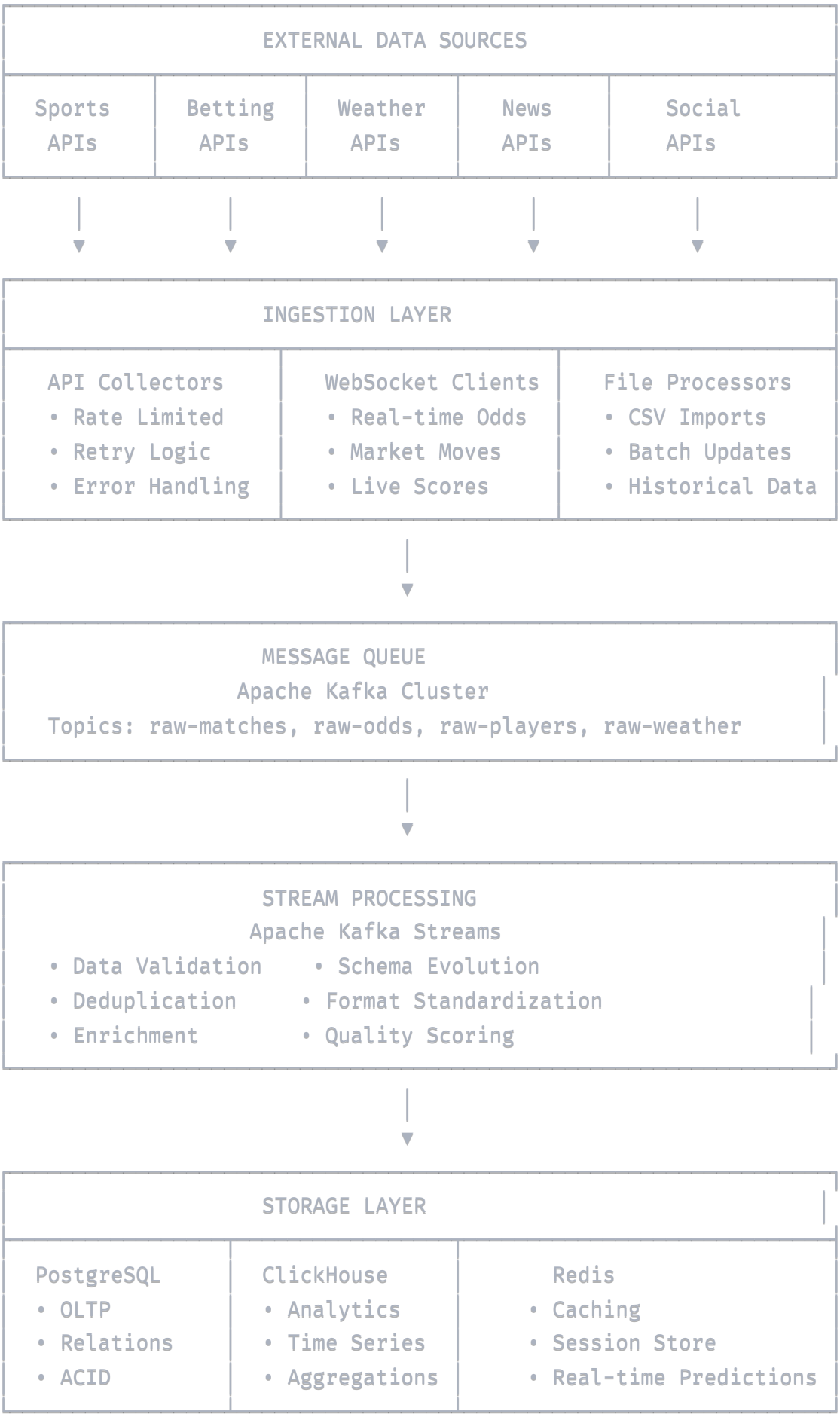
---

📊 Data Architecture

1. Data Sources & Integration

Category	Source	Frequency	Format	Critical Path
Match Data	FBref, Understat	Daily	JSON/CSV	Yes
Odds Data	OddsAPI, Betfair	Real-time	JSON	Yes
Player Stats	SofaScore, Transfermarkt	Daily	JSON	No
Weather	OpenWeatherMap	Hourly	JSON	No
News/Social	NewsAPI, Twitter	Real-time	JSON	No
Market Movements	Pinnacle, Betfair	Real-time	WebSocket	Yes

