Figures

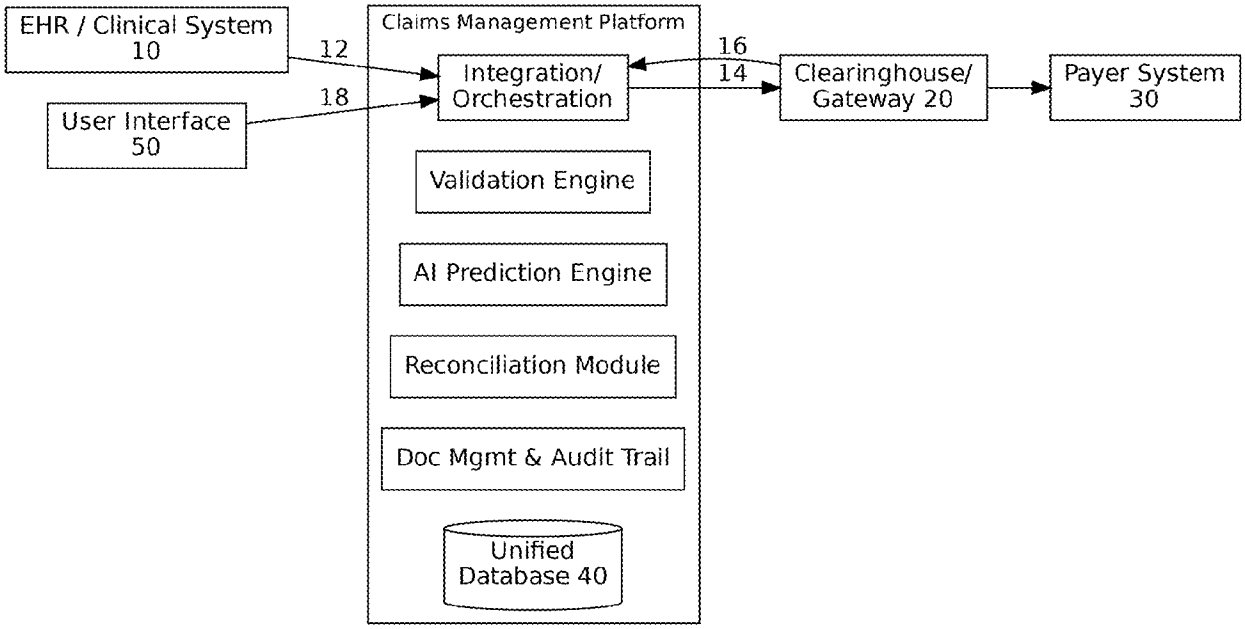

FIG. 1

Figure 1: High-Level System Architecture

Figure 1: A high-level system architecture diagram of the AI-powered claims management platform. The platform (center) interfaces with an external clinical system (EHR 10) on the left via a FHIR/HL7 data feed (arrow 12) and connects to payer systems (30) on the right through a clearinghouse gateway (20) for electronic claims submission (arrow 14) and response retrieval (arrow 16). The platform's internal modules – including a validation engine, AI prediction engine, reconciliation module, and document/audit trail manager – operate on a unified database (40) and are contained within the central box. Users interact through a web-based ***User Interface*** *(50), which communicates with the platform (arrow 18) to display real-time validations, alerts, and recommendations. All components are shown as labeled rectangles, and data flows are indicated by arrows with reference numerals corresponding to those in the patent description.*

