
Epic 1: Core Agent Functionality

This epic covers all the backend logic and autonomous capabilities of the Guardian AI agent operating within WhatsApp. It is the "brain" of the system, responsible for monitoring, analyzing, and acting on messages.

User Stories

- **User Story 1.1:** As an Administrator, I want the Guardian AI agent to securely connect to my designated WhatsApp account so that it can monitor all incoming text messages in real-time.
    - **Tasks:**
        - Integrate the Twilio API for receiving incoming message webhooks[1111].
        - Develop a message processing pipeline to queue and handle new messages asynchronously using FastAPI[2].
        - Implement secure storage for API keys and account credentials.
        - Create a persistent connection manager to handle API status and potential disconnects.
        - Log all incoming messages to the MongoDB database for record-keeping[3].
- **User Story 1.2:** As an Administrator, I want the agent to automatically moderate content by flagging messages with forbidden keywords, spam links, or flooding behavior so that our communication environment remains safe and productive.
    - **Tasks:**
        - Develop a "Moderation Agent" module that fetches the current rule set (keywords, regex patterns) from the database[4].
        - Implement real-time scanning of message content against the keyword and spam link patterns[5555].
        - Implement the rate-limiting logic to detect message flooding from a single user[6].
        - When a violation is detected, create a "violation" event and log it to the moderation log in MongoDB with context (user, message, rule violated)[7].
        - Develop the logic for the agent to issue a pre-defined, contextual warning in the chat where the violation occurred[8888].
- **User Story 1.3:** As an Administrator, I want the agent to intelligently identify and answer frequently asked questions from a knowledge base so that my support staff's workload is reduced[9999].

    - **Tasks:**
        - Develop a "Knowledge Agent" module responsible for FAQ handling.
        - Integrate the self-hosted LLM (e.g., Llama 3) to perform semantic analysis on incoming messages to identify questions[101010].
        - Implement a retrieval mechanism to find the best matching question-answer pair from the knowledge base stored in MongoDB[11111111].
        - Engineer prompts for the LLM to ensure accurate question matching.
        - Develop the logic for the agent to autonomously send the corresponding pre-written answer into the correct chat via the Twilio API[12].
        - Log all answered FAQs in the database for analytics[1313].
- **User Story 1.4:** As an Administrator, I want the agent to perform basic sentiment analysis on conversations so that I can be alerted to potential conflicts or negative trends[14141414].
    - **Tasks:**
        - Integrate the LLM to perform sentiment analysis (positive, neutral, negative) on incoming messages.
        - Develop logic to calculate an overall sentiment score for individual chats over time.
        - Store sentiment scores alongside message data in MongoDB.
        - Create a mechanism to flag conversations with sustained negative sentiment for admin review on the dashboard[15].

Epic 2: Administrator Control Dashboard

This epic covers the development of the web-based user interface that allows administrators to configure, manage, and monitor the Guardian AI agent.

User Stories

- **User Story 2.1:** As an Administrator, I want to securely log in to a web dashboard so that only authorized personnel can manage the agent's settings[16161616].
    - **Tasks:**
        - Set up a basic React project for the frontend dashboard[17].
        - Design and build a login page UI component.

        - Implement a user authentication system (e.g., JWT-based) in the FastAPI backend.
        - Create secure API endpoints for user login and session management.
        - Implement protected routes in the React application that require authentication.
- **User Story 2.2:** As an Administrator, I want a user-friendly interface to manage the moderation rules (keywords, spam patterns, rate limits) so that I can customize the agent's behavior without needing technical help[18181818].
    - **Tasks:**
        - Design the database schema for storing moderation rules in MongoDB.
        - Create full CRUD (Create, Read, Update, Delete) API endpoints in FastAPI for managing rules.
        - Build a React UI with a form and table to add, view, edit, and delete moderation rules.
        - Implement state management in the frontend to handle the rule data.
        - Connect the frontend UI to the backend API endpoints.
- **User Story 2.3:** As an Administrator, I want a simple interface to manage the FAQ knowledge base so that I can easily add, update, or remove question-and-answer pairs[19191919].
    - **Tasks:**
        - Design the MongoDB schema for the knowledge base (storing Q&A pairs).
        - Create full CRUD API endpoints in FastAPI for managing the knowledge base.
        - Build a React UI for adding, viewing, editing, and deleting FAQs.
        - Implement a search or filter functionality in the UI to easily find specific FAQs.
- **User Story 2.4:** As an Administrator, I want to view a real-time log of all agent actions (warnings issued, FAQs answered, messages flagged) so that I have complete oversight and can review its performance[202020].
    - **Tasks:**
        - Create a "read-only" API endpoint in the backend to fetch the moderation log with pagination and filtering options.
        - Design and build a "Live Moderation Log" component in React.
        - Implement real-time updates to the log, potentially using WebSockets or periodic polling.
        - Display key information for each log entry: timestamp, action type, message content, user, and the rule that was triggered[21].

