PTO/SB/16 (01-25)
Approved for use through 11/30/2027. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

PROVISIONAL APPLICATION FOR PATENT COVER SHEET – Page 1 of 2

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c).

Priority Mail Express® Label No. _____

INVENTOR(S)		
Given Name (first and middle [if any])	Family Name or Surname	Residence (City and either State or Foreign Country)
Dmitry	Pokhilko	San Francisco, California
Harvey	Sax	North Salt Lake, Utah
Kentaro	Vadney	Alameda, California

Additional inventors are being named on the _____ separately numbered sheets attached hereto.

TITLE OF THE INVENTION (500 characters max):

MULTI-TENANT FINANCIAL PLANNING PLATFORM WITH AGENT ORCHESTRATION AND PROVENANCE-AWARE DATA PROCESSING

Direct all correspondence to:

CORRESPONDENCE ADDRESS

[X] The address corresponding to Customer Number: 04743

OR

[ ] Firm or Individual Name

Address

City	State	Zip
Country	Telephone	Email

ENCLOSED APPLICATION PARTS *(check all that apply)*

[X] Application Data Sheet. See 37 CFR 1.76.

[ ] CD(s), Number of CDs _____

[X] Drawing(s) *Number of Sheets* 7

[X] Other (specify) Exhibits A, B, C

[X] Specification (e.g., description of the invention) *Number of Pages* 33

Fees Due: Filing Fee of $325 ($130 for small entity) ($65 for micro entity). If the specification and drawings exceed 100 sheets of paper, an application size fee is also due, which is $450 ($180 for small entity) ($90 for micro entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

METHOD OF PAYMENT OF THE FILING FEE AND APPLICATION SIZE FEE FOR THIS PROVISIONAL APPLICATION FOR PATENT

[X] Applicant asserts small entity status. See 37 CFR 1.27.

[ ] Applicant certifies micro entity status. See 37 CFR 1.29.
Applicant must attach form PTO/SB/15A or B or equivalent.

[ ] A check or money order made payable to the *Director of the United States Patent and Trademark Office* is enclosed to cover the filing fee and application size fee (if applicable).

[X] Payment by credit card. Form PTO-2038 is attached.

130.00

TOTAL FEE AMOUNT ($)

[X] The Director is hereby authorized to charge the filing fee and application size fee (if applicable) or credit any overpayment to Deposit Account Number: 13-2855 .

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

PTO/SB/16 (01-25)
Approved for use through 11/30/2027. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

PROVISIONAL APPLICATION FOR PATENT COVER SHEET – Page 2 of 2

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government. (NOTE: Providing this information on a provisional cover sheet, such as this Provisional Application for Patent Cover Sheet (Form PTO/SB/16), does not satisfy the requirement of 35 U.S.C. 202(c)(6), which requires that the *specification* contain a statement specifying that the invention was made with Government support and that the Government has certain rights in the invention.)

[X] No.

[ ] Yes, the invention was made by an agency of the U.S. Government. The U.S. Government agency name is:

_____

[ ] Yes, the invention was made under a contract with an agency of the U.S. Government.

The contract number is: _____

The U.S. Government agency name is: _____

_____

In accordance with 35 U.S.C. 202(c)(6) and 37 CFR 401.14(f)(4), the specifications of any United States patent applications and any patent issuing thereon covering the invention, including the enclosed provisional application, must state the following:

''This invention was made with government support under [IDENTIFY THE CONTRACT] awarded by [IDENTIFY THE FEDERAL AGENCY]. The government has certain rights in the invention."

WARNING:

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

SIGNATURE /Matthew R. Carey/ DATE June 12, 2026

TYPED OR PRINTED NAME Matthew R. Carey REGISTRATION NO. *(if appropriate)* 61,082

TELEPHONE (312) 474-6300 DOCKET NUMBER 34280/70000P

.

MULTI-TENANT FINANCIAL PLANNING PLATFORM WITH AGENT ORCHESTRATION AND PROVENANCE-AWARE DATA PROCESSING

FIELD

[0001] The present disclosure relates to multi-tenant computing platforms for financial data management, and more particularly to systems and methods for secure data isolation with immutable audit logging, intelligent agent orchestration with selective module re-execution, and provenance-aware data processing with institutional data portability.

BACKGROUND

[0002] Multi-tenant computing platforms that manage financial data for multiple organizations face technical challenges related to data isolation, audit integrity, computational efficiency, and data portability. These platforms may process sensitive financial information including user profiles, transaction records, and planning documents across organizational boundaries while maintaining compliance with regulatory retention requirements.

[0003] Conventional multi-tenant database systems typically implement data isolation through application-layer filtering, wherein queries are modified at the application level to append tenant identifiers to database operations. This approach introduces latency overhead because each query is intercepted, parsed, and modified before execution. Additionally, application-layer filtering creates security vulnerabilities because isolation logic resides in application code that can be bypassed through direct database access or through defects in the filtering logic. Conventional audit logging mechanisms in such systems typically write audit records within the same transaction context as the audited operation, which may permit application-level code to modify or delete audit entries, thereby compromising audit integrity.

[0004] Conventional systems that coordinate multiple processing modules in financial planning applications typically employ monolithic architectures that re-execute all modules upon any data change. This approach results in excessive computational resource consumption and increased response latency, particularly when user modifications affect a limited subset of input categories. Furthermore, conventional data portability mechanisms employ bulk export approaches that either transfer all user data or none, without accounting for regulatory

requirements that mandate retention of historical records at originating institutions while routing new data to receiving institutions. Such approaches fail to preserve data provenance or may require physical duplication of data across institutions, increasing storage overhead and complicating compliance with retention policies.

[0005] Therefore, there is an opportunity for systems and methods that offer improved multi-tenant data management with database-layer isolation, immutable audit logging, selective module re-execution, and provenance-aware data portability.

SUMMARY

[0006] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the detailed description. This summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

[0007] In one embodiment, a computer-implemented method for multi-tenant data isolation with immutable audit logging is provided, the method comprising: receiving, by a server, a request from a client device, the request comprising an authentication token identifying a user; extracting, by middleware executing on the server, an identifier of an organization associated with the user from the request; setting, within a database session associated with the request, a session-scoped variable to the identifier of the organization, wherein the database enforces a set of row-level security policies that restrict query results to rows matching the session-scoped variable; processing the request to generate a data operation on a tenant-scoped table, wherein the set of row-level security policies cause the database to return a set of rows associated with the identifier of the organization; generating an audit event corresponding to the data operation; and writing the audit event to an audit log by creating a savepoint within a database transaction, switching to a write-only database role authorized to insert records into the audit log, inserting the audit event into the audit log, reverting to a prior database role, and releasing the savepoint.

[0008] In one embodiment, a computer-implemented method for coordinating a plurality of specialist processing modules in a financial planning system is provided, the method comprising: receiving, by a server, a user input that modifies a set of data fields associated with a user profile; mapping, by a coordinator module of the server, each data field of the modified set of data fields

to at least one input category based on a stored mapping; determining, by an invalidation engine of the server, a subset of the plurality of specialist processing modules to execute, including selecting, for the subset, any of the plurality of specialist processing modules whose associated input categories overlap with the at least one input category, and retaining cached results for any remaining of the plurality of specialist processing modules; executing, by the server, each of the subset of the plurality of specialist processing modules to generate a set of updated outputs; receiving, from each executed specialist processing module, at least one symbolic policy signal, each of the at least one symbolic policy signal comprising a machine-readable boolean value tagged with a semantic label; evaluating, by the server, each of the at least one symbolic policy signal against a set of conflict rules, each of the set of conflict rules defining a pair of symbolic policy signals from two different specialist processing modules of the plurality of specialist processing modules that represent contradictory recommendations; and responsive to detecting a conflict, generating, by the server, a structured conflict object identifying the contradictory recommendations and presenting a set of resolution options to the user.

[0009] In one embodiment, a computer-implemented method for searching and retrieving records from an audit-class archive is provided, the method comprising: receiving, by a server from a requesting client device, a search query and a redaction mode indicator; executing, by the server, the search query against a keyword retrieval index and a semantic retrieval index in parallel to produce a keyword-ranked result list and a semantic-ranked result list; computing, by the server, a fused result list by applying a reciprocal rank fusion function to the keyword-ranked result list and the semantic-ranked result list, the reciprocal rank fusion function assigning a score to each result appearing in at least one of the keyword-ranked result list and the semantic-ranked result list based on a sum of inverse rank positions across the keyword-ranked result list and the semantic-ranked result list; selecting, based on the redaction mode indicator, between a first redaction strategy and a second redaction strategy, wherein the first redaction strategy replaces detected personally identifiable information with randomized placeholders that vary per session, and the second redaction strategy replaces detected personally identifiable information with deterministic hashed placeholders that are consistent across sessions; applying the selected redaction strategy to the fused result list to produce a redacted result list; and returning the redacted result list to the requesting client device.

[0010] In one embodiment, a computer-implemented method for routing user messages across a plurality of persona-specific computational subgraphs while maintaining conversation continuity is provided, the method comprising: receiving, by a server, a user message associated with a conversation session, the conversation session storing a persistent message history; classifying, by a deterministic intent classifier of the server, the user message into an intent category by evaluating the user message against a set of keyword-matching rules, wherein the deterministic intent classifier routes the user message responsive to a match of the set of keyword-matching rules; selecting, based on the intent category, a persona-specific computational subgraph from the plurality of persona-specific computational subgraphs, the persona-specific computational subgraph having been compiled with a bound tool set at compile time, the bound tool set defining a set of tools the persona-specific computational subgraph is authorized to invoke; loading the persistent message history from the conversation session; invoking the selected persona-specific computational subgraph with the persistent message history, a system prompt corresponding to the selected persona-specific computational subgraph, and the bound tool set of the selected persona-specific computational subgraph; and appending a response generated by the selected persona-specific computational subgraph to the persistent message history of the conversation session, whereby a subsequent user message classified into a different intent category is processed by a different persona-specific computational subgraph using the same persistent message history.

[0011] In one embodiment, a computer-implemented method for processing heterogeneous data inputs with provenance-aware conflict resolution is provided, the method comprising: receiving, by a server, a plurality of data field values for a user profile, each data field value received from one of a plurality of input modalities; associating, by the server, each data field value of the plurality of data field values with a provenance tier from a ranked set of provenance tiers, each provenance tier corresponding to the input modality from which the data field value was received; resolving, by the server, conflicts among data field values received for a same data field from different input modalities by applying a tier-based precedence rule that retains the data field value associated with a higher-ranked provenance tier; storing the resolved data field values in a canonical field store as a unified representation of the user profile; applying, by the server, a set of per-asset-class liquidity weights to asset values within the resolved data field values, each per-asset-class liquidity weight reflecting a time-to-liquidation characteristic of a corresponding asset class; computing, by the server, a set of dimension scores across a plurality of scoring

dimensions using the asset values with the set of per-asset-class liquidity weights applied thereto and the resolved data field values; and generating a composite score for the user profile from the set of dimension scores.

[0012] In one embodiment, a computer-implemented method for transferring a user record between a first institution and a second institution while preserving regulatory retention of historical data is provided, the method comprising: receiving, by a server, a portability request from a user device to transfer the user record from the first institution to the second institution; recording a switch event associated with the user record, the switch event comprising an effective timestamp, an identifier of the first institution, and an identifier of the second institution; resolving, by request middleware of the server, subsequent requests associated with the user record to the second institution based on the switch event, wherein data generated after the effective timestamp is associated with the second institution; maintaining, by the server, read access to data generated before the effective timestamp, the data generated before the effective timestamp remaining stored at the first institution and encrypted under an encryption key of the first institution; and generating a portable export bundle comprising the user record, the portable export bundle being signed under a signing key of the first institution and encrypted to a transport key of the second institution.

[0013] The foregoing general description of the illustrative embodiments and the following detailed description thereof are merely exemplary aspects of the teachings of this disclosure and are not restrictive.

BRIEF DESCRIPTION OF FIGURES

[0014] Non-limiting and non-exhaustive examples are described with reference to the following figures.

[0015] FIG. 1 illustrates a high-level architecture of a system for a multi-tenant financial planning platform, according to aspects of the present disclosure.

[0016] FIG. 2 illustrates a method for multi-tenant data isolation with immutable audit logging, according to aspects of the present disclosure.

[0017] FIG. 3 illustrates a method for coordinating a plurality of specialist processing modules in a financial planning system, according to aspects of the present disclosure.

[0018] FIG. 4 illustrates a method for searching and retrieving records with provenance-aware redaction, according to aspects of the present disclosure.

[0019] FIG. 5 illustrates a method for routing user messages across persona-specific computational subgraphs while maintaining conversation continuity, according to aspects of the present disclosure.

[0020] FIG. 6 illustrates a method for processing heterogeneous data inputs with provenance-aware conflict resolution, according to aspects of the present disclosure.