2. Data Flow Architecture



3. Database Schema Design

Core Tables (PostgreSQL)

sql

```sql
-- Leagues
CREATE TABLE leagues (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL,
    country VARCHAR(50) NOT NULL,
    tier INTEGER NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Teams
CREATE TABLE teams (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL,
    short_name VARCHAR(10),
    league_id UUID REFERENCES leagues(id),
    founded_year INTEGER,
    venue VARCHAR(100),
    city VARCHAR(50),
    country VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Players
CREATE TABLE players (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) NOT NULL,
    team_id UUID REFERENCES teams(id),
    position VARCHAR(20),
    nationality VARCHAR(50),
    birth_date DATE,
    height_cm INTEGER,
    weight_kg INTEGER,
    market_value_eur INTEGER,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Matches
CREATE TABLE matches (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    league_id UUID REFERENCES leagues(id),
    season VARCHAR(10) NOT NULL,
    matchday INTEGER,
    home_team_id UUID REFERENCES teams(id),
```

```
    away_team_id UUID REFERENCES teams(id),
    match_date TIMESTAMP NOT NULL,
    venue VARCHAR(100),
    referee VARCHAR(100),
    attendance INTEGER,
    home_score INTEGER,
    away_score INTEGER,
    match_status VARCHAR(20) DEFAULT 'scheduled',
    importance_score FLOAT DEFAULT 1.0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT unique_match UNIQUE (home_team_id, away_team_id, match_date)
);

-- Match Statistics
CREATE TABLE match_stats (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    match_id UUID REFERENCES matches(id),
    team_id UUID REFERENCES teams(id),
    possession_pct FLOAT,
    shots INTEGER,
    shots_on_target INTEGER,
    shots_off_target INTEGER,
    shots_blocked INTEGER,
    corners INTEGER,
    offsides INTEGER,
    fouls INTEGER,
    yellow_cards INTEGER,
    red_cards INTEGER,
    passes INTEGER,
    passes_accurate INTEGER,
    crosses INTEGER,
    crosses_accurate INTEGER,
    long_balls INTEGER,
    long_balls_accurate INTEGER,
    xg FLOAT,
    xga FLOAT,
    npxg FLOAT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Shots
CREATE TABLE shots (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    match_id UUID REFERENCES matches(id),
    team_id UUID REFERENCES teams(id),
```

```sql
    player_id UUID REFERENCES players(id),
    minute INTEGER NOT NULL,
    x_coord FLOAT NOT NULL,
    y_coord FLOAT NOT NULL,
    shot_type VARCHAR(20), -- 'Open Play', 'Set Piece', 'Penalty', 'Own Goal'
    body_part VARCHAR(20), -- 'Right Foot', 'Left Foot', 'Head', 'Other'
