Genomic Foundation Model AI Platform

Complete Implementation Guide

Phases 0–6

January 2026

Department of Computational Medicine & Bioinformatics
University of Michigan

Version 1.0

Contents

Chapter 1

Executive Summary

This document provides a complete implementation guide for building a secure, reproducible, AI-assisted research platform on top of a Genomic Foundation Model (EPCOT-style framework). The platform enables researchers to interact with sophisticated genomic inference tools through natural language while maintaining strict privacy controls, IRB compliance, and scientific reproducibility.

1.1 Core Design Principles

The platform is built on four foundational principles that guide every architectural and implementation decision:

1. **AI as Interpreter, Not Executor:** AI components are used exclusively for interpretation and explanation. All scientific computation occurs in deterministic backend systems, ensuring reproducibility and auditability.
2. **Privacy by Design:** Raw genomic data never crosses the AI boundary. Language models receive only structured summaries, numeric metrics, and metadata—never sequences, matrices, or patient-level information.
3. **Confidence-Aware Communication:** Every prediction includes comprehensive confidence metrics covering model certainty, data quality, context validity, and stability under perturbation. Users always know how much to trust a result.
4. **Reproducibility Guaranteed:** Every analysis run captures complete provenance including model versions, dataset versions, input parameters, and timestamps. Any result can be recreated from stored metadata.

1.2 Implementation Phases

The implementation is divided into seven phases, each building on the previous:

Phase	Name	Primary Deliverable
0	Planning & Ground Rules	Locked architecture decisions
1	Core Backend & HPC Pipeline	End-to-end genomic inference
2	Agent & AI Interpretation Layer	Natural language with safe AI
3	Frontend MVP	Researcher-usable interface
4	Confidence, Stability & Trust	Review-ready results
5	Projects, History & Versioning	Full lab notebook
6	Security, Scale & Ops	Production-safe system

Table 1.1: Implementation phases overview

1.3 Target Audience

This document is intended for:

- Technical leads responsible for implementation decisions
- Developers building individual system components
- Principal investigators overseeing the research platform
- IRB reviewers assessing data handling procedures
- Future maintainers who need to understand system design

Chapter 2

Phase 0: Planning & Ground Rules

Note

Goal: Lock all architectural decisions so you don't redesign mid-build.
Duration: 1–2 weeks
Output: Final architecture agreement, supported task list, no ambiguity going into build

2.1 Project Objective

The objective of this project is to build a secure, reproducible, AI-assisted research platform on top of a Genomic Foundation Model. The system must support:

- Natural-language interaction with genomic inference tools
- GPU-accelerated genomic inference via HPC clusters
- Structured scientific interpretation with mandatory confidence metrics
- Privacy enforcement and IRB safety by design
- Full auditability and reproducibility—every run can be recreated from metadata

2.2 MVP Scope Freeze

The MVP focuses on three core EPCOT prediction tasks. Chromatin Organization Prediction is explicitly deferred to reduce initial complexity and risk.

2.2.1 Supported Tasks for MVP

Code	Task Name	Input Requirements	Output	MVP
EFP	Epigenomic Feature Prediction	1.6 kb DNA sequence + chromatin accessibility	Predicted epigenomic signal tracks	✓
GEP	Gene Expression Prediction	∼11 kb DNA sequence + cell-type identifier	Expression score and classification	✓
EAP	Enhancer Activity Prediction	∼3 kb genomic region (coordinates or sequence)	Enhancer activity score (0–1)	✓
COP	Chromatin Organization Prediction	∼1 Mb genomic region	Contact maps and 3D structure	Post-MVP

Table 2.1: MVP task scope

2.2.2 MVP Constraints

These constraints define the boundaries of the MVP and cannot be negotiated:

- **Inference Only:** The application does not trigger model training. Users submit queries; the system returns predictions from pre-trained checkpoints.
- **Single Species:** MVP supports human genome (GRCh38/hg38) only. Multi-species support is a future enhancement.
- **Read-Only Models:** Model checkpoints are consumed but never modified by the application. Training happens offline in controlled environments.
- **Project Isolation:** Each project maintains strict isolation. No cross-project context, no persistent AI memory across projects.

2.3 Non-Negotiable Rules

These rules define immutable system behavior. They cannot be overridden by user requests, configuration changes, or feature additions.

