Project - High Level Design

On

Real Estate Support Triage Agent

Course Name: Agentic AI

Institution Name: Medicaps University – Datagami Skill Based Course

Student Name(s) & Enrolment Number(s):

Sr no	Student Name	Enrolment Number
1.	Kushagra Gupta	EN22CS301546
2.	Khushi Paliwal	EN22CS301505
3.	Kritika Nagar	EN22CS301526
4.	Kartik Gupta	EN22CS301488
5.	Kishna Sharma	EN22CS301510
6.	Jayesh Jain	EN22CS301465

Group Name: 07D5

Project Number: AAI-07

Industry Mentor Name:

University Mentor Name: Maya Yadav Baniya

Academic Year: 2025-2026

Table Content

1.INTRODUCTION

The Introduction section of a High-Level Design (HLD) document provides a foundational understanding of the system before moving into architectural and technical specifics. It explains the background of the project, the problem it addresses, and the purpose of the proposed solution. In the case of the Real Estate Support Triage Agent, the introduction establishes the need for an intelligent AI-driven system capable of handling high volumes of customer messages efficiently and safely. Real estate businesses deal with inquiries, maintenance complaints, emergency reports, and document requests across multiple communication channels. Handling these manually leads to delays, missed emergencies, and operational inefficiencies.

This section ensures that readers clearly understand:

- The business problem being solved
- The motivation for building the system
- The overall objective of the AI-based triage layer
- The importance of automation combined with human oversight

The introduction acts as a conceptual entry point and prepares the reader for the architectural explanation that follows.

1.1. Scope of the document

The Scope of the Document defines the boundaries and coverage of this High-Level Design. It clarifies what aspects of the system are included in this documentation and what areas are intentionally excluded. Defining scope ensures clarity, avoids ambiguity, and prevents unrealistic expectations from stakeholders.

This document includes:

- Overall system architecture
- Major components and their responsibilities
- Data flow and process flow
- API catalogue and integration points
- Data model and retention strategies
- Security and performance considerations

However, it excludes:

- Low-level source code implementation
- Detailed prompt engineering logic
- Infrastructure-as-code scripts

- Deployment configuration files

Clearly defining scope maintains professionalism and ensures that the document remains structured and purpose-driven.

1.2. Intended Audience

The Intended Audience section identifies the stakeholders who will use or refer to this document. Since system design involves multiple technical and business roles, it is important to clarify who the documentation is written for.

This HLD is intended for:

- Software developers implementing backend services
- AI engineers designing model interactions
- System architects reviewing scalability
- DevOps engineers planning deployment
- Academic mentors and project evaluators

By specifying the audience, the document ensures that the technical depth and explanations are appropriate for readers with different expertise levels. This improves communication between business objectives and technical implementation.

.

