SmartStop Clinical Decision Support System Specification

1. Field of the Invention

This invention relates to clinical decision support (CDS) systems for infusion therapy. In particular, it addresses a standalone, non-device CDS platform for infusion pump safety that separates clinician acknowledgment of alerts from actual resolution of the underlying risk. The field covers medical informatics and patient safety systems that log clinician responses and monitor therapy state without directly controlling any medical devices.

2. Background of the Invention

Infusion pumps in critical care settings generate numerous safety alerts (occlusions, dose limits, abnormal lab values, etc.). Traditional smart pumps and hospital alarm systems often allow a clinician to silence or acknowledge an alarm, which stops the audible alert but does not guarantee the underlying issue is corrected. In conventional systems, acknowledging an alert may be conflated with resolving it – the alarm clears even if the risky infusion continues or the abnormal physiological condition persists. This can lead to alert fatigue and unmitigated risks, as staff might assume a situation is handled when in fact the hazard remains. There is a need for a CDS solution that logs clinician acknowledgment without automatically marking the alert as resolved until an objective change in therapy or patient state occurs. Moreover, to remain an FDA-exempt “Non-Device CDS,” such a system must avoid direct pump control while providing transparent recommendations that a clinician can independently verify . The present invention addresses these gaps with a layered alert management approach that improves safety and accountability.

3. Summary of the Invention

The invention, termed SmartStop, is a clinical decision support system that improves infusion safety through a dual-state alert management mechanism. When a hazardous condition or conflict is detected (e.g. an infusion parameter violation or a dangerous lab result while a drug is infusing), SmartStop generates an alert and presents it to the clinician on a user interface (e.g. a nurse’s smartwatch or workstation). The clinician can respond by pressing an “Acknowledge” or “Override” button, indicating awareness of the alert or an intention to deviate from the recommended action. Crucially, this acknowledgment does not stop the infusion or clear the alert – it simply logs the user’s response in an immutable audit trail and halts any further escalation notifications . The system’s novel algorithmic behavior keeps the alert active

(unresolved) until the underlying risk condition is truly remedied. A dedicated monitoring module tracks the infusion pump status and relevant patient parameters in real time. Only when it detects that the infusion has been stopped (e.g. pump rate reduced to zero) or that the risky condition has normalized (e.g. lab values return to safe range) does SmartStop mark the conflict as “Resolved.” This two-stage process – Acknowledged vs. Resolved – creates a safety buffer: it distinguishes the clinician’s intent to address from the actual completion of the corrective action.

Structurally, the SmartStop system comprises layered components that work together to achieve this functionality. An alert detection engine monitors data streams (from infusion pumps and electronic health records) and triggers conflicts based on rule-based criteria. A notification and user interface layer then delivers the alert to the appropriate clinician (for example, the nearest qualified nurse) and captures acknowledgment inputs. All user actions (acknowledgments, overrides) are recorded by an audit logging subsystem as tamper-proof events, e.g. using hash-chained entries to ensure an immutable log . Separately, a therapy state evaluation module continuously or periodically checks whether the infusion in question is still running and whether the patient’s hazardous condition persists. This module implements a state-transition algorithm to automatically resolve the conflict in the system once it determines that all contributing factors have been addressed (for instance, all involved pumps turned off or offending parameters corrected). Until that point, the conflict remains in an active/unresolved state in the system’s records, even if the clinician acknowledged it minutes or hours prior. This layered architecture is novel in the CDS context: it clearly separates the human confirmation (documentation of user intent) from the system’s safety closure (objective resolution tracking). By doing so, SmartStop ensures accountability and reduces residual risk – a clinician cannot simply “paper over” a critical alert; the system will continue to flag it (or remind/escalate as appropriate) until the danger is actually gone.

