**How the Agentic AI Solution Works**

Our MindMate AI solution is architected around a single, powerful **Triage Agent** that is orchestrated by our backend application to provide an end-to-end triage workflow.

1. **Data Collection (The Frontend):** The process begins in our React.js frontend. It acts as the data collection interface, capturing the patient’s name, age, preferred language, and a free-text description of their symptoms.
2. **Orchestration (The Backend):** Upon submission, the frontend sends this data as a JSON payload to our Python/Flask backend at the /analyze endpoint. The backend validates the data and then invokes the Triage Agent.
3. **Reasoning and Action (The Triage Agent):** The Triage Agent is the core of our solution. Its entire function is defined by a sophisticated, few-shot prompt that is sent to the IBM Granite model.
   * **Agent's Role & Task:** The agent's assigned role is to act as a "mental health triage assistant." Its task is to receive the patient's symptoms and language, analyze the text for indicators of distress, and then perform two actions:
     1. **Classify** the severity of the case as "Low," "Medium," or "High."
     2. **Generate** an empathetic, actionable recommendation in the user's specified language.
   * **Agent's "Knowledge":** The agent's ability to perform this task comes from the examples provided directly in its prompt within our app.py file. It learns in real-time from examples of low, medium, and high-risk symptoms what its output should look like, ensuring a reliable and structured JSON response.
   * **Output:** The agent returns a structured JSON object containing the severity and recommendation to our Flask application.
4. **Logging and Presentation:** The Flask backend logs the agent's complete assessment to a triage\_log.csv file and returns the result to the user's browser. The separate Clinical Dashboard fetches data from this log file to provide staff with a real-time, filterable, and prioritized queue of patients.

**How We Use IBM watsonx to Build It**

Our working prototype is built with IBM watsonx at its core.

* **IBM watsonx.ai:**
  + We use watsonx.ai as the engine for our **Triage Agent**. Specifically, our Python backend calls the **IBM Granite-3 8B Instruct model** (ibm/granite-3-8b-instruct) via its REST API.
  + Our app.py script handles the complete authentication flow, obtaining an IAM token from the WATSONX\_API\_KEY and using it to authorize the generation request.
  + The **Prompt Lab** in watsonx.ai would be the ideal environment for further refining the agent's core prompt, allowing us to test different examples and parameters to enhance its accuracy and empathetic tone without changing the application code.
* **IBM watsonx.data (Recommended Production Upgrade):**
  + For a real-world clinical deployment, the triage\_log.csv file would be replaced with **watsonx.data**. This would provide a secure, scalable, and governed data lakehouse for storing sensitive Protected Health Information (PHI), ensuring compliance with privacy regulations and enabling powerful analytics on anonymized data to track community health trends.
* **IBM watsonx.governance (Essential for Clinical Trust):**
  + To move from a prototype to a trusted clinical tool, we would integrate **watsonx.governance**. This would allow us to monitor the Triage Agent for performance, drift, and potential bias. Most importantly, it would provide an **explainable audit trail** for every classification the agent makes, allowing a clinician to understand *why* a patient was flagged as "High" severity. This transparency is non-negotiable for building trust and ensuring safety in a healthcare setting.