Heart Disease Risk Assessment - Technical Architecture

System Overview

This document outlines the technical architecture of a production-grade machine learning system for cardiovascular risk assessment. The system employs modern software engineering practices with microservices architecture, containerization, and automated deployment.

Architecture Patterns

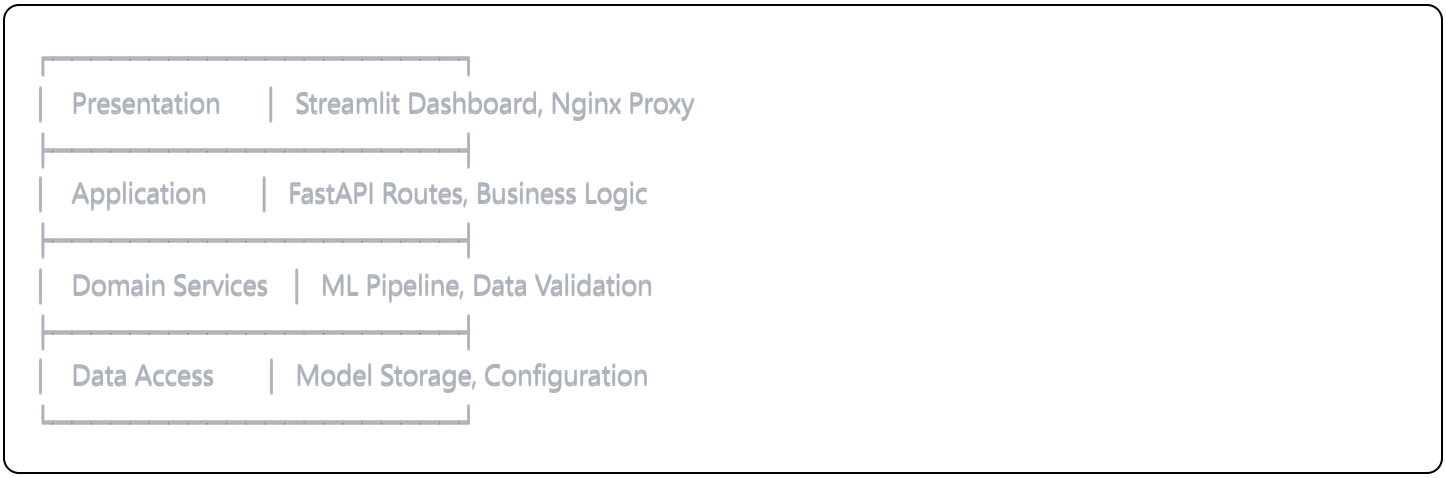
Microservices Architecture

- **API Service**: FastAPI backend for ML predictions and data processing
- **Dashboard Service**: Streamlit frontend for user interaction
- **Proxy Service**: Nginx reverse proxy for load balancing and SSL termination
- **Service Communication**: RESTful APIs with JSON data exchange

Event-Driven Design

- Asynchronous processing for batch operations
- Real-time health monitoring and alerting
- Scalable architecture supporting concurrent users

Layered Architecture



Technology Stack

Machine Learning Pipeline

Framework and Libraries

- **Core ML Framework**: scikit-learn 1.3+ for model training and inference
- **Ensemble Methods**: Random Forest, XGBoost, Logistic Regression
- **Feature Engineering**: Custom medical domain transformations
- **Model Interpretation**: SHAP 0.42+ for explainable AI
- **Data Processing**: pandas 2.0+, NumPy 1.24+ for data manipulation

Model Architecture

```
Input Features (13) → Feature Engineering (20) → Ensemble Models → Risk Assessment
                                                  ├── Random Forest
                                                  ├── XGBoost
                                                  └── Logistic Regression
```

Performance Specifications

- **Training Dataset**: UCI Heart Disease (303 patients)
- **Cross-Validation**: Stratified 5-fold with 81.39% average accuracy
- **Test Performance**: 86.89% accuracy, 95.35% AUC-ROC
- **Feature Engineering**: 20 features from 13 base clinical parameters

Backend Services

API Framework

- **Framework**: FastAPI 0.101+ for high-performance async REST services
- **Data Validation**: Pydantic 2.1+ for request/response schemas
- **Authentication**: JWT token framework (ready for implementation)
- **Documentation**: Automatic OpenAPI/Swagger generation
- **ASGI Server**: Uvicorn for production deployment