In summary, SmartStop’s primary Non-Device CDS embodiment offers a safer alert handling workflow. It does not issue any control commands to the pump when an alert is acknowledged (complying with non-device CDS regulatory criteria), yet it provides a robust safety net by monitoring the therapy. The innovation lies in the algorithm that tracks conflicts through distinct states – Active → Acknowledged → Resolved – with logging at each step and automated verification of risk mitigation. This approach shortens alarm-to-action intervals, mitigates alarm fatigue by preventing repetitive alerts (since acknowledgments halt duplicate alarms for the same event), and adds a failsafe layer of oversight without intruding on the clinician’s autonomy or the pump’s built-in controls.

4. Brief Description of the Drawings

FIG. 1 is a block diagram of the SmartStop system architecture in its non-device CDS mode. It illustrates the major components including: the Alert Detection Engine (101) that subscribes to infusion pump alarms and patient data streams, the Alert Router and User Interface module (102) that delivers notifications to clinicians (e.g. via a smartwatch app 103), and the Audit Log Database (104) for recording events. Notably, FIG. 1 shows that SmartStop sits on top of the

existing pump network and hospital IT systems without issuing any control commands to the pump, emphasizing its standalone decision support role .

FIG. 2 is a state diagram (or flowchart) illustrating the conflict state transitions and logic. Starting from a Conflict Trigger state (201) when a risky condition is detected, the flow moves to Alert Active (unacknowledged) state. Upon clinician pressing “Acknowledge” or “Override,” the system logs the action (transition 202) and the conflict enters an Acknowledged (but Unresolved) state (203). The flowchart then depicts the continuous monitoring of therapy status (204): if the pump infusion stops or the patient’s risk condition resolves, the conflict transitions to a Resolved state (205). FIG. 2 highlights that if the condition remains (pump still running, etc.), the conflict stays in the acknowledged/unresolved state, possibly triggering escalations or reminders per policy (not shown for clarity). This figure provides a visual summary of the step-by-step algorithm that governs conflict lifecycles.

FIG. 3 illustrates an example user interface and audit trail snapshot. It shows a nurse’s smartwatch display (301) with an incoming alert and an “Acknowledge” button. After the nurse taps acknowledge, the system logs an entry (shown in an audit log view 302) such as “Alert #123 acknowledged by Nurse at 14:32:20” along with a status indicator that the alert is still active. FIG. 3 also depicts how multiple acknowledgment events and a final resolution event appear in sequence in the immutable log (each entry with timestamp and digital signature). This underscores the separation between user action and resolution: e.g., the log might read “Alert triggered – 14:30:00; Acknowledged by Nurse – 14:32:20; Resolved (pump stopped) – 14:40:00.”

(Note: The drawings are described for illustrative purposes and are not to scale. They focus on the logical relationships and data flows relevant to the invention’s novelty.)

5. Detailed Description of the Invention

5.1 System Architecture Overview

Referring to FIG. 1, the SmartStop system in its standalone CDS implementation consists of several interrelated modules operating on a hospital network server or cloud platform. The Alert Detection Engine continuously monitors input data for potential conflicts. Inputs may include infusion pump telemetry (e.g. current flow rates, alarms, settings) and patient data such as lab results or vital signs from an EHR (Electronic Health Record) interface. When certain rule-based criteria are met – for example, concurrent administration of two opposing medications, or a critical lab value indicating high risk during an infusion – the engine generates a conflict alert object. Each conflict record may contain details like involved therapy/pump identifiers, the reason for alert, severity level, and a recommended action (e.g. “suggested action: stop infusion”). The conflict is initially flagged with an “Active” status (unacknowledged) and time-stamped.

Once a conflict is generated, the Notification & User Interface module (102 in FIG. 1) routes the alert to the appropriate clinician's device. In one embodiment, this involves using proximity data to find the nearest qualified caregiver and pushing an alert to their smartwatch or mobile device. The alert message includes transparent information about the condition (for example: "High potassium level while on potassium infusion – risk of hyperkalemia"), enabling the clinician to quickly grasp why the alert was raised. Crucially, at this stage the system does not attempt any automatic intervention on the pump – it purely notifies the human operator, consistent with a non-device CDS approach. The clinician's device displays options such as "Acknowledge", "Override", or "Escalate".

- Acknowledge indicates the clinician has received the alert and intends to take appropriate action (e.g. they will go check the patient or adjust the pump manually).

