# PROJECT ARCHITECTURE & SYSTEM OVERVIEW

Project CHIMERA

---

## Open Architecture, Multi-Source, Multi-Layer Quant Betting System

## 1. Introduction

This document defines the complete architectural structure of the open-architecture horse-racing intelligence platform. The system is composed of modular applications, each responsible for a defined stage of the data, modelling, compliance, and execution pipeline.

The platform is explicitly designed as an open, extensible ecosystem in which:

1. New data providers can be added to the intake stage without structural redesign
2. New modelling methodologies can be developed dynamically
3. Model layers for the Core Engine are not predetermined but emerge from the outputs of ModelForge C
4. Compliance, IP, regulatory obligations, and best-practice governance are embedded into the architecture
5. All modules remain loosely coupled, independently deployable, and horizontally scalable

Each application is identified by a functional name and an alphabetical designation, e.g. IntakeHub A, PrepLayer B, etc.

---

## 2. Modular Application Suite (Naming Conventions)

The architecture consists of the following core applications:

- A – IntakeHub
- B – PrepLayer
- C – ModelForge
- D – TestBench
- E – ArchiveFlow
- F – GuardRail

---

- Core – Core Engine

In formal documentation, modules appear as IntakeHub A, PrepLayer B, etc. In general usage and conversation, the shorter names (IntakeHub, PrepLayer, ModelForge…) are used.

---

## 3. Open Architecture Overview

The project follows a strictly open architecture model with these guiding principles:

Extensible Input Layer

- The system accepts any and all data sources on equal architectural footing
- No data provider, market feed, or external API is hard-coded, prioritised, or considered a core dependency
- Each source is handled via isolated, modular intake adapters inside IntakeHub A
- No limit on the number, type, or diversity of intake modules
- The user defines which data sources will be used at runtime; the platform adapts accordingly

Unified Normalisation Layer

- All disparate intake formats are converted into a unified structured schema in PrepLayer B
- The heterogeneous nature of multi-source data is abstracted into a consistent analytical layer

Dynamic Model Development Layer

- ModelForge C is a full model-workshop environment for building, training, comparing, and refining model logic
- No predefined model logic or agent layers exist at this stage
- All final Core Engine decision-layer definitions will be derived from validated outputs of ModelForge C and TestBench D

---

Robust Back-Testing Layer

- TestBench D operationalises long-horizon simulations, risk assessment, performance evaluation, and comparative model testing

Centralised Research & Documentation Layer

- ArchiveFlow E manages documentation, audio-to-text transcription, versioning, research catalogs, and knowledge storage

Continuous Legal & Regulatory Compliance Layer

- GuardRail F ensures all operations comply with financial regulations, sports-betting legislation, tax requirements, and IP governance
- GuardRail approval is mandatory for Core Engine deployment

Execution Layer (Core Engine)

- Executes validated strategies in real time
- No predefined agents or decision layers until the modelling lifecycle produces them
- All Core Engine structures are the direct outcome of ModelForge C and TestBench D

---

# 4. Application Specifications

## 4.1 IntakeHub A

(Open-Architecture, Provider-Agnostic Multi-Source Data Acquisition Layer)

Purpose

IntakeHub A is the universal data ingestion control centre and orchestration hub for configuring, managing, monitoring, and integrating all external data sources. It operates on a pure open-architecture principle: no data provider, market feed, API, or information source is privileged, hard-coded, or architecturally preferred. Instead, the system is constructed as a completely provider-agnostic framework wherein the user dynamically selects and configures which data sources will power the platform at runtime.

# PROJECT ARCHITECTURE & SYSTEM OVERVIEW

## Project CHIMERA

---

This represents a fundamental departure from traditional quant systems where certain data providers are baked into the architecture as "primary" or "core" sources. IntakeHub A treats every data source—whether it is a major exchange API, a specialist bookmaker feed, a form database, a historical archive, or a privately licensed dataset—as an equal, swappable component within a unified intake framework.

Architectural Philosophy

The design principle is radically simple but architecturally powerful: the system makes no assumptions about data availability, provider reliability, or market structure. Instead of forcing the user to work within the constraints of a predetermined data infrastructure, IntakeHub A inverts the relationship. The user brings their data sources; IntakeHub A adapts to them.