[0021] FIG. 7 illustrates a method for transferring a user record between institutions while preserving regulatory retention of historical data, according to aspects of the present disclosure.

DETAILED DESCRIPTION

[0022] The following description sets forth exemplary aspects of the present disclosure. It should be recognized, however, that such description is not intended as a limitation on the scope of the present disclosure. Rather, the description also encompasses combinations and modifications to those exemplary aspects described herein.

[0023] In various embodiments, the present disclosure relates to systems and methods for providing a multi-tenant financial planning platform that integrates secure data isolation, intelligent agent orchestration, and provenance-aware data processing. The platform may be configured to serve multiple organizations while maintaining strict separation of tenant data, enabling automated coordination of specialist processing modules, and facilitating secure transfer of user records between institutions. The disclosed techniques address technical challenges associated with managing sensitive financial data across organizational boundaries while preserving audit integrity and regulatory compliance.

[0024] In some embodiments, the system may include an authentication gateway configured to receive requests from client applications and validate authentication tokens that identify both a user and an associated organization. Tenant middleware may extract organization identifiers from authenticated requests and establish per-request tenant contexts by setting session-scoped variables within database sessions. The database may enforce row-level security policies that restrict query results to rows matching the session-scoped variable, thereby providing data isolation at the database layer. An audit logging mechanism may write audit events to an immutable audit log

using a savepoint-based approach that temporarily switches to a write-only database role, inserts the audit event, and reverts to a prior role, ensuring that audit records cannot be modified or deleted by application-level code.

[0025] The system may further include an agent orchestration engine configured to coordinate a plurality of specialist processing modules. A coordinator module may map modified data fields to input categories, and an invalidation engine may determine which specialist modules require re-execution based on the modified inputs while retaining cached results for unaffected modules. Each specialist module may generate symbolic policy signals comprising machine-readable boolean values tagged with semantic labels. The system may evaluate these signals against conflict rules to detect contradictory recommendations from different modules and, upon detecting a conflict, may generate structured conflict objects that present resolution options to users.

[0026] In certain embodiments, a search and redaction module may execute search queries against both keyword retrieval indexes and semantic retrieval indexes in parallel, computing fused result lists using reciprocal rank fusion. The module may apply different redaction strategies based on a redaction mode indicator, wherein a first strategy replaces personally identifiable information with randomized placeholders that vary per session and a second strategy uses deterministic hashed placeholders consistent across sessions. The system may also support routing user messages across persona-specific computational subgraphs while maintaining conversation continuity through persistent message histories, with each subgraph having a bound tool set defined at compile time.

[0027] The platform may additionally support processing heterogeneous data inputs from multiple input modalities with provenance-aware conflict resolution. Each data field value may be associated with a provenance tier from a ranked set of tiers, and conflicts among values received for the same field from different modalities may be resolved by applying tier-based precedence rules. The system may compute dimension scores across multiple scoring dimensions using per-asset-class liquidity weights and generate composite scores for user profiles. For institutional transfers, the system may record switch events with effective timestamps, resolve subsequent requests to the appropriate institution, maintain read access to pre-switch data retained at the original institution, and generate portable export bundles signed by the originating institution and encrypted to transport keys of the receiving institution.

[0028] The described systems and methods operate within the technical domain of distributed database systems, cryptographic key management infrastructure, and machine learning-based information retrieval systems. The described systems and methods address technical challenges inherent in managing sensitive data across organizational boundaries within shared computing infrastructure while maintaining data integrity, audit immutability, and computational efficiency.

[0029] Conventional multi-tenant database systems typically implement data isolation through application-layer filtering, wherein queries are modified at the application level to append tenant identifiers to database operations. This approach suffers from several technical limitations. Application-layer filtering introduces latency overhead as each query must be intercepted, parsed, and modified before execution. Additionally, this architecture creates security vulnerabilities because isolation logic resides in application code that may be bypassed through direct database access or through bugs in the filtering logic. Conventional audit logging mechanisms in such systems typically write audit records within the same transaction context as the audited operation, which permits application-level code to modify or delete audit entries, thereby compromising audit integrity. Furthermore, conventional systems that coordinate multiple processing modules typically employ monolithic architectures that re-execute all modules upon any data change, resulting in excessive computational resource consumption and increased response latency.

[0030] The disclosed system improves upon conventional technology by implementing data isolation at the database layer through session-scoped variables and row-level security policies enforced by the database engine itself. This architectural approach eliminates the latency overhead associated with application-layer query modification because the database engine applies security policies during query planning rather than requiring separate interception and modification steps. The session-scoped variable mechanism ensures that tenant context is established once per request and automatically applied to all subsequent database operations within that session, reducing the computational overhead of repeated tenant identifier lookups. The row-level security policies operate at the storage engine level, which prevents unauthorized data access even in scenarios where application-layer controls are bypassed.

[0031] The audit logging mechanism disclosed herein addresses the technical limitation of mutable audit records through a savepoint-based role-switching approach. By creating a savepoint within the database transaction, temporarily switching to a write-only database role that lacks

delete or update permissions on the audit table, inserting the audit record, and then reverting to the prior role, the system ensures that audit records cannot be modified or deleted by application-level code regardless of the permissions held by the application's primary database role. This approach leverages database-level access control mechanisms to enforce immutability without requiring separate audit infrastructure or external logging systems, thereby reducing system complexity and eliminating synchronization overhead between primary and audit data stores.

[0032] The agent orchestration engine disclosed herein improves computational efficiency through selective module re-execution based on input category mapping and invalidation analysis. Rather than re-executing all specialist processing modules upon any data modification, the coordinator module maps modified data fields to input categories, and the invalidation engine determines which modules have dependencies on the modified inputs. Modules whose input categories do not overlap with the modified fields retain their cached results, eliminating redundant computation. This selective invalidation approach reduces processor utilization and memory bandwidth consumption compared to conventional monolithic re-execution architectures. The symbolic policy signal mechanism, wherein each specialist module generates machine-readable boolean values tagged with semantic labels, enables efficient conflict detection through rule-based evaluation rather than requiring computationally expensive natural language processing or semantic analysis of module outputs.

[0033] The search and redaction module implements a hybrid retrieval architecture that executes keyword-based and semantic-based searches in parallel against separate index structures. The reciprocal rank fusion algorithm combines results from both retrieval modalities by computing scores based on inverse rank positions, which improves retrieval accuracy compared to single-modality approaches without requiring computationally expensive re-ranking models. The dual redaction strategy mechanism addresses different technical requirements for data anonymization: the randomized placeholder strategy prevents correlation attacks across sessions by generating different placeholders for the same personally identifiable information in each session, while the deterministic hashed placeholder strategy enables consistent entity tracking across sessions when such tracking is technically required. This configurable redaction architecture allows the system to adapt to different processing requirements without requiring separate processing pipelines.

[0034] The provenance-aware conflict resolution mechanism addresses technical challenges associated with processing heterogeneous data inputs from multiple input modalities. By associating each data field value with a provenance tier and applying tier-based precedence rules, the system resolves conflicts deterministically without requiring user intervention or computationally expensive reconciliation algorithms. The per-asset-class liquidity weights applied during score computation incorporate domain-specific characteristics of different asset types into the scoring algorithm, improving the accuracy of composite scores compared to uniform weighting approaches. The institutional transfer mechanism maintains data integrity across organizational boundaries through cryptographic signing and encryption of portable export bundles, wherein the originating institution's signing key provides authenticity verification and the receiving institution's transport key ensures confidentiality during transfer. The switch event recording mechanism with effective timestamps enables the system to route requests to the appropriate institution while maintaining read access to historical data, addressing regulatory retention requirements without requiring data duplication across institutions.

[0035] FIG. 1 illustrates an example high-level architecture of a system 100 for a multi-tenant financial planning platform. The system 100 may be configured to provide secure data isolation, intelligent agent orchestration, and provenance-aware data processing for multiple organizations while maintaining strict separation of tenant data. The system 100 may include a client-facing tier, a processing tier, and a data tier, each of which may include one or more components that interact to process requests and manage financial data.

[0036] A client application 102 may communicate with an authentication gateway 104 to initiate requests to the system 100. The client application 102 may be implemented as a web application, a mobile application, or another software application executing on a user device. The authentication gateway 104 may receive requests from the client application 102 and validate authentication tokens that identify both a user and an associated organization. In some embodiments, the authentication gateway 104 may verify tokens using asymmetric cryptographic signatures, such as RS256 JSON Web Tokens (JWTs), to bind requests to authenticated principals.

[0037] The authentication gateway 104 may forward authenticated requests to a tenant middleware 106. The tenant middleware 106 may extract organization identifiers from authenticated requests and establish per-request tenant contexts. In some embodiments, the tenant

middleware 106 may set session-scoped variables within database sessions to propagate tenant context to downstream components. The tenant middleware 106 may pass requests to an API server 108, which may route the requests to appropriate processing components based on request type and content.

[0038] The API server 108 may route requests to an agent orchestration engine 110. The agent orchestration engine 110 may coordinate a plurality of processing operations by invoking specialist processing modules 112 and a search and redaction module 116. The agent orchestration engine 110 may map modified data fields to input categories and determine which of the specialist processing modules 112 require execution based on the modified inputs. In some embodiments, the agent orchestration engine 110 may retain cached results for specialist processing modules 112 whose input categories do not overlap with modified fields, thereby reducing redundant computation. The specialist processing modules 112 may generate symbolic policy signals comprising machine-readable boolean values tagged with semantic labels, and the agent orchestration engine 110 may evaluate these signals against conflict rules to detect contradictory recommendations.

[0039] The search and redaction module 116 may execute search queries against both keyword retrieval indexes and semantic retrieval indexes in parallel. The search and redaction module 116 may compute fused result lists using reciprocal rank fusion and may apply different redaction strategies based on redaction mode indicators received with search requests. In some embodiments, a first redaction strategy may replace personally identifiable information with randomized placeholders that vary per session, and a second redaction strategy may replace personally identifiable information with deterministic hashed placeholders consistent across sessions.

[0040] Scheduled data pipelines 114 may support the agent orchestration engine 110 by ingesting data from external data sources 126 and performing batch precomputation operations. In some embodiments, the scheduled data pipelines 114 may include a plurality of precomputed pipelines that execute on different schedules. In some embodiments, the scheduled data pipelines 114 may include a plurality of precomputed pipelines that execute on different schedules, such as daily, multiple times per day, nightly, or weekly, depending on the data freshness requirements and computational complexity of each pipeline. Each of the scheduled data pipelines 114 may

write results to domain-specific database tables that processing components may query at low latency during request handling. This batch precomputation approach may isolate user-facing latency from the computational cost of running analysis operations on data from the external data sources 126.

[0041] An encryption and key service 118 may provide cryptographic operations for the system 100. In some embodiments, the encryption and key service 118 may implement field-level encryption using symmetric envelope encryption applied per-field based on sensitivity flags associated with field metadata. For example, the encryption and key service 118 may apply Fernet encryption, which combines AES-128-CBC with HMAC-SHA256, to fields marked as sensitive in field metadata configurations. The encryption and key service 118 may encrypt fields individually rather than at the table level, allowing different encryption policies to be applied to different fields within the same record.

[0042] The data tier of the system 100 may include a multi-tenant database 120, a vector index 122, and an archive store 124. The agent orchestration engine 110 may access the multi-tenant database 120 to read and write tenant-scoped data. In some embodiments, the multi-tenant database 120 may be implemented using PostgreSQL with Row-Level Security (RLS) policies that evaluate organization identifiers against session-scoped variables. The RLS policies may cause queries to return zero rows when the session variable is unset, thereby failing closed and preventing unauthorized data access. For example, an RLS policy may evaluate an organization identifier column against a current session setting to restrict query results to rows associated with the organization identifier established by the tenant middleware 106.

[0043] The search and redaction module 116 may access the vector index 122 to perform semantic search operations. In some embodiments, the vector index 122 may use metadata partitioning by chatbot type to segregate corpora. This metadata partitioning may allow compliance officers to scope investigations to conversational history associated with a particular persona. The vector index 122 may store vector embeddings of text content along with metadata including organization identifiers, user identifiers, chatbot type identifiers, and timestamps.

[0044] The encryption and key service 118 may access the archive store 124 to store and retrieve encrypted data for long-term retention. The archive store 124 may store audit logs, regulatory exports, and other compliance-related data separately from user data. In some

embodiments, the archive store 124 may enforce immutability constraints through a combination of application-level guards and storage-level retention policies.

[0045] The components of the system 100 may interact to provide secure data isolation across organizational boundaries. When the client application 102 sends a request, the authentication gateway 104 may validate the authentication token and forward the request to the tenant middleware 106. The tenant middleware 106 may extract the organization identifier and set the session-scoped variable before passing the request to the API server 108. The API server 108 may route the request to the agent orchestration engine 110, which may invoke the specialist processing modules 112 to process the request. Throughout this processing, the RLS policies enforced by the multi-tenant database 120 may restrict data access to rows associated with the organization identifier, providing data isolation at the database layer rather than relying on application-layer filtering.