1.3. System overview

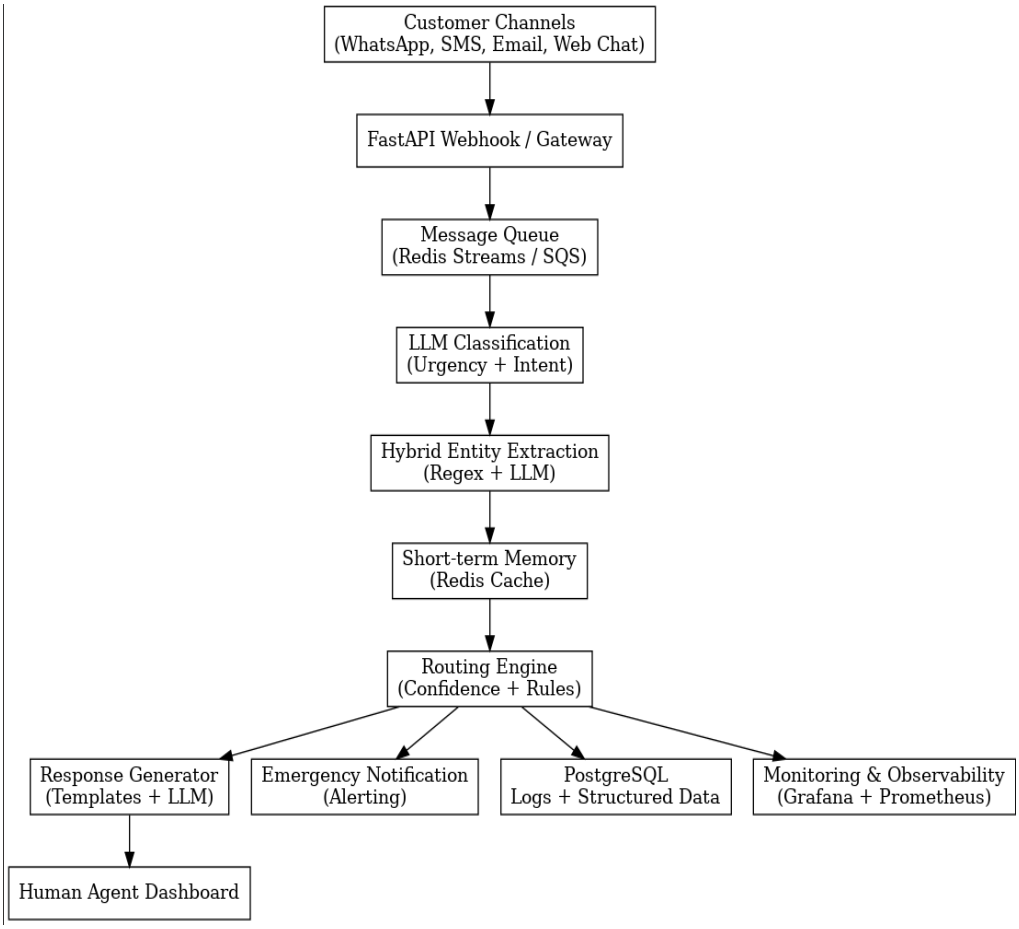
The System Overview provides a high-level description of how the system functions and how its major components interact. It gives a conceptual understanding of the overall workflow before diving into detailed architecture.

The Real Estate Support Triage Agent is an AI-powered middleware system that sits between customer communication channels and human support agents. It automatically processes incoming messages and determines the appropriate next action.

At a high level, the system performs the following tasks:

- Receives messages from multiple channels (WhatsApp, SMS, Email, Chat)
- Cleans and preprocesses the message text
- Classifies urgency and intent using AI models
- Extracts structured data such as property IDs or dates
- Applies routing logic based on confidence and risk
- Generates automatic replies or drafts
- Escalates emergencies immediately
- Logs all decisions for auditing and analytics

The overview ensures that readers understand the system's purpose, boundaries, and operational philosophy before reviewing technical design sections.



2. System Design

The System Design section explains how the system is architecturally structured to achieve its objectives. It describes how business requirements are translated into technical components and workflows. This section is central to the HLD because it demonstrates how the solution will be practically implemented.

System design focuses on:

- Application structure
- Component decomposition
- Communication mechanisms
- Data movement
- Decision-making logic

It ensures that the system is scalable, reliable, and maintainable over time.

2.1. Application Design

Application Design describes how the software is organized into logical modules and services. The system follows a modular, service-oriented architecture where each component has a clearly defined responsibility.

The main modules include:

- Message Ingestion Service
- Preprocessing Service
- AI Classification Service
- Entity Extraction Module
- Routing Engine
- Response Generation Service
- Escalation Manager
- Logging and Analytics Service

This modular approach provides:

- Independent scalability of services
- Fault isolation in case of errors
- Easier testing and debugging
- Future extensibility

By separating concerns across modules, the system becomes easier to maintain and enhance without affecting other components.

2.2. Process Flow

Process Flow describes the sequential steps that occur when a customer message enters the system. It ensures clarity in execution order and system behavior.

The process flow follows these stages:

- Message reception through webhook
- Text preprocessing and normalization
- Parallel AI tasks (classification + extraction)
- Session context retrieval

- Routing decision application
- Response generation or escalation
- Logging of all outputs

This structured workflow ensures:

- Consistency in processing
- Reduced latency through parallel execution
- Transparent decision-making
- Reliable escalation handling

Understanding process flow is crucial for validating system correctness and operational efficiency.

2.3. Information Flow

Information Flow describes how data transforms as it moves across components. It explains how unstructured text becomes structured decision data.

The transformation stages include:

- Raw message input
- Cleaned and normalized text
- AI-generated classification outputs
- Extracted structured entities
- Confidence scores
- Routing decision result
- Generated response output

This flow ensures that every piece of information is traceable and auditable. Proper information flow design improves data integrity and system transparency

2.4. Components Design

Component Design explains the responsibilities and boundaries of each subsystem. Each component is designed based on the Single Responsibility Principle, meaning it performs one core function.

