AED Final Project Group 15:

Group Members:
1. Gagana Ananda
2. Syed Hameed Uddin
3. Jayanth Muthaluri
4. Mallesh Mallikarjunaiah

TA: Chenyan Zhu

Project Title: CareLink 360

1. Problem Statement

In real-world healthcare environments, hospitals, insurance companies, and non-governmental organizations (NGOs) often operate in separate systems that lack integration and coordinated workflows. This fragmentation creates significant challenges, such as delays in patient support, inefficient communication between departments, difficulty in processing insurance claims, and limited visibility of ongoing requests.

Healthcare professionals, insurance agents, and NGO coordinators frequently rely on manual methods—emails, physical documents, or phone calls—to handle critical tasks like patient support requests, insurance approvals, and donation allocations. These disconnected processes result in miscommunication, duplication of work, slower service delivery, and poor tracking of responsibilities.

To address these issues, there is a need for a unified, ecosystem-based application that brings together multiple stakeholders—hospitals, insurance enterprises, and NGOs—under a single platform. Such a system should enable seamless communication, structured workflows, and streamlined task management between different organizations and roles.

This project aims to solve the real-world challenge of coordination across multi-organization healthcare networks by providing an integrated Java Swing application that supports inter-enterprise communication, structured work queues, role-based operations, and centralized data management. The system enhances operational efficiency, reduces delays, and ensures that all entities in the healthcare ecosystem can collaborate effectively to deliver timely and accurate support to patients and beneficiaries.

2. Project Objectives and Goals

1. Build a Multi-Enterprise Ecosystem

The system aims to create a unified healthcare ecosystem that integrates multiple types of enterprises, including hospitals, insurance companies, NGOs, donation organizations, and blood banks. Each enterprise contains several organizations with dedicated functions and workflows. This ecosystem structure enables seamless collaboration between independently functioning units, ensuring that patients, donors, insurance customers, and healthcare professionals are part of one connected platform.

2. Implement Role-Based Access and Authentication

A key requirement is to provide secure authentication for all users based on their roles. Users such as system administrators, hospital admins, doctors, lab assistants, accountants, insurance agents, NGO coordinators, patient and blood bank managers should have customized dashboards and controlled access privileges. The system ensures that each role can only access screens and perform actions that fall within their responsibilities, maintaining privacy, data integrity, and operational security.

3. Enable Workflow and Request Management

The platform must include an integrated Work Queue system to manage and track requests across the ecosystem. Users should be able to create new requests, assign them to the appropriate organization or role, and monitor the progress of these requests until completion. This facilitates smooth communication between enterprises—for example, a doctor sending lab tests to a lab assistant, an NGO requesting patient details from the hospital, or a customer applying for an insurance claim. The system ensures transparency and accountability throughout the request lifecycle.

4. Insurance Policy & Customer Management

The system should provide modules for comprehensive insurance management. This includes creating and updating insurance policies, registering insurance customers, tracking policy directories, and managing policy coverage and premium information. Insurance agents must be able to process claims, approve or decline policy requests, and view customer-specific policy histories. This ensures streamlined operations for both customers and the insurance enterprise.

5. Healthcare Services Workflow

The application must support end-to-end hospital workflow. Doctors should be able to record diagnoses, create test requests, update patient medical information, and refer cases to labs. Lab assistants should handle sample processing, update test results, and send reports back to doctors. Hospital administrators must manage employees, organizations, and patient interactions efficiently. This module ensures that the hospital functions smoothly as part of the broader ecosystem.

6. Donation & NGO Management

A dedicated module should support NGOs involved in patient assistance and resource allocation. NGO coordinators must track donation requests, maintain donor lists, and coordinate with hospitals for support. NGOs should be able to manage their own workflow, such as verifying patient needs, allocating resources, and updating request statuses. This ensures that social support systems function effectively within the healthcare network, strengthening community engagement.

7. Persist All Data Using DB4O

To ensure long-term data availability and consistency, the system must store all information using the DB4O embedded object database. This includes networks, enterprises, organizations, users, employees, roles, insurance policies, work requests, donor information, and system credentials. DB4O allows the application to store and retrieve Java objects directly without requiring complex relational schemas, ensuring high efficiency and smooth data persistence across application restarts.

