



AI-Health Project - Complete Details



Project Overview

AI-Health is a comprehensive healthcare service platform that combines AI-powered care plan generation with medical document management. It features a Python FastAPI backend service integrated with AWS services and a modern React frontend for healthcare professionals to manage patient care plans.



Architecture

Monorepo Structure

```
AI-Health/
├── ai-health-service/    # Python FastAPI backend
├── ai-health-ui/        # React frontend
├── docker-compose.yml   # Production deployment
├── docker-compose.dev.yml # Development setup
└── Documentation files
```



Backend Service (ai-health-service)

Technology Stack

- **Framework:** FastAPI (Python 3.8.1+)
- **Dependency Management:** Poetry
- **Cloud Services:**
 - AWS S3 (file storage)
 - AWS Bedrock (AI/ML)
 - AWS Textract (document extraction)
 - AWS Comprehend Medical (NER)
- **Key Libraries:**
 - `boto3` - AWS SDK
 - `pydantic` - Data validation
 - `uvicorn` - ASGI server
 - `python-multipart` - File uploads
 - `PyPDF2`, `python-docx` - Document processing

Core Modules

1. Care Plan Generation (`app/modules/care_plan.py`)

- Integrates with Amazon Bedrock for AI-powered care plan generation
- Supports **5 Foundation Models:**
 1. **Claude 4.5 Sonnet** - Premium advanced AI

2. **Claude 3.7 Sonnet** - Latest standard model
3. **Claude 3.5 Sonnet** - Proven standard performance
4. **Claude 3 Sonnet** - Cost-effective baseline
5. **Amazon Nova Micro** - Medical factors specialist

2. File Upload (**app/modules/file_upload.py**)

- S3 integration with secure file storage
- PDF validation and generic file upload support
- Presigned URL generation for secure access

3. Text Extraction (**app/modules/text_extraction.py**)

- AWS Textract integration for document OCR
- AWS Comprehend Medical for Named Entity Recognition (NER)
- Supports PDF, DOCX, images, and text files

API Endpoints

Health & System

- **GET /** - Welcome and service info
- **GET /health** - Health check with model configuration

Care Plan Generation (prefix: **/care-plan**)

- **POST /generate** - Claude 4.5 Sonnet (premium)
- **POST /claude-37-sonnet** - Claude 3.7 Sonnet
- **POST /claude-35-sonnet** - Claude 3.5 Sonnet
- **POST /claude-3-sonnet** - Claude 3 Sonnet
- **POST /nova-micro** - Amazon Nova Micro (medical factors)
- **POST /compare** - Test all 5 models simultaneously
- **GET /models** - List all configured models

Each generation endpoint has a **/sample** variant for testing with predefined data.

File Management (prefix: **/upload**)

- **POST /pdf** - Upload PDF files to S3
- **POST /file** - Upload any file type
- **GET /file/{bucket}/{key}** - Get presigned URLs
- **POST /extract-medical-data** - Extract data from medical documents

Text Extraction (prefix: **/extract**)

- **POST /text-and-ner** - Extract text and perform NER
- **POST /text-only** - Extract text only
- **POST /ner-only** - Perform NER on provided text
- **GET /supported-formats** - List supported file formats

Bedrock Diagnostics (prefix: `/bedrock`)

- `GET /access-check` - Comprehensive Bedrock access diagnostics
- `GET /permissions` - IAM permission analysis

Data Models

Key Pydantic models defined:

- **PatientInfo** - Age, gender, weight, conditions, allergies
- **PrescriptionItem** - Medication name, dosage, duration, instructions
- **DoctorPrescription** - Complete prescription with patient info
- **CarePlan** - Generated care plan structure with sections
- **CarePlanSection** - Individual care plan section with priority

Configuration (`app/config.py`)

```
# AWS Configuration
aws_access_key_id
aws_secret_access_key
aws_region: "us-east-1"
aws_role_arn (optional)

# S3
s3_bucket_name

# Bedrock Models
bedrock_model_id: "us.anthropic.claude-sonnet-4-5-20250929-v1:0"
claude_37_sonnet_model_id: "us.anthropic.claude-3-7-sonnet-20250219-v1:0"
claude_35_sonnet_model_id: "us.anthropic.claude-3-5-sonnet-20241022-v2:0"
claude_3_sonnet_model_id: "anthropic.claude-3-sonnet-20240229-v1:0"
nova_micro_model_id: "amazon.nova-micro-v1:0"

# Application
app_name: "AI Health Service"
app_version: "1.0.0"
debug: true
```