Key components include:

- Classification component → Determines urgency and intent
- Extraction component → Identifies structured entities
- Routing engine → Decides next action
- Escalation manager → Handles emergency alerts

- Logging module → Stores audit records

This separation improves:

- Maintainability
- Independent scaling
- Clear debugging boundaries
- Modular upgrades

Component design ensures that the system remains flexible and extensible over time.

2.5. Key Design Considerations

Key Design Considerations describe the important architectural decisions that shape system behavior and reliability.

Important considerations include:

- Safety-first emergency detection
- Human-in-the-loop validation
- Latency optimization (<5 seconds target)
- Cost-effective AI usage
- Full audit traceability
- Horizontal scalability

These considerations ensure that the system is not only functional but also reliable, secure, and production-ready.

.

2.6. API Catalogue

The API Catalogue documents all endpoints exposed by the system. APIs enable communication between internal modules and external platforms.

Major APIs include:

- POST /webhook/message → Receives incoming message
- GET /message/{id} → Fetches triage result

- POST /escalate → Triggers escalation
- GET /analytics → Retrieves performance metrics

Well-defined APIs ensure:

- Integration consistency
- Standardized communication
- Clear input-output contracts

.

3. Data Design

Data Design refers to the structured planning of how data is:

- Modeled
- Stored
- Accessed
- Secured
- Maintained
- Archived

It ensures that information flows efficiently across the system while maintaining consistency, integrity, and scalability.

In the Real Estate Support Triage Agent, data design is extremely important because:

- Every message must be traceable.
- Every AI decision must be auditable.
- Emergency cases must be historically verifiable.
- Analytics must be possible for model improvement.

The system converts **unstructured text data into structured relational data**, which requires a carefully designed schema.

3.1 Data Model

The Data Model describes the database structure and entity relationships. It includes entities such as Customer, Message, Intent, Routing Decision, Response, Agent, Escalation, and Session.

Key relationships:

- One Customer → Many Messages
- One Message → One Routing Decision
- One Message → Multiple Extracted Entities

A well-designed data model ensures referential integrity, consistency, and structured analytics..

3.2 Data Access Mechanism

Data Access Mechanism explains how the application interacts with the database.

It includes:

- ORM-based access (e.g., SQLAlchemy)
- Indexed queries for performance
- Transaction management
- Secure connection handling

This ensures optimized performance and secure data operations.

3.3 Data Retention Policies

Data Retention Policies define how long different types of data are stored.

For example:

- Session data → 24–48 hours
- Message logs → 1–2 years
- Escalation records → 3+ years

This balances compliance requirements, storage cost, and analytics needs.

3.4 Data Migration

Data Migration refers to the process of transferring data from one system, storage, or format to another.

Data Migration in the context of the Real Estate Support Triage Agent refers to the structured process of modifying, upgrading, and evolving the database schema and stored data as the system grows and improves over time. Since this project is AI-driven and continuously evolving, data migration is not limited to simple database changes; it also involves changes in AI model outputs, classification categories, routing rules, and extracted entity structures.

As the system matures, several types of changes may occur. For example:

- New intent categories may be added (e.g., Legal Query, Refund Request).
- Urgency levels may be expanded or redefined.
- Additional entity types such as Lease ID, Broker ID, or Complaint Type may be introduced.
- Routing decision logic may require new database fields.
- AI model versions may change and require tracking.

To support such changes safely, the system adopts a structured migration strategy that includes:

- Version-controlled migration scripts (e.g., Alembic)
- Full database backup before schema modification
- Backward compatibility maintenance
- Controlled staging environment testing
- Rollback mechanisms in case of failure

There are three primary types of migration in this project:

1. Schema Migration – Adding new columns, modifying data types, or introducing new tables such as Feedback or ModelPerformance.
2. Data Transformation Migration – Updating old records to match new classification structures (e.g., converting binary urgency levels into four-level urgency categories.
3. Model Version Migration – Storing model_version fields with each message to maintain explainability and audit traceability.

The system also follows a zero-downtime migration approach, ensuring that live message processing continues during upgrades. This is achieved through rolling updates, non-destructive schema changes, and feature flag control. Proper migration planning ensures system stability, compliance safety, and long-term extensibility.

4. Interfaces

The Interfaces section describes how the Real Estate Support Triage Agent communicates with external systems and internal services. Interfaces define structured communication boundaries and ensure seamless integration.