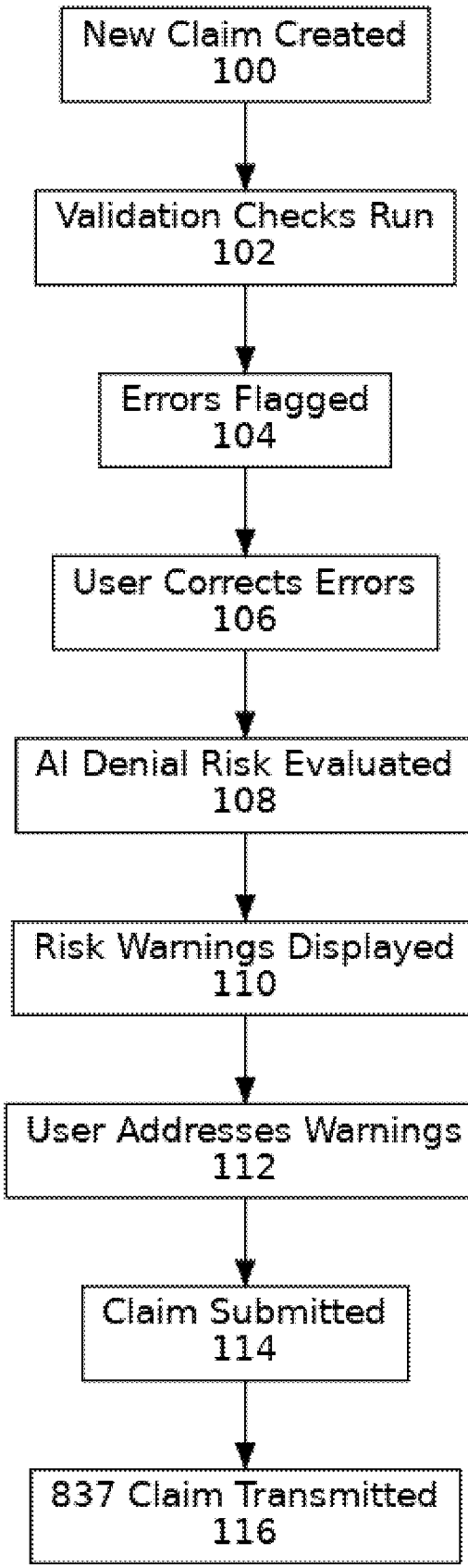
New Claim Created
100
Validation Checks Run
102
Errors Flagged
104
User Corrects Errors
106
AI Denial Risk Evaluated
108
Risk Warnings Displayed
110
User Addresses Warnings
112
Claim Submitted
114
837 Claim Transmitted
116

FIG. 2A

Figure 2A: Pre-Submission Workflow (User Interface Perspective)

*Figure 2A: A pre-submission workflow diagram illustrating how a claim is processed **before** it is sent to the payer. The steps, labeled by reference numerals 100–116, represent user and system actions in sequence. First, a **new claim** is created (100). The platform's **validation engine** immediately runs automated rule checks against payer requirements (102). If any errors or missing data are found, the system **flags the issues** (104) and prompts the user to correct them. The billing user then **corrects the errors** or enters missing information in the claim form (106). Next, the platform's **AI denial prediction engine** evaluates the claim for risk of denial (108), leveraging historical data patterns. The AI outputs a **risk warning or suggestion** (110) – for example, recommending an attachment or code modification. The user then **addresses these AI-driven warnings** by adding documentation or making adjustments (112). Once the claim is fully validated and optimized, the user **submits the claim** (114). The platform converts the claim into the standard X12 837 format and **transmits** it to the clearinghouse/payer (116). All these steps occur through the user interface in real time, with every action and outcome logged in the audit trail.*