Frontend UI (ai-health-ui)

Technology Stack

- **Framework:** React 18 with Hooks
- **UI Library:** Material-UI v5 (MUI)
- **Routing:** React Router DOM v7
- **HTTP Client:** Axios
- **Date Handling:** date-fns
- **Build Tool:** React Scripts 5.0.1

Key Components

1. `src/App.js` - Main application

- Side navigation drawer with routes
- Theme configuration
- LocalizationProvider for date pickers

2. `src/components/MedicalFactorForm.js` - Patient data input

- Comprehensive medical factor collection
- Multiple prescription management
- Medication scheduling with table-based UI
- Sample data pre-fill functionality
- Dynamic condition/allergy management

3. `src/components/CarePlanResult.js` - Display care plans

- Accordion-based section organization
- Medication schedule tables
- Print-friendly format
- Data validation and cleaning
- Warning signs and monitoring display

4. `src/components/CarePlan.js` - Main care plan workflow

- Multi-step process (upload → input → preview → generate)
- File upload integration
- Nova Micro API integration

5. `src/components/OnlinePrediction.js` - Direct prediction interface

- Real-time care plan generation
- Model selection

6. `src/components/Settings.js` - Application configuration

- User preferences
- API endpoint management

Features

✅ Medical Factor Input

- Patient demographics (age, gender, weight)
- Multiple medical conditions
- Multiple allergies
- Prescription date picker

✅ Prescription Management

- Dynamic prescription list
- Medication scheduling:
 - Frequency (daily, twice daily, etc.)
 - Time-based administration
 - Food interactions
 - Special instructions
- Add/remove prescriptions

✅ Care Plan Display

- Patient summary
- Care goals
- Medication management with schedules
- Lifestyle recommendations
- Monitoring schedules
- Warning signs
- Follow-up recommendations

✅ UI/UX

- Material Design interface
- Responsive layout
- Loading states and error handling
- Print support
- Sample data quick-fill

Routing

```
/ - Care Plan Generator  
/online-prediction - Direct prediction interface  
/settings - Application settings
```

API Integration

- Proxy configured to <http://localhost:8000>
- Axios for HTTP requests
- Integration with all backend endpoints

Deployment

Docker Setup

Development ([Dockerfile.dev](#))

- Hot reload enabled

- Poetry-based dependency management
- Volume mounts for live code updates
- Port 8000 exposed

Production (**Dockerfile**)

- Optimized build
- Multi-stage if needed
- Production-ready configuration

Docker Compose

```
# docker-compose.yml (Production)
services:
  ai-health-service:
    - Port 8000:8000
    - Env file configuration
    - Volume mounting

  ai-health-ui:
    - Port 3000:80 (nginx)
    - Depends on backend

# docker-compose.dev.yml (Development)
- Hot reload support
- Development configurations
```

Network

- Bridge network: **ai-health-network**
- Services communicate internally

Getting Started

Prerequisites

- Python 3.8.1+
- Node.js 16+ and npm
- Poetry
- AWS account with configured credentials
- S3 bucket
- Bedrock model access

Local Development

Backend

```
cd ai-health-service
poetry install
cp .env.example .env
# Edit .env with AWS credentials
poetry run uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

Frontend

```
cd ai-health-ui
npm install
npm start
```

Docker Development

```
docker-compose -f docker-compose.dev.yml up --build
```

Access Points

- Backend API: <http://localhost:8000>
- API Docs: <http://localhost:8000/docs>
- Frontend: <http://localhost:3000>



Testing

Backend Tests

