

**DYNAMIC CONNECTION ROUTING & MANAGEMENT SYSTEM**  
**HIGH-LEVEL DESIGN (HLD) & LOW-LEVEL DESIGN (LLD)**

Document Version: 1.0

Prepared by: Group-01

Date: 13-02-2026

Status: Final Design Document

**TABLE OF CONTENTS**

1. INTRODUCTION .....	1
1.1 Purpose .....	1
1.2 Scope .....	1
1.3 Definitions .....	2
1.3.1 Client	
1.3.2 Endpoint Node	
1.3.3 Central Routing Engine	
1.3.4 Redirection Policy	
1.3.5 Telemetry	
1.4 Overview .....	3
2. GENERAL DESCRIPTION .....	4
2.1 Product Perspective	
2.2 Product Functions	
2.3 Operating Environment	
2.4 Design Constraints	
2.5 Assumptions	
2.6 Special Design Prospects	
3. SYSTEM DESIGN DETAILS (HLD) .....	6
3.1 Main Design Features	
3.2 Flow Chart	
3.2.1 System Architecture	
3.3 Level 0 Design Diagram	
3.4 Level 1 Design Diagram	

3.5 Files	
3.6 User Interface	
3.7 Help	
3.8 Performance	
3.9 Security	
3.10 Reliability	
3.11 Maintainability	
3.12 Portability	
3.13 Reusability	
3.14 Application Compatibility	
3.15 Resource Utilization	
4. LOW-LEVEL DESIGN (LLD) .....	10
4.1 Purpose	
4.2 Document Conventions	
4.3 Intended Audience and Reading Suggestions	
5. DETAILED SYSTEM DESIGN .....	12
5.1 Design Descriptions	
5.2 Class Diagram	
5.2.1 Server Class Diagram	
5.2.2 Node Class Diagram	
5.2.3 Admin Class Diagram	
5.2.4 Entry Point Class Diagram	
5.3 Use Case Diagram	
5.4 Sequence Diagram	
5.5 Design and Implementation Constraints	
5.6 User Interface	

# High Level Design Document

## 1. INTRODUCTION

### 1.1 PURPOSE

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the Distributed Central Routing Management System (DCRMS). This document outlines the functional and non-functional requirements of the system and serves as a reference for developers, designers, testers, and project stakeholders during system development.

It also helps identify system constraints and ensures that all requirements are clearly defined before implementation.

### 1.2 SCOPE

The Distributed Central Routing Management System is designed to manage communication between multiple endpoint nodes through a centralized routing engine. The system authenticates nodes, monitors their availability, and intelligently routes client sessions to appropriate nodes.

The system supports:

- Secure node enrollment
- Centralized routing
- Node-level redirection policies
- Administrative overrides
- Telemetry logging
- Failover handling

DCRMS aims to improve **system reliability, scalability, and fault tolerance** in distributed environments.

## **1.3 DEFINITIONS**

### **1.3.1 CLIENT**

A client is an application or user that initiates a session request to communicate with an available endpoint node

### **1.3.2 ENDPOINT NODE**

An endpoint node is a distributed system component capable of handling client sessions. Each node maintains its own redirection policy.

### **1.3.3 CENTRAL ROUTING ENGINE**

The core component responsible for authentication, routing decisions, policy evaluation, and system monitoring.

### **1.3.4 REDIRECTION POLICY**

A predefined set of rules that determines how traffic is redirected when a node becomes unavailable, busy, or fails.

### **1.3.5 TELEMETRY**

The automated recording of system events for monitoring and diagnostics.

## **1.4 OVERVIEW**

This HLD Document is arranged in the following format:

Section 1: Introduction

A brief explanation about the purpose, aim, scope, and design format of the proposed system.

## Section 2: General Description

This section is all about the general constraints, assumptions, and design aspects associated with the proposed system. The product perspective will give an overall description of the system.

## Section 3: Design Details

This section documents the detailed design of all modules associated with the development of the proposed system.

## **2. GENERAL DESCRIPTION**

### **2.1 PRODUCT PERSPECTIVE**

DCRMS is a distributed client-server system that uses a centralized routing engine to manage communication between clients and endpoint nodes.

The system ensures high availability by dynamically selecting alternate nodes when failures occur. With node-level redirection policies, routing decisions become more flexible and efficient.

### **2.2 PRODUCT FUNCTIONS**

Major functions of the system include:

- Node authentication and enrollment
- Monitoring node health and availability
- Intelligent session routing
- Evaluation of node-specific redirection policies
- Administrative configuration management
- Telemetry logging
- Failover support

## **2.3 OPERATING ENVIRONMENT**

The system operates in a Linux-based environment and is developed using C++. It relies on stable network connectivity for communication between clients, nodes, and the routing engine.