2.3.1 AI Usage Rules

AI IS PERMITTED TO	AI IS NEVER PERMITTED TO
✓ Interpret structured numeric outputs from backend ✓ Explain predictions in natural language ✓ Summarize confidence metrics and limitations ✓ Guide users through valid input workflows ✓ Ask clarifying questions when input is ambiguous ✓ Suggest next steps based on results	✗ Perform numerical computation or model inference ✗ Generate, modify, or interpolate genomic data ✗ Make causal claims (e.g., “X causes Y”) ✗ Provide clinical recommendations or diagnoses ✗ Access raw sequences, matrices, or patient data ✗ Maintain memory across projects or sessions

Table 2.2: AI usage rules

2.3.2 Data Privacy Rules

All datasets must be explicitly classified, and classification determines what operations are permitted:

Classification	AI Access Level	Export	Cross-Project
Public	Numeric summaries only	Derived data	Never
Internal	Numeric summaries only	Derived data	Never
Restricted	No AI access	None	Never
IRB-Protected	No AI access	None	Never

Table 2.3: Data privacy rules by classification

2.3.3 Scientific Validity Rules

Every analysis run must include complete metadata for reproducibility and trust:

1. **Model Version Identifier:** Unique identifier for the exact checkpoint used
2. **Dataset Version Identifier:** Reference to the dataset version and any preprocessing applied
3. **Input Validation Status:** Confirmation that inputs met all requirements
4. **Confidence Indicators:** Model confidence, data quality score, context validity (in-distribution vs. OOD), and stability score
5. **Known Limitations:** Task-specific caveats, bias warnings, and unsupported assumptions

Important

Critical Rule: Unsupported inputs must trigger explicit refusal with clear explanation. The system must never fail silently or produce results for invalid inputs.

2.4 Technical Stack Decisions

The following technology choices are locked for the MVP:

2.4.1 Frontend Layer

Component	Technology Choice
Framework	React 18+
Language	TypeScript (strict mode)
Styling	TailwindCSS
State Management	React Query + Zustand
Deployment	Vercel

Table 2.4: Frontend technology stack

2.4.2 Agent Layer

Component	Technology Choice
Agent Framework	Parlant
Hosting	AWS EC2 m7i.large (2 vCPU, 8 GB RAM)
Primary LLM	LLaMA 3 70B
Fallback LLM	OpenAI GPT-4 Turbo
Temperature Setting	≤ 0.2 (enforced)

Table 2.5: Agent layer technology stack

2.4.3 Backend Layer

Component	Technology Choice
API Framework	FastAPI (Python 3.11+)
Hosting	AWS EC2 m7i.xlarge (4 vCPU, 16 GB RAM)
Job Queue	AWS SQS (Standard queue)
GPU Compute	Lighthouse HPC (Slurm-managed)
Database	PostgreSQL 15+ on AWS RDS
Object Storage	AWS S3

Table 2.6: Backend technology stack

2.4.4 HPC & GPU Resources

GPU Type	Primary Use Case	Memory	Cluster
V100	Legacy, debugging, small tests	16/32 GB	Great Lakes
A40	Shared inference workloads	48 GB	Great Lakes
L40 / L40S	Production inference, fine-tuning	48 GB	Lighthouse
H100	Large-scale foundation runs	80 GB	Lighthouse

Table 2.7: GPU resource allocation

2.5 Model Configuration

The EPCOT model checkpoint must be explicitly versioned and documented:

- **Model ID:** `epcot-v1.0-hg38`
- **Species:** Homo sapiens
- **Genome Build:** GRCh38/hg38
- **Supported Tasks:** EFP, GEP, EAP (COP post-MVP)
- **Backbone Status:** Frozen (no modification during inference)
- **Task Heads:** Fixed for MVP (no online adaptation)

2.6 Phase 0 Completion Checklist

Before proceeding to Phase 1, verify completion of all items:

☐ MVP task list finalized and documented (EFP, GEP, EAP)

☐ Non-negotiable rules reviewed and accepted by all stakeholders

☐ Model checkpoint available, validated, and version-locked

☐ Infrastructure accounts provisioned (AWS, HPC cluster access)

☐ Dataset classification system documented

☐ IRB protocols in place (if working with protected data)

☐ Technical stack decisions locked

☐ This document signed off by project lead, technical lead, and PI

Chapter 3

Phase 1: Core Backend & HPC Pipeline

Note

Goal: Make the system actually run genomic inference end-to-end. At the end of this phase, you can submit a request and get genomic predictions back—no UI, no AI, just correctness.
Duration: 3–4 weeks
Output: Working HPC pipeline with FastAPI orchestration

3.1 EPCOT Inference Wrappers

Create Python wrapper functions for each supported task. These wrappers provide a clean interface between the backend orchestration layer and the underlying model code.