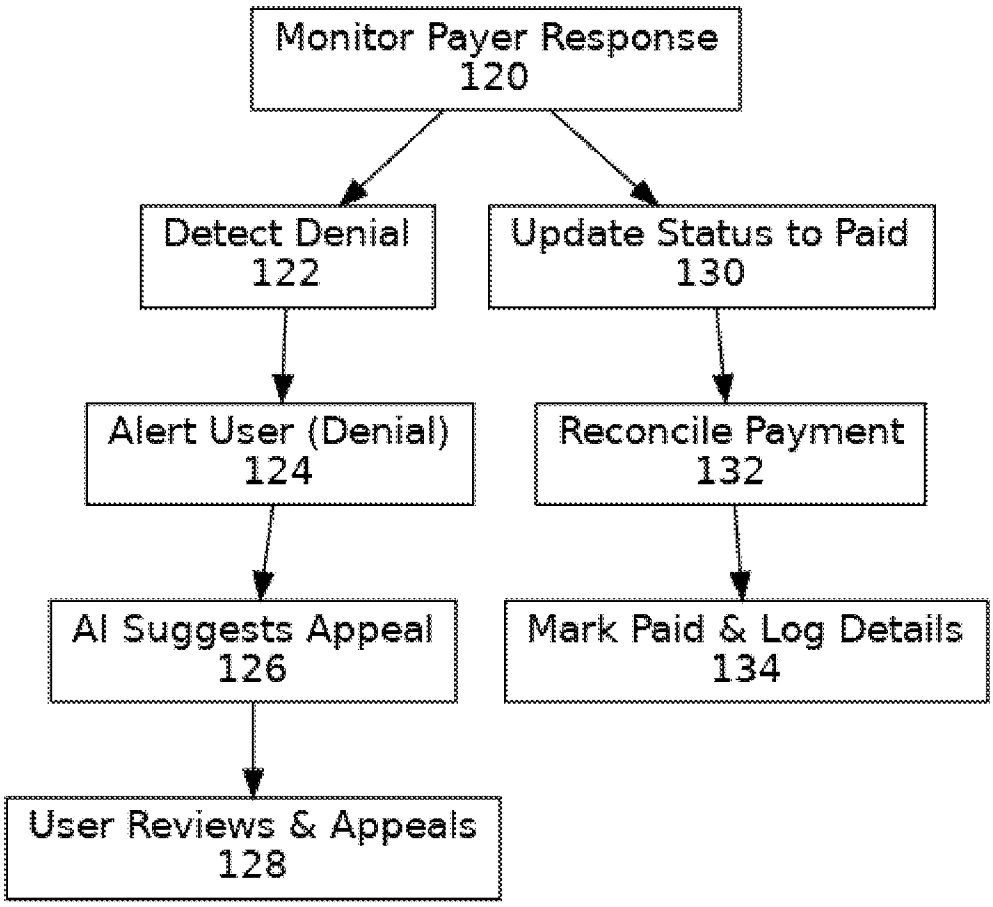

FIG. 2B

Figure 2B: Post-Submission Workflow (Denial Handling and Resolution)

Figure 2B: A post-submission workflow diagram illustrating how the system handles payer responses after a claim is submitted. The process begins with the platform ***monitoring for payer responses*** *(120) such as claim status updates (277) or remittance advice (835) messages. The flow then splits into two branches: one for a* ***denial scenario*** *(left branch) and one for a* ***clean approval*** *(right branch). In the denial branch, if a claim is denied, the system* ***detects the denial****automatically from the electronic response (122) and immediately* ***alerts the user*** *to the denial and its reason (124). The AI module then* ***suggests an appeal or remedy*** *(126) tailored to the denial reason – for example, preparing an appeal letter with supporting documentation if a missing document caused the denial. The user* ***reviews the AI's suggestions and appeal packet*** *(128) and can make corrections or add information before resubmitting the claim or sending the appeal through the platform. In the approval branch (right side), if the claim is accepted and paid without issues, the system* ***updates the claim status to paid*** *(130) and* ***reconciles the payment*** *automatically (132) by recording the payment details from the 835 remittance. The claim is then marked as paid and all details are logged (134),*

concluding the workflow. This figure highlights the platform's ability to handle both outcomes: streamlining denial management with AI-assisted appeals, and auto-posting payments for approved claims.