This flexibility is achieved through a plugin-based adapter architecture. Each data provider is integrated via a discrete, isolated intake module (adapter) that implements a standard interface contract. These adapters communicate bidirectionally with IntakeHub's orchestration core, which handles credential management, scheduling, health monitoring, error recovery, and data routing. The modular nature of this design means that:

- New providers can be added or removed without affecting existing integrations
- Providers can be swapped, upgraded, or deprecated at runtime
- Multiple instances of the same provider can run in parallel (e.g., multiple Betfair accounts or different historical data feeds)
- The system degrades gracefully if any single provider fails or becomes unavailable
- No provider lock-in or single point of failure exists

Supported Data Provider Categories (Illustrative, Not Exhaustive)

IntakeHub A is designed to accept data from the following categories of sources. This list is intentionally non-hierarchical and non-exclusive; it represents the breadth of data types the system can ingest, not a ranked set of priorities:

Exchange & Betting Market APIs

- Betfair Exchange API (tick-level market data, traded volumes, order books, historical datasets, exchange-specific data structures)
- Pinnacle Sports API (sharp bookmaker odds, line movements, closing-line data, historical datasets)
- Other licensed or proprietary exchange platforms and betting exchanges

# PROJECT ARCHITECTURE & SYSTEM OVERVIEW

## Project CHIMERA

---

Bookmaker & Odds Provider Feeds

- Direct bookmaker APIs (where available and licensed)
- Odds aggregation platforms (TheOddsAPI, OddsBoom, licensed OddsJam endpoints, similar services)
- Real-time odds streams from multiple concurrent bookmakers
- Historical odds snapshots and line-movement archives

Racing Form & Fixture Data

- Timeform API (form ratings, sectional times, going conditions, historical form databases)
- Racing Post APIs and datasets (where licensed)
- Sporting Solutions API (race cards, fixture data, historical racing information)
- Local or proprietary form databases maintained by users

Third-Party Licensing & Specialist Data

- Sportradar (licensed sports data, event metadata, historical records)
- Bloomberg, Reuters, or similar financial data feeds (if applicable to racing or gambling contexts)
- Academic or research-grade datasets
- Proprietary vendor datasets licensed directly to the user

User-Managed & Local Data Sources

- Locally maintained historical archives in standard formats (CSV, Parquet, JSON, database exports)
- Custom-built or in-house generated datasets
- Scraped or collected data from public sources (subject to legal and ToS compliance)
- Real-time feeds from user-operated data collection systems
- Any structured data file or database the user wishes to integrate

Dashboard & User Interface Features

The IntakeHub A dashboard provides comprehensive visibility and control across all data sources:

- Provider Management Interface: A dynamic control panel for configuring, enabling, disabling, and managing all integrated data providers

# PROJECT ARCHITECTURE & SYSTEM OVERVIEW

Project CHIMERA

---

- API Credential & Authentication Management: Secure input, storage, and rotation of API keys, OAuth tokens, SSL certificates, and other authentication credentials; supports multi-account setups for the same provider
- Module Discovery & Installation View: Displays installed provider adapters, available adapters ready for installation, and marketplace or registry of new providers
- Provider Health Indicators: Real-time status dashboards showing connectivity, uptime, data freshness, error rates, and performance metrics for each provider
- Real-Time Ingestion Activity Logs: Live feed of data ingestion events, provider responses, record counts, and operational events with full audit trails
- Performance Metrics & Diagnostics: Throughput statistics, latency measurements, data quality scores, schema validation results, and error frequency analysis per provider
- Integration Test Suite Results: On-demand and automated testing of provider connectivity, schema integrity, authentication, and data completeness; results displayed directly in the UI with pass/fail status and diagnostic details

Backend Architecture & Capabilities

The IntakeHub A backend runs inside Docker as a containerized Python service. It provides the orchestration, data management, and diagnostic infrastructure that enables seamless multi-provider ingestion:

- Provider Adapters: Discrete, isolated modules responsible for connecting to each data source, translating provider-specific protocols (REST APIs, WebSockets, file uploads, database connections, etc.) into a standard internal format
- Credential & Authentication Management Subsystem: Secure vault for storing credentials, tokens, and certificates; automatic refresh of time-limited tokens; support for multiple authentication schemes (API keys, OAuth 2.0, SSL certificates, basic auth, custom headers)
- Schedulers & Orchestration Engine: Manages the timing and frequency of data pulls from various providers (real-time streaming, periodic polling, event-triggered, batch downloads); handles concurrent execution of multiple provider adapters
- High-Frequency Listeners: For providers offering streaming or WebSocket data (e.g., Betfair's stream API), dedicated listener threads maintain persistent connections and capture tick-level or near-real-time market data
- Raw Data Repositories: Persistent storage of raw, unprocessed data from each provider, maintaining a complete audit trail and enabling historical reconstruction or re-processing