3.1.1 Wrapper Function Specifications

Each wrapper must follow a strict contract:

`run_efp()` — Epigenomic Feature Prediction

Parameter	Type	Validation
`dna_sequence`	`str`	Exactly 1,600 characters, ACGT only
`chromatin_accessibility`	`np.ndarray`	Required, normalized float array
Returns	`dict`	signal_tracks, metadata, timing

`run_gep()` — Gene Expression Prediction

Parameter	Type	Validation
`dna_sequence`	`str`	~11,000 characters, ACGT only
`cell_type`	`str`	Must be in supported vocabulary
Returns	`dict`	expression_score, classification, metadata

`run_eap()` — Enhancer Activity Prediction

Parameter	Type	Validation
`genomic_region`	`str`	chr:start-end OR ~3,000 bp sequence
Returns	`dict`	activity_score (0–1), metadata

3.1.2 Implementation Guidelines

- Each wrapper handles its own input preprocessing (one-hot encoding, normalization)
- Wrappers catch and translate model-specific exceptions into standard error types
- All wrappers log execution time, GPU memory usage, and model version
- Output always includes raw prediction plus metadata dictionary
- No UI or AI integration at this stage—focus purely on correctness

3.2 Slurm Job Packaging

Create standardized Slurm job templates for each task type. Jobs must be self-contained and produce consistent output structures.

3.2.1 Job Template Requirements

- **GPU Request:** Specify appropriate GPU type (V100 for debug, L40/L40S for production, H100 for large runs)
- **Memory Limits:** Set based on task requirements (EFP: 16GB, GEP: 32GB, EAP: 16GB)
- **Time Limits:** Conservative estimates with 2x buffer (EFP: 10min, GEP: 15min, EAP: 10min)
- **Output Paths:** Standardized structure in Turbo/projectup directories
- **Error Handling:** Capture stderr, set appropriate exit codes

3.2.2 Job Metadata Capture

Every job must record:

- Job ID (Slurm-assigned)
- GPU type actually allocated
- Actual runtime (wall clock and GPU time)
- Model version used
- Input hash for reproducibility
- Exit status and any error messages

3.3 FastAPI Orchestration Layer

Implement the core API endpoints that bridge the frontend/agent with the HPC cluster.

3.3.1 Endpoint Specifications

POST /submit-job

Accepts inference requests, validates inputs, and submits Slurm jobs.

Field	Type	Description
`project_id`	UUID	Project context for isolation
`task_type`	enum	EFP | GEP | EAP
`inputs`	object	Task-specific input parameters
`priority`	enum	normal | high
Returns	object	job_id, estimated_wait, status

GET /job-status/{job_id}

Returns current job state and progress information.

Field	Type	Description
`status`	enum	queued | running | completed | failed
`progress`	float	0.0 to 1.0 (if available)
`queue_position`	int | null	Position in queue
`gpu_assigned`	string | null	GPU type (if running)
`runtime_seconds`	float | null	Elapsed time

GET /job-result/{job_id}

Retrieves completed job results with full metadata.

Field	Type	Description
`prediction`	object	Task-specific prediction outputs
`confidence`	object	Confidence metrics (see Phase 4)
`limitations`	array	Known limitations for this result
`provenance`	object	Model version, timestamps, etc.

3.3.2 Input Validation Rules

FastAPI must enforce all EPCOT input constraints before job submission:

- Sequence length matches task requirements exactly
- Only valid nucleotide characters (ACGT, with configurable N threshold)
- Required modalities are present (e.g., chromatin accessibility for EFP)
- Cell type is in the supported vocabulary (for GEP)
- Genome build matches deployed model
- Dataset classification allows the requested operation

Important

Invalid inputs return HTTP 422 (Unprocessable Entity) with detailed error messages explaining exactly what failed validation.

3.4 Database Schema

Create PostgreSQL tables for job tracking and metadata:

- **jobs:** job_id, project_id, task_type, status, submitted_at, started_at, completed_at, gpu_type, runtime_seconds
- **job_inputs:** job_id, input_hash, input_metadata (JSON)
- **job_results:** job_id, result_path, confidence_metrics (JSON), provenance (JSON)
- **job_errors:** job_id, error_type, error_message, stack_trace

3.5 Phase 1 Completion Checklist

☐ All three wrapper functions pass unit tests

☐ Slurm jobs submit successfully and produce correct outputs

☐ FastAPI endpoints accept requests and return properly formatted responses

☐ End-to-end test: submit request $\rightarrow$ get prediction back

☐ Job status polling works correctly through all state transitions

- ☐ Database records created for all jobs with complete metadata
- ☐ Error cases return appropriate HTTP status codes

Chapter 4

Phase 2: Agent & AI Interpretation Layer

Note

Goal: Add natural language interaction safely, without breaking science. Users can ask questions in plain English; the system routes to the correct backend task and explains results clearly.
Duration: 3–4 weeks
Output: Working natural language interface with safe AI interpretation

4.1 Parlant Agent Setup

Configure the Parlant agent framework to handle user queries and orchestrate backend tools.