8. Provide an Extensible, Modular Architecture

The overall architecture must follow a modular pattern where the ecosystem contains multiple networks, each hosting enterprises that include various organizations. This clear hierarchy—Ecosystem → Network → Enterprise → Organization → User Role—ensures clean separation of concerns. The design allows easy integration of new enterprises or workflows in the future without affecting existing functionalities. A well-organized UI and business logic structure ensures maintainability, scalability, and system flexibility.

3. User Roles and Use Cases

Role 1: Doctor / Healthcare Staff

Doctors manage patient requests and provide medical decisions within the system.

Use Case 1: Review Patient Requests

- Access incoming WorkRequests assigned to their organization.
- View details such as symptoms, appointment needs, or insurance-related medical information.

Use Case 2: Approve / Reject Requests & Add Medical Notes

- Update diagnosis, treatment comments, or additional instructions.
- Change request status (approved, rejected, under review).
- Updates flow back to the patient through the WorkQueue.

Use Case 3: Support Insurance Verification

- Provide medical justification required for insurance claims.
- Coordinate with insurance officers via shared enterprise modules.
- Contribute to claim approval decisions through verified medical inputs.

Role 2: Patient / Insurance Customer

Patients use the system to request services and stay informed about their healthcare needs.

Use Case 1: Manage Personal Profile

- Log in through the UserAccount module.
- View or update basic personal information stored in Patient/InsuranceCustomer models.

Use Case 2: Submit Medical or Insurance Requests

- Request appointments, treatment support, or insurance claim assistance.
- A WorkRequest is created and added to the WorkQueue.
- Requests are routed to the relevant organization (Doctor or Insurance).

Use Case 3: Track Request Status

- Monitor real-time updates on their submitted requests.
- Status changes reflect actions taken by Doctor, Insurance, or Admin roles.
- Provides clear visibility into progress and next steps.

4. System Architecture and Design Approach

This project is designed using a simple, modular architecture that separates the system's logic, workflows, and user interface. The goal is to keep the system beginner-friendly, easy to maintain, and scalable for future enhancements.

High-Level Architecture

1. Business Layer (Core Logic Layer)

Location: src/Business/

This layer defines the core structure and behavior of the system. It includes:

- EcoSystem – the central controller that manages all Networks, Enterprises, and Organizations.
- Network / Enterprise / Organization – represent the healthcare structure, such as doctor units, insurance units, and patient organizations.
- UserAccount & Role – provide login, authentication, and role-based access control.
- WorkRequest & WorkQueue – manage communication between users (e.g., Patient to Doctor, Doctor to Insurance).

This layer ensures that all system rules, workflows, and data handling remain consistent and well organized.

2. User Interface Layer

Location: src/userinterface/

This layer contains all Java Swing panels used by different roles:

- Login screen
- Doctor dashboard
- Patient dashboard
- Insurance dashboard
- Admin panels

Each screen interacts with the Business Layer to display data, submit requests, update statuses, and allow users to perform their required tasks.
This separation keeps the interface simple while ensuring all logic remains centralized in the Business Layer.

3. Persistence Layer

Handled by:

- DB4OUtil.java

This module:

- Saves the entire EcoSystem state in the DB4O object database.
- Loads system data during startup.
- Ensures persistence across sessions.

Design Approach

1. Modular Multi-Tier Design

The project is divided into clear modules:

- Data models
- Business logic
- User interface components

This improves clarity, maintainability, and makes the system easy to extend with new roles or features.

---

2. Object-Oriented Principles

The system uses essential OOP concepts such as:

- Inheritance (Enterprise and Organization hierarchies)
- Composition (EcoSystem contains Networks → Enterprises → Organizations)
- Polymorphism (roles generate different UI panels)

This keeps the system structured, reusable, and flexible.

---

3. WorkQueue-Driven Workflow

The core workflow is built around the WorkQueue pattern:

- Patients submit requests
- Doctors review and update them
- Insurance staff verify claims

Each step updates the request status, enabling smooth coordination between roles and ensuring a realistic healthcare flow.