This system interacts with multiple external platforms, including:

- WhatsApp Business API
- SMS Gateway Providers
- Email Servers
- CRM Systems (future integration)
- Monitoring and Alerting tools

Each external interface follows REST-based communication over HTTPS with authentication tokens to ensure secure data exchange.

Internally, services communicate using:

- REST APIs between modules
- Message queues for asynchronous processing
- Redis cache for session state

Clear interface design ensures:

- Loose coupling between services
- Secure communication
- Scalable integration
- Reduced system dependency

Well-defined APIs act as contracts, preventing integration errors and ensuring predictable behavior.

5. State and Session Management

State and Session Management refers to how the system maintains conversational continuity across multiple customer messages. Since REST APIs are stateless by design, the system uses external session storage to maintain context.

In this project, session data is stored in Redis with a defined Time-To-Live (TTL). This session includes:

- Last 3–5 customer messages
- Previous detected intent
- Temporary context flags

For example, if a customer first asks about a 2BHK property and later says "What about the price?", the system must refer to the earlier context to understand which property is being discussed. Without session management, responses would be inconsistent or confusing.

The benefits of this approach include:

- Scalable stateless API architecture
- Fast context retrieval
- Automatic expiration of stale sessions
- Reduced database load

Session management ensures conversational coherence while maintaining distributed system scalability.

6. Caching

Caching is implemented to improve system performance and reduce processing latency. Since the Real Estate Support Triage Agent processes messages in near real-time, minimizing response time is critical.

Redis is used as the primary caching layer for:

- Session memory
- Frequently accessed configuration settings
- Rate-limiting counters
- Temporary routing states

The caching strategy follows the principle of temporal locality, meaning recently accessed data is likely to be accessed again soon. By storing such data in memory, the system achieves:

- Reduced database queries
- Lower API response time
- Improved throughput
- Better user experience

Caching also reduces infrastructure costs by limiting repeated AI calls and unnecessary database access.

7. Non-Functional Requirements

Non-Functional Requirements define the quality attributes of the system. While functional requirements describe what the system does, non-functional requirements describe how well it performs.

In this project, non-functional requirements include:

- Security
- Performance
- Scalability
- Reliability
- Observability
- Compliance

These factors determine whether the system is production-ready and enterprise-grade.

7.1 Security Aspects

Security is a critical component because the system handles customer data, property details, and emergency information.

Security measures include:

- HTTPS encryption for data in transit
- Encryption at rest for stored data
- Role-Based Access Control (RBAC) for agents
- API authentication tokens
- Secure storage of AI API keys
- Audit logging of all routing decisions

The system follows a defense-in-depth model, where multiple layers of security protect against unauthorized access, data leakage, and misuse. Emergency escalation data is also logged with traceability to ensure legal and operational accountability.

7.2 Performance Aspects

Performance ensures the system responds quickly and reliably even during high traffic.

The performance objectives include:

- Response latency below 5 seconds
- High throughput message handling
- Horizontal scalability
- Minimal downtime
- Efficient resource usage

Performance is achieved through:

- Parallel AI processing
- Queue-based architecture
- Redis caching
- Indexed database queries
- Load balancing

Monitoring tools track metrics such as:

- Average response time

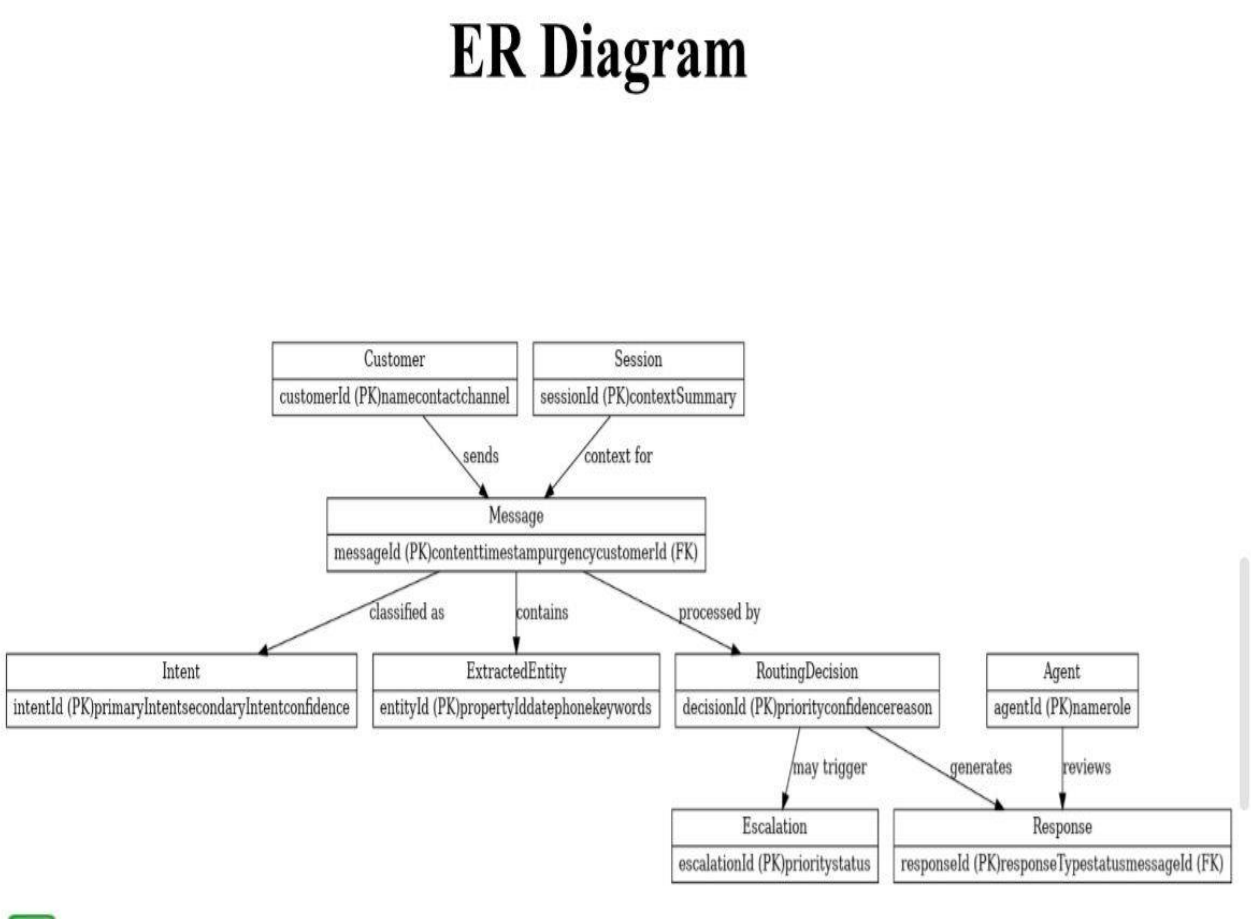
- Escalation rate
- Confidence distribution
- System uptime

This ensures continuous optimization and performance transparency.