## **2.4 DESIGN CONSTRAINTS**

- The system must be implemented using C++
- Network connectivity must be reliable
- The central server must remain operational
- Nodes must support secure communication

## **2.5 ASSUMPTIONS**

- Clients have network access
- Endpoint nodes are properly configured
- Administrators manage routing policies
- System resources are sufficient for concurrent sessions

## **2.6 SPECIAL DESIGN PROSPECTS**

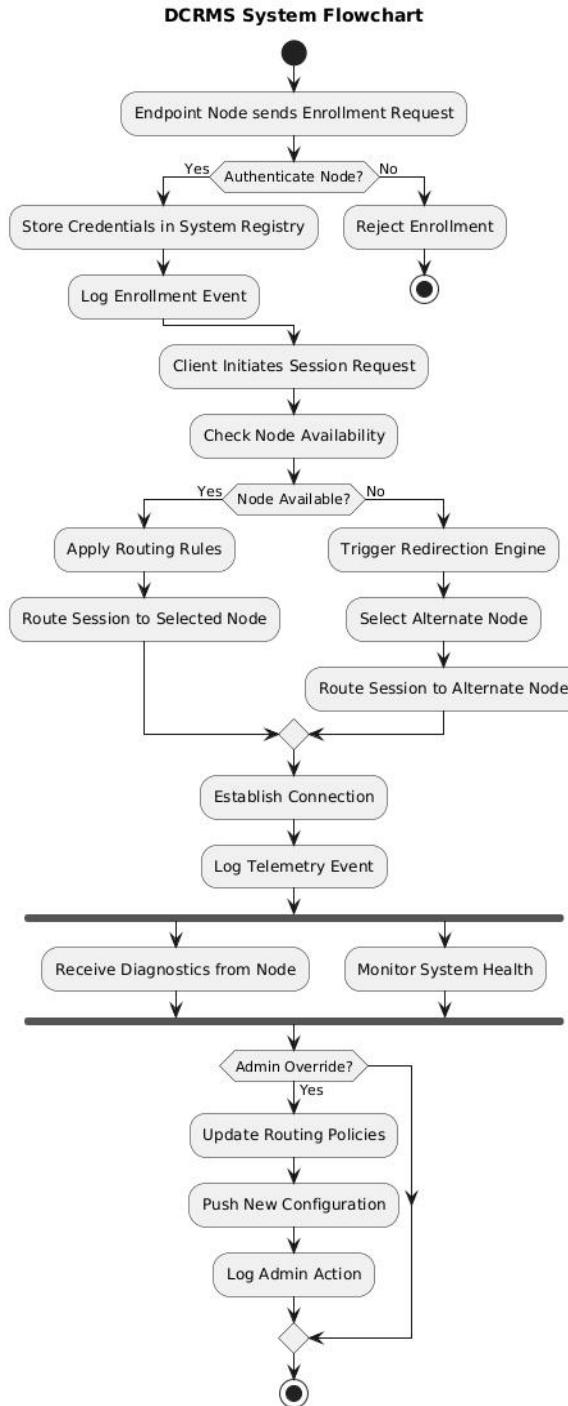
A key design feature is the implementation of node-level redirection policies, allowing each node to define preferred routing alternatives. This reduces dependency on centralized decision-making and enhances system scalability.

