I. Goal Selection and Filtering

The current frontier selection logic lacks sufficient filtering, leading to cycles and poor navigation choices.

1.1 Distance Range and Anti-Cycle Mechanisms

- **Problem:** The current minimum distance of **0.5m** (if distance_from_robot > 0.5:) is too small, causing issues, and there is no maximum limit (Issue 1). Furthermore, the robot cycles repeatedly between locations (Issue 3).
- **Solution - Distance Range:**
  - **Minimum Distance:** Increase the threshold to **2.0m** to prevent the robot from selecting trivial, unhelpful goals.
  - **Maximum Distance:** Add an upper limit, suggesting **5m to 8m**, to avoid selecting overly distant goals that may become invalid due to map updates or require long, high-cost paths.
- **Solution - Anti-Cycle:** Implement a **short-term memory** to track the last five successful goal locations and reject any new goal candidate within **0.3m** of these recent spots (Issue 3). This is separate from the blacklist.

1.2. Goal Ranking and Heuristics

- **Problem:** The scoring heuristic (size / distance) overly favors close goals, even if they offer less exploration value than a slightly farther, larger frontier (Issue 5).
- **Solution:** Replace the simple heuristic with a **weighted scoring function** that balances three key factors: **Frontier Size**, **Distance from Robot**, and **Exploration Value/Path Cost**. This requires incorporating a measure of the path's complexity (see Section II).
  - Example:

$$Score = (W_{size} \times \text{Size}) - (W_{dist} \times \text{Distance}) - (W_{cost} \times \text{Path Cost}).$$
.

II. Goal Reachability and Path Validation

The most critical missing element is verifying that the selected goal is reachable without traversing high-cost areas or walls (Issue 2).

2.1. Pre-Check Reachability

- **Problem:** Goals are selected behind walls, are unreachable, or the path to them has a prohibitively high cost (Issue 2, Jen^2).
- **Solution - Initial Filter:** Before adding a goal to the ranked list, it must pass a **Line-of-Sight/Costmap Check**.
    - **Approach:** Utilize a ROS 2 service like **nav_msgs/GetPlan** (the standard way to ask Nav2 for a path) to check if a valid path exists from the robot's pose to the candidate goal.
    - Goals that result in a failed path request or a path that is too long or contains steps through high-cost regions (e.g., near walls as identified by the local costmap) must be immediately filtered out (Issue 2, Jen^2). The path length returned can also be used in the new scoring function (Section I.2).

2.2 Ongoing Goal Recheck

- **Problem:** As the robot moves and the map updates (due to relocalisation or new sensor data), a previously valid path can become blocked or high-cost (Goal Drift) (Jen^2).
- **Solution - Periodic Validation:** While a goal is active, implement a **periodic check** (e.g., every 5-10 seconds) that re-validates the path to the current goal against the latest costmap. If the path is now detected to be high-cost or blocked, **cancel** the active goal immediately and trigger a new frontier search (Jen^2).

III. Failure Handling and Recovery

The current blacklist and failure logic are weak, leading to wasted time on impossible goals and inadequate recovery (Issue 4, Issue 6).

3.1. Blacklist Enhancement and Differentiation

- **Problem:** The current Blacklist (BL) is too small (radius **0.2m**, max length **5**) and doesn't differentiate between successful and failed locations (Issue 4, Jen^2).
- **Solution - Blacklist (Failed Goals):** Increase the BLACKLIST_RADIUS_METERS to a larger value (e.g., **0.5m** to **1.0m**) and significantly increase the max queue length (e.g., to **20-40**). This list is for goal locations that *failed* the navigation.
    - **Alternative Solution:** Trial small values e.g 0.1m also

- **Solution - Brownlist (Explored Goals):** Introduce a separate **Brownlist** to store locations of *successfully reached* goals. This prevents the robot from revisiting areas that are already explored but should be conditionally ignorable during recovery.

3.2. Goal Progress Monitoring and Timeouts

- **Problem:** The robot can get stuck on an impossible goal without canceling (Issue 6). Cancellation should be based on lack of progress, not just time elapsed (Jen^2).
- **Solution - Progress Rate Check:** Implement a monitor timer for active goals. Track the robot's distance to the goal over time. If the **rate of change of distance** (i.e., progressive movement toward the goal) is below a minimum threshold over a defined period (e.g., **30 seconds**), the goal should be **canceled** and the location added to the **Blacklist** (Jen^2). This correctly differentiates a long, valid path from a robot that is stuck or oscillating.

3.3. Stuck/Reset Recovery Mode

- **Problem:** The strict filtering rules can leave the robot stuck in a corner with no valid goals (Jen^2).
- **Solution - Corner Escape Logic:** If goal selection fails repeatedly (e.g., 5 consecutive times) due to all candidates being filtered out:
    1. **Relax Constraints:** Temporarily allow goal candidates that are in the **Brownlist** (explored) to be selected, specifically to help the robot leave a constrained area (Jen^2).
    2. **Expanded Range:** Temporarily increase the maximum goal distance to seek options farther away (Jen^2).
    3. **Last Resort Reset:** If the robot remains stuck for a long duration (e.g., **40 seconds**), perform a full **Blacklist/Brownlist reset** to force exploration of potentially missed areas (Jen^2).

IV. Debugging and Error Reporting

- **Problem:** Current debugging is poor; the robot sends the same goal repeatedly, and the user doesn't know *why* it is stuck or failing (Jen^2, Overall).

4.1 Solution: Enhance Logging and Error States.

- **In Nav2 Callbacks:** Log Nav2's internal status and any recovery attempts it makes. If the goal fails, log the specific status code received from Nav2 (GoalStatus.STATUS_...) (Overall).
- **In Goal Cycler/Service:** When a new goal cannot be found, the node must log a specific, defined reason instead of a generic message:
    - "No suitable goals found: **All Frontiers Blacklisted**."
    - "No suitable goals found: **Path Blocked to All Candidates**."
    - "Navigation failed: **No progressive movement towards goal**."
- This clarity is essential for debugging and will provide better feedback in the service response message (response.message).