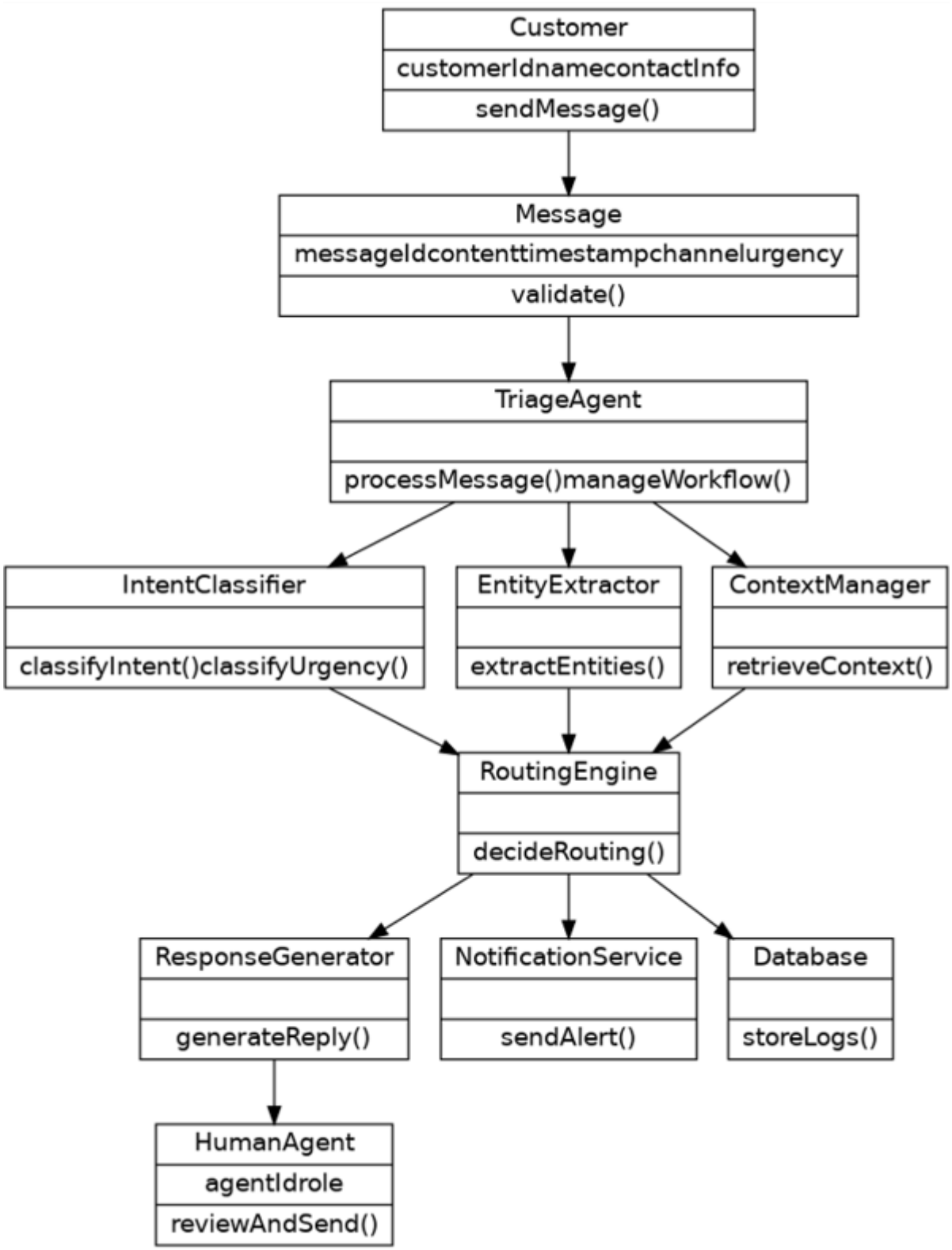
.

