



AutoCRM: AI-powered Customer Relationship Management

Background

Customer Relationship Management (CRM) applications, like Zendesk, are central to many businesses. They help support and sales teams manage diverse customer interactions while integrating with other essential tools.

CRMs often direct users to FAQs and help articles before allowing them to submit a ticket. However, many issues still require manual support, making CRMs one of the biggest sources of human labor.

AutoCRM leverages generative AI to minimize this workload and enhance the customer experience. By integrating existing help resources with the capabilities of LLMs, AutoCRM delivers an interactive support and sales experience with minimal human involvement. While some tickets may still require manual handling, the threshold is significantly raised, improving operational efficiency and boosting profitability.

Submission Guidelines

At the end of each week, you'll need to submit the following to the GauntletAI LMS:

1. A link to your project's GitHub repository.
2. The Brainlifts you used to learn about and enhance the application.
3. A 5-minute walkthrough showcasing what you've built (and, where relevant, how you built it).
4. A link to your project post on X, along with evidence of engagement with any feedback received.
5. A link to the working deployed application.

Baseline App (Week 1)

Building a Modern Customer Support System

Creating a modern customer support system requires a balanced focus on technical architecture, user experience, and customer-facing features. This document outlines the core functionalities required for a robust, scalable, and adaptable system. **Your goal is to rebuild as many of the following components as possible.**

Core Architecture

Ticket Data Model

The ticket system is central to AutoCRM, treated as a living document that captures the entire customer interaction journey. Key components include:

- **Standard Identifiers & Timestamps:** Basic fields like ticket ID, creation date, and status updates.
- **Flexible Metadata:**
 - **Dynamic Status Tracking:** Reflects team workflows.
 - **Priority Levels:** Manage response times effectively.

- **Custom Fields:** Tailor tickets to specific business needs.
- **Tags:** Enable categorization and automation.
- **Internal Notes:** Facilitate team collaboration.
- **Full Conversation History:** Includes interactions between customers and team members.

API-First Design

An API-first approach ensures accessibility and scalability, enabling:

- **Integration:** Connect seamlessly with websites, applications, and external tools.
- **Automation:** Simplify routine tasks and workflows.
- **AI Enhancements:** Lay the groundwork for future features.
- **Analytics:** Support robust reporting and insights.

API Features:

- **Synchronous Endpoints:** Handle immediate operations.
- **Webhooks:** Support event-driven architectures.
- **Granular Permissions:** Ensure secure integrations using API key authentication.

Employee Interface

Queue Management

- **Customizable Views:** Prioritize tickets effectively.
- **Real-Time Updates:** Reflect changes instantly.
- **Quick Filters:** Focus on ticket states and priorities.
- **Bulk Operations:** Streamline repetitive tasks.

Ticket Handling

- **Customer History:** Display detailed interaction logs.
- **Rich Text Editing:** Craft polished responses.
- **Quick Responses:** Use macros and templates.
- **Collaboration Tools:** Share internal notes and updates.

Performance Tools

- **Metrics Tracking:** Monitor response times and resolution rates.
- **Template Management:** Optimize frequently used responses.
- **Personal Stats:** Help agents improve efficiency.

Administrative Control

Team Management

- Create and manage teams with specific focus areas.
- Assign agents based on skills.
- Set coverage schedules and monitor team performance.

Routing Intelligence

- **Rule-Based Assignment:** Match tickets using properties.
- **Skills-Based Routing:** Assign issues based on expertise.
- **Load Balancing:** Optimize workload distribution across teams and time zones.

Data Management

Schema Flexibility

- **Easy Field Addition:** Add new fields and relationships.
- **Migration System:** Simplify schema updates.
- **Audit Logging:** Track all changes.
- **Archival Strategies:** Manage historical data efficiently.

Performance Optimization

- **Caching:** Reduce load for frequently accessed data.
- **Query Optimization:** Improve system efficiency.
- **Scalable Storage:** Handle attachments and large datasets.
- **Regular Maintenance:** Ensure smooth operation.

Customer Features

Customer Portal

- **Ticket Tracking:** Allow customers to view, update, and track their tickets.
- **History of Interactions:** Display previous communications and resolutions.
- **Secure Login:** Ensure privacy with authentication.