- Error Handling & Recovery Subsystem: Sophisticated retry logic, circuit breakers, fallback strategies, and alerting mechanisms for provider outages, authentication failures, or data quality issues
- Diagnostics & Monitoring Engine: Comprehensive logging, metrics collection, and health checks; integration with Gemini for intelligent diagnostic analysis of failures and anomalies
- Data Quality & Schema Validation: Pre-ingestion validation of provider data against expected schemas, flagging malformed records, missing fields, or unexpected data types

Open Architecture Principle in Practice

The radical openness of IntakeHub A means that:

1. No Provider Privilege: Betfair, Pinnacle, Sportradar, user-maintained archives, or any other source are all treated as equivalent components within the same framework. There is no concept of "primary" vs "secondary" data sources at the architectural level.

2. Runtime Flexibility: Users can start with Betfair and Pinnacle, add Timeform and Racing Post mid-project, swap in a custom-built real-time data feed, and simultaneously use local CSV archives—all without restarting or reconfiguring the system.

3. Graceful Degradation: If one provider fails, stops producing data, or becomes unavailable due to regulatory, licensing, or technical reasons, the system continues operating with the remaining sources. ModelForge and the downstream pipeline adapt to whatever data is available.

4. Vendor-Agnostic Growth: As new data sources become available, mature, or gain relevance, they can be integrated by developing a new adapter module. The core architecture requires no modification.

5. Compliance & Licensing Flexibility: Since no provider is baked in, it is straightforward to manage licensing agreements, data usage rights, and regulatory requirements per provider. GuardRail F validates compliance for each source independently.

6. Scalability & Distribution: Multiple instances of the same provider adapter can run in parallel (e.g., load-balanced Betfair connections, multiple concurrent data downloads

from archives), and new providers can be spun up without affecting existing ones.

This design makes IntakeHub A truly futureproof: the system does not depend on the viability or dominance of any single data provider. It is built to evolve indefinitely as the data landscape changes.

---

## 4.2 PrepLayer B

(Data Cleansing, Structuring & Schema Normalisation)

Purpose

PrepLayer B is the data transformation and standardisation layer responsible for converting the heterogeneous, disparate raw outputs from all IntakeHub A providers into a single, unified, model-ready analytical dataset. Because IntakeHub A accepts unlimited provider formats—each with its own field names, data types, temporal resolution, missing-data conventions, and structural quirks—PrepLayer B must perform comprehensive conversion, validation, and harmonisation.

Functions

1. Input Format Detection: Automatically detects the structure, schema, encoding, and format of incoming data from each provider
2. Cleansing and Data Sanitation: Removes or flags invalid records, handles missing values, corrects obvious errors, and removes duplicates
3. Schema Standardisation: Maps provider-specific field names, data types, and structures to a unified internal schema
4. Timestamp Alignment: Converts all temporal data to a common timezone and resolution, resolving timing mismatches across providers
5. Normalisation for Model-Ready Output: Transforms raw values (odds, prices, volumes, ratings) into standardised representations suitable for modelling
6. Validation and Integrity Checks: Performs multi-layered validation to ensure data completeness, consistency, and logical coherence
7. Structured Dataset Export for ModelForge C: Packages validated, normalized data into formats optimized for feature engineering and model development

# PROJECT ARCHITECTURE & SYSTEM OVERVIEW

Project CHIMERA

---

Technology Stack

- Backend: Python (Pandas/Polars for high-performance data manipulation) running in Docker
- Frontend: React dashboard for real-time monitoring and interactive exploration
- Storage: Google Cloud Storage or local database modules for staging and archiving transformed datasets

Dashboard & Monitoring Features

The PrepLayer B dashboard provides comprehensive visibility into the transformation pipeline:

- Mapping Previews: Visual representation of how provider-specific fields are mapped to the unified schema
- Before-and-After Transformation Views: Side-by-side comparison of raw input data vs. normalised output
- Error Logs & Anomaly Reports: Detailed logging of validation failures, missing data, outliers, and data quality issues
- Unified Schema Previews: Display of the target schema structure and data types
- Validation Summaries: Statistics on data completeness, record counts, and transformation success rates per provider and time period

---

## 4.3 ModelForge C

(Model Development & Strategy Generation Workspace)

Purpose

ModelForge C is the research and development laboratory where all modelling logic, trading rules, feature engineering, and strategy concepts are created, tested, iterated, and refined. It operates on the principle that no models, rules, or agents are predetermined. Instead, the system is designed to generate modelling candidates entirely from the available (normalized) data and user-guided experimentation.