[0046] FIG. 2 illustrates an example method 200 for multi-tenant data isolation with immutable audit logging. The method 200 may be performed by a server, such as the API server 108 in conjunction with the tenant middleware 106 and the multi-tenant database 120 described with reference to FIG. 1. The method 200 may provide data isolation at the database layer rather than relying on application-layer filtering, and may ensure that audit records cannot be modified or deleted by application-level code.

[0047] At a step 202, the server may receive a request from a client device. The request may comprise an authentication token identifying a user. In some embodiments, the authentication token may be a JSON Web Token (JWT) signed using an asymmetric cryptographic signature, such as RS256, that binds the request to an authenticated principal. The authentication token may include claims identifying the user. The organization may be resolved from the request context, such as via an organization header included in the request, a custom-domain lookup, or a membership lookup based on the authenticated user.

[0048] At a step 204, middleware executing on the server may extract an identifier of the organization from the request. In some embodiments, the middleware may resolve the organization identifier from the request context via an organization header, a custom-domain lookup, or a membership lookup based on the authenticated user. In some embodiments, the middleware may maintain a list of route prefixes that bypass row-level security enforcement for cross-tenant

operations. The list of route prefixes may include routes for authentication, invitations, onboarding, and platform administration endpoints. For requests whose paths match one of the route prefixes in the list, the middleware may auto-bypass row-level security enforcement. For requests whose paths do not match any of the route prefixes, the middleware may proceed to establish tenant context as described below.

[0049] At a step 206, the server may set, within a database session associated with the request, a session-scoped variable to the identifier of the organization. The session-scoped variable may be set using a database command that establishes the variable for the duration of the database session. In some embodiments, the session-scoped variable may be set using a SET LOCAL command that scopes the variable to the current transaction. The database may enforce a set of row-level security policies that restrict query results to rows matching the session-scoped variable. The row-level security policies may evaluate an organization identifier column in each tenant-scoped table against the session-scoped variable. When the session-scoped variable is unset, the row-level security policies may cause queries to return zero rows, thereby failing closed and preventing unauthorized data access.

[0050] At a step 208, the server may process the request to generate a data operation on a tenant-scoped table. The set of row-level security policies may cause the database to return a set of rows associated with the identifier of the organization. Because the row-level security policies are enforced at the database layer, the policies may apply to all queries executed within the database session, including raw SQL queries and queries executed from within stored procedures. This database-layer enforcement may eliminate cross-tenant data leakage that could otherwise occur through missing WHERE clauses, SQL injection vulnerabilities, or other application-layer bugs.

[0051] At a step 210, the server may generate an audit event corresponding to the data operation. The audit event may capture information about the data operation, including the user identifier, the organization identifier, an action type, a resource type, a resource identifier, a timestamp, an IP address, and a user agent. The audit event may be written to an audit log stored in a compliance-specific database schema that is separate from a user data schema. The compliance-specific database schema may have different encryption, retention, and access policies than the user data schema.

[0052] At a step 212, the server may create a savepoint within a database transaction as part of writing the audit event to the audit log. The savepoint may mark a point within the transaction to which the server can return after completing the audit write operation.

[0053] At a step 214, the server may switch to a write-only database role authorized to insert records into the audit log. The write-only database role may be granted INSERT privileges on the audit log table. The write-only database role may not be granted UPDATE, DELETE, or TRUNCATE privileges on the audit log table, thereby enforcing immutability at the database permission level. The role switch may be scoped to the audit write operation, with the remainder of the transaction executing under the application database role.

[0054] At a step 216, the server may insert the audit event into the audit log. Because the write-only database role lacks permissions to modify or delete existing records, the audit event may be appended to the audit log without the ability to alter previously written audit records.

[0055] At a step 218, the server may revert to a prior database role and release the savepoint. Reverting to the prior database role may restore the permissions associated with the application database role for subsequent operations within the transaction. Releasing the savepoint may complete the audit write sequence.

[0056] The savepoint-based role-switching approach described with reference to steps 212 through 218 may ensure audit immutability by temporarily elevating to a role that can write audit records but cannot modify or delete audit records. Even if application-level code is compromised through SQL injection or other vulnerabilities, the compromised code may not be able to alter audit records because the application database role used during the remainder of the request lacks UPDATE and DELETE privileges on the audit log table. The in-process role escalation may be gated to the single INSERT statement at step 216, after which privileges may be immediately revoked within the same transaction at step 218.

[0057] FIG. 3 illustrates an example method 300 for coordinating a plurality of specialist processing modules in a financial planning system. The method 300 may be performed by a server, such as the API server 108 in conjunction with the agent orchestration engine 110 and the specialist processing modules 112 described with reference to FIG. 1. The method 300 may enable selective re-execution of specialist processing modules based on input category dependencies, thereby

reducing computational overhead compared to monolithic re-execution architectures that re-execute all modules upon any data modification.

[0058] At a step 302, the server may receive a user input that modifies a set of data fields associated with a user profile. The user input may modify one or more data fields related to financial planning parameters, such as holdings, risk tolerance, income, expenses, goals, demographics, or tax information. In some embodiments, the user input may represent a life event modification, wherein a life event modeling tool may support a plurality of event types. The plurality of event types may include, for example, job loss, divorce, medical emergency, market crash, home purchase, child education, disability, natural disaster, and custom event types. Each event type may be associated with a plurality of severity levels, such as minor, moderate, high, and critical, and each combination of event type and severity level may correspond to a portfolio shock percentage that affects the user profile.

[0059] At a step 304, a coordinator module of the server may map each data field of the modified set of data fields to at least one input category based on a stored mapping. The stored mapping may define associations between data fields and input categories. In some embodiments, the stored mapping may be implemented as a dependency graph, such as an AGENT_DEPENDENCIES data structure, that specifies which input categories each specialist processing module depends upon. The input categories may include holdings, risk tolerance, income, expenses, goals, demographics, and tax information. The coordinator module may evaluate the modified data fields against the stored mapping to identify which input categories have been affected by the user input.

[0060] At a step 306, an invalidation engine of the server may determine a subset of the plurality of specialist processing modules to execute. The invalidation engine may select, for the subset, any of the plurality of specialist processing modules whose associated input categories overlap with the at least one input category identified at step 304. The invalidation engine may retain cached results for any remaining of the plurality of specialist processing modules whose associated input categories do not overlap with the modified input categories. In some embodiments, the plurality of specialist processing modules may include a risk agent, a goal-tracking agent, a tax-optimization agent, and a scenario agent. The risk agent may depend on holdings and risk tolerance input categories. The goal-tracking agent may depend on income,

expenses, goals, and demographics input categories. The tax-optimization agent may depend on income, holdings, tax information, and demographics input categories. The scenario agent may depend on income, expenses, holdings, demographics, and risk tolerance input categories. By retaining cached results for specialist processing modules whose input categories are unaffected by the user input, the invalidation engine may implement category-keyed partial recalculation that reduces processor utilization and memory bandwidth consumption compared to re-executing all specialist processing modules.

[0061] At a step 308, the server may execute each of the subset of the plurality of specialist processing modules to generate a set of updated outputs. Each specialist processing module in the subset may process the modified data fields and any relevant cached data to produce outputs specific to the domain of the specialist processing module. In some embodiments, the specialist processing modules may coordinate through a rule set to generate and update financial plans. The specialist processing modules may run Monte Carlo simulations and accommodate life stress events by dynamically creating and comparing different financial scenarios without regenerating an entire financial plan.

[0062] At a step 310, the server may receive, from each executed specialist processing module, at least one symbolic policy signal. Each of the at least one symbolic policy signal may comprise a machine-readable boolean value tagged with a semantic label. The semantic label may identify the meaning of the boolean value in the context of financial planning recommendations. For example, a tax-optimization agent may emit a symbolic policy signal with a semantic label indicating a Roth conversion recommendation, and a goal-tracking agent may emit a symbolic policy signal with a semantic label indicating a liquidity preservation recommendation. The symbolic policy signals may enable deterministic detection of contradictions without requiring natural language processing or semantic analysis of module outputs.

[0063] At a step 312, the server may evaluate each of the at least one symbolic policy signal against a set of conflict rules. Each of the set of conflict rules may define a pair of symbolic policy signals from two different specialist processing modules of the plurality of specialist processing modules that represent contradictory recommendations. For example, a conflict rule may specify that a Roth conversion recommendation from a tax-optimization agent conflicts with a liquidity preservation recommendation from a goal-tracking agent, because a Roth conversion raises near-

term tax burden while liquidity preservation seeks to maintain liquid reserves. Another conflict rule may specify that a recommendation to reduce equity exposure from a risk agent conflicts with a recommendation to maximize growth from a goal-tracking agent, because de-risking a portfolio is incompatible with maximizing expected return.

[0064] At a step 314, responsive to detecting a conflict at step 312, the server may generate a structured conflict object identifying the contradictory recommendations and presenting a set of resolution options to the user. The structured conflict object may include identifiers of the conflicting specialist processing modules, the semantic labels of the conflicting symbolic policy signals, and a set of resolution options. The set of resolution options may include prioritizing the recommendation of a first specialist processing module, prioritizing the recommendation of a second specialist processing module, or requesting user adjudication. The structured conflict object may be presented to a user interface for user selection of a resolution option.

[0065] At a step 316, when no conflict is detected at step 312, the server may apply the updated outputs to the user profile. The updated outputs from the executed specialist processing modules may be merged into the user profile, and the cached results from the non-executed specialist processing modules may be retained for subsequent operations.

[0066] In some embodiments, the system may include a circuit breaker that tracks failures per service key. The circuit breaker may have a failure threshold, such as three consecutive failures, and a reset timeout, such as sixty seconds, before transitioning from an open state to a half-open state. When the circuit breaker is in the open state, the circuit breaker may return a degraded error indicator instead of cascading an error. The circuit breaker may transition through a state machine comprising closed, open, and half-open states, wherein the circuit breaker transitions from closed to open upon reaching the failure threshold, from open to half-open after the reset timeout expires, and from half-open to closed upon a successful operation or back to open upon a failure.

[0067] The category-keyed partial recalculation approach implemented by the method 300 may reduce computational overhead by executing only those specialist processing modules whose input dependencies have been affected by user modifications. Rather than re-executing all specialist processing modules upon any data change, the invalidation engine may analyze the modified input categories and selectively invoke only the affected modules. This selective

invalidation approach may reduce processor cycles and memory bandwidth consumption, and may decrease response latency for user interactions that modify a limited subset of input categories.

[0068] FIG. 4 illustrates an example method 400 for searching and retrieving records from an audit-class archive with provenance-aware redaction. The method 400 may be performed by a server, such as the API server 108 in conjunction with the search and redaction module 116 described with reference to FIG. 1. The method 400 may implement a hybrid retrieval architecture that executes keyword-based and semantic-based searches in parallel and applies configurable redaction strategies based on request parameters.

[0069] At a step 402, the server may receive, from a requesting client device, a search query and a redaction mode indicator. The search query may comprise one or more search terms submitted by a compliance officer or other authorized user through a compliance dashboard interface. The redaction mode indicator may specify whether personally identifiable information in search results should be redacted and, if so, which redaction strategy to apply. In some embodiments, the redaction mode indicator may be a boolean value that toggles between redacted and unredacted display modes, or the redaction mode indicator may specify a particular redaction strategy from among a plurality of available strategies.

[0070] At a step 404, the server may execute the search query against a keyword retrieval index and a semantic retrieval index in parallel to produce a keyword-ranked result list and a semantic-ranked result list. The keyword retrieval index may be implemented using PostgreSQL full-text search with tsvector columns that support exact-match and lexical variant retrieval. The semantic retrieval index may be implemented using a vector index, such as a Pinecone vector index, with vector embeddings generated by an embedding model such as Voyage AI embeddings. By executing queries against both indexes in parallel, the method 400 may reduce retrieval latency compared to sequential execution while combining the strengths of both retrieval modalities. The keyword retrieval index may surface results containing exact keyword matches, while the semantic retrieval index may surface results that are semantically related to the search query even when exact keywords do not appear in the result text. For example, a search for "digital assets" may surface conversations mentioning "Bitcoin," "cryptocurrency," or "stablecoin" through the semantic retrieval index even when the literal phrase "digital assets" does not appear in those conversations.