Self-Service Tools

- **Knowledge Base:** Provide searchable FAQs and help articles.
- **AI-Powered Chatbots:** Offer instant answers to repetitive queries.
- **Interactive Tutorials:** Guide customers through common issues step-by-step.

Communication Tools

- **Live Chat:** Enable real-time support conversations.
- **Email Integration:** Allow ticket creation and updates directly via email.
- **Web Widgets:** Embed support tools on customer-facing websites or apps.

Feedback and Engagement

- **Issue Feedback:** Collect feedback after ticket resolution.
- **Ratings System:** Let customers rate their support experience.

Multi-Channel Support

- **Mobile-Friendly Design:** Ensure support tools work on all devices.
- **Omnichannel Integration:** Support interactions via chat, social media, and SMS.

Advanced Features

- **Personalized Suggestions:** Use AI to recommend relevant articles or guides.
- **Proactive Notifications:** Alert customers about ticket updates or events.
- **Multilingual Support:** Offer help in multiple languages.

AI Objectives (Week 2)

Enhancing Ticket Management with AI

You've built a ticket management system—now it's time to integrate AI to minimize repetitive tasks and elevate efficiency. The goal is to enable a highly automated customer service system, where AI resolves most tickets and significantly reduces the workload on human agents.

Baseline AI Functionality

1. LLM-Generated Responses

- Use a Large Language Model (LLM) to generate responses for customer tickets.
- Ensure responses are courteous, user-friendly, and assistive.

2. Human-Assisted Suggestions

- When human action is required, the LLM should:
 - Suggest or prepopulate a response to speed up resolution.

3. RAG-Based Knowledge Management

- Provide the LLM with relevant context to ensure factual responses.
- Use a Retrieval-Augmented Generation (RAG) system to retrieve necessary knowledge.
- Ensure the RAG system is extensible, allowing administrators to add or update knowledge sources.

4. Agentic Tool-Using AI

- Analyze and route incoming tickets by type, priority, or other criteria.
- Design an extensible system enabling the AI to interact with external APIs for better interoperability.

Advanced Features

1. Refined Self-Service Support

- Automate the entire support experience for most cases, with AI managing end-to-end ticket resolution.
- Establish clear workflows for escalation and routing to human agents when necessary.

2. Human-in-the-Loop Enhancements

- Create a streamlined queue and user experience for human review.
- Ensure seamless integration of human oversight into the AI-driven workflow.

3. Multi-Channel and Multi-Modality Support

- Allow users to request and receive support across multiple channels, including:
 - Phone
 - Chat
 - Email
- Incorporate audio and visual elements into support interactions for richer user experiences.

4. AI-Summarized Ticket and System Status

- Provide admins with a dynamically updated dashboard view.
- Include AI-generated summaries of system performance and ticket status.
- Enable interactive Q&A for admins to query system data.

5. Learning and Growth System

- Log and save outcomes when tickets require human intervention.
- Use this dataset to improve the AI's ability to handle similar cases in the future automatically.

Outcome

The result is a system where AI agents handle a significant portion of the workload, allowing human agents to:

- Guide or oversee AI processes.

- Focus on resolving complex or edge-case tickets.

By implementing these features, your system transitions from a basic ticket manager to an intelligent, scalable, and efficient customer support platform.

Important Technical Decisions (ITDs)

1. Backend Infrastructure Selection

Decision: Use Supabase as the primary backend infrastructure

Options Considered:

- Supabase
- AWS/Google/Azure cloud services
- Firebase

Reason:

- Supabase provides a narrow, focused set of operations that make it more effective for AI integration.
- Offers comprehensive foundation services in a single platform:
 - Authentication with multiple SSO providers
 - Database and object storage
 - Edge Functions for custom business logic
 - Built-in Vector data store
 - REST and GraphQL APIs
 - Row Level Security (RLS) for authorization
 - Eventing system and real-time data synchronization
- Simpler API surface area compared to alternatives (e.g., AWS's 10,000+ API calls)
- Built-in AI agent support
- Reduces complexity in IAM and security configuration
- Great tooling for multi-branch development
- Supabase is an open-source implementation of Firebase-like system and deploys on AWS or any other cloud provider.