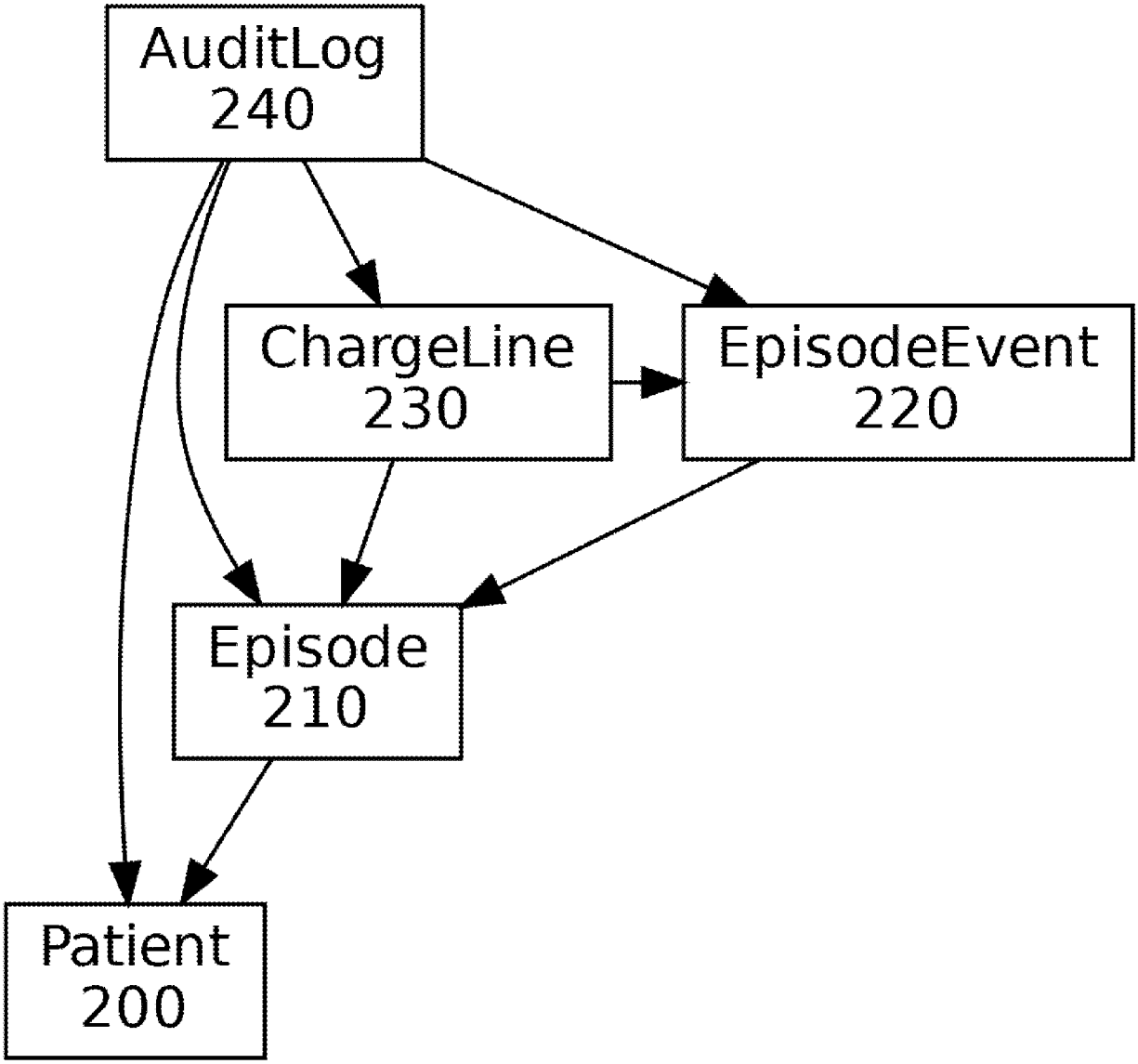

FIG. 3

Figure 3: Data Model – Patient, Episode, Event, Charge, AuditLog

Figure 3: An example data structure diagram showing key entities in the platform's unified database and their relationships. Each rectangle represents a data object with its reference numeral: ***Patient*** *(200),* ***Episode*** *(210),* ***EpisodeEvent*** *(220),* ***ChargeLine*** *(230), and* ***AuditLog*** *(240). A* ***Patient (200)*** *may have multiple associated* ***Episodes (210)*** *(each episode representing a span of care, such as a home infusion therapy episode). Each Episode links back to its Patient (indicated by the arrow from Episode 210 to Patient 200). Within an episode, various* ***clinical events*** *are recorded as EpisodeEvent entries (220) – for example, a nursing visit or a medication administration – and the diagram shows EpisodeEvents pointing to the Episode they belong to.* ***ChargeLine (230)*** *objects represent individual billable items (charges) and are associated with an Episode as well (arrow from ChargeLine to Episode). A ChargeLine can also optionally link to a specific EpisodeEvent that justifies that charge (arrow from ChargeLine 230 to EpisodeEvent 220), tying the billed item to its clinical context. The* ***AuditLog (240)*** *is an immutable log of all significant actions or changes; it stores references to the entities involved in each event. Accordingly, the AuditLog box has arrows pointing to Patient, Episode, EpisodeEvent, and ChargeLine, indicating that audit records may reference any of these (for instance, logging which user modified a charge or when an episode was created). This unified data model ensures traceability from each charge back to clinical events and provides a comprehensive audit trail for compliance.*