4.1.1 Intent Definitions

Define explicit intents that map to backend capabilities:

Intent Name	Example Triggers	Backend Endpoint
`ask_gene_expression`	"What is the expression level?"	/infer/gene-expression
`ask_epigenomic_features`	"Show epigenomic signals"	/infer/epigenomic
`ask_enhancer_activity`	"Is this an enhancer?"	/infer/enhancer-activity
`check_job_status`	"What's the status?"	/jobs/{id}/status
`explain_results`	"What does this mean?"	(LLM interpretation)

Table 4.1: Intent to endpoint mapping

4.1.2 Tool Definitions

Each Parlant tool wraps a FastAPI endpoint:

- **call_inference:** Submits inference jobs to the backend, returns job_id
- **get_job_status:** Polls job status, returns current state

- **get_job_result:** Retrieves completed results with all metadata
- **validate_input:** Pre-flight validation before submission

4.1.3 Enforcement Rules

Parlant must enforce:

- **Tool-First Execution:** Every inference intent must call a backend tool before generating any response
- **Schema-Locked Outputs:** Tool outputs follow strict schemas; agent cannot modify them
- **Refusal Rules:** Unsupported tasks trigger explicit refusal with explanation
- **No Speculation:** Agent cannot answer questions about results before tools return

4.2 LLM Integration

Configure LLM usage for interpretation only—never for computation or data generation.

4.2.1 LLM Prompt Rules

- **Temperature:** ≤ 0.2 (enforced in all prompts)
- **No New Facts:** LLM cannot introduce information not present in tool outputs
- **No Causal Claims:** LLM must use correlational language (“associated with”, “predicts”)
- **Confidence Awareness:** LLM must reflect confidence levels in language certainty

4.2.2 What LLM Receives

The LLM prompt includes only:

- Numeric summaries (scores, classifications)
- Confidence objects (model_confidence, data_quality, stability_score)
- Provenance metadata (model version, timestamps)
- Task-specific limitations

4.2.3 What LLM Never Receives

Important

These are hard boundaries that must never be crossed:

- Raw DNA sequences (even short ones) — **NEVER**
- Contact matrices or heatmaps — **NEVER**
- Raw signal tracks — **NEVER**
- Patient identifiers or PHI — **NEVER**
- IRB-protected data of any kind — **NEVER**

4.3 Intent Routing Logic

Implement routing logic that maps user queries to appropriate backend tasks:

User Query	Routing Decision
"What is the gene expression level in this region?"	→ GEP endpoint (requires cell type)
"Show predicted chromatin contacts"	→ COP endpoint (post-MVP: explain unavailability)
"Give enhancer activity score for chr1:1000-4000"	→ EAP endpoint
"What causes this gene to be expressed?"	→ REFUSE (causal claim)
"Should I treat the patient with this mutation?"	→ REFUSE (clinical recommendation)

Table 4.2: Intent routing examples

4.4 Response Generation

AI-generated responses must follow a strict structure:

4.4.1 Required Response Sections

1. **Summary:** One-sentence plain-language summary of the prediction
2. **Evidence:** Key numeric values from tool output
3. **Confidence:** Interpretation of confidence metrics
4. **Limitations:** Relevant caveats for this specific result
5. **Next Steps:** Non-prescriptive suggestions (if appropriate)

4.4.2 Confidence-Aware Language

Confidence	Language Pattern
High (> 0.8)	"The model predicts...", "Results indicate..."
Moderate (0.5–0.8)	"The model suggests...", "With moderate confidence..."
Low (< 0.5)	"The model tentatively suggests...", "Interpret with caution..."
OOD / Insufficient	"WARNING: This input appears outside the model's training distribution..."

Table 4.3: Confidence-aware language patterns

4.5 Phase 2 Completion Checklist

☐ Parlant agent successfully classifies intents from natural language

☐ All supported intents correctly route to backend tools

☐ LLM generates explanations that follow the required structure

☐ Confidence levels appropriately affect language certainty

☐ Unsupported requests trigger clear refusals

☐ No raw genomic data appears in any LLM prompt (verified by logging)

☐ End-to-end test: natural language query $\rightarrow$ correct prediction $\rightarrow$ clear explanation

Chapter 5

Phase 3: Frontend MVP

Note

Goal: Make the system usable without engineers. Researchers can interact with the platform through a web interface without touching command line tools.
Duration: 4–5 weeks
Output: Researcher-usable web interface with full query-to-result workflow

5.1 Core UI Architecture

Build a React application with the following structure:

5.1.1 Page Layout

- **Header:** Logo, project selector, user menu
- **Sidebar:** Navigation, recent jobs, quick actions
- **Main Content Area:** Results display, visualizations
- **Chat Popup:** Floating chat interface (Parlant UI)
- **Status Panel:** Job progress, queue status

5.1.2 Key Pages

1. **Dashboard:** Project overview, recent results, quick stats
2. **New Analysis:** Input forms for each task type
3. **Results Viewer:** Detailed result display with visualizations
4. **Job History:** List of past analyses with filters
5. **Settings:** User preferences, project configuration

5.2 Input Components

Build reusable input components for genomic data:

5.2.1 SequenceInput

Multi-line text input for DNA sequences with:

- Real-time validation (character checking, length display)
- FASTA format support (auto-strip headers)
- Copy/paste handling for large sequences
- Clear error messages for invalid characters
- Length indicator with task-specific requirements

5.2.2 RegionSelector

Genomic coordinate input with:

- Chromosome dropdown (chr1-22, X, Y, M)
- Start/end position inputs with formatting
- Validation against genome bounds
- Quick presets for common regions
- Gene name lookup (optional enhancement)

5.2.3 CellTypeDropdown

Searchable dropdown for cell type selection:

- Filterable list of supported cell types
- Grouped by tissue/organ
- Clear indication of commonly used types
- "Not sure" option with guidance

5.2.4 AccessibilityUpload

File upload for chromatin accessibility data:

- Drag-and-drop support
- Format validation (bigWig, bedGraph)
- Preview of uploaded data
- Normalization options

5.3 Result Components

Create card-based components for displaying results:

5.3.1 PredictionCard

Primary result display showing:

- Metric name and value (prominent)
- Score visualization (bar, gauge, or appropriate format)

- Classification label (if applicable)
- Model version badge
- Timestamp

5.3.2 ConfidenceCard

Confidence metrics breakdown:

- Overall confidence label with color coding
- Individual metrics: model confidence, data quality, stability
- Visual indicators (progress bars, icons)
- Expandable explanation of each metric

5.3.3 EvidenceCard

Supporting information:

- Genomic region context (chr:start-end)
- Links to external databases (Ensembl, UCSC)
- Input summary
- Related annotations

5.3.4 LimitationCard

Caveats and warnings:

- Task-specific limitations
- OOD warnings (if applicable)
- Bias notifications
- Suggestions for validation

5.4 Visualization Panels

Implement visualization components for different output types:

5.4.1 Signal Track Viewer

For EFP outputs:

- Interactive line/area chart
- Zoom and pan controls
- Region highlighting
- Export options (PNG, SVG, data)

5.4.2 Score Gauge

For single-value outputs (EAP, GEP):

- Circular or linear gauge
- Threshold indicators
- Comparison to baseline (if available)

5.5 Chat Interface

Implement chat popup based on parlant-chat-react:

- Floating button to open/close
- Message history within session
- Typing indicators
- Status-aware responses
- Quick action buttons for common queries
- Result cards embedded in chat

5.6 Phase 3 Completion Checklist

☐ All input components validate correctly and provide helpful feedback

☐ Users can submit analyses for all three MVP tasks without CLI

☐ Results display clearly with all required cards

☐ Chat interface connects to Parlant and handles queries

☐ Job status updates in real-time

☐ Responsive design works on desktop and tablet

☐ Accessibility requirements met

☐ End-to-end user test: non-technical researcher completes analysis

Chapter 6

Phase 4: Confidence, Stability & Trust

Note

Goal: Make results defensible. Every prediction includes comprehensive uncertainty quantification so researchers know exactly how much to trust each result.
Duration: 3–4 weeks
Output: Review-ready results with clear uncertainty communication

6.1 Confidence Metrics Framework

Implement a comprehensive confidence scoring system that accompanies every prediction.

6.1.1 Confidence Components

Metric	Description	Range
`model_confidence`	Model's internal certainty based on output distribution entropy or calibrated probability	0.0–1.0
`data_quality`	Quality assessment of input data: coverage, signal-to-noise, completeness	0.0–1.0
`context_validity`	Whether input falls within training distribution	Categorical
`stability_score`	Prediction consistency under small input perturbations	0.0–1.0
`overall_label`	Aggregated assessment combining all metrics	high | moderate | low | insufficient

Table 6.1: Confidence metrics

6.1.2 Confidence Calculation

The overall confidence label is computed using a weighted combination:

1. **Calculate individual scores:** Each metric normalized to 0–1
2. **Apply context validity modifier:** OOD inputs cap overall confidence at "moderate"

