8-Week Project Timeline: AI Privacy Firewall MVP

1. Team Members & Roles

This section outlines the core team members and their designated roles for the development of the MVP (Minimum Viable Product).

Team A: Temporal Framework (Core Framework & Temporal Context)

- **Mehara Ayisha T (Backend/Django)**: Responsible for designing 6-tuple data structures, core interfaces, and backend integration.
- **Aflah Muhammed (Linux/Security)**: Responsible for implementing the TemporalContext class, temporal rule processing, and security enforcement.

Team B: Org-Chart Integration (Graph Database & Access Control)

- **Minha K M (Database Security)**: Responsible for Neo4j setup, schema design, and organizational relationship modeling.
- **Aithel Christo Sunil (Security/ISC2)**: Responsible for access control queries, role-based permission inheritance, and boundary checking.

Team C: Ontology Integration (Ontology & System Integration/Demo)

- **Ken Mani Binu (NLP/Chatbot)**: Responsible for RDF/OWL implementation, semantic reasoning, and ontology design.
- **Nibin Thomas (Multi-language/Testing)**: Responsible for system integration, unified API development, multilingual testing, and demo preparation.

---

2. Project Goals

The primary objective is to develop a functional **AI Privacy Firewall MVP** that demonstrates contextual integrity through temporal awareness, organizational understanding, and semantic reasoning.

Core Features:

- **Temporal Context Processing**: Time-aware privacy rule validation with emergency override capabilities.
- **Organizational Graph Intelligence**: Role-based access control using Neo4j graph database.
- **Semantic Data Classification**: RDF/OWL-based ontology for intelligent data type recognition.
- **Integrated Privacy Decision Engine**: Unified system combining all three components.
- **Real-time Performance**: Sub-100ms response times for privacy decisions.

---

3. Role-Based Responsibilities

Team A: Temporal Framework (Mehara, Aflah)

- Core privacy rule engine architecture.
- Temporal context awareness and time-based validation.
- Emergency override and incident response logic.

Team B: Org-Chart Integration (Minha, Aithel)

- Organizational relationship modeling and queries.
- Role-based permission inheritance and boundary checking.
- Graph database performance and scalability.

Team C: Ontology Integration (Ken, Nibin)

- Semantic data classification and reasoning.
- Unified API development and integration.
- Demo scenarios and comprehensive testing.

---

4. Weekly Development Timeline & Assigned Tasks

Weeks 1–2: Foundation & Setup

- **Team A (Mehara, Aflah):**
    - Design 6-tuple data structures and interfaces.
    - Set up development environment and repos.
    - Create TemporalContext class with mock data.
    - Implement basic privacy rule matching.
    - Unit tests for core structures.
    - Build EnhancedContextualIntegrityTuple class.
    - Temporal rule engine with time window validation.
- **Team B (Minha, Aithel):**
    - Set up Neo4j environment.
    - Create sample organizational data (30+ people, 5 departments).
    - Design graph schema for people, roles, and relationships.
    - Implement Cypher queries for reporting relationships.
    - Test Python-Neo4j connectivity.
    - Implement department boundary checking.
    - Add project membership relationships.
- **Team C (Ken, Nibin):**
    - Research RDF/OWL tools (rdflib, owlready2).
    - Design simplified ontology schema for classifications.
    - Create sample ontology (3–4 types + relationships).
    - Set up reasoning engine.
    - Define system architecture & API contracts.

- Implement semantic classification hierarchy.

Weeks 3–4: Core Components

- **Team A (Mehara, Aflah):**
  - Data freshness & expiry validation.
  - Incident response temporal patterns.
  - Audit/legal hold handling.
  - Temporal role permission inheritance.
  - Optimize for real-time performance.
  - Integrate with Team B's org-chart queries.
- **Team B (Minha, Aithel):**
  - Role-based access query functions.
  - Temporal role support (acting roles, coverage).
  - Add classification properties to departments.
  - Skip-level manager detection.
  - Contractor & external user handling.
  - Cross-functional team detection.
- **Team C (Ken, Nibin):**
  - Concept equivalence matching (SSN = Social Security Number).
  - Domain context disambiguation.
  - Test scenarios across all components.
  - Start integration layer build.
  - Transitive relationship reasoning.

- Regulatory compliance rule mapping.

Weeks 5–6: System Integration

- **Team A (Mehara, Aflah):**
  - Integrate with ontology reasoning.
  - Edge case handling & error conditions.
  - Logging and metrics.
  - Performance testing with realistic data.
  - Bug fixes & optimization.
- **Team B (Minha, Aithel):**
  - Permission inheritance through hierarchy.
  - Caching for frequent relationships.
  - REST API for integration.
  - Concurrent access & connection pooling.
  - Graph validation & consistency.
  - Large org performance optimization.
- **Team C (Ken, Nibin):**
  - Domain-specific filtering logic.
  - Unified API layer integration.
  - Test data generators for complex cases.
  - Unified privacy decision engine.
  - Request routing & aggregation.
  - End-to-end testing.

Weeks 7–8: MVP Completion & Documentation

- **Team A (Mehara, Aflah):**
    - Support for complex time patterns.
    - Rule conflict resolution.
    - Temporal analytics & reporting.
    - Advanced emergency scenarios.
    - Documentation: privacy patterns, config guides.
- **Team B (Minha, Aithel):**
    - Real-time org chart updates.
    - Admin interface for management.
    - Data export for auditing.
    - External partner org support.
    - Documentation: schema, HR integration guides.
- **Team C (Ken, Nibin):**
    - Web-based demo interface.
    - Comprehensive test suite.
    - Error handling & recovery.
    - System health monitoring.
    - Demo scenarios & user stories.
    - Documentation: architecture & deployment.

---

5. Success Metrics

- **Functionality**: All PRD components work together seamlessly.
- **Performance**: <100ms decision latency under realistic load.
- **Accuracy**: 90%+ correctness in test scenarios.
- **Integration**: Smooth flow between temporal, org, and semantic modules.
- **Demo Quality**: Compelling, realistic business scenarios.
- **Robustness**: Handles edge cases and provides audit trails.

Demo Scenarios to Showcase

1. Emergency Access: ER doctor accessing records at 2 AM.
2. Temporal Boundaries: Contractor access expires with project end.
3. Organizational Hierarchy: Manager accessing team data.
4. Semantic Understanding: “Diagnosis” meaning in medical vs IT contexts.
5. Complex Integration: All 3 systems making a nuanced decision.