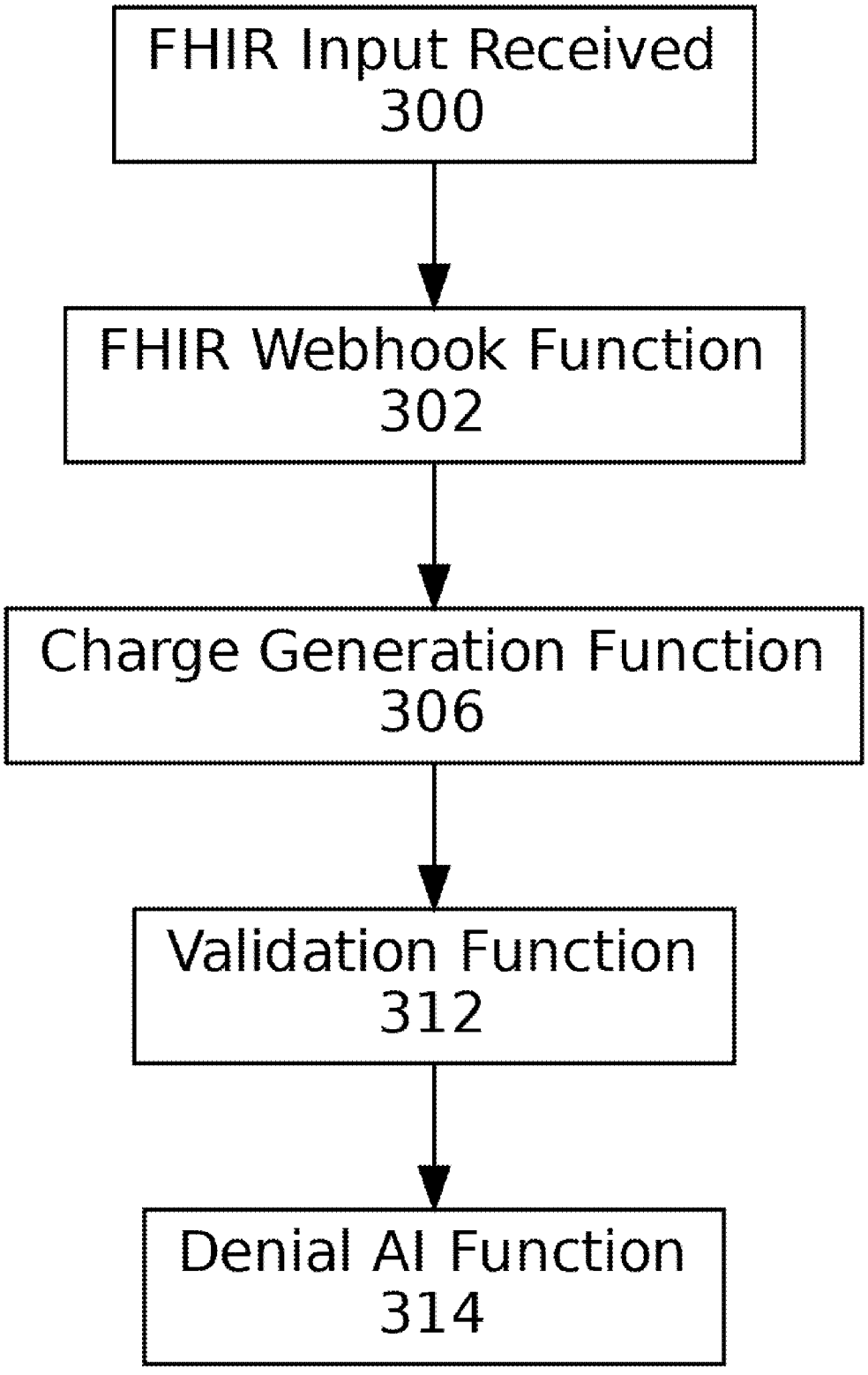
FHIR Input Received
300
FHIR Webhook Function
302
Charge Generation Function
306
Validation Function
312
Denial AI Function
314

FIG. 4

Figure 4: Orchestration of Serverless Functions (Edge Pipeline)