---

4. Problem-Solving Strategy

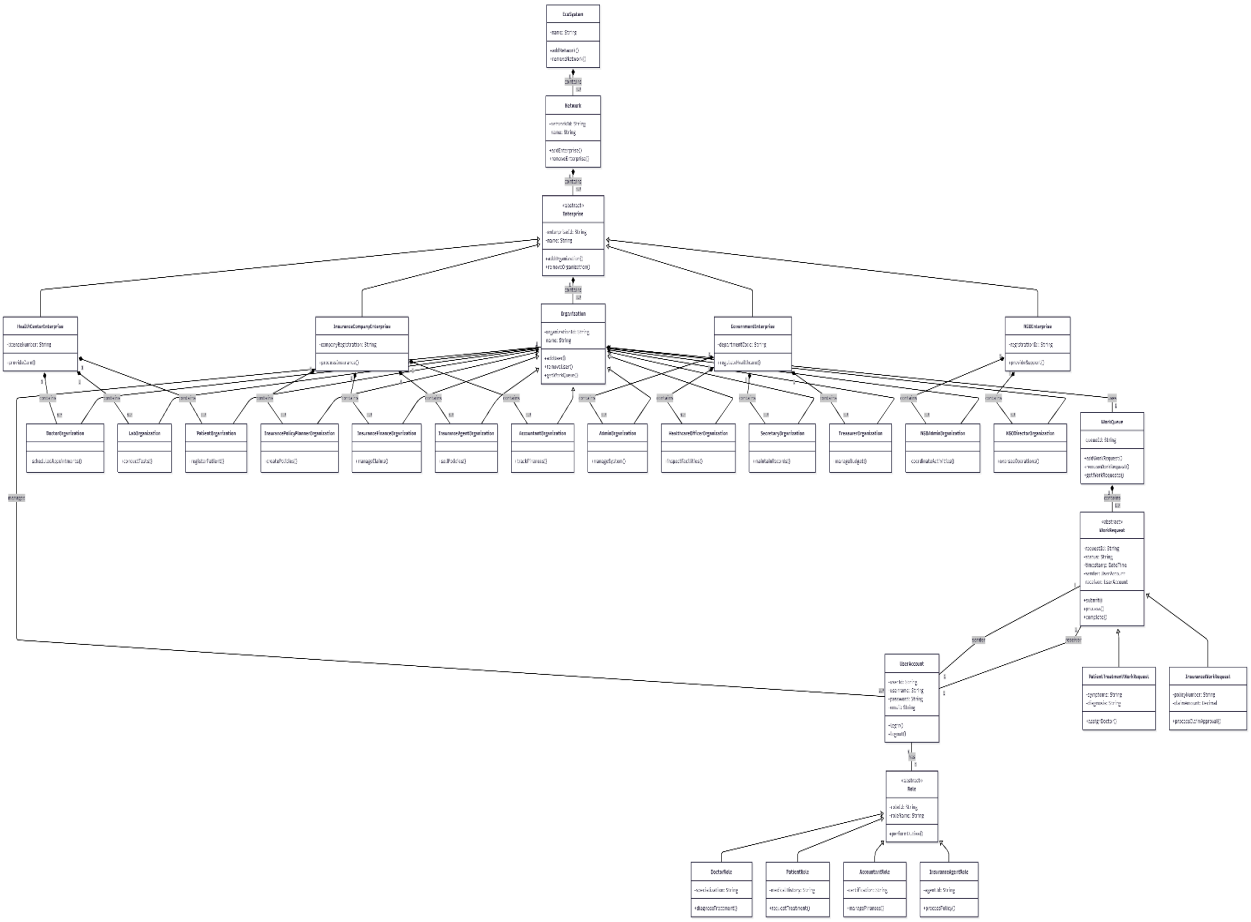
The design solves the healthcare workflow challenge by:

- Clearly defining user roles
- Creating dashboards for each role
- Organizing the system into networks, enterprises, and organizations
- Using WorkQueue to support request movement and status updates

This makes the application intuitive for users and easy for developers to enhance over time.

