WATER LLM ENGINE – FULL FUNCTION OVERVIEW (LAYMAN EXPLANATI

post_scada_command

Purpose: Sends a control command to a SCADA system to operate assets (e.g., valves, pumps).
Inputs: SCADA API endpoint, command to send, and authentication token.
Logic: Makes a POST request to the SCADA endpoint with the given command.
Actions: Triggers a real-world asset action like opening a valve.
Verification: Retries up to 3 times if the command fails to send.
How to Call: Internally used within `actuate_asset()` function.

send_mqtt_message

Purpose: Publishes a message (command) to a specific MQTT topic for device communication.
Inputs: MQTT broker address, topic name, command string.
Logic: Sends the payload to the topic using MQTT protocol.
Actions: Used for notifying edge devices or controllers.
Verification: Retries 3 times if MQTT transmission fails.
How to Call: Internally used inside `actuate_asset()`.

send_opcua_command

Purpose: Issues commands via the OPC-UA protocol to compatible devices.
Inputs: OPC-UA server URL, command to execute.
Logic: Connects to OPC-UA device, finds the target node, sets its value.
Actions: Writes values (like "open", "start") to industrial devices.
Verification: Retries 3 times if OPC-UA fails.
How to Call: Called by `actuate_asset()` only if OPCUAClient is available.

send_modbus_command

Purpose: Sends control instructions to Modbus-compatible PLCs.
Inputs: IP, port of PLC, command string.
Logic: Connects to the PLC and writes a register value.
Actions: Used in water infrastructure to start/stop hardware like pumps.
Verification: Retries 3 times on failure.
How to Call: Called by `actuate_asset()` if Modbus config is present.

get_integration_config

Purpose: Fetches all API endpoints, protocols, and tokens needed for a location from the database.
Inputs: Location name (default is 'London').
Logic: Queries SQLite database and maps values into a dictionary.
Actions: Provides configuration for all actuator/sensor functions.
Verification: Returns empty dict if no config found.
How to Call: Used across nearly all major functions to get connection details.

actuate_asset

Purpose: Sends operational commands to real-world water infrastructure assets using various protocol
s.
Inputs: Command string (like 'open_valve'), Location name.
Logic: Fetches config, then sends the command using available protocols (SCADA, MQTT, OPC-UA, Modbus
).
Actions: Activates pumps, opens valves, etc.
Verification: Confirms each protocol's response and logs success/failure.
How to Call: Directly or within overflow/storm response functions.

fetch_weather_data

Purpose: Pulls weather forecast data from a configured API.
Inputs: Location name.
Logic: Calls external API from config and parses JSON response.
Actions: Supplies rainfall forecast for decision-making.
Verification: Returns error if API fails or not configured.
How to Call: Called by overflow, prediction, storm functions.

fetch_sensor_data

Purpose: Collects live sensor values like tank fill %, inflow rate, etc.
Inputs: Location name.
Logic: Sends HTTP request to the sensor API from config.
Actions: Retrieves telemetry for downstream analysis.
Verification: Returns parsed JSON or error message.
How to Call: Used widely in advisory, storm, overflow, etc.

call_gpt

Purpose: Invokes GPT-4 to generate expert-level advisory using a prompt.
Inputs: Prompt string, temperature setting.
Logic: Sends structured messages to OpenAI's API.
Actions: Generates natural language advice or predictions.
Verification: Retries up to 3 times; logs error on failure.
How to Call: Used within advisory or report generation.

calculate_overflow_risk

Purpose: Calculates overflow risk based on rainfall and tank utilization.
Inputs: Rainfall in mm, tank fill percent.
Logic: Applies a rule-based threshold logic.
Actions: Returns risk score - LOW, MEDIUM, HIGH.
Verification: Simple logical if-else checks.
How to Call: Used in prediction, storm response, overflow control.

overflow_control

Purpose: Controls water infrastructure based on overflow risk.
Inputs: Location name.
Logic: Fetches data, calculates risk, acts accordingly (e.g., open valve).
Actions: Sends commands, logs result, generates advisory.
Verification: Returns risk level, action taken, GPT summary.
How to Call: Called during storm or overflow conditions.

get_real_time_inputs

Purpose: Consolidates real-time weather and sensor data.
Inputs: Location name.
Logic: Calls fetch functions and returns structured data.
Actions: Serves as input collector for other functions.
Verification: Fallbacks to defaults if missing data.
How to Call: Used in storm response, advisory, prediction.

predict_overflow

Purpose: Predicts whether overflow is likely based on thresholds.
Inputs: Rainfall in mm, tank fill percent.
Logic: Checks if either input crosses danger limits.
Actions: Returns True/False.
Verification: Boolean result.
How to Call: Called by storm coordinator, validator, etc.

dynamic_control_advice

Purpose: Recommends control strategy based on tank fill level.
Inputs: Tank fill percent.
Logic: Suggests pump or valve actions.
Actions: Returns advisory string.
Verification: Threshold logic.
How to Call: Used in storm response and diagnostics.

detect_anomalies

Purpose: Flags out-of-bound sensor readings.
Inputs: Dictionary of sensor readings.
Logic: Looks for overfill, negative inflow, etc.
Actions: Returns list of issues.
Verification: Hardcoded rule checks.
How to Call: Part of storm logic or validator.

compliance_check

Purpose: Checks if required overflow actions were taken during heavy rainfall.
Inputs: Rainfall in mm, overflow_triggered (boolean).
Logic: If rainfall exceeds threshold but no action, it's non-compliant.

