Customer Ticket Classification System Documentation

This document provides comprehensive documentation for the AI-powered Customer Ticket Classification System. It covers the project's overview, features, installation, API endpoints, project structure, configuration, and testing.

Project Overview

An AI-powered customer support ticket classification system using LangChain, OpenAI GPT-4o Mini, and MongoDB. The system automatically categorizes tickets by severity and generates appropriate responses based on priority levels.

Features

- **Intelligent Classification**: Uses OpenAI GPT models via LangChain to classify tickets by category and priority
- **Severity-Based Responses**: Generates contextually appropriate responses based on ticket severity
- **Memory Integration**: Maintains conversation context using LangChain memory for improved accuracy
- **Confidence Scoring**: Provides confidence scores and flags low-confidence classifications for manual review
- **REST API**: Complete API with validation, error handling, and rate limiting
- **Batch Processing**: Supports processing multiple tickets efficiently
- **Analytics**: Real-time statistics and trends
- **Admin Controls**: Configuration management and system monitoring

Priority Levels & Response Styles

Priority	Use Cases	Response Style	SLA
Critical	System outages, security breaches, data loss	Urgent, apologetic, action-oriented	Immediate
High	Major functionality broken, VIP customer issues	Professional, empathetic, solution-focused	2 hours
Medium	Minor bugs, standard customer requests	Helpful, friendly, informative	24 hours
Low	General questions, feature requests	Courteous, educational, patient	48 hours

Quick Start

Prerequisites

- Node.js 18+
- OpenAI API key
- MongoDB (optional - uses in-memory storage for demo)

Installation & Setup

1. **Extract the project files**
2. **Install dependencies:**
3. **Configure environment:**
4. **Start the server:**
5. **Test the API:**
   - Import `postman-collection.json` into Postman

- Or use curl commands below

API Endpoints

Core Ticket Operations

Shell

```
# Submit a new ticket
POST /api/tickets
Content-Type: application/json

{
  "subject": "System is down",
  "description": "Complete system outage affecting all users",
  "customerInfo": {
    "email": "customer@example.com",
    "name": "John Doe",
    "priority": "VIP"
  },
  "source": "email"
}

# Get specific ticket
GET /api/tickets/{ticketId}

# List tickets with filters
GET /api/tickets?priority=Critical&category=Technical&limit=10

# Submit feedback
PUT /api/tickets/{ticketId}/feedback
Content-Type: application/json

{
  "isCorrect": false,
  "correctedCategory": "Billing",
  "correctedPriority": "High",
  "agentId": "agent-001"
}

# Batch process tickets
POST /api/tickets/batch
Content-Type: application/json

[
```

```
  {"subject": "Login issue", "description": "Cannot log in..."},
  {"subject": "Billing question", "description": "Charged twice..."}
]
```

Analytics

Shell

```
# Get classification statistics
GET /api/analytics/stats

# Get trends over time
GET /api/analytics/trends?period=day
```

Admin (requires x-api-key: demo-api-key-12345)

Shell

```
# Get system configuration
GET /api/admin/config
x-api-key: demo-api-key-12345

# Update confidence threshold
PUT /api/admin/config/confidence-threshold
x-api-key: demo-api-key-12345
Content-Type: application/json

{"threshold": 80}

# Clear AI memory
POST /api/admin/memory/clear
x-api-key: demo-api-key-12345

# System health check
GET /api/admin/health
x-api-key: demo-api-key-12345
```

Testing Examples

Test Different Severity Levels

Critical Priority:

Shell

```
curl -X POST http://localhost:3000/api/tickets \
  -H "Content-Type: application/json" \
  -d '{
    "subject": "URGENT: Complete system outage - all users affected",
    "description": "Our entire platform is down. All customers are getting
500 errors when trying to log in. This started 30 minutes ago and is
affecting all users.",
    "customerInfo": {"priority": "VIP"},
    "source": "email"
  }'
```

Low Priority:

Shell

```
curl -X POST http://localhost:3000/api/tickets \
  -H "Content-Type: application/json" \
  -d '{
    "subject": "How to export my data?",
    "description": "Hi, I would like to know how I can export all my data
from your platform. I need it for my records.",
    "source": "web"
  }'
```

Expected Response Format

JSON

```
{
  "success": true,
  "data": {
    "ticketId": "TICKET-1704123456789-abc123def",
    "classification": {
      "category": "Technical",
      "priority": "Critical",
      "confidence": 95,
      "summary": "Complete system outage affecting all users",
      "suggestedResponse": "We are immediately escalating this critical
system outage to our senior engineering team. We understand the severity of
this issue and are treating it as our highest priority. Our team is actively
```

```
working to restore service and we will provide updates every 15 minutes until
resolved.",
      "tags": ["outage", "system", "critical", "urgent"],
      "reasoning": "This is a critical system failure requiring immediate
attention due to complete service disruption affecting all users."
    },
    "processingTime": 1250,
    "needsManualReview": false
  }
}
```


Project Structure

Plain Text

```
src/
├── classification/
│   ├── chains.ts          # LangChain orchestration
│   ├── engine.ts          # Main classification engine
│   ├── memory.ts          # Context memory management
│   └── prompts.ts         # AI prompt templates
├── middleware/
│   ├── auth.ts            # Authentication middleware
│   ├── error.ts           # Error handling
│   └── validation.ts      # Request validation
├── routes/
│   ├── admin.ts           # Admin endpoints
│   ├── analytics.ts       # Analytics endpoints
│   └── tickets.ts         # Core ticket endpoints
├── types/
│   └── ticket.ts          # TypeScript interfaces
├── app.ts                 # Express app setup
├── demo.ts                # Demo script
└── server.ts              # Server startup
```

Configuration

Environment Variables

Shell

```
OPENAI_API_KEY=your_openai_api_key_here    # Required
MONGODB_URI=mongodb://localhost:27017/...  # Optional (uses in-memory for
demo)
NODE_ENV=development                       # Optional
PORT=3000                                  # Optional
CONFIDENCE_THRESHOLD=70                    # Optional
API_KEY=your_admin_api_key_here           # Optional (for admin endpoints)
```

Classification Categories

- **Technical**: Software bugs, system errors, integration issues
- **Billing**: Payment problems, subscription issues, refunds
- **General Inquiry**: Questions, information requests, how-to
- **Bug Report**: Confirmed software defects requiring fixes
- **Feature Request**: New functionality suggestions
- **Account**: Login issues, profile changes, permissions

Available Scripts

Shell

```
npm run dev     # Start development server with hot reload
npm run demo    # Run classification demo with sample tickets
npm run build   # Build TypeScript to JavaScript
npm run start   # Start production server
npm run test    # Run unit tests
```

Performance Metrics

- **Classification Time**: 800-2000ms per ticket
- **Confidence Score**: 70-95% for clear tickets
- **Throughput**: 30-60 tickets/minute (with rate limiting)

- **Memory Usage**: Stable with periodic cleanup