3. **Compute weighted average:** model_confidence (40%), data_quality (25%), stability_score (35%)
4. **Map to label:** > 0.8 = high, 0.5–0.8 = moderate, 0.3–0.5 = low, < 0.3 = insufficient

6.2 Stability Testing

Implement perturbation-based stability assessment:

6.2.1 Perturbation Methods

- **Coordinate Shift:** Slide input region by ±10bp, ±50bp, ±100bp and measure prediction variance
- **Sequence Perturbation:** Introduce random SNPs at 0.1%, 0.5%, 1% rates and measure impact
- **Bootstrap Resampling:** For accessibility inputs, resample and rerun (where applicable)

6.2.2 Stability Score Computation

Stability score $= 1 - \frac{variance}{maxexpectedvariance}$

- Score > 0.9: Highly stable—small changes don't affect prediction
- Score 0.7–0.9: Stable—minor variation within expected bounds
- Score 0.5–0.7: Moderately stable—some sensitivity to input changes
- Score < 0.5: Unstable—prediction varies significantly with small changes

6.3 Out-of-Distribution Detection

Implement OOD detection to flag inputs that may produce unreliable results:

6.3.1 Detection Methods

- **Embedding Distance:** Measure distance from input embedding to training distribution centroid
- **Cell Type Coverage:** Flag cell types not seen during training
- **Sequence Composition:** Detect unusual GC content, repeat regions, or ambiguous bases
- **Region Annotation:** Flag regions with known data quality issues

6.3.2 OOD Response Protocol

When OOD is detected:

1. Set `context_validity` to "ood"
2. Cap overall confidence at "moderate" maximum
3. Add specific warning to limitations
4. AI explanation must explicitly acknowledge OOD status
5. UI displays prominent warning banner

6.4 UI Confidence Display

Implement clear visual communication of confidence:

Confidence	Color	Visual Treatment
High	Green (#22C55E)	Solid badge, full opacity
Moderate	Yellow (#EAB308)	Outlined badge, info icon
Low	Orange (#F97316)	Warning styling, caution icon
Insufficient	Red (#EF4444)	Alert banner, stop icon

Table 6.2: Confidence color coding

6.5 Phase 4 Completion Checklist

☐ All four confidence metrics computed for every prediction

☐ Stability testing runs automatically

☐ OOD detection flags unusual inputs with clear warnings

☐ AI language adapts appropriately to confidence levels

☐ UI displays confidence clearly with color coding and icons

☐ No silent failures—low confidence is always communicated

☐ Researchers in user testing correctly interpret confidence displays

Chapter 7

Phase 5: Projects, History & Versioning

Note

Goal: Turn the system into a real research platform. Researchers can organize work into projects, track history, compare results, and export reproducible records.
Duration: 4–5 weeks
Output: Full lab notebook functionality with guaranteed reproducibility

7.1 Project Model

Implement project-based organization:

7.1.1 Project Entity

Field	Type	Description
`project_id`	UUID	Unique identifier
`name`	string	Human-readable project name
`description`	text	Project description and goals
`created_at`	timestamp	Creation time
`owner_id`	UUID	User who created the project
`collaborators`	array<UUID>	Users with access (Phase 6)
`default_classification`	enum	Default dataset classification

Table 7.1: Project entity schema

7.1.2 Project Isolation

Strict isolation is enforced:

- Each project has its own chat history
- AI has no memory of other projects

- Results cannot be viewed across projects
- Exports are project-scoped

7.2 Experiment Tracking

Record every analysis as a versioned experiment:

7.2.1 Experiment Record

Field	Type	Description
`experiment_id`	UUID	Unique identifier
`project_id`	UUID	Parent project
`name`	string	User-assigned experiment name
`description`	text	Experiment notes
`runs`	array<run_id>	All analysis runs
`created_at`	timestamp	Creation time
`tags`	array<string>	User-defined tags

Table 7.2: Experiment record schema

7.2.2 Run Record

Each individual analysis is a run:

Field	Type	Description
`run_id`	UUID	Unique identifier
`experiment_id`	UUID	Parent experiment
`task_type`	enum	EFP | GEP | EAP
`inputs`	JSON	Complete input parameters
`input_hash`	string	Hash for deduplication
`outputs`	JSON	Complete prediction outputs
`confidence`	JSON	All confidence metrics
`provenance`	JSON	Model version, etc.
`timestamp`	timestamp	Execution time

Table 7.3: Run record schema

7.3 Version Management

Track versions of models and datasets:

7.3.1 Model Registry

- Catalog of all deployed model versions

- Supported tasks per version
- Known limitations per version
- Deprecation notices for old versions

7.3.2 Run Comparison

Enable comparison of results across versions:

- Side-by-side result display
- Diff highlighting for changed predictions
- Version mismatch warnings
- "Fork run" to rerun with new model version