    technique VARCHAR(20), -- 'Normal', 'Volley', 'Half Volley', 'Lob', 'Overhead'
    under_pressure BOOLEAN DEFAULT FALSE,
    one_on_one BOOLEAN DEFAULT FALSE,
    outcome VARCHAR(20), -- 'Goal', 'Saved', 'Off Target', 'Blocked', 'Post'
    xg FLOAT NOT NULL,
    psxg FLOAT, -- Post-shot xG
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Odds (Current and Historical)
CREATE TABLE odds (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    match_id UUID REFERENCES matches(id),
    bookmaker VARCHAR(50) NOT NULL,
    market_type VARCHAR(30) NOT NULL, -- '1X2', 'Over/Under', 'Asian Handicap'
    selection VARCHAR(50) NOT NULL,
    odds_decimal FLOAT NOT NULL,
    odds_american INTEGER,
    implied_probability FLOAT,
    timestamp TIMESTAMP NOT NULL,
    is_closing BOOLEAN DEFAULT FALSE,
    volume BIGINT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Odds Movements (High-frequency data)
CREATE TABLE odds_movements (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    match_id UUID REFERENCES matches(id),
    bookmaker VARCHAR(50) NOT NULL,
    market_type VARCHAR(30) NOT NULL,
    selection VARCHAR(50) NOT NULL,
    old_odds FLOAT NOT NULL,
    new_odds FLOAT NOT NULL,
    change_pct FLOAT NOT NULL,
    timestamp TIMESTAMP NOT NULL,
    trigger_reason VARCHAR(100), -- 'Large bet', 'News', 'Injury', 'System'
    volume_before BIGINT,
    volume_after BIGINT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```sql
-- Player Ratings & Performance
CREATE TABLE player_performances (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    match_id UUID REFERENCES matches(id),
    player_id UUID REFERENCES players(id),
    team_id UUID REFERENCES teams(id),
    minutes_played INTEGER DEFAULT 0,
    position VARCHAR(20),
    rating FLOAT,
    goals INTEGER DEFAULT 0,
    assists INTEGER DEFAULT 0,
    shots INTEGER DEFAULT 0,
    shots_on_target INTEGER DEFAULT 0,
    key_passes INTEGER DEFAULT 0,
    passes INTEGER DEFAULT 0,
    passes_accurate INTEGER DEFAULT 0,
    crosses INTEGER DEFAULT 0,
    crosses_accurate INTEGER DEFAULT 0,
    dribbles INTEGER DEFAULT 0,
    dribbles_successful INTEGER DEFAULT 0,
    tackles INTEGER DEFAULT 0,
    tackles_successful INTEGER DEFAULT 0,
    interceptions INTEGER DEFAULT 0,
    clearances INTEGER DEFAULT 0,
    aerial_duels INTEGER DEFAULT 0,
    aerial_duels_won INTEGER DEFAULT 0,
    fouls_committed INTEGER DEFAULT 0,
    fouls_drawn INTEGER DEFAULT 0,
    yellow_cards INTEGER DEFAULT 0,
    red_cards INTEGER DEFAULT 0,
    xg FLOAT DEFAULT 0,
    xa FLOAT DEFAULT 0, -- Expected Assists
    npxg FLOAT DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Team Form & Ratings