```
cd ai-health-service
pytest tests/ -v

# Specific tests
python tests/test_care_plan.py
python demo_complete.py
```

Test Files

- [tests/test_api.py](#) - API endpoint tests
- [tests/test_care_plan.py](#) - Care plan module tests
- [tests/test_s3_access.py](#) - S3 integration tests
- [demo_complete.py](#) - Complete functionality demo



Key Features Summary

AI Capabilities

- ✓ **5 Foundation Models** - Claude 4.5, 3.7, 3.5, 3 Sonnet + Amazon Nova Micro
- ✓ **Medical Specialization** - Nova Micro for complex multi-comorbidity cases
- ✓ **Drug Interaction Analysis** - Comprehensive medication safety
- ✓ **Multi-Model Comparison** - Test all models simultaneously
- ✓ **Age-Specific Care** - Elderly care optimization

Document Management

- ✓ **S3 Integration** - Secure file storage
- ✓ **PDF Validation** - Format verification
- ✓ **Text Extraction** - AWS Textract integration
- ✓ **Medical NER** - AWS Comprehend Medical
- ✓ **Presigned URLs** - Secure file access

User Interface

- ✓ **Material UI Design** - Modern, responsive
- ✓ **Multi-Step Workflow** - Guided care plan creation
- ✓ **Medication Scheduling** - Table-based management
- ✓ **Sample Data** - Quick testing
- ✓ **Print Support** - Print-friendly layouts

Developer Experience

- ✓ **Auto-Generated Docs** - Swagger/OpenAPI
- ✓ **Hot Reload** - Development efficiency
- ✓ **Modular Architecture** - Clean separation
- ✓ **Comprehensive Testing** - pytest suite
- ✓ **Docker Support** - Containerized deployment

Important Files

Documentation

- **README.md** - Main project documentation
- **DEVELOPMENT.md** - Development guide
- **ARCHITECTURE_SUMMARY.md** - Architecture details
- **HOT_RELOAD_GUIDE.md** - Development workflow
- **CARE_PLAN_GUIDE.md** - Care plan features
- Various model-specific guides (Claude, Nova)

Configuration

- **ai-health-service/pyproject.toml** - Python dependencies
- **ai-health-ui/package.json** - Node.js dependencies
- **ai-health-service/.env** - Environment variables

Sample Data

- `sample-medical-documents.json` - Sample medical documents
 - `sample-medical-text.txt` - Sample medical text
-

Use Cases

1. **Healthcare Professionals** - Generate comprehensive care plans from prescriptions
 2. **Medical Documentation** - Upload and extract data from medical documents
 3. **Patient Care Management** - Manage medications and monitoring schedules
 4. **AI Model Comparison** - Test different AI models for best results
 5. **Medical Research** - Analyze medical documents with NER
-

Project Statistics

Backend

- **Lines of Code:** ~2000+ (Python)
- **Modules:** 3 core modules
- **Routes:** 4 route groups
- **API Endpoints:** 20+ endpoints
- **Test Files:** 4 test suites

Frontend

- **Lines of Code:** ~3000+ (JavaScript/React)
- **Components:** 6+ main components
- **Routes:** 3 main routes
- **Dependencies:** 15+ npm packages

Infrastructure

- **Docker Images:** 4 (2 services × 2 environments)
 - **AWS Services:** 4 integrated services
 - **AI Models:** 5 foundation models
-

Security Features

- AWS credential management
 - Presigned URLs for secure file access
 - Environment-based configuration
 - CORS configuration for API security
 - Secure file upload validation
-

Project Highlights

1. **Multi-Model AI Integration** - First healthcare platform to integrate 5 different foundation models
 2. **Specialized Medical AI** - Amazon Nova Micro specifically for complex medical cases
 3. **Comprehensive NER** - AWS Comprehend Medical for medical entity extraction
 4. **Modern Tech Stack** - FastAPI + React with Material UI
 5. **Production Ready** - Docker containerization with dev/prod environments
 6. **Extensive Documentation** - 10+ markdown guides
 7. **End-to-End Workflow** - From document upload to care plan generation
-



License & Ownership

- **Repository:** AI-Health
 - **Owner:** pandianvasantharajan
 - **Branch:** main
 - **Last Updated:** November 7, 2025
-



Development Status



Completed Features

- Multi-model AI integration
- Document management system
- Care plan generation
- Medical data extraction
- React UI with Material Design
- Docker deployment setup
- Comprehensive testing



Potential Enhancements

- Additional AI models
 - Real-time collaboration
 - Enhanced analytics dashboard
 - Mobile application
 - Integration with EHR systems
 - Multi-language support
-

This is a production-ready, enterprise-level healthcare AI platform with robust AWS integration, multiple AI models, and a modern user interface designed for healthcare workflows.