- Override indicates the clinician is consciously bypassing the recommendation (for instance, choosing to continue the infusion despite the alert, perhaps for justified clinical reasons).

- (Escalate would forward the alert to another or higher-level clinician, but for this invention's core logic, we focus on acknowledge/override pathways.)

When the clinician taps Acknowledge or Override, the User Interface module sends this response back to the SmartStop server. The system then invokes its Audit Logging subsystem to record the event. Each acknowledgment or override is recorded as an immutable log entry containing the conflict ID, user ID, action type, and timestamp . For example, an "ACK" event might be logged as: {conflict #123, event: ACKNOWLEDGE, user: NurseJones, time: 2025-11-17T14:32:20Z}. The audit log is preferably implemented as an append-only ledger – for instance, using a blockchain or hash-chain technique – so that entries cannot be altered or removed once written . This ensures a tamper-evident record of who responded to the alert and when, which is valuable for both clinical oversight and regulatory compliance. It is important to note that acknowledging an alert in SmartStop has no direct effect on the infusion device or on the patient's therapy parameters; it merely updates the system's records and notification workflow . The infusion pump will continue running as it was, and if it was alarming, the clinician would still need to silence or address that alarm on the device itself. SmartStop's non-device mode thus preserves the existing device behavior and only adds a parallel layer of communication and logging.

After logging the acknowledgment, the system transitions the conflict's state to "Acknowledged" (meaning a user has seen it) but notably not "Resolved." In the SmartStop database, for example, the conflict may have a status field that remains "0" or "active" rather than "1" or "closed." The user interface might indicate this by showing the alert as acknowledged (perhaps with an icon or label) yet still requiring resolution. This intermediate state signals that the ball is now in the clinician's court to follow through with corrective action, and it also prevents duplicate alerts: SmartStop's logic can suppress further redundant alarms for the same issue once acknowledged, avoiding alarm fatigue while the clinician works on it.

5.2 Conflict Resolution Tracking Algorithm

The core innovation of SmartStop lies in the Conflict Resolution Engine, which runs a state-transition algorithm to determine when an alert can truly be marked resolved. Unlike conventional systems that mark an alert resolved as soon as it's acknowledged, SmartStop keeps the conflict open until an objective resolution criterion is met. This criterion generally is: the hazardous condition that triggered the alert is no longer present. The system evaluates this by monitoring two main factors in the conflict record: therapy state and risk state.

- Therapy State: For any conflict tied to an infusion, the system tracks the status of the relevant pump(s) or medication infusion. This can be done by subscribing to pump telemetry or polling the infusion management system. The key parameter is typically the infusion rate (or an on/off status). The Conflict Resolution Engine checks if the infusion rate has been set to zero or if the pump has been stopped for all therapies involved in the conflict. If the conflict involves multiple infusions (e.g. two interacting drugs), it will check all associated pumps. Only if all implicated infusions are halted (rate = 0) does the therapy state indicate safety. If even one pump is still running above zero rate, the therapy state condition for resolution is false .

- Risk State: Some conflicts involve patient conditions (lab values, vital signs) that may take time to normalize even after therapy is adjusted. The engine also monitors these relevant patient parameters. For example, if the alert was triggered by a high potassium level (hyperkalemia) while a potassium infusion was running, stopping the infusion is one step, but the conflict might also optionally track if the potassium lab value falls back into normal range. In many cases, halting the offending therapy will eventually resolve the physiological risk, but the design can include checks like "if latest lab result for potassium is back under threshold." In general, the conflict remains unresolved until either the therapy is stopped or the risky condition itself clears (whichever is appropriate for the given rule).

