Lecture 20 - Contents

An overview of the main sections in this lecture.

This outline is for guidance. Navigate the slides with the left/right arrow keys.

Lecture 20:

Capstone Project

End-to-End Medical AI System

Ho-min Park

homin.park@ghent.ac.kr

powersimmani@gmail.com

Project Overview

Project Scope

- Experience the entire medical AI system development process
- End-to-End implementation from data collection to deployment
- Design considering actual clinical environment
- Team collaboration and project management practice

Final Goals

- Completed medical AI system prototype
- Detailed technical documentation and presentation materials
- Performance evaluation and verification results
- Deployment-ready level code

Evaluation Criteria

- System design and implementation quality (40%)
- Performance and stability (30%)
- Documentation and presentation (20%)

Key Milestones

1. System design and planning (Week 1-2)
2. Implementation and development (Week 3-6)

- Creativity and innovation (10%)

3. Deployment and verification (Week 7-8)
4. Final presentation and evaluation (Week 9)

Part 1/3:

System Design and Planning

1. Requirements Analysis
2. Architecture Planning
3. Technology Stack Selection
4. Data Pipeline Design
5. Model Selection Criteria
6. Integration Points
7. Security Architecture

Requirements Analysis

Functional Requirements

- Medical image analysis and diagnostic support
- Real-time prediction results
- User-friendly interface
- Data management and storage functionality
- Performance monitoring dashboard

Non-Functional Requirements

- Response time < 5 seconds
- 99.9% system availability
- HIPAA compliance
- Scalable architecture
- Data encryption and security