# **3. DESIGN DETAILS**

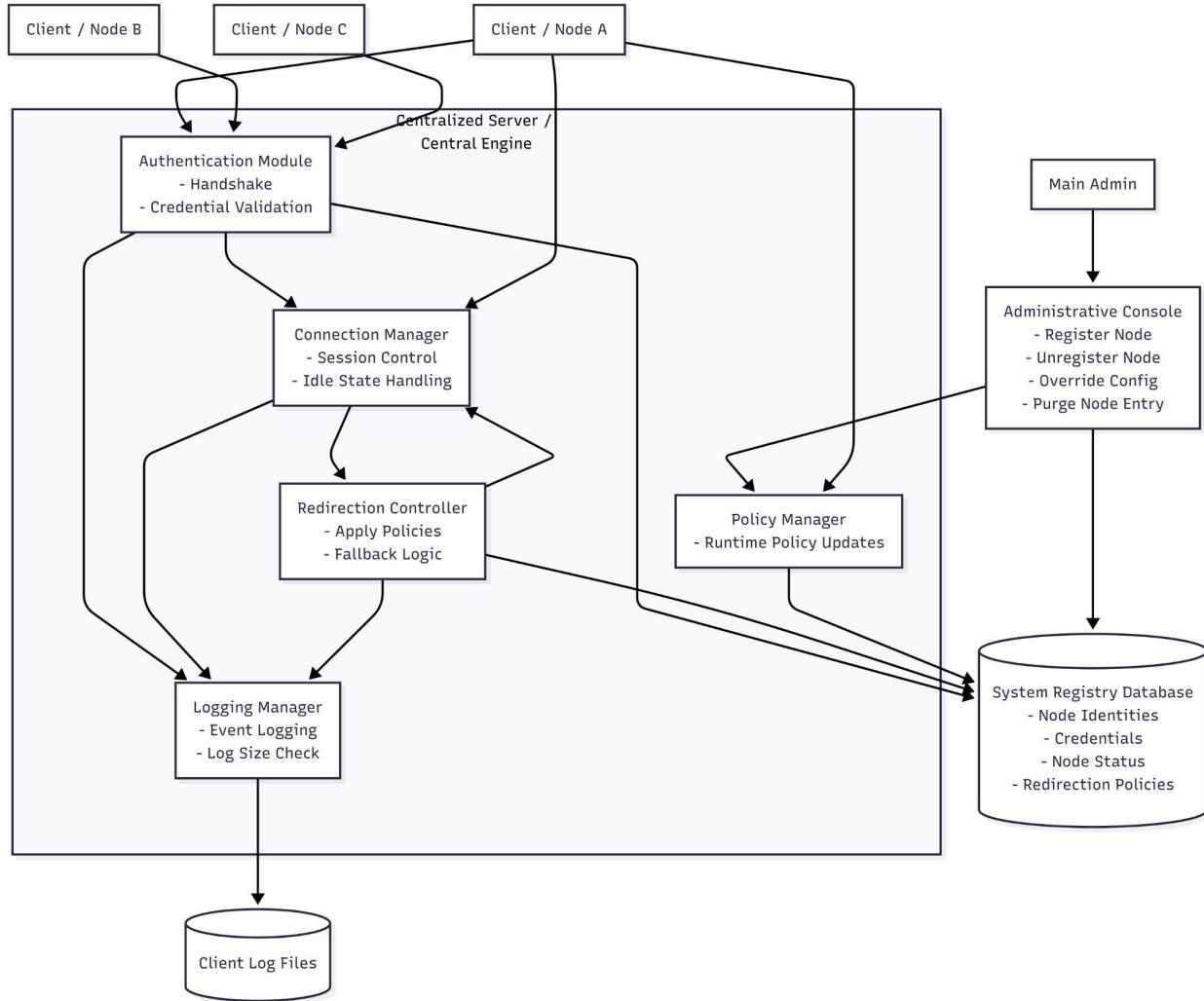
## **3.1 MAIN DESIGN FEATURES**

The main design features include four major parts: the architecture, the user interface design, the text processing modules, process relation, and automation. In order to make these designs easier to understand, the design has been illustrated in attached diagrams (Use Case Diagram, Data Flow Diagrams, and Sequence Diagrams).