CREATE TABLE team_ratings (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    team_id UUID REFERENCES teams(id),
    rating_date DATE NOT NULL,
    elo_rating FLOAT,
    spi_rating FLOAT,
    attack_strength FLOAT,
    defense_strength FLOAT,
    home_advantage FLOAT DEFAULT 0.0,
```

```
    form_l5 VARCHAR(5), -- 'WWLDW' format
    goals_scored_l10 INTEGER,
    goals_conceded_l10 INTEGER,
    xg_l10 FLOAT,
    xga_l10 FLOAT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT unique_team_date UNIQUE (team_id, rating_date)
);

-- Injuries & Suspensions
CREATE TABLE player_unavailability (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    player_id UUID REFERENCES players(id),
    type VARCHAR(20) NOT NULL, -- 'Injury', 'Suspension', 'International Duty'
    reason VARCHAR(200),
    start_date DATE NOT NULL,
    expected_return_date DATE,
    actual_return_date DATE,
    severity VARCHAR(20), -- 'Minor', 'Moderate', 'Major', 'Season Ending'
    body_part VARCHAR(50), -- For injuries
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Weather Conditions
CREATE TABLE match_weather (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    match_id UUID REFERENCES matches(id),
    temperature_celsius FLOAT,
    humidity_pct FLOAT,
    wind_speed_kmh FLOAT,
    wind_direction_degrees FLOAT,
    precipitation_mm FLOAT,
    weather_description VARCHAR(100),
    visibility_km FLOAT,
    pressure_hpa FLOAT,
    recorded_at TIMESTAMP NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Travel & Fatigue Analysis
CREATE TABLE team_travel (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    match_id UUID REFERENCES matches(id),
    team_id UUID REFERENCES teams(id),
    previous_match_city VARCHAR(100),
```

```
    current_match_city VARCHAR(100),
    travel_distance_km FLOAT,
    travel_time_hours FLOAT,
    days_rest INTEGER,
    timezone_change_hours FLOAT DEFAULT 0,
    travel_method VARCHAR(50), -- 'Bus', 'Flight', 'Train'
    fatigue_score FLOAT, -- Calculated composite score
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tactical Analysis
CREATE TABLE tactical_setups (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    match_id UUID REFERENCES matches(id),
    team_id UUID REFERENCES teams(id),
    formation VARCHAR(10) NOT NULL,
    style_attacking VARCHAR(50), -- 'Counter-attack', 'Possession', 'Direct'
    style_defensive VARCHAR(50), -- 'High Press', 'Mid Block', 'Low Block'
    pressing_intensity FLOAT, -- 0-100 scale
    tempo VARCHAR(20), -- 'Slow', 'Medium', 'Fast'
    width VARCHAR(20), -- 'Narrow', 'Medium', 'Wide'
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Analytics Tables (ClickHouse)

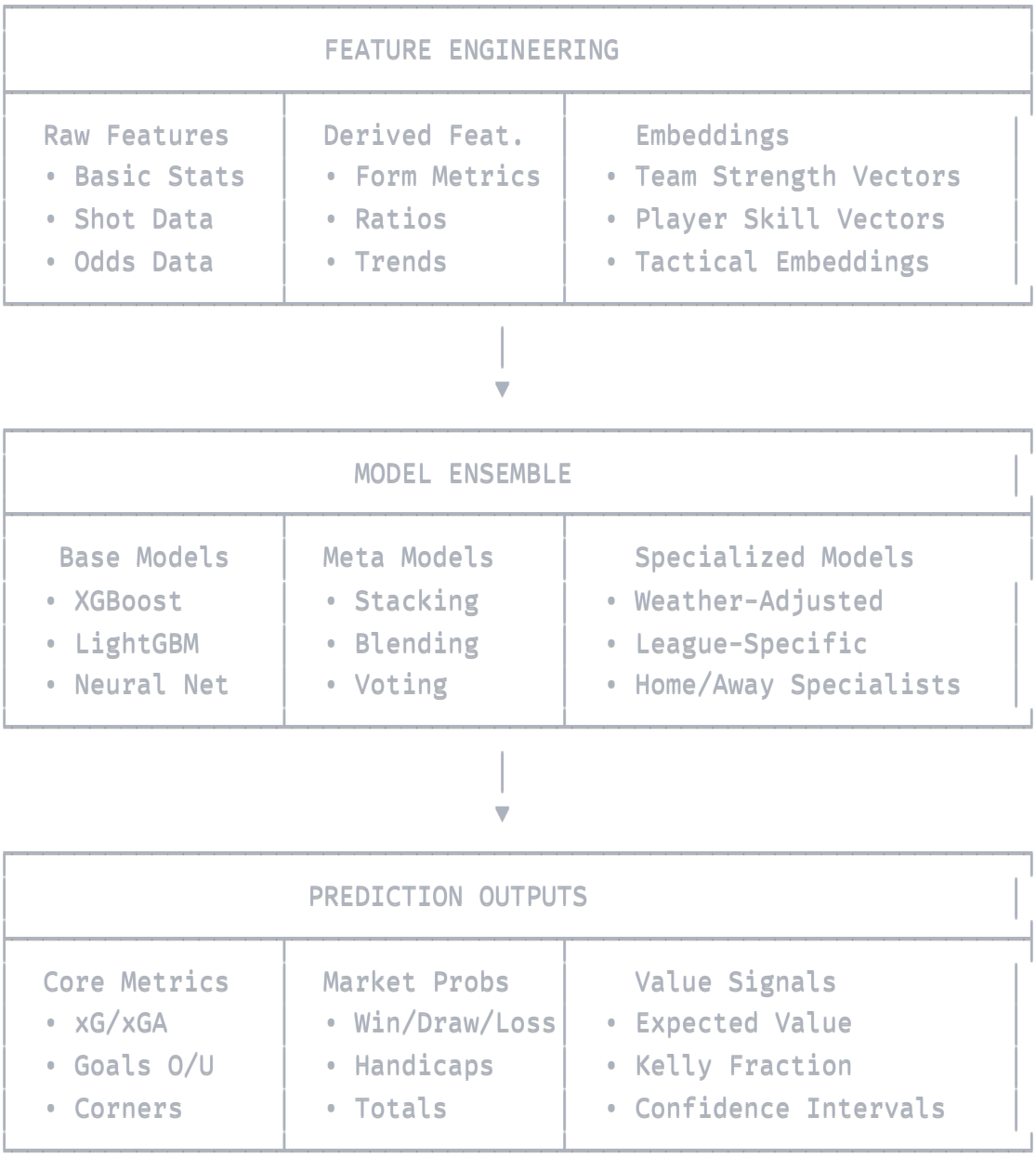
sql