Object Model (UML Class Diagram)

Automated_HealthCare-2025-11-19-053913.png



Data Input and Output

1. User Login and Work Area Access

The system initiates the workflow by authenticating a user and directing them to their specialized workspace.

- A user provides credentials to log in via the UserAccount module.
- The system verifies the associated Role (e.g., Doctor, Patient, Insurance Agent).
- The system uses the Role class to generate the correct Work Area Panel for the user (e.g., Patient Work Area Panel or Doctor Work Area Panel), ensuring Role-Based Access Control (RBAC).

2. Work Request Initiation (Input)

Work begins when a patient or another organization submits a need.

- Initiator: Typically the Patient Role or a staff member.
- Action: The user requests a service (e.g., appointment, claim, fund request).
- Creation: A specialized WorkRequest object is created (e.g., Patient Treatment Work Request, Insurance Work Request, Government Fund Request).
- Routing: The WorkRequest is assigned a sender and a receiver (both are User Account objects) and its status is set to "Pending."
- Queuing: The request is added to the relevant Organization's WorkQueue (e.g., a patient request goes to the Doctor Organization's WorkQueue).

3. Task Processing and Inter-Organizational Flow

Stage	Role Involved	Action/Business Logic	WorkRequest Update
Medical Processing	Doctor Role	Reviews the Patient Treatment Work Request. If approved, they may use the Provide Prescription Panel to add details or create a new Lab Test Work Request and send it to the **Lab Assistant Role**.	Status changes from "Pending" to "Approved" or "Under Review."

Lab/Support	Lab Assistant Role	Processes the Lab Test Work Request.	Updates the request with results and sends it back to the Doctor.
Insurance Claim	Insurance Agent Role	Processes the Insurance Work Request. Uses the Process Request Panel to verify policy details and medical justification provided by the Doctor.	Status updated based on claim verification.
Financial/Billing	Accountant Role	Review the Accountant Billing Request. Uses the Process Medical Billings Panel to calculate the final amount and may use the Email Billing Information To Patient Panel to send the output.	Request closed, billing finalized.
Administrative/Funds	NGO Director Role / Government Treasurer Role	Processes NGO Fund Request or Government Fund Request by reviewing documents. The request may be routed internally (e.g., Secretary to Treasurer to Director).	Status updated at each approval stage.
Blood Bank	Blood Bank Manager	Manages blood unit requests from hospitals/patients	Requested → Available/Unavailable → Reserved → Delivered
NGO Patient Assistance	NGO Coordinator	Processes patient requests for financial aid or resources	Under Review → Approved/Denied → Resource Allocated → Support Complete
Insurance Policy Registration	Insurance Policy Planner	Sets up new insurance policies for customers	Application Received → Policy Created → Active
Government Facility Inspection	Healthcare Officer	Conducts compliance inspections of healthcare facilities	Scheduled → In Progress → Report Submitted →

			Compliant/Requires Follow-up

Category	Planned Items	Count
Enterprises	1. HealthCenterEnterprise 2.InsuranceCompany Enterprise 3. GovernmentEnterprise 4. NGOEnterprise	4 Enterprises
Organizations	HealthCenterEnterprise: 1. DoctorOrganization 2. LabOrganization 3. PatientRegistrationOrganization InsuranceCompanyEnterprise: 4. InsurancePolicyOwnerOrganization 5. InsuranceBillingOrganization 6. InsuranceAgentOrganization GovernmentEnterprise: 7. AccountantOrganization 8. AdminOrganization 9. HealthCommOfficerOrganization 10. SecretaryOrganization NGOEnterprise: 11. TreasurerOrganization 12. NGOAdminOrganization 13. NGODoctorOrganization	13 Organizations
Roles	1. DoctorRole 2. LabAssistantRole 3. PatientRole 4. AccountantRole 5. InsuranceAgentRole 6. SecretaryRole 7. PolicyOwnerRole 8. InsuranceBillingRole 9. AdminRole	13 Roles

	10. HealthCommOfficerRole 11. TreasurerRole 12. NGOAdminRole 13. NGODoctorRole	
Work Request Types	1. PatientTreatmentWorkRequest 2. InsuranceBillingWorkRequest 3. InsuranceVerificationWorkRequest 4. BloodBankWorkRequest 5. NGOPatientAssistanceWorkRequest 6. InsurancePolicyRegistrationWorkRequest 7. GovernmentFacilityInspectionWorkRequest	7 Work Requests

4. Resolution and Feedback (Output)

The final stage ensures the initiator is informed, and records are updated.