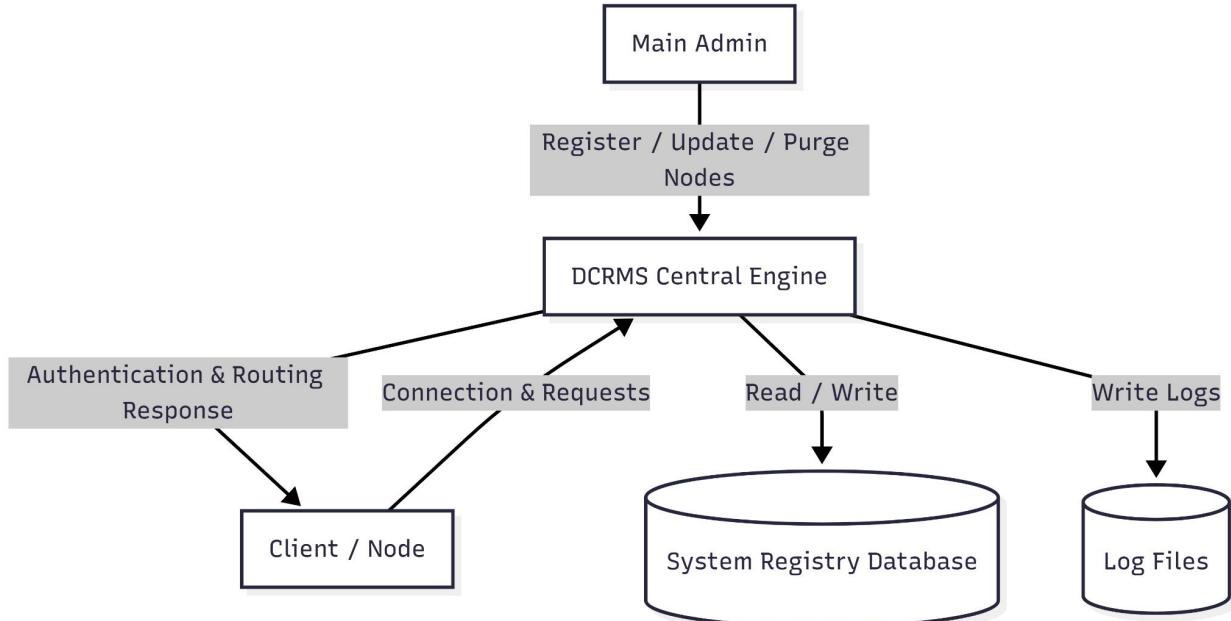
## 3.2 FLOW CHART



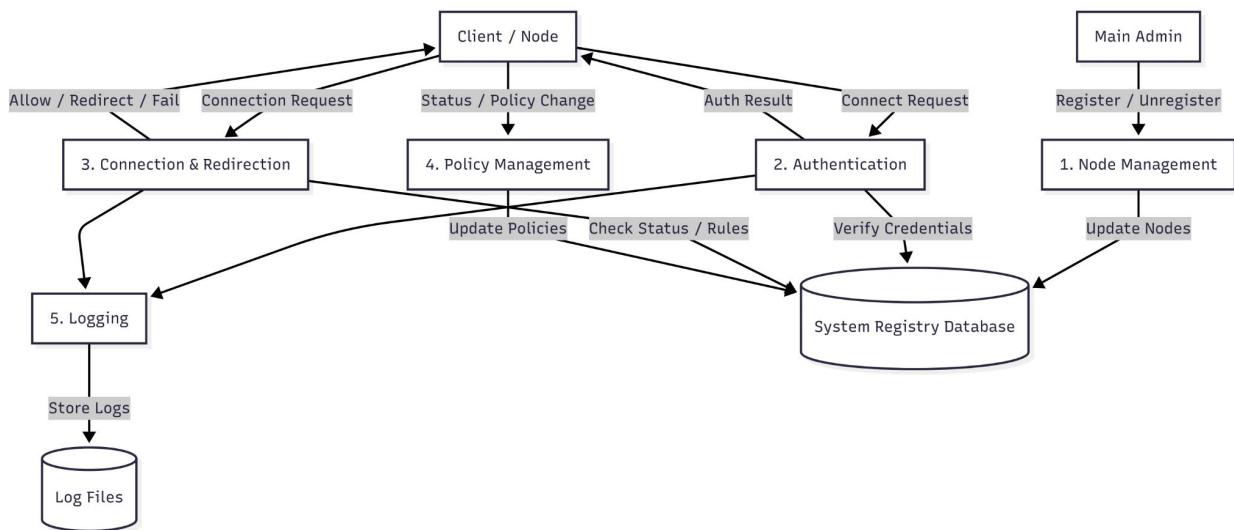
### 3.2.1 SYSTEM ARCHITECTURE



### 3.3 LEVEL 0 DESIGN DIAGRAM



### 3.4 LEVEL 1 DESIGN DIAGRAM



### **3.5 FILES**

The DCRMS system manages configuration files, policy files, and log files to ensure proper system operation.

- **Configuration Files:** Store server settings, node parameters, routing rules, and timeout values.
- **Node Policy Files:** Maintain node-level redirection policies that guide routing decisions when a node is unavailable or busy.
- **Log Files:** Generated for each node and server activity, containing timestamps, connection history, routing events, and diagnostic information.

Log files are subject to size constraints. When a log file exceeds the predefined limit, the system provides a mechanism to create a new log file to ensure uninterrupted logging.

### **3.6 USER INTERFACE**

- Linux-based machine with network connectivity.
- Command Line Interface (CLI) for server, admin, and node operations.
- Local Admin Console available on each node to dynamically update node status (Available, Busy, Unavailable) and modify redirection policies at runtime.

The interface is designed to be lightweight and efficient for distributed environments.

### **3.8 HELP**

Help documentation will include setup instructions, configuration guidelines, and operational procedures for administrators and nodes. The documentation explains how to register nodes, configure routing policies, monitor logs, and manage system behavior.

Future enhancements may include publishing the system as an open-source distributed routing framework with detailed user guides.

### **3.9 PERFORMANCE**

The system is designed to provide fast routing decisions with minimal latency.

Performance goals include:

- Efficient authentication and handshake processing
- Rapid node availability checks
- Intelligent redirection without noticeable delay
- Optimized logging mechanisms
- Support for concurrent client-node connections

The architecture ensures that routing decisions do not become a performance bottleneck

### **3.10 SECURITY**

Security is a critical aspect of DCRMS.

Key security measures include:

- Secure authentication handshake before node enrollment
- Validation of node credentials
- Controlled administrative access
- Restricted configuration updates
- Optional encrypted communication between nodes

These measures prevent unauthorized nodes from participating in the network.