The Conflict Resolution Engine may run periodically (e.g. every minute) or be event-driven (triggered by updates in pump status or lab results). FIG. 2 (state flow) provides a high-level view of this logic loop. Below is a pseudocode representation of the conflict state management algorithm, integrating the above concepts:

```
on ConflictTriggered(conflict):
   conflict.status = ACTIVE   // Unacknowledged, unresolved
   notifyClinician(conflict)

on UserResponse(conflict, action):
   if action in {ACKNOWLEDGE, OVERRIDE}:
      logAckEvent(conflict.id, action, user, timestamp)
      conflict.ack_state = action       // mark as acknowledged (or overridden)
      // Do NOT change conflict.status to resolved here
```

```
        // (conflict.status remains ACTIVE/0, indicating unresolved)
        stopEscalationTimers(conflict)    // prevent further alerts/escalation for this conflict

// Separate monitoring process (runs periodically or on data update):
process UnresolvedConflicts():
    for each conflict in conflicts where conflict.status != RESOLVED:
        checkTherapy = all_infusion_rates_zero(conflict.related_therapies)
        checkRisk   = conflict.risk_factor_resolved()  // e.g., latest lab back to normal
        if checkTherapy or checkRisk:
            // Resolution criteria met
            conflict.status = RESOLVED    // Mark conflict as resolved
            conflict.resolution_time = now()
            appendResolutionNote(conflict, "Resolved by condition cleared at " + now())
            logEvent(conflict.id, "RESOLVED", systemUser, now())
        else:
            // Still unresolved, remain in acknowledged state if previously acknowledged
            continue monitoring (conflict remains active)
```

In the above logic, when a conflict is first triggered, it is set to ACTIVE and a notification is sent out. The clinician's response via UserResponse leads to logging an acknowledgment or override. Notably, the conflict's status is not set to RESOLVED at that point – it stays in an unresolved state, though we might record an ack_state for UI purposes (to indicate it's been acknowledged). The system stops any further escalation alerts for that conflict once it's acknowledged, under the assumption that a human is addressing it. However, the conflict remains in the monitoring loop of UnresolvedConflicts().

The monitoring process iterates over all active conflicts and evaluates if the resolution conditions are satisfied. The function all_infusion_rates_zero() checks the latest pump readings for each therapy involved in the conflict . SmartStop can obtain these readings via integrations with pump data (for example, reading a therapy_readings database or an API that gives current rates). If every relevant infusion's rate is 0 (or the pump channel is off), checkTherapy becomes true. Separately, risk_factor_resolved() is a placeholder for checking patient data depending on the conflict type – for some conflicts this may always return true if not applicable, or it may check things like "most recent lab < threshold" or "heart rate back to safe range," etc. If either condition is true (therapy off or risk gone), the conflict is considered objectively resolved. The engine then updates the conflict's status to RESOLVED (e.g. setting a database field from 0 to 1 for that record) and timestamps it. It also appends a note to the conflict's resolution field indicating how it was resolved, for example: "Auto-resolved by pump state at 14:40:00 (all therapies off)" . Additionally, the system logs a "RESOLVED" event to the audit trail, marking the official closure of the alert. At this point, in the UI, the conflict would move to a resolved list or get a "resolved" badge, etc.

If the resolution checks fail (meaning the infusion is still running or the condition still abnormal), the conflict remains active. The system can continue to monitor it. Optionally, if a conflict remains acknowledged but unresolved beyond a certain duration, the system could escalate the

issue (for instance, notify a supervisor or, in the device-integrated version, initiate a failsafe pump stop). In the primary non-device mode, escalation might involve sending a reminder or alert to additional personnel that "Conflict X was acknowledged but therapy not stopped after Y minutes." Such policies can be configured to ensure no critical alert is forgotten after acknowledgment. The key distinction, however, is that SmartStop will not by itself stop the pump or declare the situation safe until it has evidence of actual risk mitigation. This contrasts with prior art where an alarm silence or acknowledgment might prematurely conclude the alert workflow.

5.3 Separation of Acknowledgment and Resolution – Novelty and Advantages

By dividing the alert lifecycle into these two layers – user acknowledgment vs. system-verified resolution – the SmartStop system introduces a new safety paradigm in clinical decision support. In traditional pump alarm workflows, once a nurse presses the silence button or acknowledges an alert in software, that alert often disappears from active view. The pump might still be infusing a dangerous medication, but the system assumes the clinician will handle it, and no further tracking of that specific alert occurs. SmartStop, on the other hand, continues to track the alert post-acknowledgment. This layered architecture (alerting layer vs. resolution layer) is novel in the CDS context and yields several advantages:

- Improved Patient Safety: The patient remains protected because the system ensures closure. Even if a busy clinician gets distracted after acknowledging, SmartStop can remind or escalate since the conflict stays unresolved in the system. The risk of an alert being lost in the shuffle is reduced.