Requirements Matrix

Accuracy (> 95%)	HIGH
Response Speed (< 5s)	HIGH
Data Security	HIGH

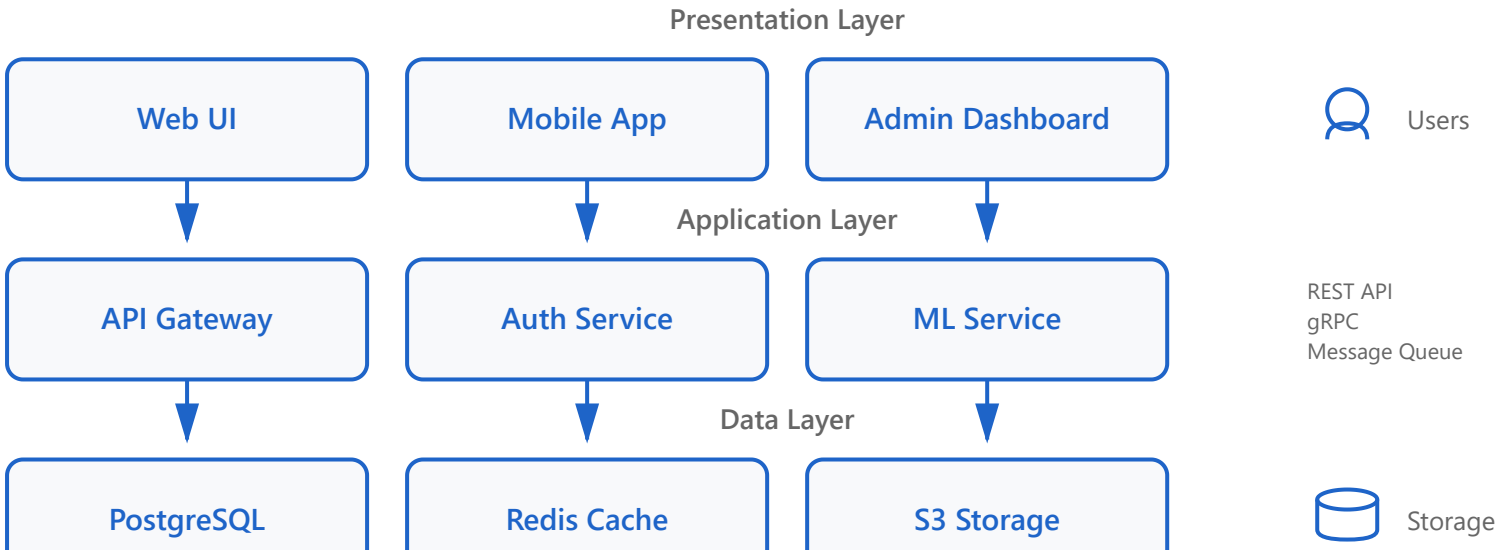
Priority Setting

- Must Have: Core diagnostic features
- Should Have: Monitoring system
- Nice to Have: Advanced visualization

User Experience MEDIUM

- Future: Automatic AI model updates

Architecture Planning - System Components



Core Design Principles

- Microservice Architecture
- Scalable Structure
- Loose Coupling

Communication Protocols

- REST API for synchronous
- Message Queue for async
- gRPC for ML inference

Technology Stack Selection

Backend

- Python (FastAPI)
- Node.js (Express)
- PostgreSQL Database
- Redis Cache

Frontend

- React.js
- TypeScript
- TailwindCSS
- D3.js for visualization

ML/AI

- PyTorch / TensorFlow
- ONNX Runtime
- MLflow for tracking
- Weights & Biases

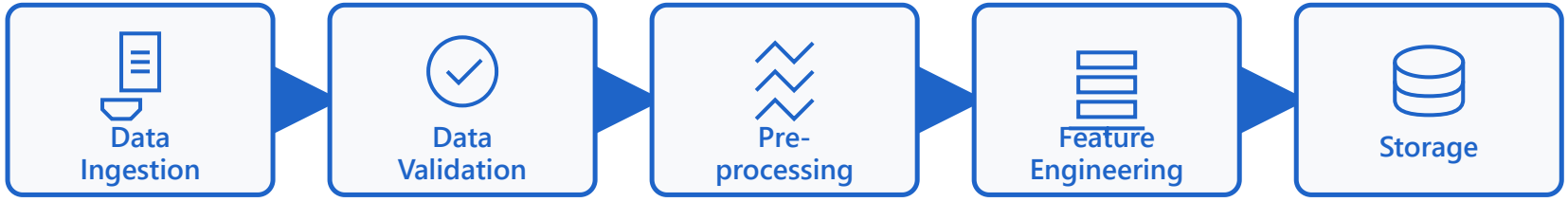
Infrastructure

- Docker / Kubernetes
- AWS / GCP
- CI/CD (GitHub Actions)
- Terraform

Selection Criteria

Community Support	Performance & Scalability	Security & Compliance
Development Productivity	Cost Efficiency	Maintainability

Data Pipeline Design



ETL Process

- Extract: Collect data from various sources
- Transform: Cleansing and transformation
- Load: Store in database

Key Considerations

- Data quality validation
- Scalable architecture
- Real-time processing support

Model Selection Criteria

Model Comparison

- Accuracy vs Speed Tradeoff
- Model Size and Memory Usage
- Inference Time and Throughput

Selection Criteria

- Model Architecture Suitable for Task
- Utilizing Pre-trained Models
- Fine-tuning Strategy