### **3.11 RELIABILITY**

DCRMS is designed to provide continuous and dependable service as long as the central server remains operational.

Reliability is achieved through:

- Policy-based redirection
- Failover mechanisms
- Node health monitoring
- Telemetry logging

- Administrative overrides

Even when nodes become unavailable, the system attempts to redirect connections to maintain service continuity.

### **3.12 MAINTAINABILITY**

The system follows a modular architecture, making it easy to maintain and extend.

Maintenance activities may include:

- Updating routing algorithms
- Modifying redirection policies
- Adjusting configuration settings
- Enhancing security mechanisms

Minimal downtime is expected during updates due to centralized configuration management.

### **3.13 PORTABILITY**

DCRMS is portable across Linux-based environments and can be deployed on various distributions with minimal setup.

The system relies on standard networking libraries and C++ compilation support, allowing it to operate consistently across compatible platforms

### **3.14 REUSABILITY**

The system is built using reusable components such as:

- Configuration parser
- Logging module
- Security utilities
- Communication protocol

These modules can be integrated into other distributed systems requiring routing, authentication, or node management.

### **3.15 APPLICATION COMPATIBILITY**

DCRMS operates as a distributed client-server application and is compatible with systems that support standard network communication protocols.

It can be integrated with applications requiring:

- centralized routing
- node orchestration
- session management
- distributed communication

The architecture allows future expansion into larger distributed infrastructures.

### **3.16 RESOURCE UTILIZATION**

The system is designed for efficient utilization of computational and network resources.

- Memory usage is optimized through controlled session handling.
- CPU overhead is minimized using lightweight routing logic.
- Log rotation prevents excessive disk consumption.
- Network bandwidth is efficiently used through structured packet communication.
- The system ensures stable performance even as the number of nodes increases.

# Low Level Design Document

## 4. INTRODUCTION

The Distributed Central Routing Management System (DCRMS) is a distributed client-server application developed in C++ that manages communication between multiple endpoint nodes through a centralized routing engine. The system authenticates nodes before allowing them to participate in the network and continuously monitors their availability to ensure reliable session routing.

DCRMS implements intelligent routing by evaluating node status and applying node-level redirection policies whenever a target node is busy or unavailable. The system also supports administrative overrides, telemetry logging, and failover mechanisms to maintain high availability.

This Low-Level Design (LLD) document provides a detailed technical description of the internal components, class responsibilities, communication protocols, data handling mechanisms, and operational constraints of the system.

### 4.1 PURPOSE

- To describe the low-level design and internal workflow of the Distributed Central Routing Management System.
- To translate the functional and non-functional requirements defined in the SRS into detailed technical components.
- To explain the internal modules such as the Central Routing Engine, Authentication Manager, State Controller, Redirection Engine, Configuration Manager, and Telemetry Logger.
- To describe the communication protocol used for packet exchange between the server and endpoint nodes.
- To define the authentication handshake process for secure node enrollment.

- To explain routing algorithms and node-level redirection policy evaluation.
- To provide guidance on configuration management, logging mechanisms, and error handling.
- To serve as a technical reference for developers during implementation.
- To assist testers in understanding internal workflows for unit and integration testing.
- To help evaluators verify that the system design satisfies the requirements defined in the SRS.

## **4.2 DOCUMENT CONVENTIONS**

The document uses the “Times New Roman ” font with bold for the heading. The main heading size is 18. The Sub-heading size is 14point and the sub heading under the Subheading is 12points. The content under the sub-headings is of 12 points size. All the spacing are normal/default spacing of MS Word.

## **4.3 INTENDED AUDIENCE AND READING SUGGESTIONS**

This document is intended for developers, testers, system architects, project managers, and evaluators involved in the design and implementation of distributed systems.

It provides a technical overview of the system components and their interactions. Readers are encouraged to review the SRS before reading this document to better understand how the design supports the defined requirements.

The terminology used assumes basic familiarity with networking concepts, distributed architectures, and client-server communication models.

## **5.DETAILED SYSTEM DESIGN**

The DCRMS architecture is designed to ensure secure, reliable, and efficient communication across distributed nodes. The system follows a centralized management approach while allowing decentralized policy control at the node level.