```sql
-- High-frequency odds data for market analysis
CREATE TABLE odds_ticks (
    match_id String,
    bookmaker String,
    market_type String,
    selection String,
    odds_decimal Float64,
    timestamp DateTime64(3),
    volume UInt64,
    created_date Date DEFAULT toDate(timestamp)
) ENGINE = MergeTree()
PARTITION BY created_date
ORDER BY (match_id, bookmaker, market_type, timestamp);

-- Aggregated market movements for analysis
CREATE MATERIALIZED VIEW market_movements_hourly
ENGINE = AggregatingMergeTree()
PARTITION BY toYYYYMM(timestamp)
ORDER BY (match_id, bookmaker, market_type, toStartOfHour(timestamp))
AS SELECT
    match_id,
    bookmaker,
    market_type,
    selection,
    toStartOfHour(timestamp) as hour,
    avgState(odds_decimal) as avg_odds,
    minState(odds_decimal) as min_odds,
    maxState(odds_decimal) as max_odds,
    sumState(volume) as total_volume,
    countState() as tick_count
FROM odds_ticks
GROUP BY match_id, bookmaker, market_type, selection, hour;

-- Model predictions and performance tracking
CREATE TABLE model_predictions (
    id String,
    match_id String,
    model_name String,
    model_version String,
    prediction_type String, -- 'xG', 'Match_Outcome', 'Goals_O/U'
    home_prediction Float64,
    away_prediction Float64,
    draw_prediction Float64,
    confidence_score Float64,
    features Map(String, Float64),
    prediction_timestamp DateTime64(3),
```

```
    actual_outcome Nullable(String),
    actual_home_score Nullable(UInt8),
    actual_away_score Nullable(UInt8),
    created_date Date DEFAULT toDate(prediction_timestamp)
) ENGINE = MergeTree()
PARTITION BY created_date
ORDER BY (match_id, model_name, prediction_timestamp);
```

🤖 AI/ML Architecture

1. Model Pipeline Overview



2. Model Specifications

A. Expected Goals (xG) Model

Model Type: Gradient Boosting (XGBoost) + Neural Network Ensemble

Features (47 total):

python

```
SHOT_FEATURES = [
    # Spatial features
    'x_coord', 'y_coord', 'distance_to_goal', 'angle_to_goal',

    # Temporal features
    'minute', 'time_since_last_shot', 'match_state',

    # Shot characteristics
    'shot_type_encoded', 'body_part_encoded', 'technique_encoded',
    'under_pressure', 'one_on_one', 'fast_break',

    # Player features
    'player_xg_avg_l10', 'player_conversion_rate', 'player_fatigue',
    'player_position_encoded', 'player_form_rating',

    # Team features
    'team_attack_strength', 'team_xg_avg_l10', 'opponent_defense_strength',
    'team_shots_per_game', 'team_possession_avg',

    # Match context
    'score_differential', 'home_advantage', 'weather_impact',
    'rivalry_factor', 'match_importance', 'referee_strictness',

    # Tactical features
    'formation_attacking_width', 'pressing_intensity', 'tempo',
    'defensive_line_height', 'midfield_density'
]

TRAINING_CONFIG = {
    'xgboost_params': {
        'max_depth': 6,
        'learning_rate': 0.1,
        'n_estimators': 1000,
        'subsample': 0.8,
        'colsample_bytree': 0.8,
        'random_state': 42
    },
    'neural_net_params': {
        'hidden_layers': [128, 64, 32],
        'dropout_rate': 0.3,
        'learning_rate': 0.001,
        'batch_size': 512,
        'epochs': 100
    },
    'ensemble_weights': {
        'xgboost': 0.7,
```

