Healthcare SOA - Implementation Plan

Phase 1: Foundation and Enterprise Database Strategy (Weeks 1-2)

Week 1: Project Initialization

- ☑ Set up project structure and repository
- ☑ Establish coding standards and linting rules
- ☑ Create Docker container setup
- ☑ Configure shared PostgreSQL database
- ☑ Define enterprise database adaptation strategy

- ☑ Set up MongoDB and Redis for appointment service

Week 2: MuleSoft ESB Integration

- ☑ Implement MuleSoft ESB with healthcare-integration-app
- ☑ Configure service routing through ESB
- ☑ Set up domain sharing in MuleSoft
- ☑ Create health check endpoints
- ☑ Implement service gateway flows
- ☑ Configure proper logging for the ESB

Phase 2: Microservices Development (Weeks 3-6)

Week 3-4: Patient Service Development

- ☑ Implement enterprise database schema adaptation for Patient Service
  - ☑ Create entity models with column definition overrides
  - ☑ Configure Hibernate for existing schema compatibility
  - ☑ Implement proper naming strategies
  - ☑ Use Integer IDs with serial columns instead of Long
- ☑ Develop Patient Service core functionality
  - ☑ Patient demographics management
  - ☑ Patient search capabilities
  - ☑ REST API endpoints for CRUD operations
  - ☑ Integration with MuleSoft ESB

Week 5-6: Appointment Service Development

- ☑ Implement complex database adaptation strategy for Appointment Service
  - ☑ Create custom repository implementation for date-range queries
  - ☑ Implement @Transient fields for missing database columns
  - ☑ Add feature flags for controlled deployment
  - ☑ Create custom getters/setters for property encoding
- ☑ Add additional data store integration

    - ☑ MongoDB integration for appointment metadata
    - ☑ Redis for appointment caching and real-time updates

Phase 3: Integration and Enhancement (Weeks 7-9)

Week 7-8: System Integration and APIs

- ☐ Enhance MuleSoft integration flows
    - ☐ Add error handling and retry mechanisms
    - ☐ Implement data transformation for different clients
    - ☐ Add support for healthcare standards (HL7, FHIR)
    - ☐ Configure additional endpoint security
- ☐ Enhance appointment features
    - ☐ Scheduling engine improvements
    - ☐ Notification system for appointments
    - ☐ Calendar integration
    - ☐ Redis-based real-time updates

Week 9: Documentation and API Management

- ☐ Create comprehensive API documentation with Swagger/OpenAPI
- ☐ Implement API versioning strategy
- ☐ Add proper rate limiting and throttling
- ☐ Create developer portal for API consumers
- ☐ Develop integration examples for third-party systems

Phase 4: Testing, Security & Optimization (Weeks 10-12)

Week 10-11: Testing and Security

- ☐ Unit testing (min 80% code coverage)

- ☐ Integration testing across services
- ☐ Load testing for enterprise database adaptation
- ☐ Security implementation
  - ☐ OAuth 2.0 / JWT authentication
  - ☐ Role-based access control
  - ☐ Data encryption in transit and at rest
  - ☐ API security policies

Week 12: Monitoring and Finalization

- ☐ Performance optimization
  - ☐ Query optimization for enterprise schema
  - ☐ Redis caching strategy refinement
  - ☐ Connection pooling tuning
- ☐ Monitoring implementation
  - ☐ MuleSoft operational monitoring
  - ☐ Service health metrics
  - ☐ Database performance tracking
- ☐ Documentation finalization

Technology Stack

Backend Services

- Java 8 with Spring Boot 2.5.x
- MuleSoft 3.8.0 for ESB implementation
- Hibernate/JPA with Enterprise Schema Adaptation
- Spring Framework for dependency injection and REST endpoints

Databases

- PostgreSQL 13 for shared relational data (enterprise schema)
- MongoDB 4.4 for appointment metadata and history
- Redis 6.2 for caching and real-time updates

Integration Patterns

- Enterprise Database Adaptation Pattern
  - Column definition overrides
  - Custom repository implementations
  - Transient fields with property encoding
  - Feature flagging system
- Service Gateway Pattern in MuleSoft
- Domain Sharing in MuleSoft ESB
- Repository Pattern with custom extensions

Infrastructure

- Docker for containerization
- Docker Compose for local deployment
- Configurable logging with logback
- Metrics collection via Actuator endpoints

Resource Allocation

Development Team

- 1 Technical Architect (full-time)
- 2 Senior Java Developers (full-time)
- 1 Database Specialist with PostgreSQL expertise (part-time)
- 1 MuleSoft Integration Specialist (full-time)
- 1 DevOps Engineer (part-time)

Infrastructure Requirements

- Development environment: Local Docker containers
- Testing environment: AWS t3.medium instances
- Production environment: AWS t3.large instances with proper sizing
- PostgreSQL RDS instance with proper backup
- MongoDB Atlas for appointment data (M10 cluster)

- Redis ElastiCache for caching and real-time functionality
- Container registry for Docker images