- **User Story 2.5:** As an Administrator, I want a master control switch on the dashboard to activate or deactivate the agent's services for the connected account so that I have ultimate control[22222222].
    - **Tasks:**
        - Add an "isActive" field to the main agent configuration in the database.
        - Create an API endpoint to toggle this "isActive" status.
        - Implement a simple toggle switch component in the React dashboard.
        - Ensure the core agent logic in the backend checks this flag before processing any message.
- **User Story 2.6:** As an Administrator, I want to see an analytics overview with visualizations of key metrics so that I can understand the agent's impact and effectiveness at a glance[23].
    - **Tasks:**
        - Develop backend API endpoints to aggregate data for analytics (e.g., count of moderated messages per day, most frequent rule violations, most asked FAQs).
        - Design a dashboard UI layout for displaying charts and key performance indicators (KPIs).
        - Integrate a charting library (e.g., Chart.js or D3.js) into the React frontend.
        - Build components to visualize the aggregated data, such as bar charts for violations and pie charts for FAQ categories.

Epic 3: System Foundation & Deployment

This technical epic covers the foundational setup, architecture, and deployment infrastructure necessary for the project to run reliably across different environments.

User Stories

- **User Story 3.1:** As the Development Team, we need to establish the core project structure and version control so that we can collaborate effectively.
    - **Tasks:**
        - Initialize a Git repository.
        - Create separate folder structures for the FastAPI backend and the React frontend.
        - Establish branching policies (e.g., GitFlow).
        - Set up a project board (e.g., Jira, Trello) to track tasks.
- **User Story 3.2:** As the Development Team, we need to set up and deploy the self-hosted AI engine so that our application has a reliable and controllable NLP service[24].

    - **Tasks:**
        - Choose and download a quantized open-source LLM (e.g., Llama 3 8B) from Hugging Face[25].
        - Set up a Hugging Face Space to host the model[26].
        - Deploy the model behind a private API endpoint within the Space[27].
        - Create a client in the FastAPI backend to securely communicate with the hosted model API.
        - Define a fallback strategy to use the Google AI Gemini API's free tier if self-hosting fails[28].
- **User Story 3.3:** As the Development Team, we need to create a containerized deployment setup so that we can ensure consistency across development, testing, and production environments.
    - **Tasks:**
        - Write a `Dockerfile` for the FastAPI backend application.
        - Write a `Dockerfile` for the React frontend application.
        - Create a `docker-compose.yml` file to orchestrate the backend, frontend, and MongoDB services.
        - Write deployment scripts for different environments (Dev, Staging, Go-live)[29].

Epic 4: Advanced Agent Capabilities

Reasoning: This epic moves the agent from being purely reactive (moderating, answering FAQs) to being proactive. It starts to build a personality and an intelligence that can understand and engage with the community on a deeper level, identifying key members and summarizing conversations to provide value back to the administrators.

- **User Story 4.1:** As a Community Manager, I want the agent to proactively identify and engage with potential brand advocates so that we can nurture a loyal community.
    - **Tasks:**
        - Develop an "Advocate Identification" module that tracks user engagement metrics (e.g., message frequency, positive sentiment, helpfulness to others).
        - Implement logic to score users based on these metrics to identify top community members.
        - Create a set of pre-approved, personalized engagement messages for the agent to use.

        - Develop an action for the agent to initiate non-intrusive, positive conversations with identified advocates.
        - Flag identified advocates on the administrator dashboard for human review and further relationship-building.
- **User Story 4.2:** As an Administrator, I want to be able to request an on-demand summary of any long conversation thread so that I can quickly catch up on key points without reading everything.
    - **Tasks:**
        - Create a command (e.g., "/summarize") that administrators can use in WhatsApp.
        - Develop a "Summarization Agent" that can retrieve the conversation history for a specific chat.
        - Engineer a prompt for the LLM to generate a concise, executive summary of the conversation.
        - Implement the logic for the agent to send the summary as a private message to the requesting administrator.
- **User Story 4.3:** As a Community Manager, I want the agent to autonomously detect and alert me to potential PR crises before they escalate so that we can respond quickly.
    - **Tasks:**
        - Enhance the sentiment analysis module to detect rapid shifts towards negative sentiment across multiple users.
        - Implement keyword tracking for sensitive topics or brand mentions in a negative context.
        - Create a "Crisis Alert" threshold based on the volume and velocity of negative messages.
        - Develop an alert system (e.g., email, SMS, high-priority dashboard notification) to immediately inform administrators when the threshold is crossed.

Epic 5: Multi-Platform Integration & Orchestration

Reasoning: A marketing team doesn't just operate on one channel. This epic is critical for scaling the agent's operations, allowing it to manage a brand's presence across the digital landscape consistently. The Orchestrator Agent is the "manager" that ensures all the specialized agents work in concert.

- **User Story 5.1:** As a Marketing Manager, I want to connect the agent system to our company's Facebook, Twitter, and Instagram accounts so that it can manage all social channels from one place.
    - **Tasks:**
        - Research and integrate the APIs for Facebook Pages, Twitter (X), and Instagram.