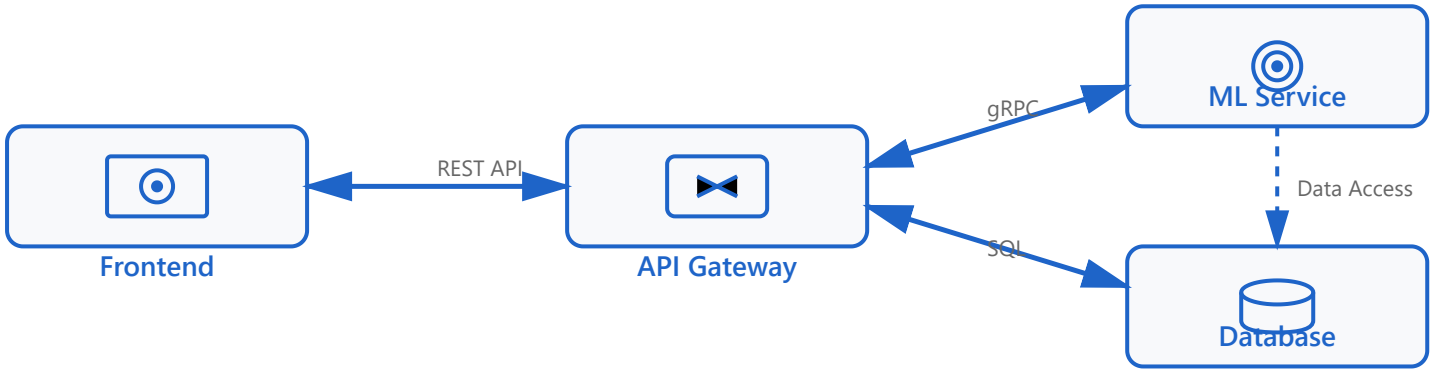
Performance Metrics

- Accuracy, Precision, Recall
- F1 Score, AUC-ROC
- Inference Time and FLOPs

Practical Considerations

- Deployment Environment Constraints
- Maintenance and Updates
- Cost Efficiency

Integration Points



System Integration Map

- Frontend ↔ API Gateway
- API Gateway ↔ ML Service
- ML Service ↔ Database

API Design

- RESTful API Endpoints
- Request/Response Format
- Error Handling Strategy

Interface Definition

- Data Exchange Format (JSON)
- Authentication and Authorization

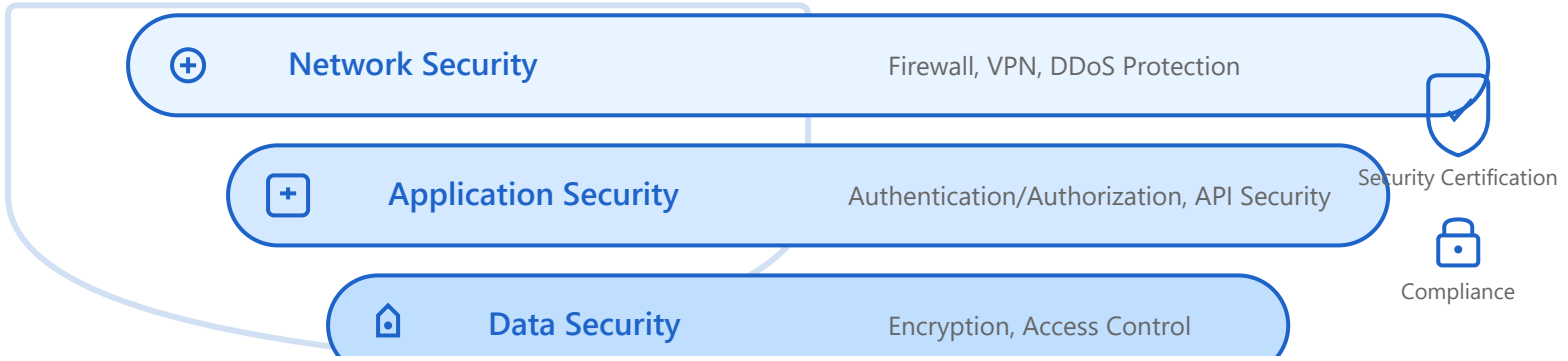
Integration Testing

- End-to-End Testing
- Performance Testing

- Version Management Strategy

- Load Testing

Security Architecture



Security Layers

- Network Security (Firewall, VPN)
- Application Security (Authentication/Authorization)
- Data Security (Encryption)

Threat Model

- SQL Injection Prevention
- XSS/CSRF Attack Prevention
- Data Breach Prevention

Access Control

- Role-Based Access Control (RBAC)
- Multi-Factor Authentication (MFA)