*Figure 4: A flow diagram illustrating the **event-driven orchestration** of the platform's microservice (serverless) functions. The process starts when new clinical data is received via the FHIR API – shown as **FHIR Input Received**(block 300). This triggers the **FHIR Webhook Function** (302), which parses the incoming FHIR/HL7 data (e.g. patient info, encounters, orders) and maps it into the platform's internal records. Upon completion, the next step is the **Charge Generation Function** (306), which is invoked to automatically create draft billing entries (ChargeLines) based on the clinical events (for example, generating appropriate HCPCS codes for recorded procedures or medications). After charges are generated, the **Validation Function** (312) is triggered. This function applies payer-specific rules and edits, "scrubbing" the claim by checking for missing or incorrect data, and flagging any issues or making minor corrections. Finally, the pipeline calls the **Denial AI Function** (314), which runs an AI model on the validated claim to predict the risk of denial and attach any recommended preventive actions (such as adding documentation). Each function in this chain runs independently and is event-triggered by the completion of the previous one, forming a streamlined pipeline from initial data ingestion (block 300) through charge creation (306), rule validation (312), and AI risk scoring (314). The figure emphasizes how these discrete modules are orchestrated sequentially to process a claim in near-real-time.*

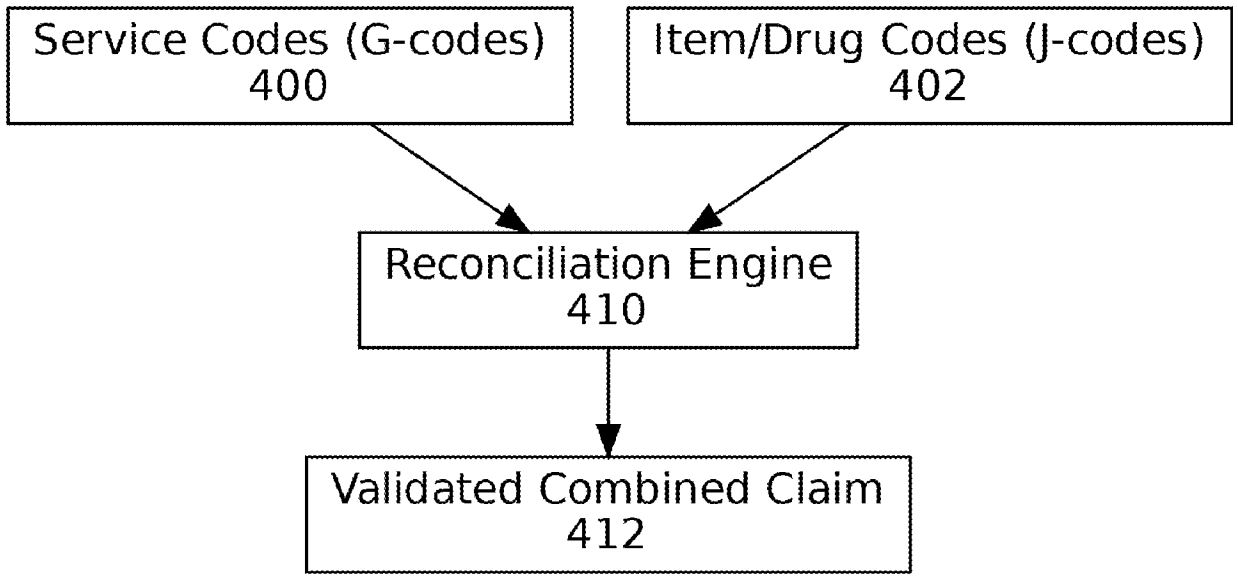

FIG. 5

Figure 5: Reconciliation Engine for Time-Linked Codes

Figure 5: A schematic diagram of the platform's ***reconciliation engine*** *(410) that ensures required code pairings are present within a specified time window (here, a 30-day home infusion episode). The inputs to the engine are two sets of billing codes:* ***Service codes (G-codes)*** *400 (e.g. codes for nursing visits in home infusion therapy) on the left, and* ***Item/Drug codes (J-codes)*** *402 (e.g. codes for the infused medication or supplies) on the right. Both sets of codes, detected for a given patient and episode, feed into the* ***Reconciliation Engine*** *(block 410). The engine applies configured rules to* ***match each G-code with its corresponding J-code*** *(and vice versa) within the allowed timeframe. If a service code is missing its required drug code (or vice versa), the engine flags the discrepancy. In some cases, it can even auto-generate a placeholder for a missing code if sufficient clinical evidence exists. The output of this process is a* ***validated combined claim*** *(412) that contains all necessary code pairings. The figure shows the two input code groups merging into the reconciliation module, and a single validated claim output emerging. By enforcing code pair rules (for example, ensuring that a nursing visit code is linked to a drug supply code within 30 days), the reconciliation engine helps prevent billing errors and denials due to incomplete code sets.*