AI BrainFrame System

Technical Requirements Document (TRD)

Document Information

Project: AI BrainFrame Field Service Intelligence Platform
Version: 1.0
Date: August 3, 2025
Classification: Technical Specification - Development Team
Dependencies: Functional Requirements Document v1.0

---

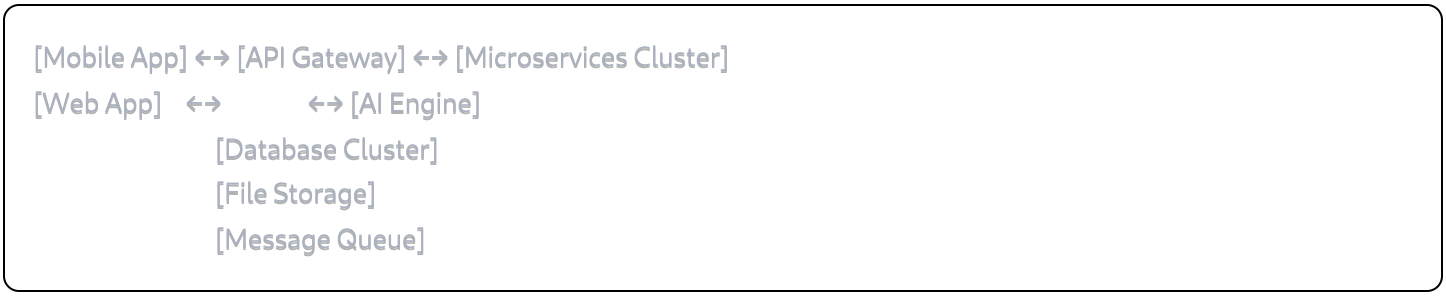
1. Technical Architecture Overview

1.1 System Architecture Pattern

Microservices Architecture with API Gateway pattern

- Containerized services using Docker
- Kubernetes orchestration for scalability
- Event-driven communication between services
- CQRS (Command Query Responsibility Segregation) for data operations

1.2 High-Level Architecture Components



1.3 Technology Stack Selection

Frontend Applications:

- **Mobile:** React Native 0.72+ with TypeScript
- **Web:** React 18+ with TypeScript and Next.js 13+
- **State Management:** Redux Toolkit with RTK Query
- **UI Framework:** React Native Elements / Material-UI

Backend Services:

- **API Framework:** FastAPI 0.100+ with Python 3.11+
- **Authentication:** Auth0 or custom JWT with refresh tokens
- **API Gateway:** Kong or AWS API Gateway
- **Service Mesh:** Istio for service-to-service communication

AI/ML Stack:

- **LLM:** Llama 2 70B (local deployment) + OpenAI GPT-4 (hybrid)
- **Embeddings:** Sentence-BERT, OpenAI Ada-002
- **Computer Vision:** YOLOv8, OpenCV, PyTorch
- **Vector Database:** Pinecone or Weaviate
- **ML Pipeline:** MLflow for model versioning and deployment

Data Layer:

- **Primary Database:** PostgreSQL 15+ with pgvector extension
- **Document Store:** MongoDB 6.0+ for unstructured data
- **Cache:** Redis 7+ for session and query caching
- **Search Engine:** Elasticsearch 8+ for full-text search
- **File Storage:** MinIO (S3-compatible) or AWS S3

Infrastructure & DevOps:

- **Containerization:** Docker with multi-stage builds
- **Orchestration:** Kubernetes 1.27+
- **CI/CD:** GitLab CI/CD or GitHub Actions
- **Monitoring:** Prometheus + Grafana + ELK Stack
- **Message Queue:** Apache Kafka or RabbitMQ

---

2. Detailed Technical Requirements

2.1 Mobile Application Technical Specifications

TR-001: Mobile App Framework

- React Native with TypeScript for cross-platform compatibility

- Minimum supported OS: iOS 14+, Android 8+ (API 26+)
- Offline-first architecture with background sync
- Native modules for camera, GPS, biometrics

TR-002: Mobile Performance Requirements

- App startup time: <3 seconds on mid-range devices
- AI response rendering: <1 second after data received
- Battery optimization: <5% drain per hour during active use
- Memory usage: <200MB peak usage

TR-003: Mobile Security

- Certificate pinning for API communications
- Biometric authentication (fingerprint, face ID)
- Local data encryption using AES-256
- Root/jailbreak detection with graceful degradation

2.2 Backend API Technical Specifications

TR-004: API Gateway Requirements

- Rate limiting: 1000 requests/minute per user
- Request/response validation using JSON Schema
- API versioning strategy (v1, v2, etc.)
- Automatic API documentation with OpenAPI 3.0

TR-005: Microservices Architecture

User Service:

- Authentication and authorization
- User profile management
- Role-based access control (RBAC)
- Session management with Redis

Job Service:

- Job lifecycle management
- Document association and versioning

- Real-time job status updates via WebSocket
- Integration with external dispatch systems

AI Service:

- Natural language processing pipeline
- Computer vision processing
- Model inference and response generation
- Response caching and optimization

Knowledge Service:

- Document indexing and search
- Content versioning and approval workflows
- Metadata extraction and tagging
- Full-text search with Elasticsearch

Notification Service:

- Real-time notifications via WebSocket
- Push notifications for mobile devices
- Email notifications for critical events
- SMS alerts for emergency escalations

2.3 AI Engine Technical Specifications

TR-006: Large Language Model Implementation

- **Primary LLM:** Llama 2 70B hosted on local GPU cluster
- **Fallback LLM:** OpenAI GPT-4 for complex queries
- **Fine-tuning:** Custom domain adaptation with field service data
- **Context Window:** 4096 tokens minimum with conversation memory

TR-007: Computer Vision Pipeline

- **Object Detection:** YOLOv8 for parts identification
- **OCR:** Tesseract + PaddleOCR for text extraction
- **Image Classification:** Custom CNN for equipment categorization
- **Real-time Processing:** <5 seconds for image analysis

TR-008: Vector Database and Embeddings

- **Embedding Model:** Sentence-BERT for document similarity
- **Vector Storage:** Pinecone or Weaviate with 768-dimensional vectors
- **Similarity Search:** Cosine similarity with <100ms query time
- **Index Size:** Support for 1M+ document embeddings

TR-009: Knowledge Retrieval System

- **Hybrid Search:** Vector similarity + keyword matching + filters
- **Ranking Algorithm:** Custom scoring based on relevance, recency, success rate
- **Context Injection:** Dynamic prompt engineering with retrieved context
- **Feedback Loop:** User feedback integration for ranking improvement

2.4 Database Architecture

TR-010: Primary Database Schema

PostgreSQL Tables:

```sql
-- Core entities
users (id, email, role, company_id, created_at, updated_at)
jobs (id, job_number, customer_id, status, location, assigned_tech)
documents (id, job_id, type, file_path, metadata, version)
conversations (id, user_id, job_id, messages, context)
solutions (id, problem_description, solution_steps, success_rate)

-- AI-specific tables
embeddings (id, document_id, vector, model_version)
ai_responses (id, query, response, feedback, context)
knowledge_base (id, content, tags, approval_status)
```

TR-011: Data Relationships

- Users → Jobs (many-to-many through assignments)
- Jobs → Documents (one-to-many)
- Jobs → Conversations (one-to-many)
- Documents → Embeddings (one-to-one)
- Solutions → Jobs (many-to-many through resolutions)

2.5 Security Technical Requirements

TR-012: Authentication & Authorization

- **OAuth 2.0 + PKCE** for mobile authentication
- **JWT tokens** with 15-minute access, 7-day refresh
- **Role-based permissions** with hierarchical inheritance
- **Multi-factor authentication** for administrative users

TR-013: Data Protection

- **Encryption at Rest:** AES-256 for all databases
- **Encryption in Transit:** TLS 1.3 for all communications
- **PII Protection:** Tokenization of sensitive customer data
- **Audit Logging:** Immutable logs for all data access

TR-014: Network Security

- **VPC Isolation** with private subnets for databases
- **WAF Protection** against common web attacks
- **DDoS Protection** with rate limiting and geo-blocking
- **Intrusion Detection** with automated response

2.6 Performance and Scalability

TR-015: Performance Benchmarks

- **API Response Time:** <200ms for 95% of requests
- **Database Query Time:** <50ms for indexed queries
- **AI Inference Time:** <3 seconds for complex queries
- **File Upload Speed:** Support for 100MB files with resumable uploads

TR-016: Scalability Requirements

- **Horizontal Scaling:** Auto-scaling based on CPU/memory usage
- **Database Scaling:** Read replicas and connection pooling
- **CDN Integration:** Global content delivery for static assets
- **Load Balancing:** Application-level load balancing with health checks

TR-017: Monitoring and Observability

- **Application Metrics:** Custom business metrics with Prometheus
- **Log Aggregation:** Centralized logging with ELK Stack
- **Distributed Tracing:** Jaeger for request tracing across services
- **Health Checks:** Kubernetes liveness and readiness probes

---

3. Integration Specifications

3.1 External System Integrations

TR-018: CRM/Dispatch System Integration

- **API Standards:** RESTful APIs with webhook support
- **Data Sync:** Real-time job updates with conflict resolution
- **Authentication:** API keys with IP whitelisting
- **Fallback:** Offline mode with background sync

TR-019: Manufacturer Database Integration

- **Equipment APIs:** Direct integration with major manufacturers
- **Parts Catalogs:** Automated parts lookup and pricing
- **Documentation Sync:** Automatic manual and bulletin updates
- **Rate Limiting:** Respectful API usage with caching