Compliance

- HIPAA Compliance
- GDPR Compliance

- Audit Logging

- Data Protection Policy

Part 2/3:

System Implementation

1. Data Collection & Processing
2. Model Training Pipeline
3. Evaluation Framework
4. API Development
5. Frontend Interface
6. Testing Strategies

Data Collection & Processing

Data Collection

- Securing medical image datasets
- Metadata collection and labeling
- Data quality verification

Data Processing

- Image normalization and resizing
- Data augmentation
- Batch processing optimization

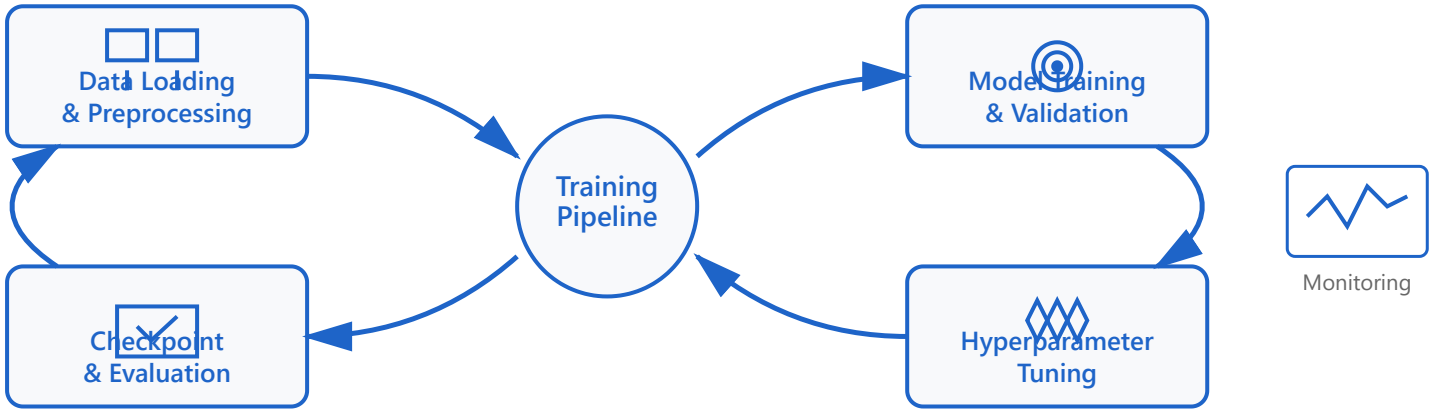
Data Flow

- Real-time data pipeline
- Batch processing scheduling
- Data version management

Implementation Tools

- Pandas, NumPy
- OpenCV, PIL
- Apache Airflow

Model Training Pipeline



Training Process

- Data loading and preprocessing
- Model training and validation
- Hyperparameter tuning