- Completion: The final staff member updates the WorkRequest status to "Complete" or "Rejected."
- Data Persistence: New data (e.g., medical notes, policy approvals) is stored in the core Business Layer models (e.g., Patient, Insurance).
- Notification: The original sender (e.g., the Patient Role) sees the real-time status change on their dashboard, providing clear visibility into progress. The output is the final decision, justification, or required next steps.

System Maintenance: System Admin Role uses panels like Manage Network Panel and Manage Enterprise Panel to ensure the core hierarchical structure (EcoSystem $\to$ Network $\to$ Enterprise $\to$ Organization) is properly maintained for future workflows.

Implementation Plan and Timeline

Week / Dates	Tasks / Milestones	Deliverables
Week 1Nov 18 – Nov 24	1. Finalize architecture hierarchy. 2. Complete UML & component diagrams. 3. TA review & proposal approval. 4. Create GitHub repo + branches. 5. Set coding/commit rules.	1. Approved proposal. 2. UML diagrams. 3. NetBeans skeleton project. 4. GitHub structure prepared.
Week 2Nov 25 – Dec 1	1. Implement Business Layer classes. 2. Setup DB4OUtil. 3. Create test networks & enterprises. 4. Seed users & sample requests. 5. Write Milestone Update (Dec 1)	1. Business layer functional. 2. DB4O persistence working. 3. Sample test data ready. 4. Milestone Update submitted.
Week 3Dec 1 – Dec 5	1. Develop all UI panels. 2. Connect UI → Business layer. 3. Implement doctor–insurance–NGO workflows. 4. Create test scenarios. 5. Mid-testing & validation.	1. Functional UI. 2. End-to-end workflows. 3. Test scenarios executed.
Week 4Dec 5 – Dec 7	1. Full functional testing. 2. DB4O save/load testing. 3. Fix layout and validation issues. 4. Resolve Git conflicts. 5. Prepare PPT, diagrams, final folders.	1. Fully working project. 2. All branches merged. 3. Submission package ready.

Assumptions and Limitations

Assumption	Explanation
Valid User Inputs	1. Users provide correct details 2. Minimal validation required
Single-System Usage	1. Application runs locally 2. No multi-user concurrency
Linear Workflow	1. Requests follow fixed path: Patient → Doctor → Insurance → NGO
DB4O Stability	1. Database file remains intact 2. No corruption due to IDE version mismatch
Role-Based Access	1. Users only perform allowed actions 2. No privilege escalation
GitHub Workflow	1. Each member uses own branch 2. PRs used to merge to main
Stable Test Data	1. Seeded data reused for consistency 2. Testing controlled
IDE Compatibility	1. All members use similar NetBeans version
WorkQueue Workflow	1. Suitable to simulate real tasks 2. Supports multi-enterprise flow

Limitation	Explanation
No Online Database	1. Only DB4O used 2. No MySQL/Mongo support
No Multi-User Access	1. Single-machine usage 2. Not networked or cloud-based
UI Constraints	1. Swing limitations 2. Hard to create modern responsive UI
DB File Issues	1. DB4O file can corrupt if merged 2. Should not be pushed to GitHub
Frequent Merge Conflicts	1. .form files conflict easily 2. Hard to resolve XML UI conflicts
Limited Security	1. No encryption 2. Simple username/password
Not Production-Ready	1. For academic simulation only 2. No HIPAA compliance
Dependency Bottlenecks	1. UI depends on Business Layer 2. Some tasks cannot run in parallel
Time Constraints	1. Deadline limits advanced features 2. Testing time reduced
Complex Workflows	1. Multi-enterprise tasks require extra testing 2. Hard to debug with limited time

Approval Status: Approved
Approved by: Chenyan Zhu
Approval date: 11/21/2025