7.4 Export Functionality

Implement comprehensive export capabilities:

Format	Contents	Use Case
JSON	Structured results with all metadata	Programmatic access
CSV	Tabular summary (derived data only)	Spreadsheet analysis
PDF	Formatted report with visualizations	Publication, archival

Table 7.4: Export formats

7.4.1 Export Restrictions

- **Raw data exports are never allowed:** Only derived, aggregated outputs
- **IRB-protected content cannot be exported:** Export buttons disabled
- **Audit trail required:** All exports logged

7.5 Phase 5 Completion Checklist

☐ Projects can be created, edited, and deleted

☐ Experiments organize runs within projects

☐ All runs include complete provenance for reproducibility

☐ Export produces valid JSON, CSV, and PDF outputs

☐ Search finds relevant past experiments

☐ Run comparison works across model versions

☐ Audit log captures all exports

☐ Researchers can reproduce any past result

Chapter 8

Phase 6: Security, Scale & Operations

Note

Goal: Make the system production-safe. Multiple users can work simultaneously with proper access controls, and the system scales to handle demand.
Duration: 4–6 weeks
Output: Stable, secure multi-user system ready for production use

8.1 Role-Based Access Control (RBAC)

Implement granular permissions:

Role	Permissions	Typical User
Viewer	View results; no submission or export	Auditors
Analyst	Submit analyses, view results, export derived data	Students, staff
Researcher	Create projects, manage experiments	Grad students
PI	Manage membership, access IRB data	PIs
Admin	User management, system configuration	IT staff

Table 8.1: User roles and permissions

8.2 Authentication & Authorization

Implement secure authentication:

8.2.1 Authentication Options

- **University SSO:** SAML integration with campus identity provider
- **OAuth 2.0:** Google/Microsoft accounts for external collaborators
- **API Keys:** For programmatic access (rate-limited)

8.2.2 Session Management

- JWT tokens with short expiration (1 hour)
- Refresh token rotation
- Session invalidation on permission changes
- Audit logging of all authentication events

8.3 Audit Logging

Comprehensive audit trail for compliance:

8.3.1 Logged Events

- User authentication (login, logout, failed attempts)
- Data access (which datasets, which fields)
- Analysis submissions (inputs, parameters)
- Result access (who viewed what, when)
- Exports (format, content hash, destination)
- Administrative actions (role changes, project modifications)

8.3.2 Log Retention

- Security events: 2 years minimum
- Access logs: 7 years (IRB requirement)
- Analysis logs: Permanent (for reproducibility)

8.4 Scaling Infrastructure

Implement horizontal scaling:

8.4.1 Backend Scaling

- **Load Balancer:** AWS ALB distributes requests across instances
- **Auto Scaling Group:** EC2 instances scale based on CPU/memory
- **Minimum instances:** 2 (for availability)
- **Maximum instances:** 10 (cost control)

8.4.2 Database Scaling

- Read replicas for query distribution
- Connection pooling (PgBouncer)
- Query optimization and indexing
- Automated backups

8.5 Async Job Queue

Production-ready job handling:

8.5.1 Queue Architecture

- **Primary Queue:** Standard jobs (SQS Standard)
- **Priority Queue:** Urgent jobs from PI users
- **Dead Letter Queue:** Failed jobs for investigation
- **Worker Pool:** Slurm jobs pulling from queue

8.5.2 Job Lifecycle

1. Job submitted → SQS message created
2. Worker claims message → visibility timeout starts
3. Slurm job submitted → status updated to "running"
4. Inference completes → results stored
5. Message deleted → status updated to "completed"
6. On failure → message returns to queue (up to 3 retries)
7. After max retries → dead letter queue + alert

8.6 Monitoring & Alerting

Implement comprehensive observability:

8.6.1 Alerting Rules

Condition	Severity	Response
Error rate $> 5\%$	Critical	Page on-call
Job queue > 100	Warning	Slack notification
GPU utilization $< 20\%$	Info	Weekly report
Database connections $> 80\%$	Warning	Slack notification
SSL certificate expiring	Critical	Page on-call

Table 8.2: Alerting rules

8.7 Disaster Recovery

Plan for failure scenarios:

8.7.1 Backup Strategy

- **Database:** Automated daily snapshots, 30-day retention

- **Object Storage:** Cross-region replication
- **Configuration:** Infrastructure as Code (Terraform)
- **Model Checkpoints:** Versioned storage in Data Den

8.7.2 Recovery Procedures

- Database restore: < 1 hour RTO
- Full system rebuild: < 4 hours RTO
- Documented runbooks for common failures
- Quarterly disaster recovery drills

8.8 Phase 6 Completion Checklist

☐ RBAC enforces permissions correctly for all roles

☐ SSO authentication works with university identity provider

☐ Audit logs capture all required events

☐ Auto-scaling handles 10x normal load

☐ Job queue processes 1000+ jobs/day without issues

☐ Monitoring dashboards provide real-time visibility

☐ Alerting notifies on-call for critical issues

☐ Disaster recovery tested and documented

☐ Security review completed and findings addressed

☐ System operates stably for 2 weeks under production load

Appendix A

API Reference

A.1 Inference Endpoints

A.1.1 POST /infer/epigenomic

Submit an Epigenomic Feature Prediction job.