[0071] At a step 406, the server may compute a fused result list by applying a reciprocal rank fusion function to the keyword-ranked result list and the semantic-ranked result list. The reciprocal rank fusion function may assign a score to each result appearing in at least one of the keyword-ranked result list and the semantic-ranked result list based on a sum of inverse rank positions across the keyword-ranked result list and the semantic-ranked result list. In some embodiments, the reciprocal rank fusion function may compute the score for each result using the formula score $= \sum \frac{1}{k+\text{rank}}$, where k is a smoothing parameter and rank is the position of the result in each ranked list in which the result appears. The smoothing parameter k may have a default value of 60 and may be configurable per request within a range of 1 to 200. The reciprocal rank fusion function may merge results from the semantic retrieval index and the keyword retrieval index without requiring either retriever to expose calibrated scores, thereby enabling fusion of heterogeneous retrieval systems that produce incomparable score distributions.

[0072] At a step 408, the server may select, based on the redaction mode indicator, between a first redaction strategy and a second redaction strategy. The selection may be performed by evaluating the redaction mode indicator received at step 402 to determine which redaction strategy is appropriate for the requesting context. In some embodiments, the first redaction strategy may be applied for human-facing surfaces such as administrator dashboards, while the second redaction strategy may be applied for machine-facing analytics or semantic search contexts.

[0073] At a step 410, when the first redaction strategy is selected, the server may replace detected personally identifiable information with randomized placeholders that vary per session. The randomized placeholders may be generated using a cryptographically secure random selection function, such as a secrets.choice() function, to produce a 6-character alphanumeric string for each detected item of personally identifiable information. Because the randomized placeholders vary per session, the same underlying personally identifiable information may yield different placeholder values in different sessions, thereby preventing cross-session correlation by users viewing the redacted results.

[0074] At a step 412, when the second redaction strategy is selected, the server may replace detected personally identifiable information with deterministic hashed placeholders that are consistent across sessions. The deterministic hashed placeholders may be generated using a cryptographic hash function, such as SHA-256, with a salt value stored in an environment variable.

The hash output may be truncated to produce placeholders of appropriate length for different types of personally identifiable information. For example, email addresses may be replaced with placeholders comprising 16 hexadecimal characters derived from the truncated hash, while names may be replaced with placeholders comprising 8 hexadecimal characters. Because the deterministic hashed placeholders are consistent across sessions, the same personally identifiable information may yield the same placeholder value in each session, thereby enabling consistent entity tracking and joinability across analytics tables while preventing exposure of the underlying personally identifiable information.

[0075] At a step 414, the server may apply the selected redaction strategy to the fused result list to produce a redacted result list. The redaction may be applied at the display layer rather than at the storage layer, such that the underlying data remains stored in encrypted form and the redaction pipeline operates on decrypted text at response time. This display-layer redaction approach may allow the same underlying archive to support both redacted and unredacted access modes without requiring separate indexes or re-indexing operations when toggling between modes.

[0076] At a step 416, the server may return the redacted result list to the requesting client device. The redacted result list may be transmitted to a compliance dashboard or other client interface for display to the requesting user. In some embodiments, the server may also generate an audit log entry recording the search operation, including the redaction mode that was applied, to maintain a record of archive access for regulatory compliance purposes.

[0077] The archive store 124 described with reference to FIG. 1 may implement Write Once Read Many (WORM) compliance through a triple-layer approach. A first layer may comprise storage-level retention policies that prevent deletion or modification of archived objects at the storage infrastructure level. A second layer may comprise application-layer immutability guards that raise permission errors when application code attempts to modify archived conversations, thereby preventing modification through application-level code paths. A third layer may comprise database triggers that block UPDATE and DELETE operations on archived rows at the database level, thereby preventing modification even by privileged administrators with direct database access. This triple-layer WORM compliance implementation may provide defense-in-depth

protection for archived records by enforcing immutability at multiple levels of the system architecture.

[0078] FIG. 5 illustrates an example method 500 for routing user messages across a plurality of persona-specific computational subgraphs while maintaining conversation continuity. The method 500 may be performed by a server, such as the API server 108 in conjunction with the agent orchestration engine 110 described with reference to FIG. 1. The method 500 may enable domain-appropriate conversational experiences by routing user messages to persona-specific subgraphs that have different tool sets, system prompts, and processing configurations, while preserving conversational context across persona switches.

[0079] At a step 502, the server may receive a user message associated with a conversation session. The conversation session may store a persistent message history that accumulates messages across multiple interactions. In some embodiments, the conversation session may also store a context summary and a context summarized through identifier that compact prior conversation turns into a single system message when the persistent message history grows large. The context summary may enable cross-persona context retention without re-sending raw transcripts of prior conversation turns.

[0080] At a step 504, a deterministic intent classifier of the server may classify the user message into an intent category by evaluating the user message against a set of keyword-matching rules. The deterministic intent classifier may route the user message responsive to a match of the set of keyword-matching rules. In some embodiments, the deterministic intent classifier may evaluate user messages through four ordered stages. A first stage may perform short follow-up preservation for user messages having eight or fewer words that contain assent keywords, such as "yes," "ok," or "proceed," to preserve a prior intent for consent-gate replies. A second stage may perform consent reply detection for user messages having ten or fewer words when a consent marker is present in the message history. A third stage may perform ordered keyword matching against intent rules, wherein a first match determines the intent category. A fourth stage may apply a fallback to a general intent category when no rules match in the preceding stages. Because the deterministic intent classifier evaluates deterministic rules rather than invoking a language model for routing decisions, the method 500 may reduce nondeterminism in compliance-sensitive processing paths.

[0081] At a step 506, the server may select, based on the intent category, a persona-specific computational subgraph from the plurality of persona-specific computational subgraphs. The persona-specific computational subgraph may have been compiled with a bound tool set at compile time. The bound tool set may define a set of tools the persona-specific computational subgraph is authorized to invoke. In some embodiments, the plurality of persona-specific computational subgraphs may include three named personas. Merely as an example, a first persona, referred to as Freddy, may be configured as a financial planner persona with 27 tools and a temperature parameter of 0.0. As another example, a second persona, referred to as Harvey, may be configured as a do-it-yourself investor persona with 11 tools and a temperature parameter of 0.8. As a further example, a third persona, referred to as Anna, may be configured as a female customer persona with 27 tools and a temperature parameter of 0.5. The tool permissions may be enforced at graph-compile time, such that subgraph compilation binds the permitted tools to each persona. This compile-time binding may provide a structural boundary that prevents unauthorized tool invocation even in scenarios involving prompt injection attacks, because tools not bound to a persona cannot be invoked by that persona regardless of the content of user messages.

[0082] In some embodiments, the bound tool set may include tools tagged with a mission-critical flag. Tools tagged as mission-critical may not be disabled by organization administrators through configuration settings. By preventing disabling of mission-critical tools, the system may preserve plan-correctness invariants regardless of organization-level configuration changes.

[0083] At a step 508, the server may load the persistent message history from the conversation session. The persistent message history may include messages from prior interactions, including interactions processed by different persona-specific computational subgraphs. When the persistent message history exceeds a size threshold, the server may load a compacted representation comprising the context summary rather than the full raw transcript.

[0084] At a step 510, the server may invoke the selected persona-specific computational subgraph with the persistent message history, a system prompt corresponding to the selected persona-specific computational subgraph, and the bound tool set of the selected persona-specific computational subgraph. In some embodiments, the system prompt may be selected through a three-layer cascade. A first layer of the cascade may comprise a persona instruction stored in a database with per-organization override capability. A second layer of the cascade may comprise

platform chatbot configuration instructions that provide a platform-wide override. A third layer of the cascade may comprise hardcoded core prompts that serve as a fallback when neither the first layer nor the second layer provides a prompt. Each layer of the cascade may be independently configurable, and version identifiers associated with each layer may enable auditors to reproduce the system prompt that was provided to the model on any past date.

[0085] At a step 512, the server may append a response generated by the selected persona-specific computational subgraph to the persistent message history of the conversation session. By appending the response to the persistent message history, a subsequent user message classified into a different intent category may be processed by a different persona-specific computational subgraph using the same persistent message history. This conversation continuity mechanism may enable a user to transition between personas, such as from Freddy to Harvey and back to Freddy, while each persona retains the full context of what was communicated to other personas in intervening interactions. The system prompt, the bound tool set, and the temperature parameter may change when the user transitions between personas, but the conversational state stored in the persistent message history may persist across persona switches.

[0086] FIG. 6 illustrates an example method 600 for processing heterogeneous data inputs with provenance-aware conflict resolution. The method 600 may be performed by a server, such as the API server 108 in conjunction with the agent orchestration engine 110 described with reference to FIG. 1. The method 600 may enable aggregation of data field values from multiple input modalities into a unified user profile while resolving conflicts based on provenance tier precedence and computing composite scores across multiple scoring dimensions.

[0087] At a step 602, the server may receive a plurality of data field values for a user profile. Each data field value may be received from one of a plurality of input modalities. In some embodiments, the plurality of input modalities may include manual form entries submitted through a user interface, natural language disclosures extracted from chat sessions via field aliases, account balance data received from financial aggregation webhooks, and values extracted from uploaded documents via optical character recognition. The same data field may receive values from multiple input modalities, such as when a user manually enters an income value that differs from an income value extracted from a linked payroll account.

[0088] At a step 604, the server may associate each data field value of the plurality of data field values with a provenance tier from a ranked set of provenance tiers. Each provenance tier may correspond to the input modality from which the data field value was received. In some embodiments, the ranked set of provenance tiers may include four tiers with integer authority weights. A first tier, USER_TYPED, may have an authority weight of 4 and may correspond to values explicitly confirmed by a user through manual entry. A second tier, PLAID_AGGREGATED, may have an authority weight of 3 and may correspond to values received from institutional application programming interfaces such as financial account aggregation services. A third tier, PDF_EXTRACTED, may have an authority weight of 2 and may correspond to values extracted from uploaded documents via optical character recognition or document parsing. A fourth tier, INFERRED_FROM_ASSUMPTION, may have an authority weight of 1 and may correspond to values imputed from domain-specific assumption tables when user-provided data is unavailable. The provenance tier metadata may distinguish values supplied by the user from values derived from third-party aggregation or inferred via assumption tables.

[0089] At a step 606, the server may resolve conflicts among data field values received for a same data field from different input modalities by applying a tier-based precedence rule that retains the data field value associated with a higher-ranked provenance tier. The tier-based precedence rule may cause a lower-authority writer to be unable to silently overwrite a higher-authority value. For example, when a first institution provides an annual income value from the PLAID_AGGREGATED tier at a first timestamp and a second source provides an annual income value from the USER_TYPED tier at a second timestamp earlier than the first timestamp, the USER_TYPED value may be retained because the authority weight of 4 outranks the authority weight of 3, regardless of the relative timestamps. This provenance-aware conflict resolution may implement an evidence-weighted truth model that prioritizes user-confirmed values over aggregated or inferred values.

[0090] At a step 608, the server may store the resolved data field values in a canonical field store as a unified representation of the user profile. In some embodiments, the canonical field store may be implemented as a FormValueUnion JSONB column that serves as a single source of truth across multiple form surfaces. The canonical field store may contain field metadata entries, such as 474 field metadata entries in some implementations. Each field metadata entry may declare attributes including a union column identifier, a label, a display configuration, a data type, a

category, a subcategory, keywords, a unit, and chat aliases. The category attribute may assign each field to one of the four scoring quadrants: Person, Assets, Goals, or Obstacles. The chat aliases may enable natural language field extraction, wherein a chat agent may map natural language phrases to the corresponding union column identifier for storage in the canonical field store.

[0091] At a step 610, the server may apply a set of per-asset-class liquidity weights to asset values within the resolved data field values. Each per-asset-class liquidity weight may reflect a time-to-liquidation characteristic of a corresponding asset class. In some embodiments, the set of per-asset-class liquidity weights may include default values that vary based on the liquidity profile of each asset class. Bank accounts may have a liquidity weight of 1.00, reflecting same-day settlement under normal market conditions. Brokerage accounts may have a liquidity weight of 1.00. Cash may have a liquidity weight of 1.00. Bonds may have a liquidity weight of 1.00. Crypto assets may have a liquidity weight of 0.80, reflecting higher bid-ask spreads and potential slippage. Collectibles may have a liquidity weight of 0.60. Real estate may have a liquidity weight of 0.30, reflecting title transfer friction and extended sale timelines. Vehicles may have a liquidity weight of 0.30. Gold may have a liquidity weight of 0.10, reflecting physical sale, assay, and buyer-finding delays. The per-asset-class liquidity weights may produce a contextual net worth wherein a position in a less liquid asset class contributes less to scoring dimensions than an equivalent position in a more liquid asset class. In some embodiments, the per-asset-class liquidity weights may be overridable on a per-user basis without requiring code changes.