- Accountability and Auditability: Every step is logged. The immutable audit trail captures the exact timeline: when the alert fired, who acknowledged it, and when the issue was actually resolved . This provides transparency for quality improvement and legal/regulatory review. For example, in an adverse event investigation, the hospital can see that an alert was acknowledged at a certain time but the infusion wasn't stopped until much later – highlighting process gaps.

- Workflow Compatibility: The system achieves the above without adding undue burden or device integration. Clinicians interact with SmartStop through simple notifications and buttons. The pump continues to function normally; SmartStop doesn't interfere with pump controls or require any special hardware modifications in the non-device mode. This means hospitals can deploy SmartStop as a software-only safety layer on top of existing infusion pumps (which remain unaltered and FDA-approved on their own). SmartStop qualifies as a Non-Device CDS tool by meeting criteria such as providing transparent rule-based advice and leaving final control to the human user .

- Alarm Fatigue Mitigation: Because SmartStop logs the acknowledgment and can suppress duplicate alerts for the same ongoing issue, clinicians aren't bombarded with

the same alarm repeatedly while they are addressing it. For example, if a pump would normally alarm every few minutes for a threshold breach, SmartStop can act as an intermediary by recording that "Nurse is aware" and not re-notifying unless the condition worsens or a set time passes. This adaptive throttling of alerts (acknowledge → pause alerts) can be built into the rules, as long as the conflict remains active in the background (some implementations may include a cool-down period as a dependent feature).

To illustrate the layered operation, consider a use case: A patient is on a high-dose vasopressor infusion and develops an arrhythmia indicated by telemetry. SmartStop triggers a conflict alert recommending to check the infusion. The nurse on duty hits "Acknowledge" on her smartwatch, logging that she's aware. She then assesses the patient. If she determines the infusion should indeed be stopped and clamps the IV line at 14:40, the pump's data (or a manual input in SmartStop) soon reflects rate = 0. SmartStop detects this and at 14:41 marks the conflict resolved. If, however, the nurse decides to continue the infusion (perhaps the arrhythmia is minor or due to a different cause) – she might have chosen "Override" when acknowledging. The conflict stays active in SmartStop despite the override. The system might not remind that same nurse again immediately (since she overrode it), but the override is recorded, and the conflict could be escalated to a physician if the dangerous condition persists. In any case, SmartStop would only resolve the conflict if the arrhythmia subsides or the infusion is later stopped. This ensures that an override decision doesn't simply hide the problem; it remains visible for follow-up.

In summary, the detailed operation of SmartStop underscores a fundamental principle: logging user intent is separated from enforcing patient safety outcomes. The invention achieves this through a combination of software modules and algorithms that monitor, log, and update conflict states accordingly. This approach can be applied broadly to many CDS scenarios, not only infusion pumps – any alert that a clinician can acknowledge can benefit from having a monitored "closure" condition. SmartStop's design is particularly tailored to infusion therapy because of the high stakes of continuous drug delivery and the regulatory opportunity to intervene (or not) in a layered fashion. By maintaining a clear boundary between the non-device alert handling and any device control (which is relegated to an optional secondary mode), SmartStop provides a flexible platform: it can function purely as an intelligent alarm mediator, or be extended into a closed-loop safety system if regulatory clearance is obtained.

For the purposes of this patent specification, all descriptions have focused on the standalone Non-Device CDS mode of SmartStop. In this mode, the invention does not perform any automatic pump stops or adjustments – it relies on the clinician to carry out interventions, while the system orchestrates communication and verification. This yields a novel and useful system that enhances infusion safety and documentation without stepping into regulated device functionality. The following section outlines exemplary system and method claims that capture the inventive concepts of SmartStop.