# PROJECT ARCHITECTURE & SYSTEM OVERVIEW

## Project CHIMERA

---

Functions

1. Feature Generation and Selection: Tools for creating derived features from raw market data (e.g., moving averages, volatility measures, odds movement indicators, form-based metrics)
2. Rule-Based Logic Builder: Interface for defining conditional logic, entry/exit criteria, and decision trees without requiring code
3. Multi-Parameter Testing Framework: Systematic exploration of parameter combinations to optimize strategy performance
4. Comparative Model Scoring: Side-by-side evaluation and ranking of different model versions and methodologies
5. Access to ArchiveFlow E's Research Database: Direct integration with the research library to inform model development with prior work, regulatory insights, and domain knowledge
6. Structured Export of Candidate Model Configurations: Packaging of completed models for downstream testing in TestBench D
7. No Predetermined Agent or Strategy Structures: The environment is agnostic to any particular modelling paradigm
8. Responsible for Producing All Candidate Logic: Every strategy that may eventually power the Core Engine originates here

Technology Stack

- Backend: Python (scikit-learn, TensorFlow, custom algorithmic libraries) running in Docker for compute-intensive operations
- Frontend: React interactive model-builder interface with drag-and-drop logic composition and real-time performance feedback
- Storage: Local or Cloud-based model configuration repository, version control for model iterations

---

# PROJECT ARCHITECTURE & SYSTEM OVERVIEW

Project CHIMERA

---

## 4.4 TestBench D

(Back-Testing, Evaluation & Stress Testing Layer)

### Purpose

TestBench D is the operational validation platform responsible for rigorously evaluating candidate models produced by ModelForge C against historical datasets, simulated market paths, and real-world cross-provider comparisons. It measures model robustness, stability, and realistic performance under a variety of market conditions.

### Functions

1. Full Historical Back-Testing: Replay of model logic against complete historical datasets from any combination of providers
2. Multi-Year Simulations: Extended backtests spanning multiple years of data to assess consistency and regime changes
3. Cross-Provider Testing: Evaluation of the same model logic against Betfair, Pinnacle, blended provider data, or any other provider combinations to assess strategy sensitivity to data source
4. Drawdown Analysis: Detailed assessment of peak-to-trough losses, recovery times, and risk-adjusted performance metrics
5. Volatility, Stability, Risk Profiling: Comprehensive risk analytics including Sharpe ratio, Sortino ratio, max drawdown, volatility clustering, and tail-risk analysis
6. Monte Carlo Simulation Support: Generation of synthetic market paths to stress-test strategies under scenarios not present in historical data
7. Performance Summary: Aggregated reporting of returns, win rates, risk metrics, and comparative rankings
8. Export of Validated Model Results to GuardRail F for Compliance Verification: Passing of backtest reports and model specifications for legal and regulatory review
9. Export to Core Engine for Deployment (Only After Approval): Successful models, post-GuardRail certification, are packaged for live execution

### Technology Stack

- Backend: Python (heavy compute-intensive operations using NumPy, Pandas, and specialized backtesting libraries) running in Docker
- Optional Frontend: React dashboards for interactive exploration of backtest results

- Storage: Local or cloud repository for historical datasets, backtest results, and model artifacts

Dashboard Features

- Equity Curves: Visual representation of cumulative returns over time
- Drawdown Charts: Visualization of underwater plots and peak-to-trough declines
- Volatility Distributions: Histograms and density plots of returns
- Parameter Outcome Tables: Heatmaps and tables showing performance across parameter ranges
- Finalised Validation Reports: Comprehensive PDF or HTML exports documenting all testing results

---

## 4.5 ArchiveFlow E

(Document Management, Research Processing & Automation)

Purpose

ArchiveFlow E is the centralised research, documentation, transcription, and version control system serving as the institutional knowledge repository for the entire Project Chimera ecosystem. It manages and organizes all documentation, research data, regulatory texts, audio transcripts, and project intelligence, making this body of knowledge fully searchable and directly accessible to downstream modules.