[0092] At a step 612, the server may compute a set of dimension scores across a plurality of scoring dimensions using the asset values with the set of per-asset-class liquidity weights applied thereto and the resolved data field values. In some embodiments, the plurality of scoring dimensions may comprise four quadrants: Person, Assets, Goals, and Obstacles. The Person quadrant may incorporate demographic and personal information fields. The Assets quadrant may incorporate financial holdings with liquidity-weighted values. The Goals quadrant may incorporate financial planning objectives and target metrics. The Obstacles quadrant may incorporate risk factors, liabilities, and behavioral engagement signals. The server may compute each dimension score using a weighted sum formula combined with domain-specific assumption tables. The domain-specific assumption tables may include Social Security Administration life-expectancy curves, Internal Revenue Service standard-deduction tables, age-banded income multipliers, and configurable discount rates. The use of domain-specific assumption tables rather

than machine learning inference may enable deterministic score reproduction for advisor defensibility and regulatory scrutiny.

[0093] In some embodiments, behavioral signals may contribute to the dimension scores. The behavioral signals may include login frequency representing logins per a rolling window such as 30 days, plan edit cadence representing plan edits per the rolling window, session duration average representing mean session length, total sessions representing lifetime session count, advisor meetings count representing completed scheduled advisor meetings, personal financial statement completion percentage, fact finder completion percentage, documents uploaded count, and number of linked financial aggregation accounts. The behavioral signals may be folded into the Obstacles quadrant as engagement indicators, wherein a user profile that has not engaged within a threshold period such as 90 days may score lower on goal follow-through even when asset values are strong.

[0094] In some embodiments, each dimension score may be decomposable to show contributing fields for explainability. Further, in some embodiments, a dimension driver function may reverse-map each dimension score back to a set of top contributing fields, such as the top three contributing fields, enabling a user interface to render explanations of why a particular dimension received a particular score with field-level attribution.

[0095] At a step 614, the server may generate a composite score for the user profile from the set of dimension scores. In some embodiments, the composite score may be computed on a scale of 0 to 999. The composite score may be generated using a weighted sum of the dimension scores normalized to the 0 to 999 scale. The server may assign a score band to the composite score based on score thresholds.

[0096] In some embodiments, the server may persist score snapshots for historical trend reporting. Score snapshot persistence may be gated by a minimum delta threshold check that writes a new snapshot when the composite score has changed or when a threshold time period such as 24 hours has elapsed since a previous snapshot. This threshold gating may reduce snapshot churn while preserving an append-only history when the score genuinely changes. In some embodiments, the minimum delta threshold check may evaluate per-dimension deltas, such as a threshold of plus or minus 5 points on any dimension, to trigger snapshot persistence.

[0097] In some embodiments, the system may implement a k-anonymity service for data exports. The k-anonymity service may enforce a k value of 5 with quasi-identifiers including age

in 10-year buckets, zip code as a 3-digit prefix, and income in bracket ranges such as 8 income bands. The k-anonymity service may suppress any equivalence class with fewer than k members to prevent re-identification of individuals in exported datasets.

[0098] FIG. 7 illustrates an example method 700 for transferring a user record between a first institution and a second institution while preserving regulatory retention of historical data. The method 700 may be performed by a server, such as the API server 108 in conjunction with the encryption and key service 118 described with reference to FIG. 1. The method 700 may enable data portability across institutional boundaries while maintaining compliance with regulatory retention requirements by partitioning data at a temporal boundary defined by an institution-switch event.

[0099] At a step 702, the server may receive a portability request from a user device to transfer the user record from the first institution to the second institution. The portability request may be initiated by a user through a client interface, such as a single-click transfer mechanism. The portability request may identify the user record to be transferred, the first institution from which the user record originates, and the second institution to which the user record is to be transferred.

[0100] At a step 704, the server may record a switch event associated with the user record. The switch event may comprise an effective timestamp, an identifier of the first institution, and an identifier of the second institution. In some embodiments, the switch event may be recorded across three related database rows. A first database row in a portability consents table may record user consent to the portability operation, including a source organization identifier corresponding to the first institution, a destination organization identifier corresponding to the second institution, a consent timestamp, and a retention expiration timestamp. A second database row in a data exports table may record bundle metadata, including a date range, an encryption key identifier, a bundle hash, a bundle size, a record count, and a manifest. A third database row in an export records table may record per-export index entries, including a bundle identifier, a user identifier, an organization identifier, and storage object paths. The effective timestamp recorded in the switch event may define a temporal boundary that partitions the user record into data generated before the effective timestamp and data generated after the effective timestamp.

[0101] At a step 706, request middleware of the server may resolve subsequent requests associated with the user record to the second institution based on the switch event. Data generated

after the effective timestamp may be associated with the second institution. In some embodiments, the request middleware may compute the primary tenant for the user per request by evaluating the latest non-revoked consent record associated with the user. The request middleware may resolve the primary tenant to the second institution without physically moving any data rows between institutions. Data rows created under the license of the first institution may remain stored at the first institution, while new data operations, such as tool calls, plan edits, financial aggregation synchronizations, and chat turns, may be routed to the second institution. This consent-based tenant resolution approach may enable forward-stream routing to the second institution while preserving the historical partition at the first institution.

[0102] At a step 708, the server may maintain read access to data generated before the effective timestamp. The data generated before the effective timestamp may remain stored at the first institution and encrypted under an encryption key of the first institution. In some embodiments, the encryption key of the first institution may be managed through a cloud key management service with per-tenant envelope encryption. The cloud key management service may maintain a keyring per organization, wherein each organization has a dedicated cryptographic key for encrypting data associated with that organization. The per-tenant envelope encryption may generate a random data encryption key, encrypt the payload with the data encryption key using a symmetric encryption algorithm such as AES-256-GCM, encrypt the data encryption key with the organization-specific key management service key, and store the wrapped data encryption key along with the ciphertext and a nonce. In some embodiments, the encryption keys may be subject to automatic key rotation on a periodic schedule, such as every 90 days. The automatic key rotation may rotate the wrapping key while permitting wrapped data encryption keys to continue to decrypt under any active key version. The first institution may retain read access to the pre-switch data for regulatory retention purposes, and the second institution may not receive the encryption key of the first institution.

[0103] At a step 710, the server may generate a portable export bundle comprising the user record. The portable export bundle may be signed under a signing key of the first institution and encrypted to a transport key of the second institution. In some embodiments, the portable export bundle may include a signed manifest using JSON Web Signature (JWS). The signed manifest may contain a field-bag comprising data field values associated with the user profile, plan snapshots representing financial planning states, an audit-log slice containing audit events relevant

to the user record, and chat-message references or content depending on receiving-institution policy. The signing operation may enable the second institution to verify the origin and authenticity of the portable export bundle by validating the signature against a published verification key of the first institution. The encryption to the transport key of the second institution may ensure confidentiality during transfer by encrypting a symmetric data encryption key to the transport public key of the second institution using an asymmetric encryption algorithm. The second institution may unwrap the data encryption key using a private key corresponding to the transport key and decrypt the portable export bundle.

[0104] In some embodiments, the system may store a retention policy in a mapping table that associates organization identifiers and data classes with retention periods and legal bases. The mapping table may have a composite primary key comprising an organization identifier and a data class identifier, and may map to retention years and a legal basis for each combination. Different data classes may have different retention periods at each institution. For example, chat messages may have a retention period of three years, order records may have a retention period of six years, tax documents may have a retention period of seven years, and portability exports may have a retention period of seven years. This per-organization-per-data-class retention policy structure may enable institutions to comply with different regulatory obligations for different record types.

[0105] The method 700 may address regulatory scenarios where a user transfers from the first institution to the second institution but the first institution is required to retain records created under the license of the first institution. By recording the switch event with an effective timestamp, resolving subsequent requests to the second institution via request middleware, maintaining read access to pre-switch data encrypted under the encryption key of the first institution, and generating a signed and encrypted portable export bundle, the method 700 may enable data portability while preserving regulatory retention of historical data at the originating institution.

[0106] Additional details related to this disclosure are included in Exhibits A, B, and C which accompany this filing.

[0107] The detailed description set forth above is intended as a description of various configurations and is not intended to represent the only configurations in which the concepts described herein may be practiced. The detailed description includes specific details for the purpose of providing a thorough understanding of the various concepts. However, the concepts

may be practiced without these specific details. In some instances, well-known structures and components are shown in block diagram form in order to avoid obscuring such concepts. The scope of the present disclosure is defined by the claims appended hereto, and the detailed description should not be taken as limiting the scope of the claims. Features described in connection with one embodiment may be applicable to other embodiments, and the description of features in one context should not be taken as excluding such features from other contexts. Alternative embodiments, including embodiments that employ later-developed technology, may fall within the scope of the claims.

[0108] The components, modules, and systems described herein may be implemented in various configurations. Components described as separate or distinct may be combined into a single component, module, or system. Conversely, components described as a single unit may be implemented as multiple separate components that communicate with one another. Operations described as being performed in a particular sequence may be performed in different orders, may be performed concurrently, or may be interleaved with other operations. The specific order of operations described herein is illustrative and not limiting. Variations, modifications, additions, and substitutions to the described embodiments are contemplated and fall within the scope of the claims.

[0109] The systems and methods described herein may be implemented using hardware, software, firmware, or any combination thereof. Hardware implementations may include processors, application-specific integrated circuits, field-programmable gate arrays, digital signal processors, or other circuitry configured to perform the described operations. Software implementations may include executable instructions stored on machine-readable media that, when executed by one or more processors, cause the processors to perform the described operations. Firmware implementations may include executable instructions stored in non-volatile memory that configure hardware to perform specific functions. A general-purpose processor may be configured by software to become a special-purpose processor that performs the functions described herein.

[0110] Hardware modules may be permanently configured to perform specific operations, such as through hardwired circuitry in an application-specific integrated circuit or through configuration of a field-programmable gate array. Alternatively, hardware modules may be

temporarily configured to perform specific operations, such as through execution of software instructions by a processor. A processor may be configured differently at different times to perform different functions, and the same processor may serve as part of different modules or systems at different times. The choice between permanent and temporary configuration, and the allocation of functionality between hardware and software, may be made based on considerations such as cost, time, power consumption, and performance requirements.

[0111] The operations described herein may be performed by one or more processors located in a single computing system or distributed across multiple computing systems. The multiple computing systems may communicate via one or more networks, including local area networks, wide area networks, or the Internet. Components of the systems described herein may communicate via buses, shared memory structures, message passing interfaces, or other communication mechanisms. Functionality described as being performed by a single component may be distributed across multiple components located at different physical locations. Conversely, functionality described as being performed by multiple components may be consolidated into a single component.

[0112] References to processing, computing, calculating, determining, displaying, or similar terms refer to operations performed by one or more machines that manipulate and transform data represented as physical quantities within the memory, registers, or other storage devices of the machines. Data may be represented as electrical, magnetic, optical, or other physical signals within machine-readable media. Machine-readable media may include volatile memory, non-volatile memory, removable storage media, or non-removable storage media. The operations described herein may transform data from one physical state to another physical state within such media.

[0113] As used herein, the terms "comprising," "including," and "having" are open-ended terms that do not exclude the presence of additional elements or steps. References to "a" or "an" element include both singular and plural instances of the element unless the context clearly indicates otherwise. References to "one embodiment," "an embodiment," "some embodiments," or similar phrases indicate that a particular feature, structure, or characteristic is included in at least one embodiment, but do not imply that the feature, structure, or characteristic is present in all embodiments or is limited to a single embodiment. Features described in connection with one embodiment may be combined with features described in connection with other embodiments.

[0114] The systems, methods, and components described herein, including the authentication gateway, tenant middleware, API server, agent orchestration engine, specialist processing modules, search and redaction module, encryption and key service, multi-tenant database, vector index, and archive store, may be implemented in configurations other than those specifically described. The specific architectures, data structures, algorithms, and protocols described herein are illustrative and may be replaced with alternative architectures, data structures, algorithms, and protocols that achieve similar functionality. The claims are not limited to the specific implementations described in the detailed description.

[0115] A number of implementations have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the disclosure. Accordingly, other implementations are within the scope of the following claims.

CLAIMS

1. A computer-implemented method for multi-tenant data isolation with immutable audit logging, the computer-implemented method comprising:

receiving, by a server, a request from a client device, the request comprising an authentication token identifying a user;

extracting, by middleware executing on the server, an identifier of an organization associated with the user from the request;

setting, within a database session associated with the request, a session-scoped variable to the identifier of the organization, wherein the database enforces a set of row-level security policies that restrict query results to rows matching the session-scoped variable;

processing the request to generate a data operation on a tenant-scoped table, wherein the set of row-level security policies cause the database to return a set of rows associated with the identifier of the organization;

generating an audit event corresponding to the data operation; and

writing the audit event to an audit log by:

creating a savepoint within a database transaction,

switching to a write-only database role authorized to insert records into the audit log,

inserting the audit event into the audit log,

reverting to a prior database role, and

releasing the savepoint.