The design focuses on:

- Secure node authentication
- Intelligent session routing

- Policy-based redirection
- Administrative configuration
- System monitoring and logging
- Failover handling

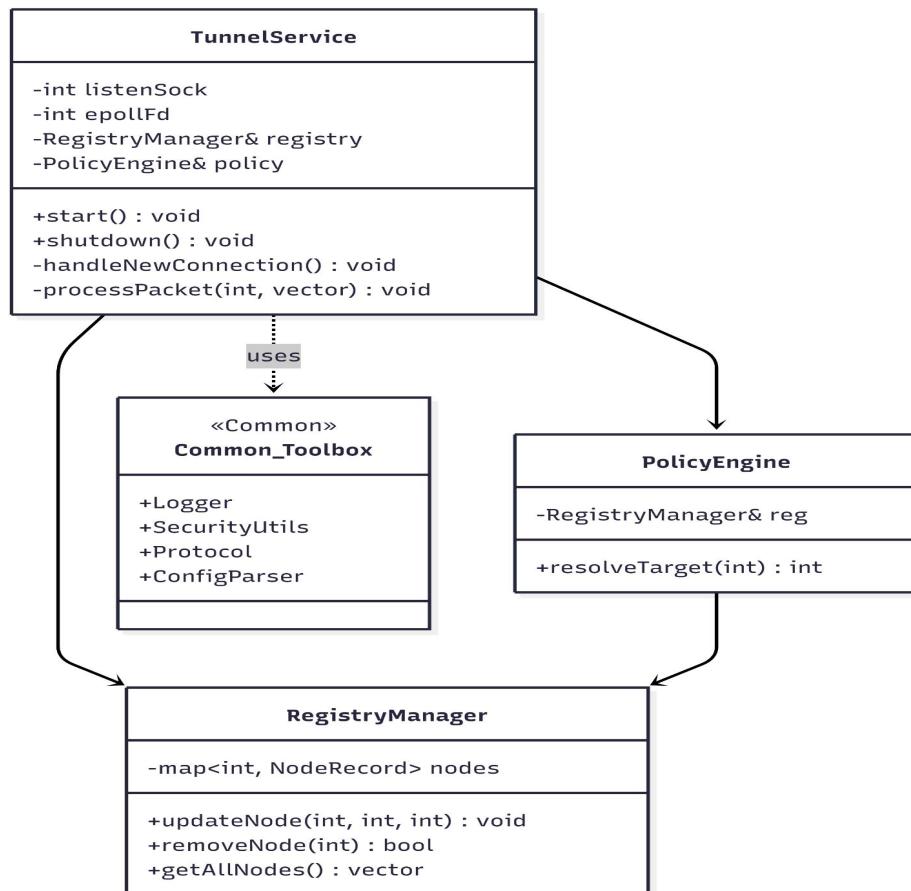
The structural and behavioral aspects of the system are illustrated using architectural diagrams, use case diagrams, sequence diagrams, flowcharts, and data flow diagrams.

## **5.1 DESIGN DESCRIPTIONS**

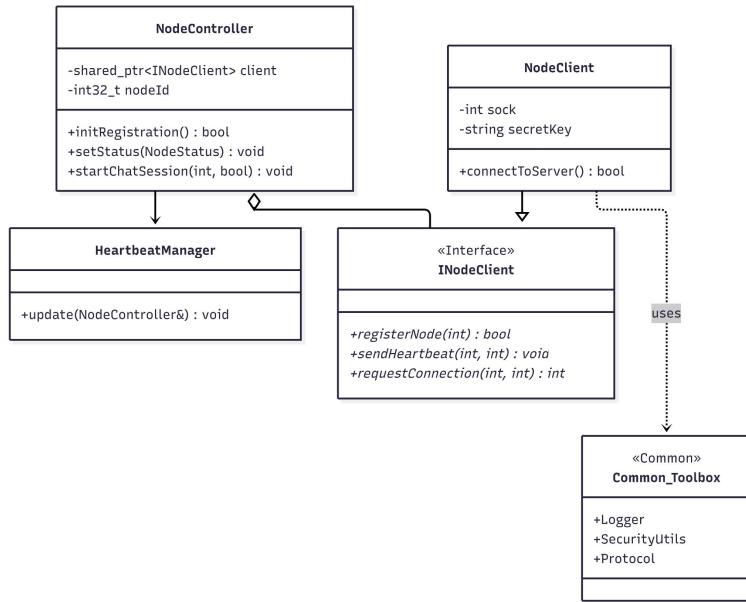
The main design features include four major parts: the architecture, the user interface design, the files, process relation, and automation. In order to make these designs easier to understand, the design has been illustrated in attached diagrams ( Use Case, Data flow diagrams).