```
        'neural_net': 0.3
    }
}
```

B. Match Outcome Model

Model Type: Multi-class Classification (LightGBM + Logistic Regression Ensemble)

Features (63 total):

```python
MATCH_FEATURES = [
    # Team strength metrics
    'home_elo_rating', 'away_elo_rating', 'elo_difference',
    'home_spi_rating', 'away_spi_rating', 'spi_difference',
    'home_attack_strength', 'away_attack_strength',
    'home_defense_strength', 'away_defense_strength',

    # Recent form (last 10 matches)
    'home_points_l10', 'away_points_l10',
    'home_goals_scored_l10', 'away_goals_scored_l10',
    'home_goals_conceded_l10', 'away_goals_conceded_l10',
    'home_xg_l10', 'away_xg_l10', 'home_xga_l10', 'away_xga_l10',

    # Head-to-head history
    'h2h_home_wins_l5', 'h2h_draws_l5', 'h2h_away_wins_l5',
    'h2h_avg_goals', 'h2h_home_advantage',

    # Player availability
    'home_key_players_missing', 'away_key_players_missing',
    'home_squad_value_available', 'away_squad_value_available',

    # Contextual factors
    'rest_days_difference', 'travel_distance_difference',
    'weather_score', 'referee_home_bias', 'crowd_size_expected',
    'match_importance_home', 'match_importance_away',

    # Market intelligence
    'opening_odds_home', 'opening_odds_draw', 'opening_odds_away',
    'odds_movement_home_24h', 'odds_movement_draw_24h', 'odds_movement_away_24h',
    'market_confidence', 'sharp_money_direction'
]
```

C. Value Betting Model

Model Type: Binary Classification + Regression for Expected Value

Purpose: Identify bets where model probability > implied odds probability

python

```python
VALUE_BET_FEATURES = [
    # Model predictions
    'model_home_prob', 'model_draw_prob', 'model_away_prob',
    'model_confidence_score', 'ensemble_agreement',

    # Market features
    'best_odds_home', 'best_odds_draw', 'best_odds_away',
    'avg_odds_home', 'avg_odds_draw', 'avg_odds_away',
    'odds_std_deviation', 'market_efficiency_score',

    # Historical performance
    'model_accuracy_similar_matches', 'market_overreaction_history',
    'bookmaker_bias_home', 'bookmaker_bias_away',

    # Timing features
    'hours_to_kickoff', 'odds_stability', 'late_money_direction',

    # Kelly Criterion inputs
    'bankroll_fraction_recommended', 'expected_value',
    'downside_risk', 'upside_potential'
]

def calculate_value_bet_score(model_prob, best_odds, confidence):
    implied_prob = 1 / best_odds
    edge = model_prob - implied_prob
    kelly_fraction = (model_prob * best_odds - 1) / (best_odds - 1)

    # Adjust for confidence
    adjusted_kelly = kelly_fraction * confidence

    # Risk-adjusted value score
    value_score = edge * confidence * np.log(best_odds)

    return {
        'value_score': value_score,
        'edge': edge,
        'kelly_fraction': max(0, adjusted_kelly),
        'recommended_stake': min(adjusted_kelly, 0.02)  # Max 2% of bankroll
    }
```

3. Model Training Pipeline

python

```python
# Airflow DAG for model training
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from datetime import datetime, timedelta

def train_xg_model():
    """Train xG model with latest data"""
    # Data preparation
    df = load_shot_data(days_back=365)
    df = engineer_shot_features(df)

    # Train-test split (temporal)
    train_df = df[df['match_date'] < '2025-01-01']
    test_df = df[df['match_date'] >= '2025-01-01']

    # Model training
    xgb_model = train_xgboost_model(train_df)
    nn_model = train_neural_net_model(train_df)

    # Ensemble creation
    ensemble = create_xg_ensemble(xgb_model, nn_model)

    # Validation
    test_predictions = ensemble.predict(test_df)
    metrics = calculate_metrics(test_df['xg_actual'], test_predictions)

    # Model registration
    register_model(ensemble, metrics, 'xg_model_v2')

    return metrics

dag = DAG(
    'model_training_pipeline',
    description='Daily model retraining pipeline',
    schedule
```