2. A computer-implemented method for coordinating a plurality of specialist processing modules in a financial planning system, the computer-implemented method comprising:

receiving, by a server, a user input that modifies a set of data fields associated with a user profile;

mapping, by a coordinator module of the server, each data field of the modified set of data fields to at least one input category based on a stored mapping;

determining, by an invalidation engine of the server, a subset of the plurality of specialist processing modules to execute, including:

selecting, for the subset, any of the plurality of specialist processing modules whose associated input categories overlap with the at least one input category, and

retaining cached results for any remaining of the plurality of specialist processing modules;

executing, by the server, each of the subset of the plurality of specialist processing modules to generate a set of updated outputs;

receiving, from each executed specialist processing module, at least one symbolic policy signal, each of the at least one symbolic policy signal comprising a machine-readable boolean value tagged with a semantic label;

evaluating, by the server, each of the at least one symbolic policy signal against a set of conflict rules, each of the set of conflict rules defining a pair of symbolic policy signals from two different specialist processing modules of the plurality of specialist processing modules that represent contradictory recommendations; and

responsive to detecting a conflict, generating, by the server, a structured conflict object identifying the contradictory recommendations and presenting a set of resolution options to the user.

3. A computer-implemented method for searching and retrieving records from an audit-class archive, the computer-implemented method comprising:

receiving, by a server from a requesting client device, a search query and a redaction mode indicator;

executing, by the server, the search query against a keyword retrieval index and a semantic retrieval index in parallel to produce a keyword-ranked result list and a semantic-ranked result list;

computing, by the server, a fused result list by applying a reciprocal rank fusion function to the keyword-ranked result list and the semantic-ranked result list, the reciprocal rank fusion function assigning a score to each result appearing in at least one of the keyword-ranked result list and the semantic-ranked result list based on a sum of inverse rank positions across the keyword-ranked result list and the semantic-ranked result list;

selecting, based on the redaction mode indicator, between a first redaction strategy and a second redaction strategy, wherein:

the first redaction strategy replaces detected personally identifiable information with randomized placeholders that vary per session, and

the second redaction strategy replaces detected personally identifiable information with deterministic hashed placeholders that are consistent across sessions;

applying the selected redaction strategy to the fused result list to produce a redacted result list; and

returning the redacted result list to the requesting client device.

4. A computer-implemented method for routing user messages across a plurality of persona-specific computational subgraphs while maintaining conversation continuity, the computer-implemented method comprising:

receiving, by a server, a user message associated with a conversation session, the conversation session storing a persistent message history;

classifying, by a deterministic intent classifier of the server, the user message into an intent category by evaluating the user message against a set of keyword-matching rules, wherein the

deterministic intent classifier routes the user message responsive to a match of the set of keyword-matching rules;

selecting, based on the intent category, a persona-specific computational subgraph from the plurality of persona-specific computational subgraphs, the persona-specific computational subgraph having been compiled with a bound tool set at compile time, the bound tool set defining a set of tools the persona-specific computational subgraph is authorized to invoke;

loading the persistent message history from the conversation session;

invoking the selected persona-specific computational subgraph with the persistent message history, a system prompt corresponding to the selected persona-specific computational subgraph, and the bound tool set of the selected persona-specific computational subgraph; and

appending a response generated by the selected persona-specific computational subgraph to the persistent message history of the conversation session, whereby a subsequent user message classified into a different intent category is processed by a different persona-specific computational subgraph using the same persistent message history.

5. A computer-implemented method for processing heterogeneous data inputs with provenance-aware conflict resolution, the computer-implemented method comprising:

receiving, by a server, a plurality of data field values for a user profile, each data field value received from one of a plurality of input modalities;

associating, by the server, each data field value of the plurality of data field values with a provenance tier from a ranked set of provenance tiers, each provenance tier corresponding to the input modality from which the data field value was received;

resolving, by the server, conflicts among data field values received for a same data field from different input modalities by applying a tier-based precedence rule that retains the data field value associated with a higher-ranked provenance tier;

storing the resolved data field values in a canonical field store as a unified representation of the user profile;

applying, by the server, a set of per-asset-class liquidity weights to asset values within the resolved data field values, each per-asset-class liquidity weight reflecting a time-to-liquidation characteristic of a corresponding asset class;

computing, by the server, a set of dimension scores across a plurality of scoring dimensions using the asset values with the set of per-asset-class liquidity weights applied thereto and the resolved data field values; and

generating a composite score for the user profile from the set of dimension scores.

6. A computer-implemented method for transferring a user record between a first institution and a second institution while preserving regulatory retention of historical data, the computer-implemented method comprising:

receiving, by a server, a portability request from a user device to transfer the user record from the first institution to the second institution;

recording a switch event associated with the user record, the switch event comprising an effective timestamp, an identifier of the first institution, and an identifier of the second institution;

resolving, by request middleware of the server, subsequent requests associated with the user record to the second institution based on the switch event, wherein data generated after the effective timestamp is associated with the second institution;

maintaining, by the server, read access to data generated before the effective timestamp, the data generated before the effective timestamp remaining stored at the first institution and encrypted under an encryption key of the first institution; and

generating a portable export bundle comprising the user record, the portable export bundle being signed under a signing key of the first institution and encrypted to a transport key of the second institution.

ABSTRACT

Systems, methods, and platforms are provided for multi-tenant data isolation with immutable audit logging. A server receives a request from a client device comprising an authentication token identifying a user. Middleware extracts an identifier of an organization associated with the user from the request. A session-scoped variable is set to the identifier within a database session, wherein the database enforces row-level security policies restricting query results to rows matching the session-scoped variable. The request is processed to generate a data operation on a tenant-scoped table, wherein the row-level security policies cause the database to return rows associated with the organization identifier. An audit event corresponding to the data operation is generated and written to an audit log by creating a savepoint within a database transaction, switching to a write-only database role, inserting the audit event, reverting to a prior database role, and releasing the savepoint.

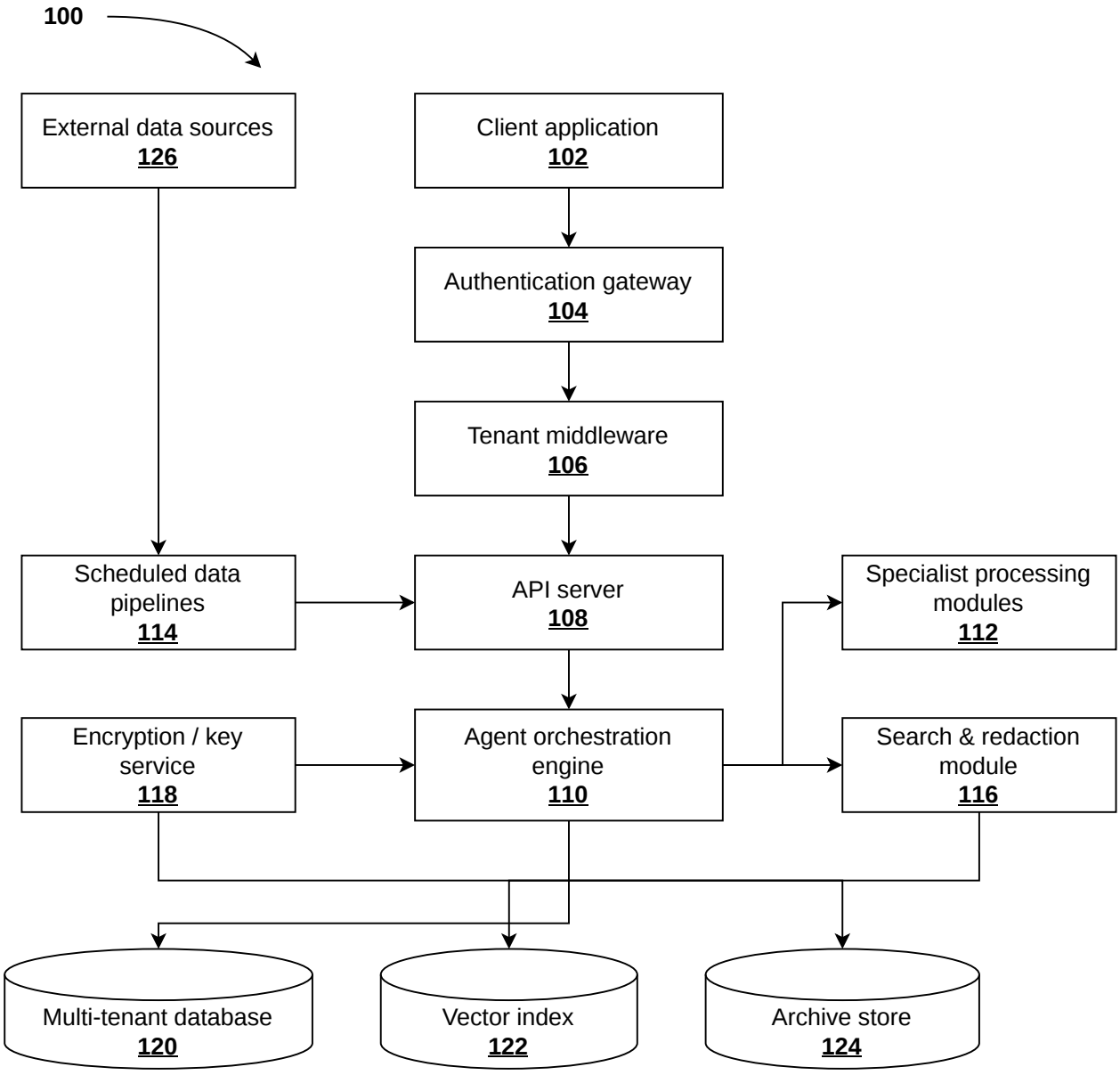

100
External data sources
126
Client application
102
Authentication gateway
104
Tenant middleware
106
Scheduled data pipelines
114
API server
108
Specialist processing modules
112
Encryption / key service
118
Agent orchestration engine
110
Search & redaction module
116
Multi-tenant database
120
Vector index
122
Archive store
124

FIG. 1

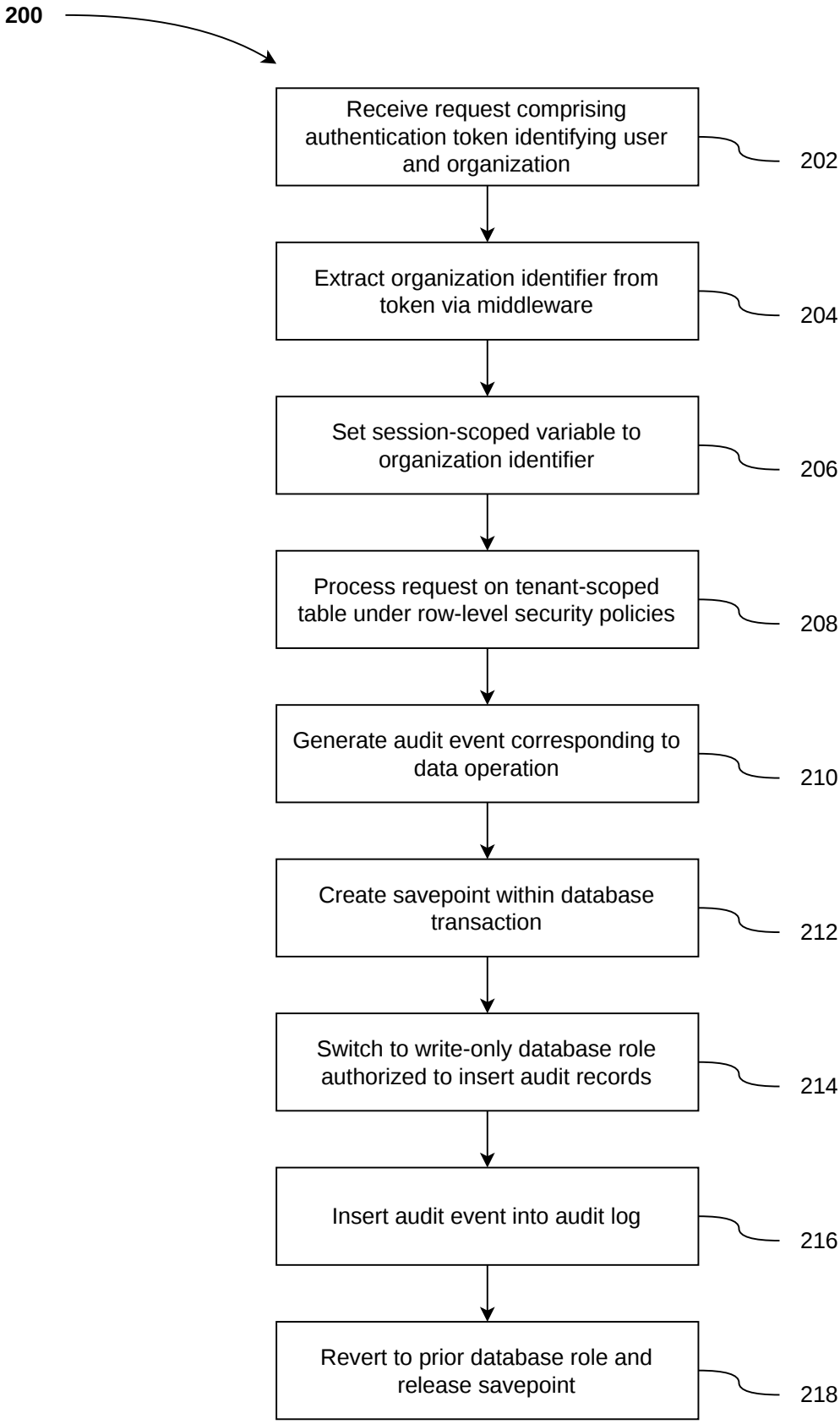

200
Receive request comprising authentication token identifying user and organization
202
Extract organization identifier from token via middleware
204
Set session-scoped variable to organization identifier
206
Process request on tenant-scoped table under row-level security policies
208
Generate audit event corresponding to data operation
210
Create savepoint within database transaction
212
Switch to write-only database role authorized to insert audit records
214
Insert audit event into audit log
216
Revert to prior database role and release savepoint
218