Functions

1. Continuous Audio-to-Text Transcription: Automated conversion of audio recordings (research notes, strategy discussions, compliance reviews) into indexed, searchable text
2. Auto-Organization of Research Data: Intelligent categorisation and tagging of documents, papers, and research materials
3. Cross-Machine Sync for Multi-Location Workflow: Synchronisation of research libraries across multiple machines and locations, supporting distributed development teams
4. Version Control and Archiving: Complete version history and audit trails for all documents and research artifacts

5. Structured Library Accessible Directly by ModelForge C: Integration allowing researchers and model developers to reference prior work, validated features, and institutional knowledge directly within the modelling environment
6. Repository for Regulatory Documents Used by GuardRail F: Centralized storage of compliance frameworks, regulatory documents, and legal assessments used by the compliance layer

Technology Stack

- Backend: Python running in Docker
- Transcription: Whisper (OpenAI) or equivalent speech-to-text service
- Frontend: React interface for browsing, searching, and managing the research library
- Storage: Synced cloud repository (Google Cloud Storage, Git-backed systems, or equivalent) enabling real-time synchronization and version control

---

## 4.6 GuardRail F

(Legal, Compliance, IP Governance & Regulatory Enforcement)

Purpose

GuardRail F ensures that the entire ecosystem—from data ingestion through model development, validation, and real-world execution—adheres to all applicable legislation, regulatory obligations, ethical standards, and intellectual property frameworks. It serves as the mandatory approval gate through which no model can pass into the Core Engine without explicit certification.

Compliance Domains

GuardRail F operates across seven distinct but interconnected compliance domains:

1. Financial-Sector Standards: Ensures adherence to FCA regulations, financial conduct rules, and any applicable banking or securities regulations
2. Sports-Betting Regulations: Validates compliance with Gambling Commission regulations, Licencing and Advertising standards, and jurisdiction-specific betting legislation

3. Data Privacy and PII Protection: Validates anonymisation, data minimization, and storage compliance under GDPR, CCPA, and other privacy frameworks
4. Tax Duties and Obligations: Ensures proper documentation of income, assessment of tax liability, and alignment with 2026–2027 sports betting tax reforms (UK) and other emerging tax frameworks
5. Intellectual Property Verification, Conflict Detection, and Novel-IP Claims: Registers new model logic internally, checks against external IP claims or prior art, and maintains defensive IP library
6. Licensing Agreements and External API Usage Restrictions: Validates compliance with third-party API terms of service, licensing requirements, and data usage restrictions
7. Ethical AI Governance and Security Standards: Ensures model logic does not introduce bias, manipulative strategies, or security vulnerabilities; enforces responsible AI practices

Functions

Compliance Document Processor

- Ingests all regulatory documents, terms of service, API agreements, and licensing terms
- Parses and indexes compliance-relevant text
- Performs semantic compliance checks using NLP (powered by Gemini)
- Flags inconsistencies, violations, or areas of ambiguity for human review

Sports-Betting & Taxation Monitor

- Stores and validates current and projected tax obligations
- Ensures model execution aligns with legal criteria and regulatory requirements
- Maintains up-to-date assessment of changing tax landscape and sports-betting regulations

Financial Conduct Alignment

- Ensures system operations do not violate financial conduct rules, market abuse regulations, or insider trading prohibitions
- Validates that no model employs manipulative, abusive, or deceptive practices

# PROJECT ARCHITECTURE & SYSTEM OVERVIEW

## Project CHIMERA

---

Privacy & Data Protection Enforcement

- Validates anonymisation techniques and storage security
- Ensures compliance with GDPR right-to-erasure, data minimization, and consent frameworks
- Audits handling of personally identifiable information throughout the system

Intellectual Property Management

- Registers new model logic internally with full metadata (creator, date, methodology, backtest results)
- Cross-references against external IP databases and prior art to avoid infringement
- Maintains internal IP library for defensible ownership of novel strategies and methodologies
- Generates IP defensibility reports for institutional governance

Mandatory Approval Layer

- Enforces the principle that no model can enter Core Engine execution without explicit GuardRail F certification
- Produces formal compliance certificates documenting the legal and regulatory basis for strategy deployment
- Maintains audit trails of all approvals, denials, and compliance modifications