8. Diagram & Flowcharts :

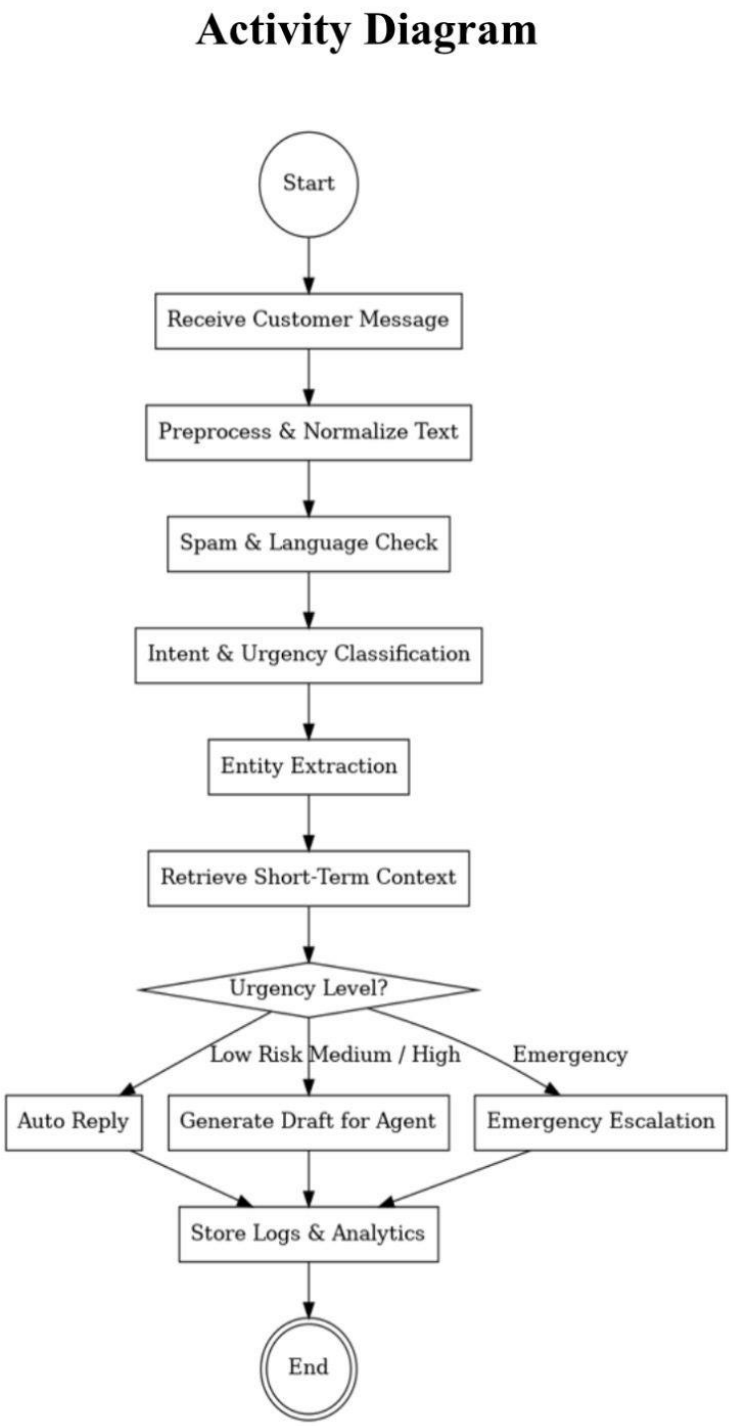
8.1 ER Diagram



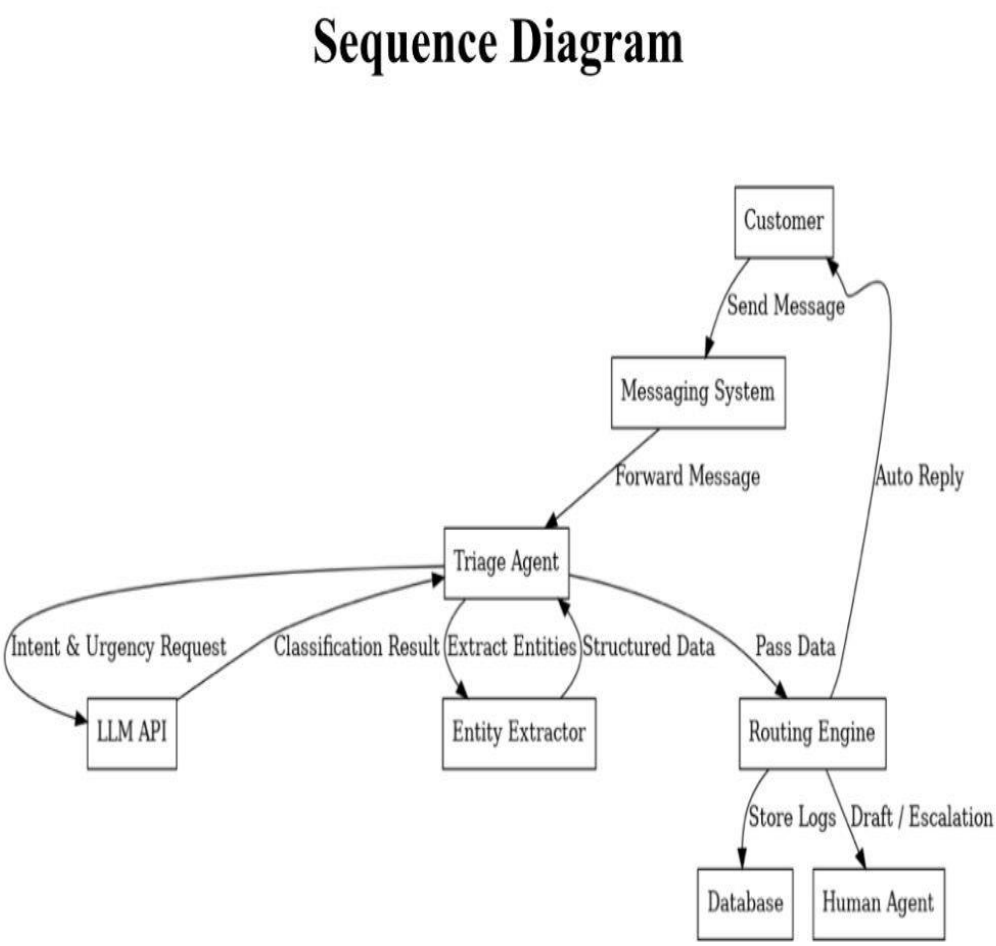
8.2 Class Diagram



8.3 Activity Diagram

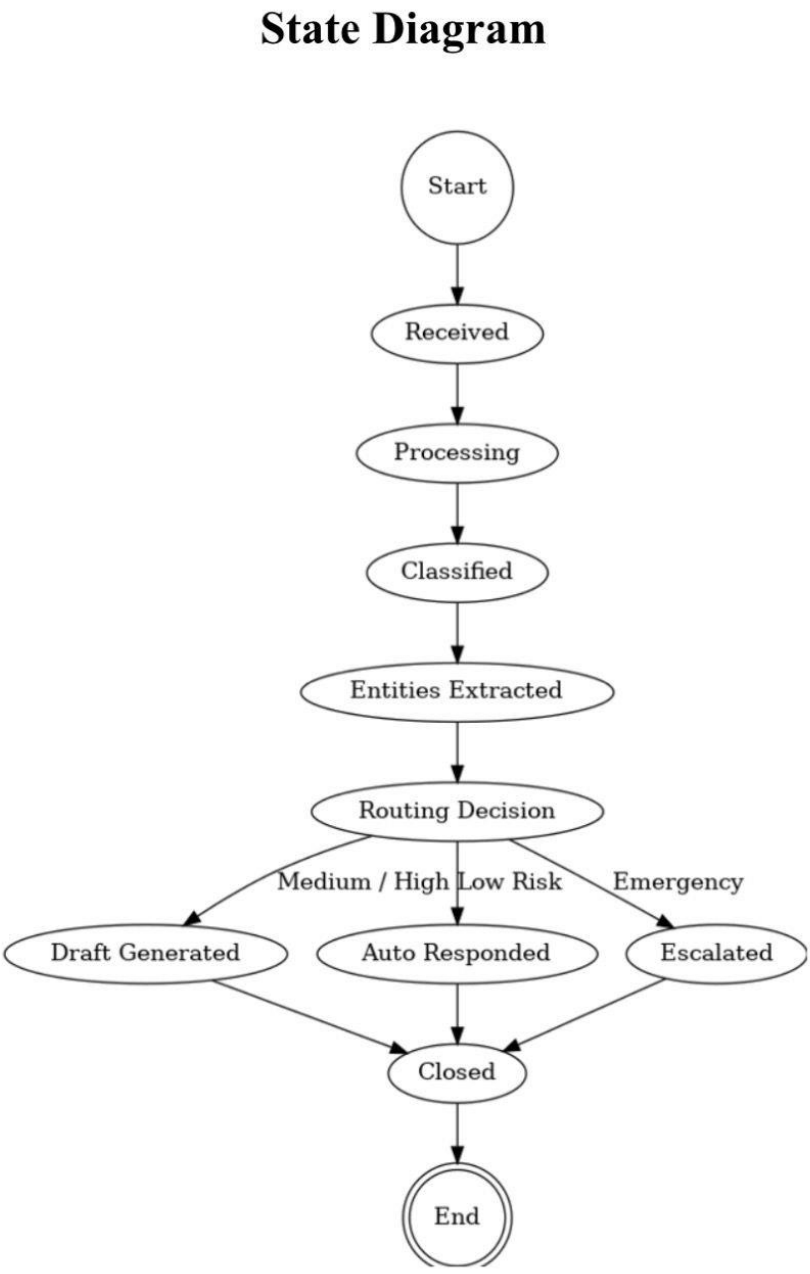


8.4 Sequence Diagram



8.5 Component Diagram

8.6 State Diagram



9.References

The references section lists the technical resources and documentation used for understanding and designing the system.

Fowler, M. (2014). *Microservices: A Definition of This New Architectural Term*. Available at: https://martinfowler.com

1. Newman, S. (2015). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media.
2. Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media.
3. Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral Dissertation, University of California, Irvine.
4. Elmasri, R., & Navathe, S. (2016). *Fundamentals of Database Systems*. Pearson Education.
5. PostgreSQL Global Development Group. *PostgreSQL Official Documentation*. Available at: https://www.postgresql.org/docs/
6. Redis Labs. *Redis Documentation*. Available at: https://redis.io/documentation
7. Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing*. Stanford University.
8. Brown, T. et al. (2020). "Language Models are Few-Shot Learners." *Advances in Neural Information Processing Systems (NeurIPS)*.
9. OpenAI. *OpenAI API Documentation*. Available at: https://platform.openai.com/docs
10. OWASP Foundation. *OWASP Top 10 Web Application Security Risks*. Available at: https://owasp.org
11. Hohpe, G., & Woolf, B. (2003). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley.