# Project – High Level Design (HLD)

## on Telecom Support Triage Agent (AAI-39)

**Course Name:** Datagami Skill Based Course
**Institution Name:** Medi-Caps University


| Sr No | Student Name | Enrollment Number |
|---|---|---|
| 1 | Sanidhya Yadav | EN22CS301866 |
| 2 | Samiksha Saxena | EN22CS301853 |
| 3 | Sanand Alshi | EN22CS301859 |
| 4 | Sanidhya Somani | EN22CS301865 |
| 5 | Sarthak Chhajed | EN22CS301874 |


## 1. Introduction

The Telecom Support Triage Agent (AAI-39) is an AI-powered system designed to automate the initial customer support triaging process in telecommunication companies. The system leverages Large Language Models (LLMs) and Natural Language Processing (NLP) techniques to classify customer messages, extract critical information, generate draft responses, and route issues to appropriate support teams.

This project aims to improve response efficiency, reduce manual workload, and ensure faster issue resolution by implementing a production-style AI triage workflow.

---

### 1.1 Scope of the Document

This document provides a **High-Level Design (HLD)** for the Telecom Support Triage Agent. It outlines the system architecture, major components, data flow, APIs, and non-functional requirements. The document serves as a reference for understanding system behavior and design decisions.

---

### 1.2 Intended Audience

- Faculty evaluators
- Project reviewers
- Developers and system designers
- Students studying AI-based software systems

---

### 1.3 System Overview

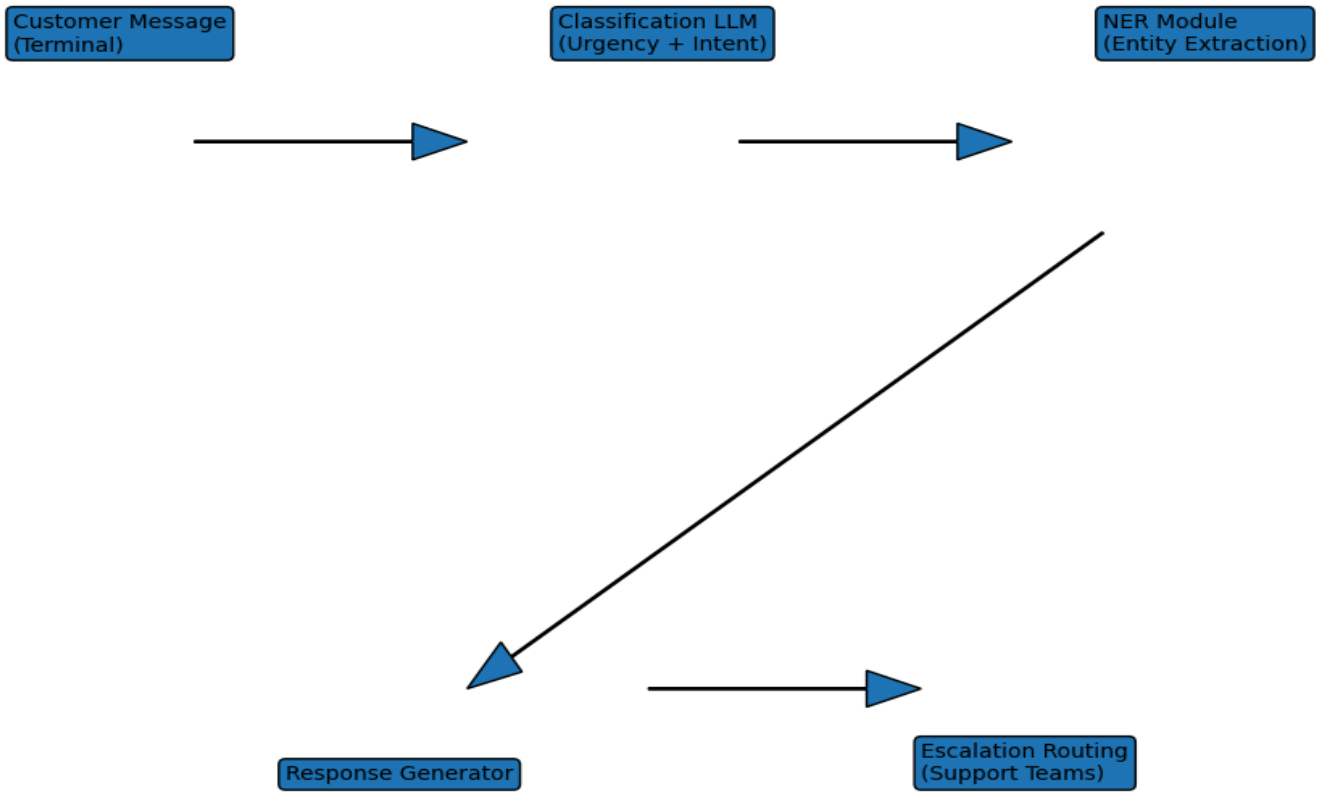
The system is a terminal-based AI application that accepts telecom customer messages as input. Using an LLM-powered workflow, it classifies message urgency and intent, extracts entities, generates professional responses, and routes the issue to the appropriate internal team.