Pipeline Configuration

- Automated training workflow
- Checkpoint saving
- Early Stopping

Monitoring

- Training loss tracking

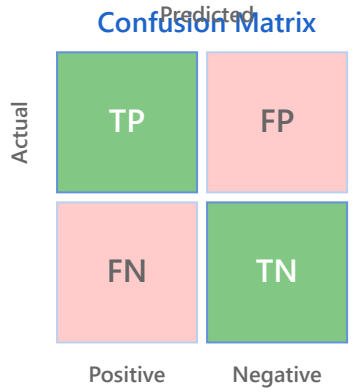
Tools & Utilities

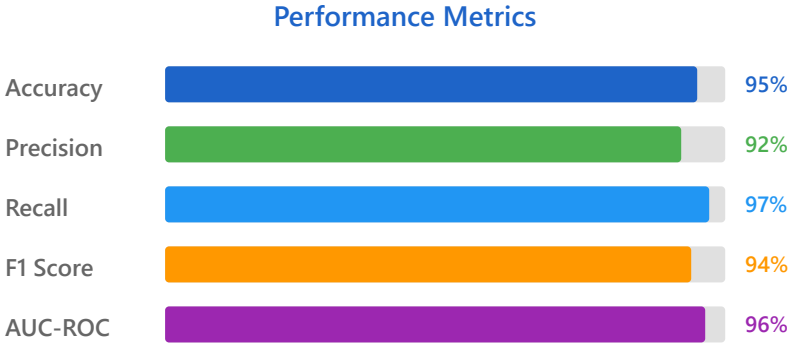
- PyTorch Lightning

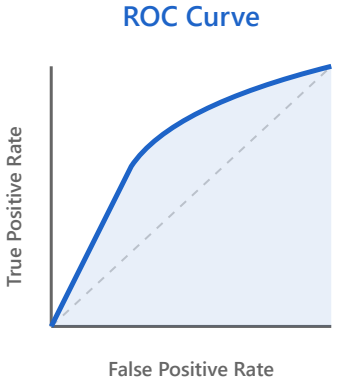
- Validation metrics monitoring
- Resource usage monitoring

- MLflow
- Weights & Biases

Evaluation Framework







Evaluation Metrics

- Accuracy, Precision, Recall
- F1 Score, ROC-AUC
- Confusion Matrix

Testing Strategy

- Hold-out Test Set
- K-Fold Cross Validation
- Stratified Sampling

Performance Analysis

- Error Analysis
- Feature Importance

Validation Process

- Independent Test Data
- Real Environment Simulation

- Model Interpretation

- Statistical Significance Testing

https://claude.ai/chat/459ded5a-adb0-4efd-bc5d-ef4b837650f7

Frontend Interface

UI Design

- Intuitive user interface
- Responsive design
- Accessibility considerations

Core Features

- Image upload
- Real-time prediction results
- Result visualization

User Flow

- Login and authentication
- Data upload
- View and download results

Tech Stack

- React.js
- TypeScript
- TailwindCSS

Testing Strategies

Unit Testing

- Function-level testing
- Independent testing by module
- Using Pytest, Jest

Integration Testing

- Inter-component interaction
- API endpoint testing
- Database integration

E2E Testing

- User scenario testing
- Complete workflow verification
- Selenium, Cypress

Test Coverage

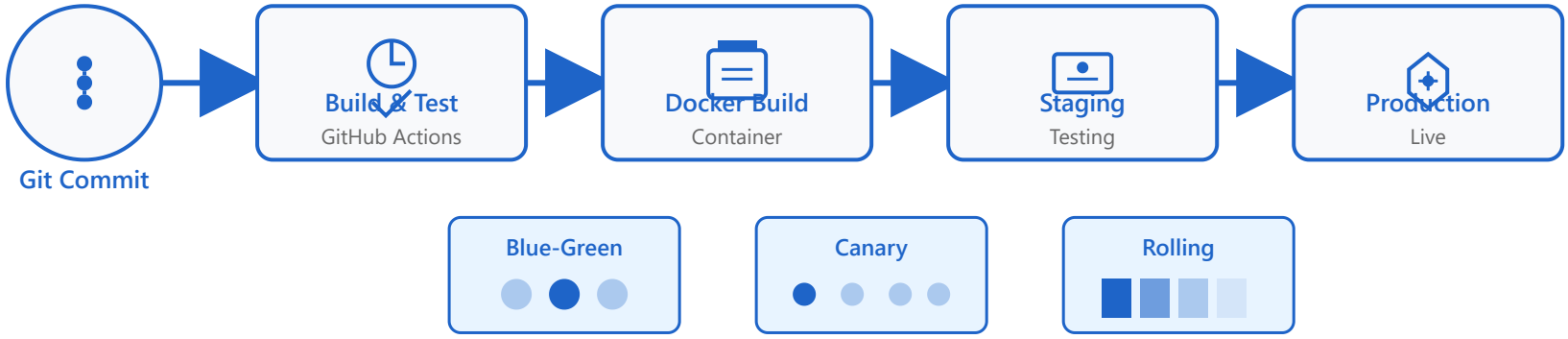
- Code coverage > 80%
- Critical Path 100% coverage
- Automated test execution