        - Abstract the core agent logic (moderation, FAQ) to be platform-agnostic.
        - Develop platform-specific adapters to handle the unique message formats and API requirements of each channel.
        - Update the dashboard to allow administrators to add and authenticate new social media accounts.
- **User Story 5.2:** As a Marketing Manager, I want an "Orchestrator Agent" that can deploy other specialized agents (e.g., Moderator, Knowledge Agent) to the appropriate channel based on my rules.
    - **Tasks:**
        - Design the architecture for a high-level Orchestrator Agent that manages a pool of specialized agents.
        - Create a rules engine where administrators can define which agents are active on which platforms.
        - Develop the internal communication bus for the Orchestrator to pass tasks to the correct specialized agent.
        - Update the dashboard to provide a central control panel for managing agent deployment across all connected platforms.

Epic 6: Predictive Analytics & AI-Driven Strategy

Reasoning: This is where the agent becomes a strategist. Instead of just executing tasks, it starts to think, analyze data, and provide recommendations. This moves its value from saving time to actively improving marketing ROI, making it an indispensable part of the team.

- **User Story 6.1:** As a Content Strategist, I want the agent to analyze historical engagement data and forecast which content topics and formats will perform best next month so that I can optimize my content calendar.
    - **Tasks:**
        - Develop a "Predictive Insights Agent" module.
        - Create data pipelines to ingest historical performance data (likes, shares, comments, etc.) from all connected platforms.
        - Train a machine learning model to identify correlations between content attributes (topic, format, sentiment, time of day) and engagement.
        - Develop a feature on the dashboard to present these insights, such as "Recommended Topic: AI in Marketing" or "Optimal Post Time: Tuesday at 10 AM."
- **User Story 6.2:** As a Performance Marketer, I want the agent to use reinforcement learning to autonomously manage my paid ad campaigns on Facebook so that it can optimize my budget for maximum ROI without constant human intervention.
    - **Tasks:**
        - Integrate with the Facebook Ads API.

        - Build a "Reinforcement Learning Agent" that can create and modify ad campaigns (e.g., change copy, adjust targeting, alter bids).
        - Define the reward function for the agent (e.g., maximize click-through rate or conversions).
        - Implement a "safe mode" with a capped budget for the agent to run its experiments.
        - Create a dashboard to monitor the agent's ad performance and compare it against a human-managed baseline.

New Epics: Building the Autonomous Marketing Team

The following epics introduce the core functions that elevate the system from a powerful tool to a self-sufficient marketing department.

Epic 7: Autonomous Content Generation & Campaign Execution

Reasoning: The most fundamental role of a marketing team is to create and distribute content. This epic gives the agent the hands and voice to execute campaigns from start to finish. By integrating multimodal generation, it ensures the content is engaging and not limited to text.

- **User Story 7.1:** As a Marketing Manager, I want to provide the agent with a high-level campaign brief (e.g., "Promote our new product launch for Q4"), and have it generate a series of draft social media posts, including text and relevant images.
    - **Tasks:**
        - Develop a "Content Generation Agent" that can interpret a campaign brief.
        - Integrate a text-to-image model (e.g., Stable Diffusion, Midjourney API).
        - Engineer prompts that combine the campaign goal with brand voice guidelines to create compelling copy.
        - Develop logic for the agent to generate multiple variations of text and images for A/B testing.
        - Create a "Drafts" section in the dashboard where a human can review, edit, and approve the AI-generated content before it's scheduled.
- **User Story 7.2:** As a Marketing Manager, once I approve the content, I want the agent to autonomously schedule and publish it across all relevant social channels at the optimal times it previously identified.
    - **Tasks:**
        - Integrate the content generation and scheduling functionalities.
        - Create an automated workflow: `Brief -> Generate -> Await Approval -> Schedule -> Publish`.
        - Build a content calendar view on the dashboard showing all scheduled and published posts across all platforms.

        - Implement a post-publication monitoring function to track initial engagement on new content.

Epic 8: Lead Nurturing & Sales Funnel Automation

Reasoning: Marketing's ultimate goal is to generate leads and revenue. This epic transforms the agent into a sales development representative (SDR). It can identify potential customers, guide them through a predefined funnel, and hand them off to a human sales team when they are qualified, directly contributing to the bottom line.

- **User Story 8.1:** As a Sales Manager, I want the agent to identify potential leads who express purchase intent in conversations and automatically engage them with a pre-defined nurturing sequence.
    - **Tasks:**
        - Develop an "Intent Detection" module using NLP to recognize buying signals (e.g., "How much does this cost?", "Where can I buy?").
        - Create a lead nurturing workflow builder on the dashboard where managers can define a sequence of messages and resources to send to potential leads.
        - Implement logic for the agent to tag users as "leads" in the system and initiate the appropriate nurturing sequence.
- **User Story 8.2:** As a Salesperson, I want to be notified with the full conversation history when a lead has been qualified by the agent so that I can take over the conversation for the final sale.
    - **Tasks:**
        - Define "lead qualification" criteria (e.g., asked for a demo, visited the pricing page).
        - Create an automated handoff process, such as sending an email to the sales team with the lead's contact information and a summary of their interaction with the agent.
        - Integrate with a CRM (like Salesforce or HubSpot) to automatically create a new lead record upon qualification.