Request Body:

- `project_id` (UUID, required): Project context
- `dna_sequence` (string, required): Exactly 1,600 bp
- `chromatin_accessibility` (array<float>, required): Normalized signal array
- `priority` (enum, optional): "normal" | "high"

Response (202 Accepted):

- `job_id` (UUID): Unique job identifier
- `status`: "queued"
- `estimated_wait_seconds` (int): Estimated queue time

A.1.2 POST /infer/gene-expression

Submit a Gene Expression Prediction job.

Request Body:

- `project_id` (UUID, required): Project context
- `dna_sequence` (string, required): ∼11,000 bp
- `cell_type` (string, required): Must be in supported vocabulary
- `priority` (enum, optional): "normal" | "high"

A.1.3 POST /infer/enhancer-activity

Submit an Enhancer Activity Prediction job.

Request Body:

- `project_id` (UUID, required): Project context

- `genomic_region` (string, required): chr:start-end OR ∼3,000 bp sequence
- `priority` (enum, optional): "normal" | "high"

A.2 Job Management Endpoints

A.2.1 GET /jobs/{job_id}/status

Get current job status.

Response:

- `job_id` (UUID)
- `status`: "queued" | "running" | "completed" | "failed"
- `queue_position` (int | null)
- `gpu_assigned` (string | null)
- `runtime_seconds` (float | null)

A.2.2 GET /jobs/{job_id}/result

Get completed job results.

Response:

- `prediction` (object): Task-specific outputs
- `confidence` (object): All confidence metrics
- `limitations` (array<string>)
- `provenance` (object): Model version, timestamps, etc.

A.3 Validation Endpoint

A.3.1 POST /validate/input

Pre-flight validation before job submission.

Request Body:

- `task_type` (enum): EFP | GEP | EAP
- `inputs` (object): Task-specific inputs

Response:

- `valid` (boolean)
- `errors` (array<string>): Validation error messages
- `warnings` (array<string>): Non-blocking warnings

Appendix B

Confidence Metrics Deep Dive

B.1 Model Confidence

Model confidence reflects the model's internal certainty about its prediction. For classification tasks, this is typically derived from softmax probabilities. For regression tasks, it may come from prediction intervals or ensemble disagreement.

Calculation:

- Classification: $1 - \frac{entropy(softmax)}{maxentropy}$
- Regression: Based on predicted variance or ensemble std dev
- Calibration: Scores are calibrated against held-out validation set

B.2 Data Quality Score

Assesses the quality of input data:

- **Sequence quality:** Fraction of ambiguous bases (N), unusual patterns
- **Coverage (accessibility):** Signal-to-noise ratio, missing regions
- **Completeness:** Whether all required modalities are present

B.3 Context Validity

Determines whether the input is within the model's training distribution:

- **in_distribution:** Input resembles training data
- **ood:** Input differs significantly from training data
- **unknown:** Cannot determine (missing reference)

B.4 Stability Score

Measures prediction consistency under perturbation:

Process:

1. Generate N perturbed versions of input (N=10 default)

2. Run inference on all versions
3. Calculate variance of predictions
4. Score $= 1 - \frac{variance}{maxexpectedvariance}$

Appendix C

Glossary

Term	Definition
EPCOT	A generalizable deep learning framework for predicting multiple genomic modalities from DNA sequence and chromatin accessibility
EFP	Epigenomic Feature Prediction—predicts epigenomic signal tracks from sequence and accessibility
GEP	Gene Expression Prediction—predicts gene expression levels from sequence and cell type
EAP	Enhancer Activity Prediction—predicts enhancer activity scores for genomic regions
COP	Chromatin Organization Prediction—predicts 3D chromatin structure (post-MVP)
OOD	Out-of-Distribution—input that differs significantly from training data
Parlant	Agent framework used for natural language interaction and tool orchestration
Provenance	Complete metadata about a prediction enabling reproducibility
Slurm	Workload manager for HPC clusters handling GPU job scheduling
IRB	Institutional Review Board—oversees research involving human subjects

End of Document