Part 3/3:

Deployment and Validation

1. Deployment Pipeline
2. Performance Monitoring
3. Clinical Validation Study
4. User Acceptance Testing
5. Documentation & Training
6. Maintenance Planning

Deployment Pipeline



CI/CD Configuration

- GitHub Actions Workflow
- Automated Build & Test
- Docker Image Creation

Deployment Flow

- Development → Staging → Production
- Blue-Green Deployment
- Canary Release

Infrastructure Management

- Infrastructure as Code (Terraform)

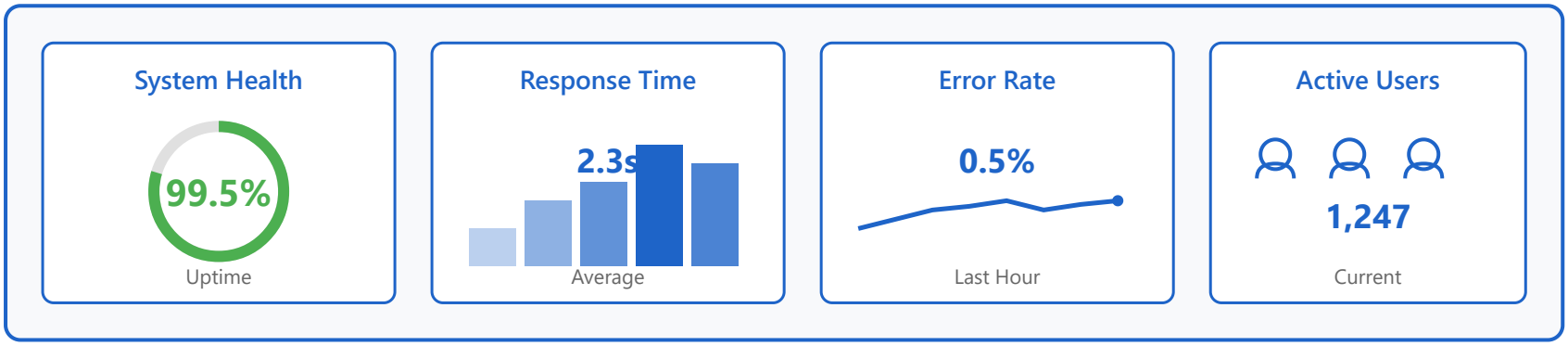
Deployment Automation

- Automatic Rollback Mechanism

- Kubernetes Orchestration
- Auto Scaling Configuration

- Health Check Monitoring
- Deployment Notification System

Performance Monitoring



Monitoring Dashboard

- Real-time System Status
- API Response Time
- Error Rate Tracking

Performance Metrics

- Throughput
- Latency
- Resource Usage

Alert System

- Threshold-based Alerts
- Anomaly Detection

Monitoring Tools

- Prometheus
- Grafana

- Slack/Email Integration

- ELK Stack

Clinical Validation Study

Research Protocol

- Study Design and Objectives
- Inclusion/Exclusion Criteria
- Sample Size Calculation

Validation Metrics

- Diagnostic Accuracy
- Sensitivity and Specificity
- Positive/Negative Predictive Value

Clinical Evaluation

- Expert Comparison Study
- Real Clinical Environment Testing
- Patient Safety Assessment

Regulatory Compliance

- IRB Approval
- FDA/CE Certification Preparation
- Clinical Trial Documentation

User Acceptance Testing (UAT)

UAT Planning

- Write test scenarios
- Involve actual users
- Build test environment

Feedback Collection

- Usability evaluation
- Feature completeness verification
- Identify improvements