Resources:

- <https://www.youtube.com/watch?v=dU7GwCOgvNY&pp=ygUPc3VwYWJhc2UgY291cnNI>
- <https://www.youtube.com/watch?v=ydz7Dj5QHKY&list=PL4cUxeGkcC9hUb6sHthUEwG7r9VDPBMKQ>
- <https://www.youtube.com/watch?v=zBZgdTb-dns&pp=ygUPc3VwYWJhc2UgY291cnNI>

2. Development Tool Selection

Decision: [Use Lovable + Cursor combination for development](#)

Options Considered:

- Single LLM interactions
- Traditional IDE-based development
- Lovable + Cursor combination
- Supabase UI Agent

Reason:

- Single LLM interactions require too much manual effort for feedback loops
- Traditional development is slower
- Lovable + Cursor combination provides:
 - 20-50x improvement in development speed
 - Great support and understanding of Supabase API and subsystems.
 - Simple Supabase library available for most common platforms.
 - Comprehensive feedback loops across all system aspects
 - Superior debugging and bug-fixing capabilities
 - Complementary strengths without conflicting assumptions
 - Natural language interface for rapid development
 - Github integration for version control
 - Automated deployment capabilities
- Lovable is your primary interface that force a product manager thinking
- When Lovable can't dig itself out of a problem, you can open up the repo in cursor and get the cursor agent to fix it

3. Cursor Composer vs Agent

Decision: [Use cursor agent](#)

Options Considered:

- Cursor Composer
- Cursor Agent

Reason:

- Cursor Composer is a one-shot approach to solving any problem
- Cursor Agent takes an agentic approach to solving a problem

4. Code Organization Strategy

Decision: Prioritize AI-optimized code organization over traditional human-centric patterns. Don't keep trying to refactor the code for human readability.

Options Considered:

- Traditional clean code architecture
- Strict frontend/backend separation
- AI-optimized organization

Reason:

- AI agents, not humans, primarily maintain code
- Traditional separation of concerns may hinder AI performance (disconnected context)
- Lots of code in a single file is good for AI and terrible for humans.
- Benefits include:
 - More efficient context handling for AI
 - Standardized hooks and function calls
 - Seamless flow of business logic across frontend and backend
 - Faster development and maintenance cycles

5. Multi-Frontend Architecture

Decision: When you need multiple small UIs with a single backend, use a centralized edge function repository.

Options Considered:

- Distributed edge functions across repos
- Centralized edge function repository - MonoRepo
- Hybrid distribution

Reason:

- Prevents logic conflicts across multiple frontends
- Maintains a single source of truth for business logic
- Reduces risk of overwriting logic across repositories
- Simplifies maintenance and updates
- Allows other repos to focus on frontend-specific code
- Switching to Monorepo will prevent the use of Lovable, but Cursor will do well.

6. Source control

Decision: Use GitHub

Options Considered:

- Github
- Gitlab
- Other Git providers

Reasons:

- GitHub has native support in both Cursor and Lovable
- Extensive support across development tools and CI/CD systems

7. CI/CD

Decision: Use [AWS Amplify 2.0 as the deployment platform](#)

Options Considered:

- Amplify 2.0
- S3 + Cloudfront
- Vercel

Reason:

- S3 + CloudFront is too low-level to deal with
- This is a close call between Vercel and Amplify 2.0.
- Both are two-click integrations with GitHub, and the deployments are fast and simple
- Amplify has a slight edge over Vercel, given its Route53 integration that makes it easy to set up custom domains. Vercel isn't much more complex, but you must log into two different systems.

8. Framework Selection

Decision:

[While we recommend using LangChain as a teaching tool and for implementing your agent framework, it is not a requirement for this project. You are free to choose the framework or tools that best suit your preferences and project needs.](#)

Options Considered:

- LangChain
- LlamaIndex
- Custom Frameworks (e.g., Python's asyncio or FastAPI)
- Cloud Platforms (e.g., AWS SageMaker, Azure AI, Google Cloud AI)

Reason:

LangChain is recommended because it:

- Offers pre-built integrations for LLMs, RAG workflows, and workflow chaining.
- Features a modular design that simplifies experimentation with different agent architectures.
- Provides extensive documentation and community support, making it a great starting point for learning agent frameworks.