Actions: Returns compliance status.
Verification: Logical evaluation.
How to Call: Part of validation, storm coordination.

run_all_analyses

Purpose: Runs all assessments in one go for a location.
Inputs: Location name.
Logic: Combines predictions, compliance check, anomaly detection, advisory.
Actions: Returns structured output for dashboard use.
Verification: Internally validates every logic block.
How to Call: Used in dashboards or API endpoints.

get_logged_interventions

Purpose: Reads file containing manual or automated interventions.
Inputs: None.
Logic: Loads from log file.
Actions: Lists actions taken previously.
Verification: File read fallback if log absent.
How to Call: Used in validation or audit features.

get_parameter_descriptions

Purpose: Explains all key sensor parameters in human terms.
Inputs: None.
Logic: Hardcoded dictionary.
Actions: Returns dictionary of parameter meanings.
Verification: None needed.
How to Call: Helpful for UI/UX or help tab.

generate_regulatory_report

Purpose: Creates a formatted report text string for regulators.
Inputs: Location name, rainfall, risk level.
Logic: Generates a summary string including compliance status.
Actions: Used for display or logging.
Verification: Checks if risk is high.
How to Call: Part of storm response or reporting tab.

recommend_infrastructure_upgrades

Purpose: Suggests ways to improve the system based on observations.
Inputs: Location name.
Logic: Returns pre-defined improvements.
Actions: Part of advisory generation.
Verification: Hardcoded list.
How to Call: Used in storm advisory or dashboard.

alert_operator

Purpose: Sends alerts to the operator or control center.
Inputs: Message string (default alert).
Logic: Prints and logs alert.
Actions: Human intervention is triggered.
Verification: Print confirmation.
How to Call: Used in risk escalation paths.

suggest_action_for_risk

Purpose: Gives human-readable steps to deal with different risk levels.
Inputs: Risk level (LOW, MEDIUM, HIGH).
Logic: Rule-based mapping.
Actions: Returns textual advisory.
Verification: None needed.
How to Call: Used by many coordinator and validation modules.

fetch_tank_config

Purpose: Loads tank configuration from CSV for a location.
Inputs: Location name.
Logic: Filters a dataframe and returns zone/capacity mappings.
Actions: Used in balancing logic.
Verification: Catches file read failures.
How to Call: Used in load balance tanks logic.

load_balance_tanks

Purpose: Redistributes water among tanks if any are overfilled.
Inputs: Location name.
Logic: Reads tank config, checks sensor fill per tank, acts if > 90%.
Actions: Sends actuation, logs actions, returns balance summary.
Verification: Logs actions, fallbacks on bad sensor data.
How to Call: Part of storm response or manual validation.

forecast_weather_with_gpt

Purpose: Uses GPT to generate multi-day weather forecasts and warnings.
Inputs: Location name, forecast horizon in days.
Logic: Creates prompt and calls GPT-4 for contextual weather outlook.
Actions: Returns advisory with insights.
Verification: Fallback if GPT call fails.
How to Call: Used in dashboards, summary or storm modules.

storm_response_coordinator

Purpose: Executes all necessary actions during a storm event.
Inputs: Location name.
Logic: Checks for ongoing storm, reads impact zones, collects data, actuates assets, logs outcomes.

Actions: Full orchestration of real-time storm control.
Verification: Duplicate storm check, detailed output returned.
How to Call: Central entry point for simulating storm handling.

check_asset_availability

Purpose: Verifies whether each infrastructure asset is functioning as expected.
Inputs: Location name.
Logic: Reads config file and compares expected vs actual sensor values.
Actions: Flags working, non-working or missing telemetry.
Verification: Tabulates and reports per-asset status.
How to Call: Used in diagnostics, validation or storm flow.

compare_predictions_with_actuals

Purpose: Logs whether real events matched overflow predictions.
Inputs: Location name.
Logic: Pulls live data, makes prediction again, logs result.
Actions: Saves results in JSON log line format.
Verification: Used for post-storm audit accuracy.
How to Call: Background job or dashboard validator.

generate_contextual_advisory

Purpose: Produces a rich contextual advisory using current conditions and river risk.
Inputs: Location name.
Logic: Gathers data and builds a tailored GPT-4 prompt.
Actions: Returns GPT-generated advisory text.
Verification: Includes river severity data in prompt.
How to Call: Triggered in dashboard or storm analysis.

get_river_impact_severity

Purpose: Retrieves flood risk level for river zones near a location.
Inputs: Location name.
Logic: Loads CSV, filters by location, formats into list of zones.
Actions: Used for early action and impact prediction.
Verification: Logs file read failure if invalid.
How to Call: Storm functions and advisory modules use this.