Test Cases

- General user scenarios
- Expert workflows
- Exception handling

Result Analysis

- Satisfaction survey
- Prioritize issues
- Establish improvement plan

User Acceptance Testing (UAT)

UAT Planning

- Write test scenarios
- Involve actual users
- Build test environment

Feedback Collection

- Usability evaluation
- Feature completeness verification
- Identify improvements

Test Cases

- General user scenarios
- Expert workflows
- Exception handling

Result Analysis

- Satisfaction survey
- Prioritize issues
- Establish improvement plan

Maintenance Planning

Maintenance Schedule

- Regular Update Schedule
- Security Patch Application
- Performance Optimization

Monitoring and Support

- 24/7 System Monitoring
- User Support System
- Issue Tracking

Update Process

- New Feature Addition
- Bug Fixes
- Model Retraining and Deployment

Long-term Planning

- Technical Debt Management
- System Improvement Roadmap
- Scalability Planning

Project Presentations

Presentation Structure

- Project overview and objectives
- System architecture explanation
- Key features demo
- Results and achievements

Presentation Tips

- Clear and concise delivery
- Effective use of visual materials
- Interaction with audience
- Q&A preparation

Evaluation Criteria

- Technical completeness
- Presentation quality
- Creativity and innovation
- Teamwork and collaboration

Time Allocation

- Presentation 15 minutes
- Demo 5 minutes
- Q&A 5 minutes
- Total 25 minutes

Peer Review Session

Evaluation Rubric

- Feature Completion (25%)
- Code Quality (25%)
- Documentation (20%)
- Creativity (30%)

Feedback Format

- Constructive Criticism
- Specific Improvement Suggestions
- Emphasize Positive Aspects
- Share Learning Points

Peer Evaluation

- Mutual Team Evaluation
- Write Feedback Documents
- Discussion and Questions
- Share Best Practices

Learning Outcomes

- Understanding Different Approaches
- Gaining Problem-Solving Ideas
- Improving Collaboration Skills
- Developing Critical Thinking

Lessons Learned

Technical Lessons

- Effective architecture patterns
- Optimization techniques
- Pitfalls to avoid
- Importance of tool selection

Project Management

- Importance of time management
- Risk response strategies
- Team communication
- Schedule adjustment experience

Improvement Recommendations

- Better design methods
- Test strategy improvement
- Documentation process
- Deployment automation

Personal Growth

- New skill acquisition
- Problem-solving abilities
- Collaboration skills
- Leadership development

Career Opportunities in Medical AI

Key Positions

- ML Engineer
- Data Scientist
- Research Scientist
- Clinical AI Specialist

Required Skills

- Machine Learning Expertise
- Medical Domain Knowledge
- Software Engineering
- Communication Skills

Career Path

- Junior → Mid → Senior Engineer
- Specialist → Lead → Principal
- Research → Product → Management
- Startup Founder

Industry Trends

- Medical AI Market Growth
- Regulatory Environment Changes
- Telemedicine Expansion
- Personalized Medicine

Course Reflection

Learning Journey

- Understanding fundamental concepts
- Hands-on project experience
- Team collaboration experience
- Professional networking

Key Achievements

- End-to-End system development
- Technology stack mastery
- Problem-solving skills
- Portfolio building

Challenges and Overcoming

- Technical difficulties
- Time management
- Team coordination
- Successful resolution

Future Plans

- Continuous learning
- Open source contribution
- Network expansion
- Career development

Congratulations!

Introduction to Biomedical Datascience 과정 수료

여러분은 의료 AI 시스템 구축의 전 과정을 성공적으로 완료하였습니다.

이제 여러분은 의료 데이터 과학 분야의 전문가로서 첫 걸음을 내디뎠습니다.

🏆 수료증이 발급되었습니다

🌟 포트폴리오 프로젝트 완성

Alumni Network

Career Support

Continuous Learning