Alternative frameworks can also be effective, depending on your goals:

- LlamaIndex: Focused on retrieval-augmented generation workflows and data-centric operations.
- Custom Frameworks: Provide maximum control, leveraging libraries like Python's asyncio or web frameworks like FastAPI.
- Cloud Platforms: Offer scalable and pre-built orchestration tools, such as AWS SageMaker or Azure AI, for agent workflows.

Resources:

- <https://x.com/kregenrek/status/1879230971171733944>
- <https://x.com/nutlope/status/1879587920744788172>

Test2Pass (T2P) requirements

. Brainlift Documentation

- **Required Sections:**
 - **Purpose:** Clearly outlines the goal of the application and what it aims to solve.
 - **Experts:** Details expertise consulted (e.g., domain experts or credible references).
 - **Spiky POVs:** Highlights key insights or unique perspectives that guided the approach.
 - **Knowledge Tree:** Maps key concepts and their interconnections.
- **External Resources:**
 - References at least 5 relevant, high-quality external resources.
 - Sources should be credible and directly inform the project's implementation.
- **Impact:**
 - Demonstrates how the Brainlift meaningfully shapes LLM behavior when provided as contextual input.

Video Walkthrough (e.g., Loom)

- **Length:** 3-5 minutes.
- **Accessibility:** Shared publicly (via a platform link).
- **Content:**
 - Demonstrates the entire ticket lifecycle from creation to resolution.
 - Highlights AI agent support, including response generation, routing, and human-in-the-loop workflows.

Git Repository

- **Source Code:**
 - High-quality, production-grade code.
 - Passes autograder checks for style, formatting, and best practices.
- **Testing:**
 - Tests written and passing for all critical path code (e.g., logic impacting ticket management workflows).
 - Includes unit tests, integration tests, and edge cases.
- **CI/CD:**
 - Some CI/CD pipelines are implemented and functional.
 - Automated build, test, and deploy processes integrated into the workflow.



AI Tools and Resources

You'll need to dive into agents, tools, and data models:

- [Build an Agent | !\[\]\(7a8011739ec4e250e2f89a547d75fb0a_img.jpg\) LangChain](#) - tutorial for building your first LLM agent
- [Tools Concepts | !\[\]\(07dce76283bf618e2364d95ae0021e26_img.jpg\) LangChain](#) - conceptual guide to LLM tools
- [Tools Integrations | !\[\]\(44ee86b940d3a0ca166486da8985875e_img.jpg\) LangChain](#) - list of existing tools to potentially integrate

The data modeling (e.g. schema and migrations for tickets and other db entities) is not specific to AI - but again it is worth taking time to understand and plan. You can refer to open-source implementations such as [FreeScout](#) to draw inspiration and design ideas.

Additionally, here are resources on Zendesk and CRMs in general, to help you plan:

- [Zendesk Tutorials - YouTube](#)
-  COMPLETE Zendesk Tutorial For Beginners 2021 - How To Use Zendesk Cu...
-  Zendesk Tutorial: Customer Service Software & Sales CRM

Scope and deliverables

Deliverable	Description
CRM app	Working CRM app that enables, at minimum, ticket filing, management, and response, with three types of users (customers, workers, admins)
AI augmentation - baseline	CRM app automatically routes and resolves tickets that do not require human interaction
AI augmentation - stretch	AI experience is more refined - self-service, multimodal, admin summary, learning system, or come up with your own idea!

Milestones

Completion date	Project phase	Description
Jan 21, 2025	CRM app MVP	You should have a working CRM app with at least ticket entities and creation
Jan 22, 2025	Check in 1	
Jan 24, 2025	App complete	On Friday you should have completed your app.
Jan 27, 2025	AI Objectives start	
Jan 29, 2025	Check in 2	
Jan 31, 2025	AI features complete	On Friday you should have completed the AI objectives