Service Architecture

```
mermaid
```

Prediction Service Components

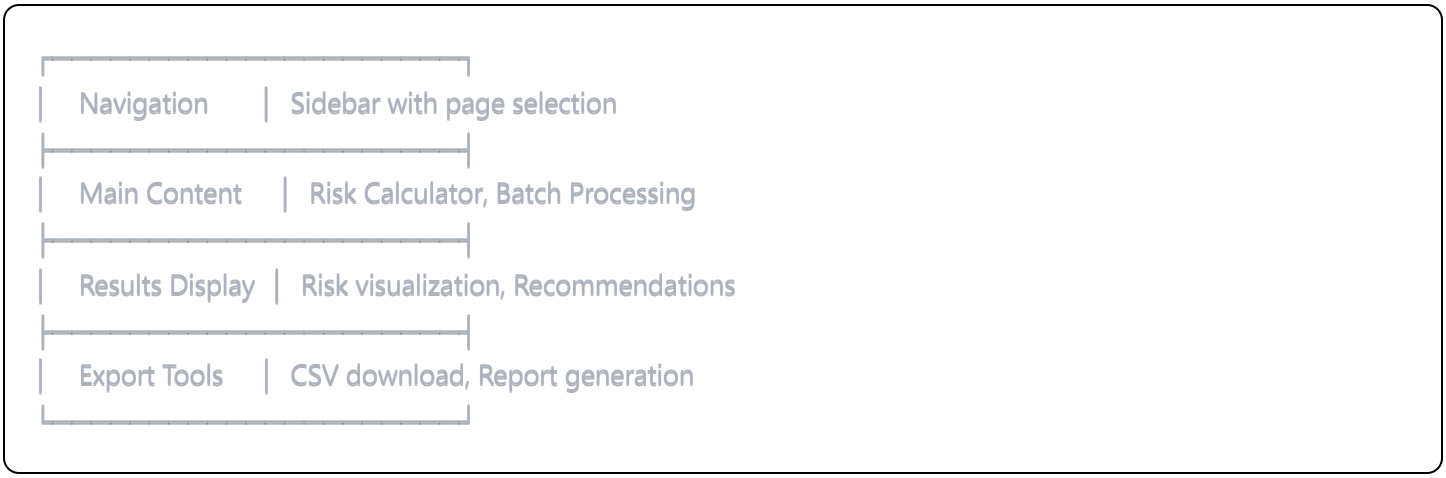
- **Data Validator**: Medical parameter range checking
- **Feature Engineer**: Domain-specific transformations
- **Model Manager**: Ensemble model orchestration
- **Interpretation Engine**: SHAP-based explainable predictions
- **Recommendation Generator**: Personalized health advice

Frontend Application

Framework and Components

- **Framework**: Streamlit 1.25+ for rapid ML application development
- **Styling**: Custom CSS for professional medical interface
- **Visualization**: Plotly for interactive charts and risk gauges
- **Responsiveness**: Mobile-optimized design patterns
- **State Management**: Session state for multi-page navigation

User Interface Architecture



Infrastructure and Deployment

Containerization Strategy

- **Multi-Service Architecture**: Separate containers for API, Dashboard, Proxy
- **Docker Images**: Optimized Python 3.9 slim base images
- **Build Strategy**: Multi-stage builds for production optimization
- **Volume Management**: Persistent storage for models and logs

Container Architecture

Orchestration

- **Docker Compose**: Multi-service orchestration for development and production
- **Service Dependencies**: Proper startup ordering and health checks

- **Network Isolation**: Custom bridge networks for service communication
- **Volume Persistence**: Data and model persistence across container restarts

Production Deployment

- **Platform**: Ubuntu 22.04 LTS on Contabo VPS
- **SSL/TLS**: Let's Encrypt automatic certificate management
- **Domain Configuration**: heartdisease.duminduthushan.com
- **Monitoring**: Docker health checks and structured logging

Data Flow Architecture

Request Processing Pipeline

mermaid

```
graph TD
    A[Patient Data Input] --> B[Input Validation Layer]
    B --> C{Validation Passed?}
    C -->|No| D[Error Response]
    C -->|Yes| E[Feature Engineering Pipeline]
    E --> F[Medical Domain Transformations]
    F --> G[Feature Scaling and Normalization]
    G --> H[ML Ensemble Models]
    H --> I[Prediction Aggregation]
    I --> J[SHAP Interpretation Engine]
    J --> K[Risk Stratification Logic]
    K --> L[Medical Recommendations Generator]
    L --> M[Response Formatting]
    M --> N[Client Application]
```

Feature Engineering Pipeline

1. **Data Validation**: Medical parameter range checking
2. **Missing Value Handling**: Imputation strategies for clinical data
3. **Domain Feature Creation**:
   - Age stratification groups
   - Blood pressure categories
   - Cholesterol risk levels
4. **Interaction Features**:

- Age × Cholesterol interaction
- Blood pressure × Age interaction

5. **Composite Metrics**:
   - Heart rate reserve calculation
   - Cardiovascular risk scoring
6. **Feature Scaling**: RobustScaler for outlier handling

Model Inference Pipeline

1. **Feature Preprocessing**: Apply trained scalers and transformations
2. **Ensemble Prediction**: Aggregate predictions from multiple models
3. **Probability Calibration**: Ensure reliable probability estimates
4. **Risk Classification**: Convert probabilities to risk categories
5. **Interpretation Generation**: SHAP-based feature importance
6. **Recommendation Engine**: Generate personalized health advice

Security Architecture

Data Protection

- **Encryption in Transit**: TLS 1.3 for all client-server communications
- **Data Retention**: No persistent storage of patient health information
- **Input Sanitization**: Comprehensive parameter validation and sanitization
- **Rate Limiting**: Request throttling to prevent abuse (100 req/min)

Access Control

- **Network Security**: Container network isolation
- **Service Authentication**: Inter-service communication security
- **CORS Configuration**: Cross-origin resource sharing controls
- **SSL Configuration**: Strong cipher suites and security headers

Compliance Considerations

- **HIPAA Readiness**: Architecture supports PHI handling requirements
- **Data Processing Transparency**: Clear data flow documentation
- **Audit Capability**: Comprehensive logging for compliance reporting

- **Privacy by Design**: Minimal data collection and processing

Performance Architecture

Scalability Design

- **Horizontal Scaling**: Stateless services for easy horizontal scaling
- **Load Distribution**: Nginx reverse proxy with upstream load balancing
- **Resource Optimization**: Efficient memory usage and CPU utilization
- **Caching Strategy**: Model and configuration caching for performance

Performance Specifications

- **Response Time Targets**:
  - Single prediction: <2 seconds (95th percentile)
  - Batch processing: <1 second per patient
  - Dashboard rendering: <3 seconds initial load
  - API documentation: <1 second response
- **Throughput Capacity**:
  - Concurrent users: 100+ simultaneous assessments
  - Daily predictions: 10,000+ risk assessments
  - Batch processing: Up to 100 patients per request
- **Resource Requirements**:
  - Memory usage: <2GB per service container
  - CPU utilization: <80% under normal load
  - Storage requirements: <10GB total system storage

Monitoring and Observability

- **Health Monitoring**: Comprehensive health check endpoints
- **Performance Metrics**: Response time and throughput monitoring
- **Error Tracking**: Structured error logging and alerting
- **Resource Monitoring**: Container resource usage tracking

Deployment Architecture

Environment Configuration

```yaml
Production Environment:
  ├── Operating System: Ubuntu 22.04 LTS
  ├── Container Runtime: Docker 24.0+
  ├── Orchestration: Docker Compose
  ├── Web Server: Nginx 1.18+
  ├── SSL Management: Certbot/Let's Encrypt
  └── Domain: heartdisease.duminduthushan.com
```

Service Discovery and Communication

- **Internal Communication**: Container names for service resolution
- **External Access**: Nginx proxy with SSL termination
- **Health Checks**: Automated service health monitoring
- **Graceful Degradation**: Fallback mechanisms for service failures

Backup and Recovery

- **Model Artifacts**: Regular backup of trained models and scalers
- **Configuration Backup**: Version-controlled configuration management
- **Container Image Storage**: Versioned container images for rollback
- **Recovery Procedures**: Documented disaster recovery processes

Development and Operations

CI/CD Pipeline (Ready for Implementation)

```mermaid
graph LR
    A[Code Commit] --> B[Automated Testing]
    B --> C[Container Building]
    C --> D[Security Scanning]
    D --> E[Staging Deployment]
    E --> F[Integration Testing]
    F --> G[Production Deployment]
    G --> H[Health Monitoring]
```

Quality Assurance

- **Automated Testing**: Unit, integration, and end-to-end test suites
- **Code Quality**: Linting, formatting, and static analysis
- **Security Scanning**: Container vulnerability assessment
- **Performance Testing**: Load testing and benchmarking

Maintenance and Updates

- **Rolling Updates**: Zero-downtime deployment strategy
- **Configuration Management**: Environment-specific configurations
- **Logging Strategy**: Centralized logging with log rotation
- **Monitoring Alerts**: Proactive issue detection and notification

This architecture provides a robust, scalable, and maintainable foundation for the Heart Disease Risk Assessment System, demonstrating enterprise-grade software engineering practices suitable for production healthcare environments.