FIG. 2

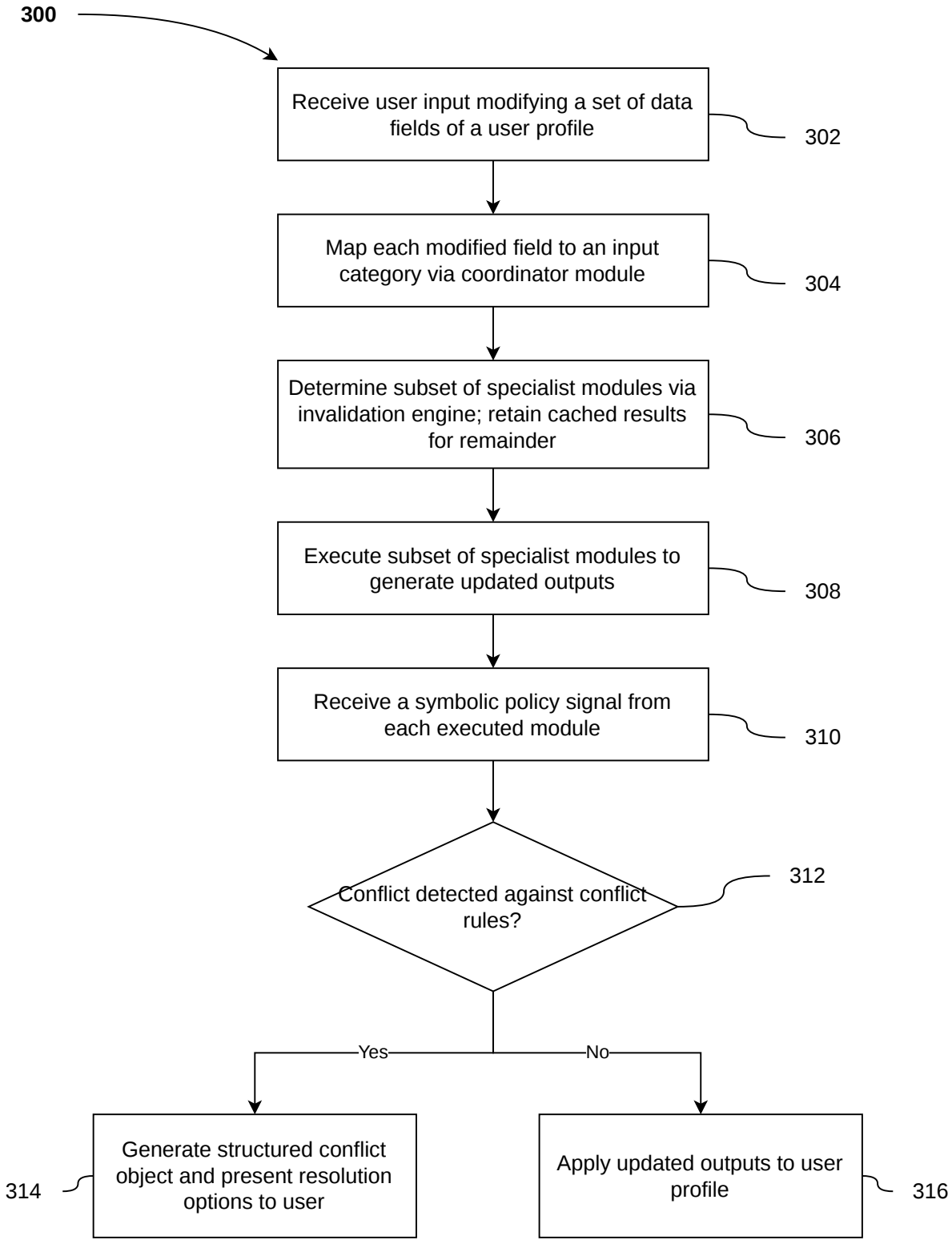

300
Receive user input modifying a set of data fields of a user profile
302
Map each modified field to an input category via coordinator module
304
Determine subset of specialist modules via invalidation engine; retain cached results for remainder
306
Execute subset of specialist modules to generate updated outputs
308
Receive a symbolic policy signal from each executed module
310
Conflict detected against conflict rules?
312
Yes
No
Generate structured conflict object and present resolution options to user
314
Apply updated outputs to user profile
316

FIG. 3

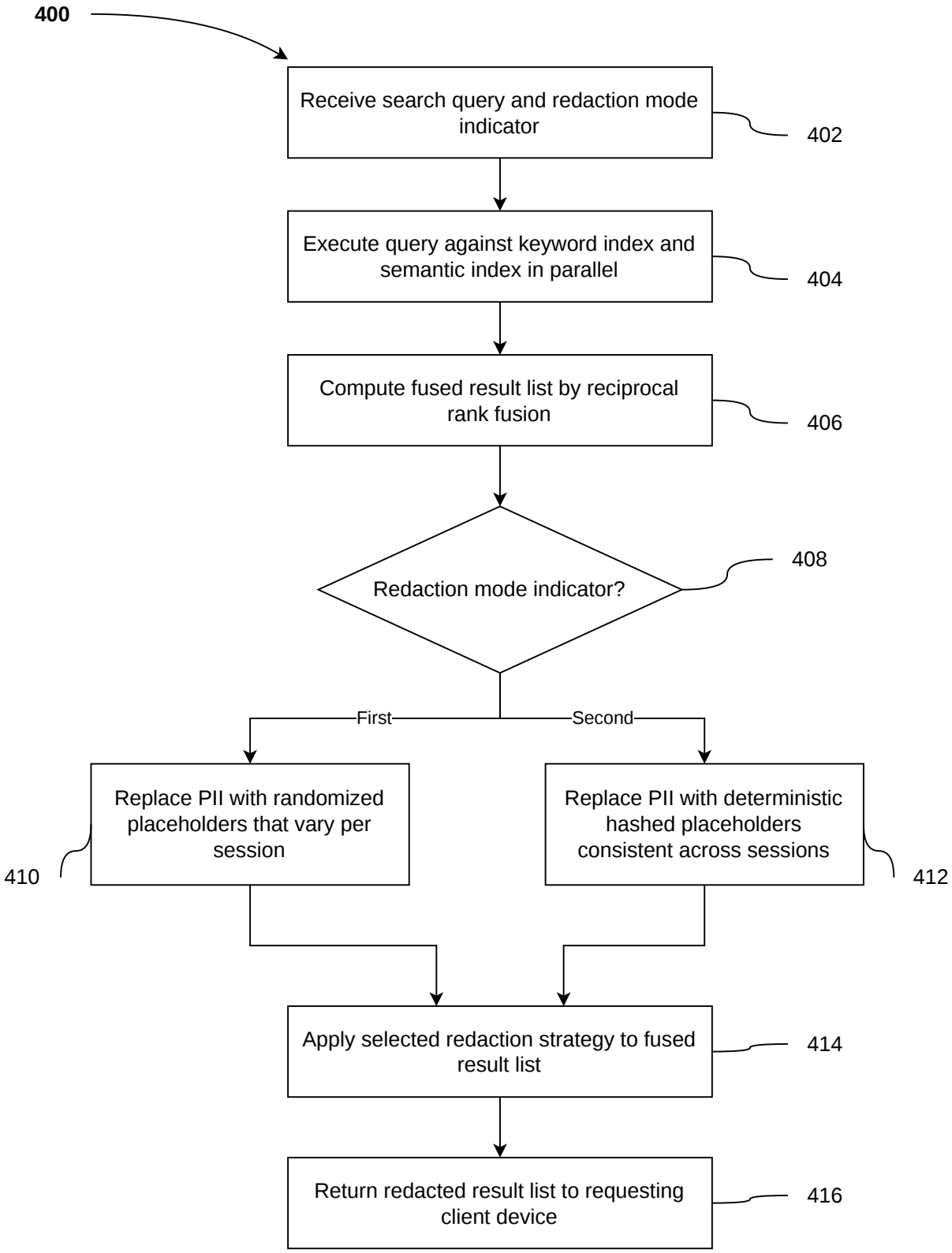

400
Receive search query and redaction mode indicator
402
Execute query against keyword index and semantic index in parallel
404
Compute fused result list by reciprocal rank fusion
406
Redaction mode indicator?
408
First
Second
Replace PII with randomized placeholders that vary per session
410
Replace PII with deterministic hashed placeholders consistent across sessions
412
Apply selected redaction strategy to fused result list
414
Return redacted result list to requesting client device
416

FIG. 4

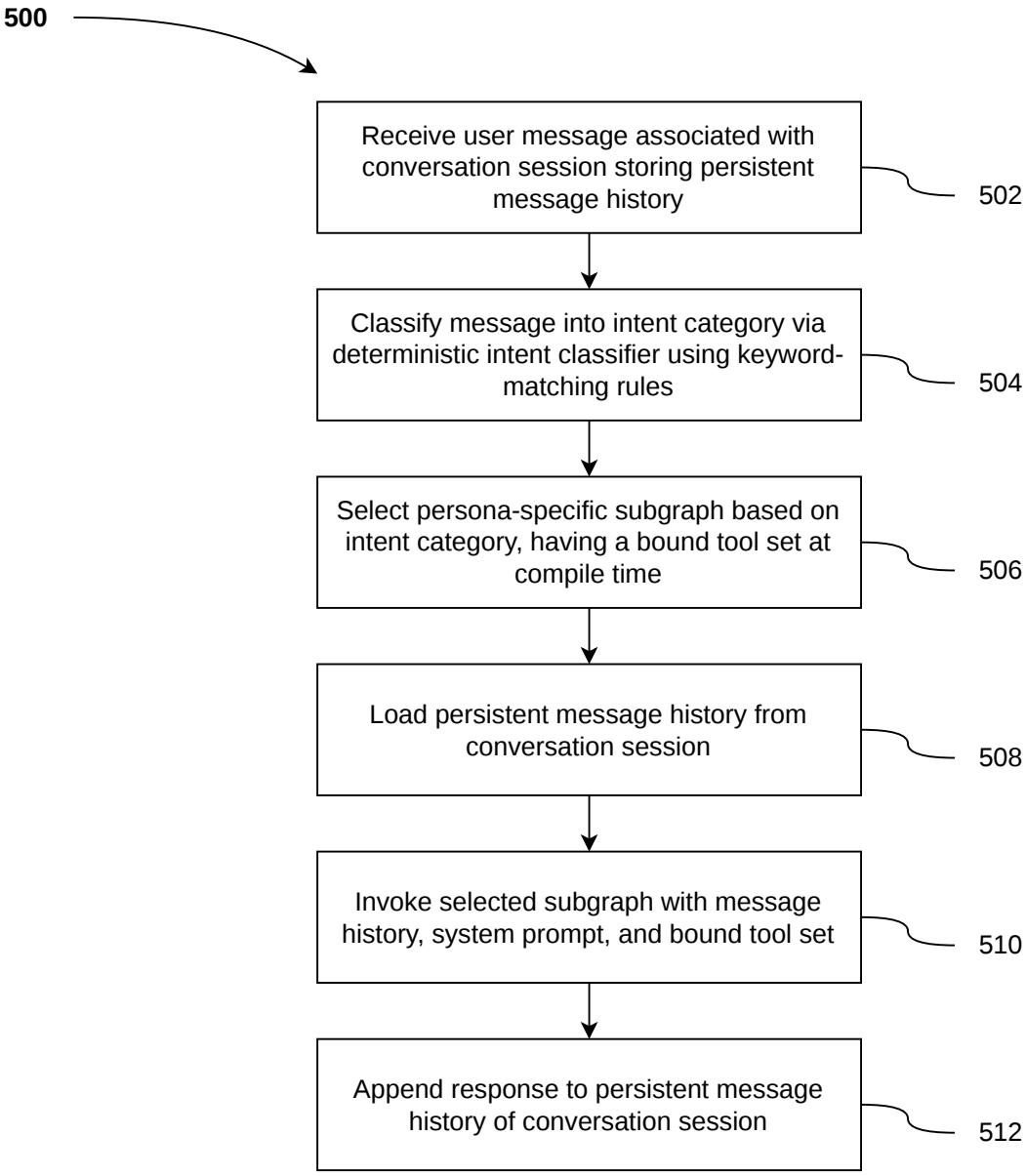

500
Receive user message associated with conversation session storing persistent message history
502
Classify message into intent category via deterministic intent classifier using keyword-matching rules
504
Select persona-specific subgraph based on intent category, having a bound tool set at compile time
506
Load persistent message history from conversation session
508
Invoke selected subgraph with message history, system prompt, and bound tool set
510
Append response to persistent message history of conversation session
512