3.2 Third-Party Service Integration

TR-020: Cloud Services

- **AWS/GCP/Azure:** Multi-cloud deployment capability
- **CDN:** CloudFlare or AWS CloudFront for global delivery
- **Backup:** Automated backup to multiple geographic regions
- **Disaster Recovery:** RTO < 4 hours, RPO < 1 hour

---

4. Development Environment

4.1 Local Development Setup

TR-021: Development Environment

- **Docker Compose:** Complete local environment with one command

- **Hot Reloading:** Real-time code changes for frontend and backend
- **Database Seeding:** Sample data for consistent testing
- **Mock Services:** Simulated external APIs for offline development

TR-022: Code Quality Standards

- **Linting:** ESLint + Prettier for JavaScript/TypeScript, Black + isort for Python
- **Type Safety:** 100% TypeScript coverage for frontend, Python type hints
- **Testing:** 80%+ code coverage with unit, integration, and E2E tests
- **Documentation:** Automated API docs and code comments

4.2 Deployment Pipeline

TR-023: CI/CD Pipeline

- **Automated Testing:** Run all tests on every commit
- **Security Scanning:** Dependency vulnerability checks
- **Performance Testing:** Load testing for critical API endpoints
- **Deployment:** Blue-green deployment with automatic rollback

---

5. Data Migration and Legacy Support

5.1 Data Migration Strategy

TR-024: Migration Requirements

- **ETL Pipeline:** Extract data from existing systems
- **Data Validation:** Integrity checks and validation rules
- **Incremental Migration:** Phased approach with minimal downtime
- **Rollback Capability:** Ability to revert to previous state

---

6. Compliance and Regulatory Requirements

6.1 Industry Compliance

TR-025: Regulatory Compliance

- **SOC 2 Type II:** Security and availability controls
- **GDPR Compliance:** Data privacy and right to deletion

- **Industry Standards:** NFPA, IFC, NEC code compliance databases
- **Audit Trail:** Immutable logging for compliance reporting

---

7. Maintenance and Support

7.1 System Maintenance

TR-026: Maintenance Requirements

- **Automated Updates:** Zero-downtime deployment capability
- **Health Monitoring:** 24/7 system health monitoring
- **Performance Optimization:** Automatic query optimization
- **Capacity Planning:** Predictive scaling based on usage patterns

7.2 Support Infrastructure

TR-027: Support Systems

- **Error Tracking:** Sentry for real-time error monitoring
- **User Analytics:** Anonymous usage analytics for product improvement
- **Support Integration:** Built-in support ticket creation
- **Knowledge Base:** Self-service help system

---

8. Success Metrics and KPIs

8.1 Technical Performance KPIs

- **System Uptime:** 99.9% availability target
- **Response Time:** <3 seconds for AI responses
- **Error Rate:** <0.1% error rate for API calls
- **User Satisfaction:** >4.5 app store rating

8.2 Development KPIs

- **Deployment Frequency:** Daily deployments capability
- **Lead Time:** <24 hours from commit to production
- **Mean Time to Recovery:** <1 hour for critical issues
- **Bug Resolution:** 90% of bugs fixed within 48 hours

---

9. Risk Assessment and Mitigation

9.1 Technical Risks

High Risk: AI Model Performance

- **Risk:** Inaccurate responses leading to safety issues
- **Mitigation:** Human oversight, confidence scoring, feedback loops
- **Monitoring:** Response accuracy tracking and user feedback

Medium Risk: Scalability Challenges

- **Risk:** System performance degradation under load
- **Mitigation:** Auto-scaling, load testing, performance monitoring
- **Contingency:** Cloud bursting for peak demand

Medium Risk: Integration Complexity

- **Risk:** External system integration failures
- **Mitigation:** Robust error handling, fallback mechanisms, monitoring
- **Testing:** Comprehensive integration testing

---

10. Implementation Phases

10.1 Phase 1: Core Infrastructure (Months 1-2)

- Database setup and basic API framework
- User authentication and basic mobile app
- Development environment and CI/CD pipeline

10.2 Phase 2: AI Engine (Months 2-3)

- LLM integration and vector database setup
- Basic Q&A functionality
- Document ingestion pipeline

10.3 Phase 3: Advanced Features (Months 3-4)

- Computer vision integration
- Job management system
- Real-time features and notifications

10.4 Phase 4: Production Readiness (Month 4)

- Performance optimization and security hardening
- Monitoring and alerting setup
- Load testing and production deployment

---

11. Conclusion

This Technical Requirements Document provides the detailed technical foundation for building the AI BrainFrame system. The architecture emphasizes scalability, security, and maintainability while leveraging modern open-source technologies.

The microservices approach ensures modularity and allows for independent scaling of different system components. The hybrid AI approach balances cost and performance while maintaining high availability.

Document Prepared By: Technical Architecture Team
Review Required By: Senior Development Team, DevOps Team
Next Steps: System Architecture Diagrams and Database Schema Design