---

## 2. System Design

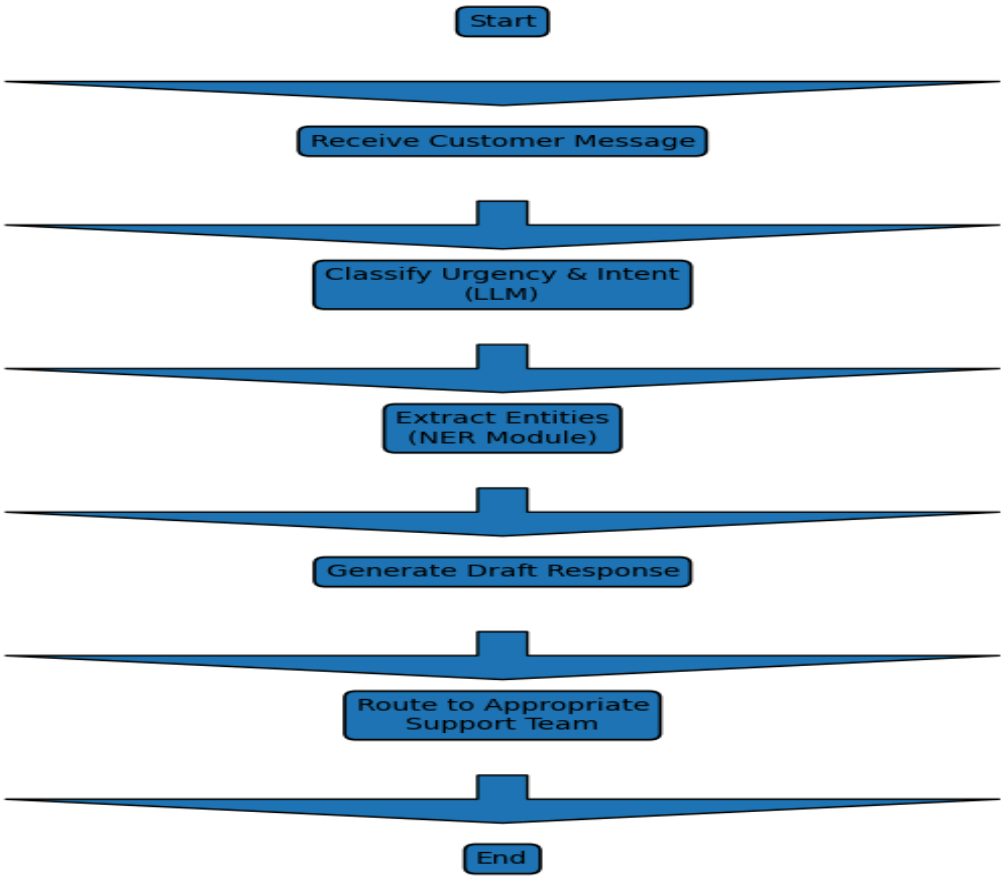
### 2.1 Application Design

The Telecom Support Triage Agent follows a **modular and layered architecture**, ensuring separation of concerns and scalability. Each functional responsibility is handled by independent modules such as classification, entity extraction, response generation, and routing.



## 2.2 Process Flow

1. Customer enters a support message via terminal
2. Message is passed to the classification module
3. Urgency and intent are determined using LLM
4. NER module extracts structured entities
5. Draft response is generated
6. Issue is routed to the appropriate support team
7. Output is displayed in structured format



---

## 2.3 Information Flow

- Input: Raw customer message (text)
- Processing: LLM classification + entity extraction
- Output: Structured data (urgency, intent, entities, routing, response)

---

## 2.4 Components Design

| Component | Description |
|---|---|
| Classification Agent | Identifies urgency and intent |
| NER Module | Extracts customer ID, phone number, ticket ID, date |
| Response Generator | Creates professional draft replies |
| Escalation Router | Routes issues to relevant teams |
| Parser Utility | Cleans and validates structured output |
| Terminal Interface | Handles user input and output |

---

## 2.5 Key Design Considerations

- Modular architecture for maintainability
- Secure handling of API keys
- Structured JSON output for reliability
- Production-like workflow simulation
- Terminal-based lightweight execution

### 2.6 API Catalogue

| API | Purpose |
|---|---|
| Groq LLM API | Text classification and response generation |

---

## 3. Data Design

### 3.1 Data Model

| Field | Description |
|---|---|
| customer_message | Raw input message |
| urgency | Low / Medium / High |
| intent | Type of telecom issue |
| entities | Extracted structured data |
| routed_team | Assigned support department |
| draft_response | Generated reply |

---

### 3.2 Data Access Mechanism

- In-memory data processing
- No persistent database used
- Data handled during runtime only

---

### 3.3 Data Retention Policies

- No customer data is stored

- All data exists temporarily during execution
- Ensures privacy and security compliance

---

### 3.4 Data Migration

Not applicable, as no persistent data storage is used.

---

## 4. Interfaces

- **User Interface:** Terminal-based CLI
- **External Interface:** Groq LLM API

---

## 5. State and Session Management

- Stateless system
- Each message processed independently
- No session memory maintained

---

## 6. Caching

- No caching implemented
- All responses generated dynamically in real-time

---

## 7. Non-Functional Requirements

### 7.1 Security Aspects

- API keys stored in environment variables
- `.env` file excluded via `.gitignore`
- Structured output validation prevents malformed responses

---

### 7.2 Performance Aspects

- Fast response time using optimized LLM calls
- Lightweight terminal-based execution
- Efficient prompt engineering to reduce latency

---

## 8. References


1. LangChain Documentation
2. Groq LLM API Documentation
3. Python Official Documentation
4. Natural Language Processing Concepts