## 5.2 CLASS DIAGRAM

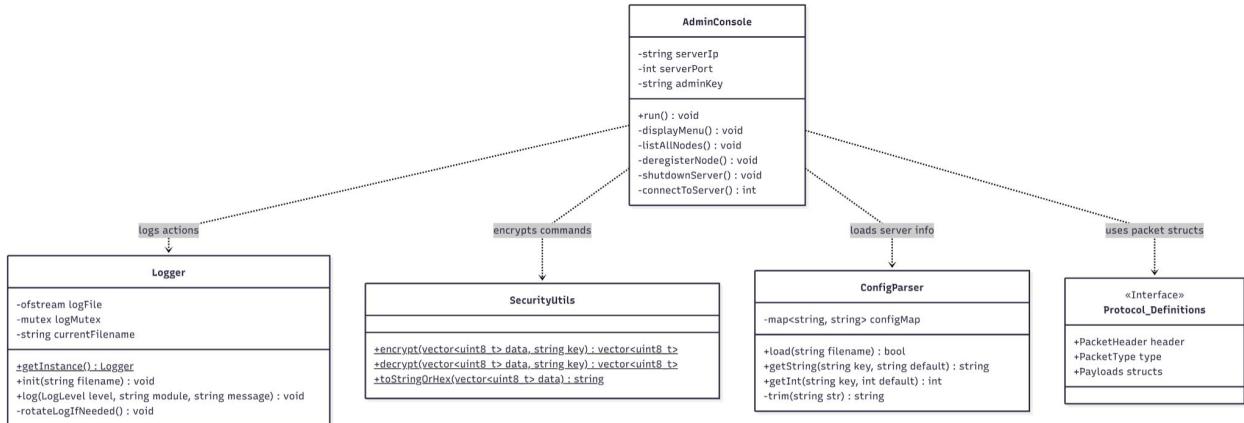
### 5.2.1 Server Class Diagram



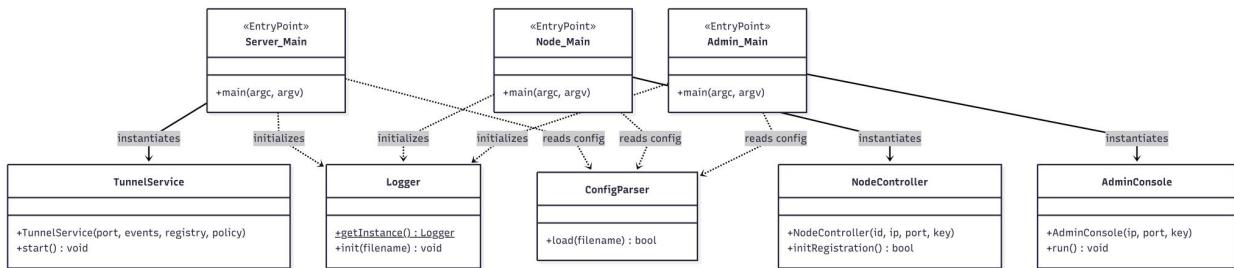
## 5.2.2 Node Class Diagram



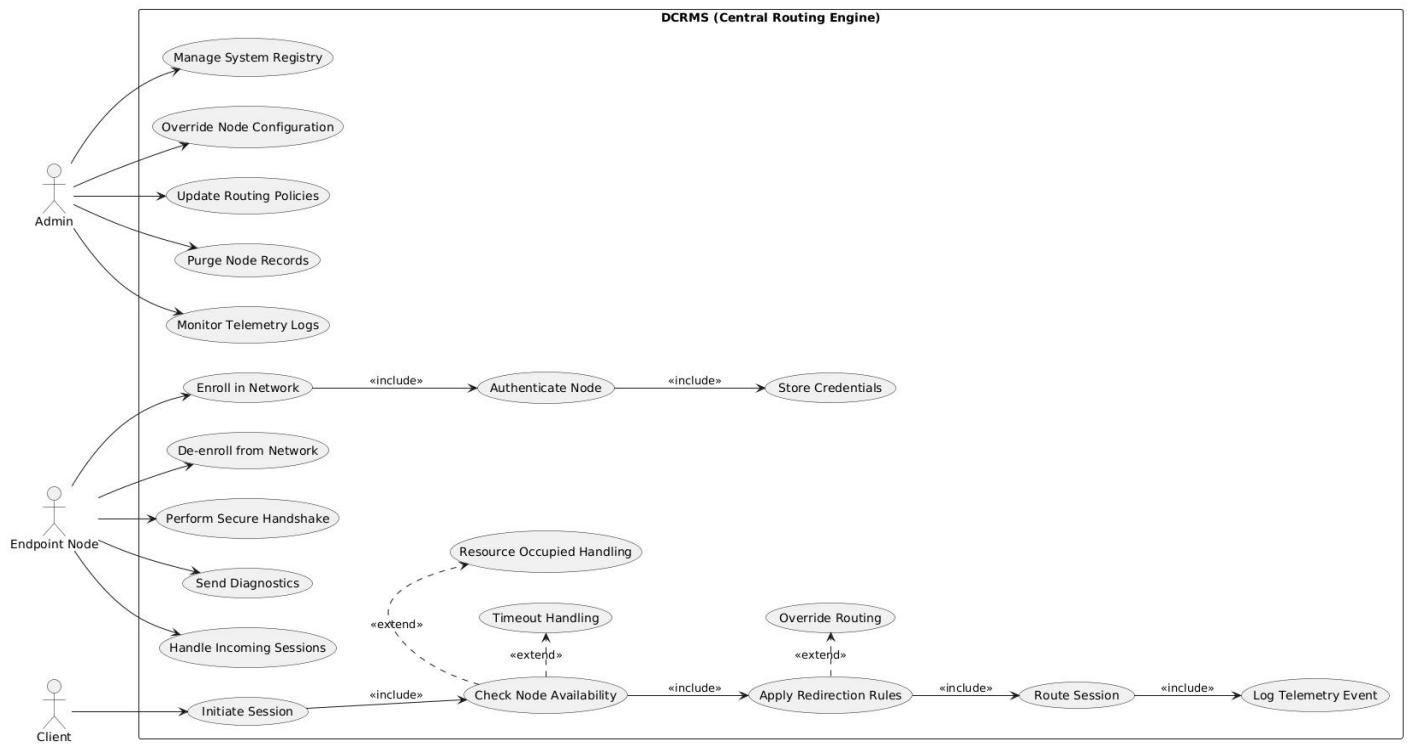
## 5.2.3 Admin Class Diagram



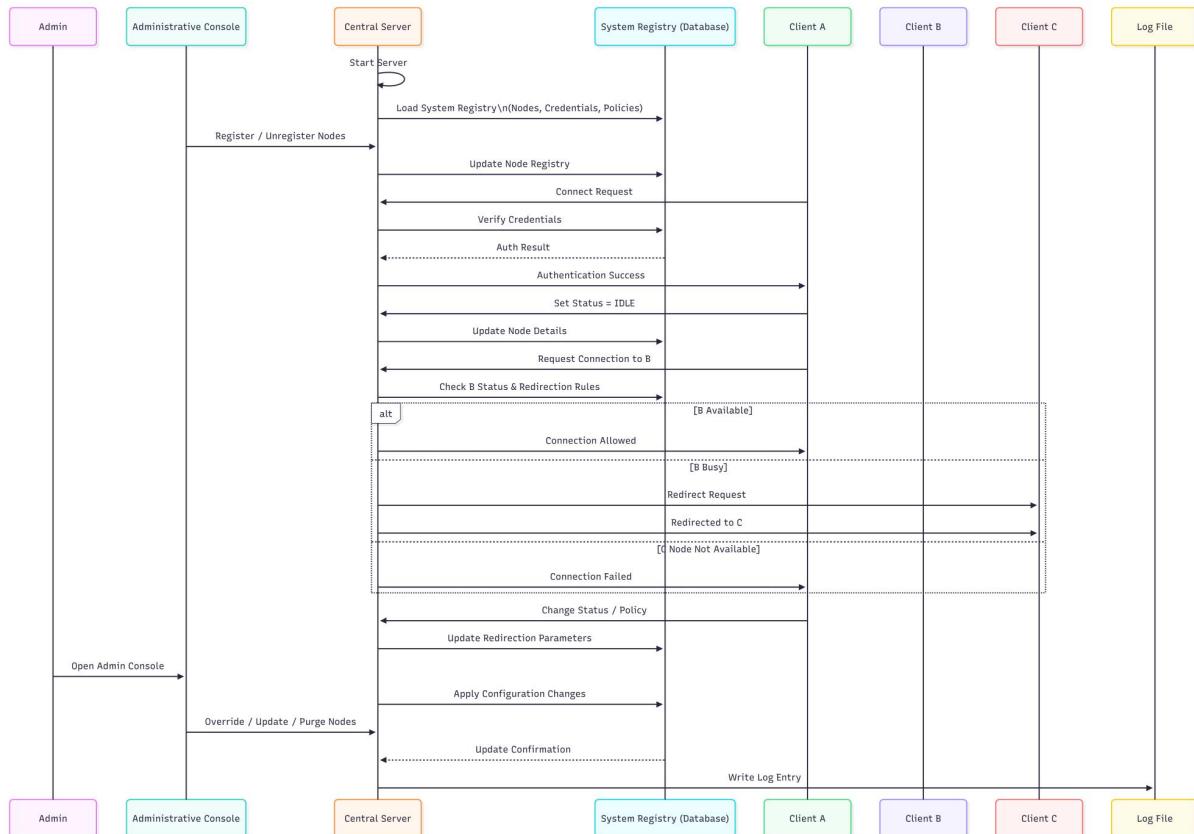
## 5.2.4 Entry Point Class Diagram



## 5.3 USE CASE DIAGRAM



## 5.5 SEQUENCE DIAGRAM



## 5.6 DESIGN AND IMPLEMENTATION CONSTRAINTS

The system must be implemented using C++ only

## 5.7 USER INTERFACE

- Desktop or a Linux machine with internet connection.
- Command Line Interface (CLI).