Technology Stack

- Backend: Python running in Docker
- NLP & Analysis: Gemini for regulatory text analysis, semantic compliance checking, and anomaly detection
- Frontend: React dashboards for compliance officers and legal reviewers
- Storage: Versioned compliance repository with audit logging and immutable records

---

# PROJECT ARCHITECTURE & SYSTEM OVERVIEW

Project CHIMERA

---

## 4.7 Core Engine

(Final Strategy Execution System)

Purpose

The Core Engine is the real-time execution layer responsible for running validated, approved betting strategies in live market conditions. It does not contain predefined agent layers, decision logic, or execution rules. Instead, every strategy that runs in the Core Engine is the direct output of the ModelForge C → TestBench D → GuardRail F pipeline.

Architectural Principles

1. No Predefined Agent Layers: The Core Engine is a blank canvas for strategy execution; all logic is derived from upstream modules
2. Strategy Logic Entirely Derived From:
    - ModelForge C outputs (the actual trading rules and decision logic)
    - TestBench D validation reports (performance metrics and risk assessment)
    - GuardRail F compliance approval (legal and regulatory clearance)

Functions

1. Real-Time Provider Monitoring: Continuous monitoring of live data streams from all configured data providers (Betfair, Pinnacle, custom feeds, etc.)
2. Real-Time Signal Evaluation: Evaluation of live market data against model logic to generate trading signals
3. Execution of Validated Model Logic: Placement of bets, orders, or positions according to the approved strategy
4. Risk-Control Enforcement: Implementation of position limits, exposure caps, drawdown stops, and other risk management rules
5. Live Logging and Analytics: Comprehensive recording of all trades, signals, and operational events for post-trade analysis and compliance audit
6. Compliance-Checked Operational Posture: Continuous validation that live execution remains aligned with GuardRail F requirements

Technology Stack

- Frontend: React interface (deployed on Cloudflare Edge) for monitoring and manual intervention

---

- Backend: Python multi-agent engine running in Docker, with specialized modules for each data provider and strategy variant
- Storage: Google Cloud Storage for live logs; local caches for low-latency signal computation
- Logging: Centralised structured logging system capturing all operational events for analytics and compliance review

---

## 5. Inter-Application Relationships

Operational Flow

The primary data and model lifecycle flows in the following sequence:

1. IntakeHub A → Raw data acquisition from all configured providers
2. PrepLayer B → Unified, normalised structured datasets
3. ModelForge C → Candidate models and strategies
4. TestBench D → Performance validation and backtesting results
5. GuardRail F → Compliance approval and certification
6. Core Engine → Real-world execution of approved strategies

Research & Compliance Support Flow

ArchiveFlow E provides critical support to both ModelForge C and GuardRail F:

- ArchiveFlow E → ModelForge C: Provides research database, prior work, validated features, institutional knowledge, and domain insights
- ArchiveFlow E → GuardRail F: Supplies regulatory documents, compliance frameworks, legal opinions, and audit trails
- ArchiveFlow E → Core Engine Lifecycle Support: Maintains documentation snapshots and historical records of all deployed strategies for post-trade analysis and audit defensibility

Data Sovereignty & Feedback Loops

While the primary flow is unidirectional (IntakeHub → PrepLayer → ModelForge → TestBench → GuardRail → Core Engine), feedback mechanisms exist:

---

- ModelForge C can request additional features or raw data from PrepLayer B
- TestBench D may identify data quality issues, prompting review in PrepLayer B
- Core Engine operational results are logged and can inform new model hypotheses in ModelForge C
- GuardRail F findings may require additional documentation or modification to model logic

---

## 6. Infrastructure Overview

Execution Environment

The entire system operates within a containerized, cloud-native infrastructure:

1. Backend Computation: Python microservices running in Docker containers, deployable to local hardware or cloud infrastructure (Google Cloud, AWS, Azure, etc.)
2. Frontend Deployment: React + Vite applications deployed to Cloudflare Edge for globally distributed, low-latency access
3. Local Execution: Where performance, confidentiality, or regulatory requirements dictate, services run on local hardware (e.g., on-premises backtesting to avoid cloud exposure)
4. Cloud Execution: Where scalability and distributed computing are required, services scale horizontally in cloud environments

Storage & Data Management

1. Google Cloud Storage: Primary repository for raw datasets, structured analytical data, and processed results from PrepLayer B, ModelForge C, and TestBench D
2. Local Secure Storage: Sensitive compliance data, API credentials, and regulatory documents stored in secure local repositories with encryption and access controls
3. Version-Controlled Research Repository: ArchiveFlow E maintains a fully version-controlled, synced repository (backed by Git or equivalent) for research materials, documentation, and institutional knowledge

# PROJECT ARCHITECTURE & SYSTEM OVERVIEW

Project CHIMERA

---

AI & Language Model Services

1. Gemini: Powers intake-related logic (schema detection, error diagnosis), regulatory text analysis in GuardRail F, and anomaly detection across the platform
2. Whisper: Handles continuous audio-to-text transcription for ArchiveFlow E, converting verbal research and strategy discussions into indexed, searchable documents
3. Cloudflare AI: Lightweight inference tasks appropriate for edge deployment, such as real-time feature computation or signal preprocessing in the Core Engine

Networking & Deployment

- All services are containerized and independently deployable
- Inter-service communication via REST APIs, message queues, and WebSocket connections
- API gateway and load balancing provided by Cloudflare for frontend traffic
- Secure credential management via environment variables, vault services, and OAuth 2.0 where applicable

---

## 7. Governance & Long-Term Operational Strategy

Core Governance Principles

The platform adheres to ongoing principles of:

1. Regulatory Compliance (GuardRail F): All operations are subject to continuous compliance review; regulatory changes trigger automatic model audit and re-certification
2. Ethical AI Use: Strategies are audited for bias, market manipulation, and unintended consequences; responsible AI governance is embedded in GuardRail F
3. Transparent Model Lifecycle Tracking: Every model, from creation through deployment and deactivation, is fully documented and traceable
4. Audit-Ready Documentation via ArchiveFlow E: Complete record of all decisions, approvals, rejections, and modifications available for regulatory or internal audit

---

5. Modular Expansion Without Structural Redesign: New data providers, modelling methodologies, and compliance frameworks can be integrated without architectural changes
6. Intellectual Property Defensibility: Internal IP library, prior-art checking, and documentation ensure defensible ownership of novel strategies
7. Institutional-Grade Risk Controls: Position limits, drawdown stops, circuit breakers, and manual intervention capabilities ensure financial safety

Deployment Criteria for Core Engine

A model or strategy cannot be deployed to the Core Engine execution layer until it satisfies all of the following:

1. TestBench D Validation: Successful completion of multi-year backtests, stress testing, and risk profiling demonstrating acceptable performance characteristics
2. GuardRail F Legal & Regulatory Approval: Explicit certification that the strategy complies with all applicable financial regulations, sports-betting law, data privacy requirements, tax obligations, licensing agreements, and IP considerations
3. ArchiveFlow E Documentation Snapshot: Complete documentation package including model specification, backtest results, compliance certificates, research basis, and audit trail

Accountability & Defensibility

This three-layer deployment gate ensures:

- Accountability: Clear responsibility for each approval stage with full audit trails
- Reproducibility: Complete documentation enables re-running, re-testing, and independent verification
- Defensibility: If questioned by regulators, tax authorities, or other stakeholders, the organisation can demonstrate comprehensive due diligence, proper governance, and responsible AI practices

---

# PROJECT ARCHITECTURE & SYSTEM OVERVIEW

Project CHIMERA

---

## 8. Long-Term Evolutionary Capability

Designed for Longevity

The architecture is intentionally designed to remain relevant and extensible over a multi-decade horizon:

Provider Flexibility: As data sources mature, decline, or new sources emerge, the open-architecture nature of IntakeHub A ensures the platform can evolve without redesign. The loss or gain of a data provider affects only that provider's adapter; the entire system remains functional.

Methodological Openness: ModelForge C makes no assumptions about modelling paradigms. Machine learning, statistical arbitrage, rules-based systems, ensemble methods, and future methodologies can all coexist within the same environment.

Regulatory Agility: GuardRail F is designed to accommodate evolving regulatory landscapes. Tax reforms, new betting regulations, and changing financial conduct rules can be incorporated by updating compliance rules without architectural modifications.

Technological Evolution: The containerized, microservice-based architecture allows individual components to be upgraded, replaced, or refactored without disrupting the entire system. Python versions, library upgrades, and infrastructure migrations can proceed independently per module.

---

### Document Control

- Version: 2.0 (Open-Architecture IntakeHub A Redesign)
- Last Updated: December 2024
- Next Review: Q1 2025