FIG. 5

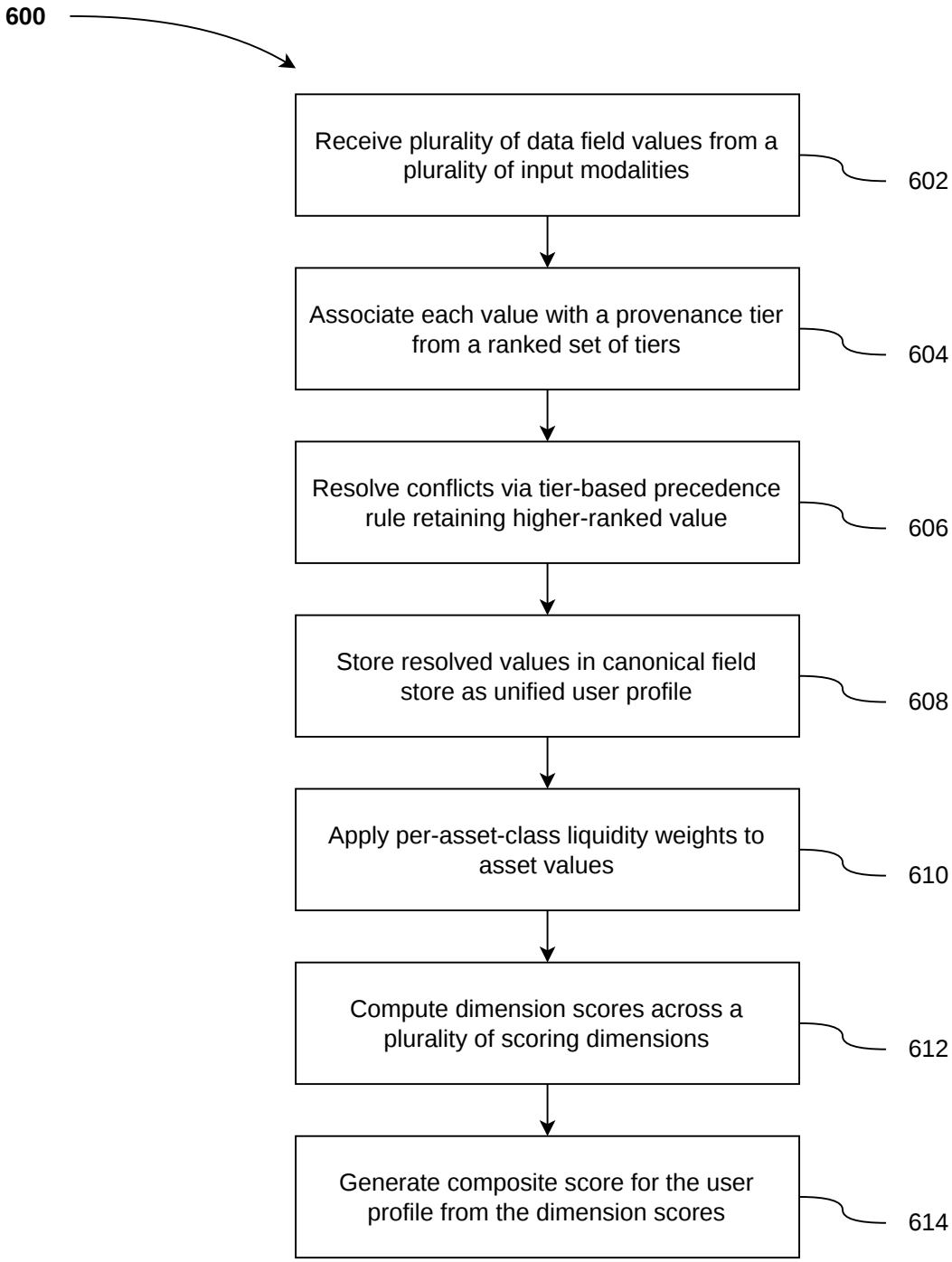

600
Receive plurality of data field values from a plurality of input modalities
602
Associate each value with a provenance tier from a ranked set of tiers
604
Resolve conflicts via tier-based precedence rule retaining higher-ranked value
606
Store resolved values in canonical field store as unified user profile
608
Apply per-asset-class liquidity weights to asset values
610
Compute dimension scores across a plurality of scoring dimensions
612
Generate composite score for the user profile from the dimension scores
614

FIG. 6

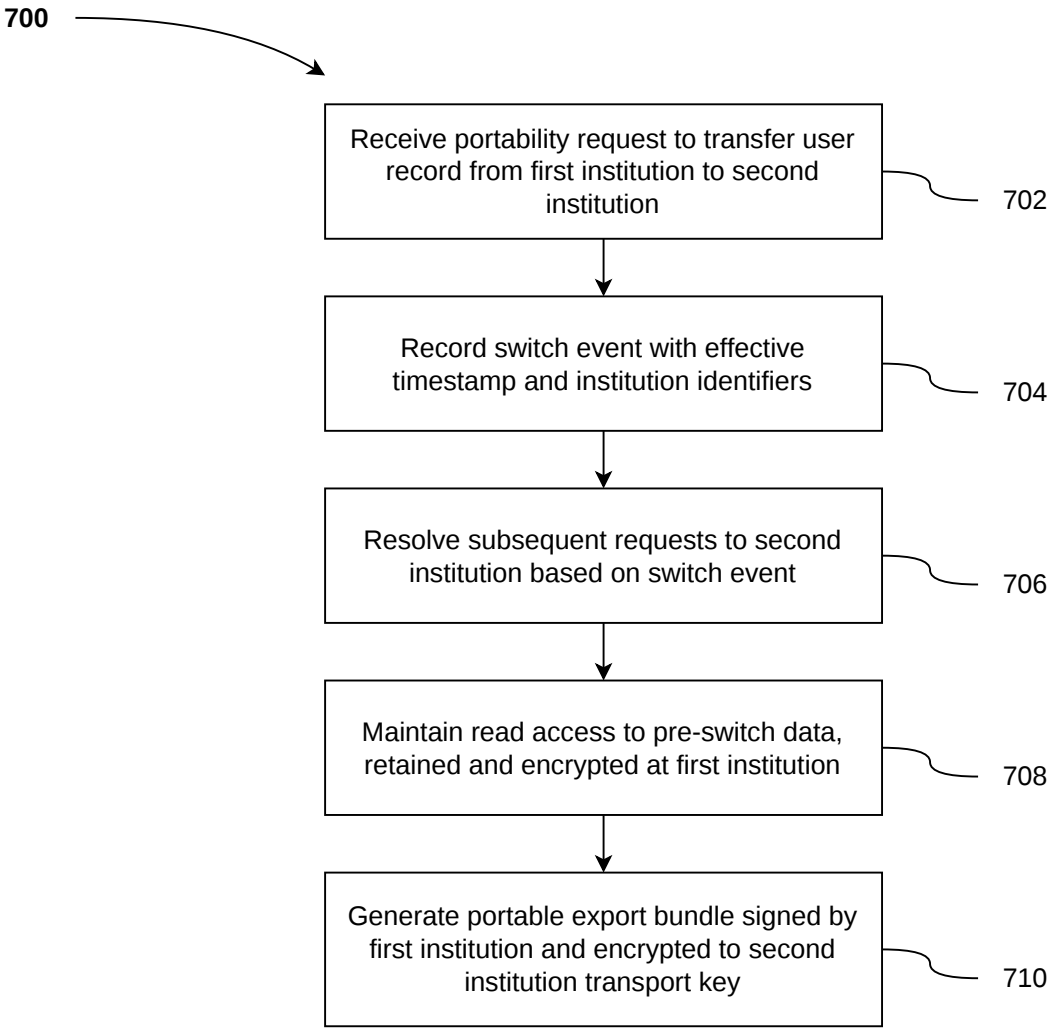

700
Receive portability request to transfer user record from first institution to second institution
702
Record switch event with effective timestamp and institution identifiers
704
Resolve subsequent requests to second institution based on switch event
706
Maintain read access to pre-switch data, retained and encrypted at first institution
708
Generate portable export bundle signed by first institution and encrypted to second institution transport key
710

FIG. 7

Application Data Sheet

Inventor Information

Inventor Number::	1
Given Name::	Dmitry
Family Name::	Pokhilko
City of Residence::	San Francisco
State or Province of Residence::	CA
Country of Residence::	US
Street of mailing address::	100 Van Ness Ave, Apt. 2210
City of mailing address::	San Francisco
State or Province of mailing address::	CA
Country of mailing address::	US
Postal or Zip Code of mailing address::	94102

Inventor Number::	2
Given Name::	Harvey
Family Name::	Sax
City of Residence::	North Salt Lake
State or Province of Residence::	UT
Country of Residence::	US
Street of mailing address::	845 Winter Lane
City of mailing address::	North Salt Lake
State or Province of mailing address::	UT

Country of mailing address::	US
Postal or Zip Code of mailing address::	84054

Inventor Number::	3
Given Name::	Kentaro
Family Name::	Vadney
City of Residence::	Alameda
State or Province of Residence::	CA
Country of Residence::	US
Street of mailing address::	116 Brunswick Way
City of mailing address::	Alameda
State or Province of mailing address::	CA
Country of mailing address::	US
Postal or Zip Code of mailing address::	94502

Correspondence Information

Correspondence Customer Number::	04743

Application Information

Application Type::	Provisional
Subject Matter::	Utility
CD-ROM or CD-R?::	None
Sequence submission?::	None

Computer Readable Form (CRF)?::	No
Title::	MULTI-TENANT FINANCIAL PLANNING PLATFORM WITH AGENT ORCHESTRATION AND PROVENANCE-AWARE DATA PROCESSING
Attorney Docket Number::	34280/70000P
Request for Early Publication?::	No
Request for Non-Publication?::	No
Total Drawing Sheets::	7
Small Entity?::	Yes
Petition included?::	No
Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2::	No

Representative Information

Representative Customer Number::	04743

Domestic Priority Information

Foreign Priority Information

Applicant Information

Applicant Number::	1

Applicant Type::	Assignee
Organization Name::	Veru365 LLC
Street of mailing address::	845 Winter Ln S
City of mailing address::	North Salt Lake
State or Province of mailing address::	UT
Country of mailing address::	US
Postal or Zip Code of mailing address::	84054

Assignee Information Including Non-Applicant Assignee Information

Authorization or Opt-Out of Authorization to Permit Access

When this Application Data Sheet is properly signed and filed with the application, applicant has provided written authority to permit a participating foreign intellectual property (IP) office access to the instant application-as-filed (see paragraph A in subsection 1 below) and the European Patent Office (EPO) access to any search results from the instant application (see paragraph B in subsection 1 below).

Should applicant choose not to provide an authorization identified in subsection 1 below, applicant **must opt-out** of the authorization by checking the corresponding box A or B or both in subsection 2 below.

NOTE: This section of the Application Data Sheet is **ONLY** reviewed and processed with the **INITIAL** filing of an application. After the initial filing of an application, an Application Data Sheet cannot be used to provide or rescind authorization for access by a foreign IP office(s). Instead, Form PTO/SB/39 or PTO/SB/69 must be used as appropriate.

1. Authorization to Permit Access by a Foreign Intellectual Property Office(s)

A. Priority Document Exchange (PDX) - Unless box A in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the European Patent Office (EPO), the Japan Patent Office (JPO), the Korean Intellectual Property Office (KIPO), the State Intellectual Property Office of the People's Republic of China (SIPO), the World Intellectual Property Organization (WIPO), and any other foreign intellectual property office participating with the USPTO in a bilateral or multilateral priority document exchange agreement in which a foreign application claiming priority to the instant patent application is filed, access to: (1) the instant patent application-as-filed and its related bibliographic data, (2) any foreign or domestic application to which priority or benefit is claimed by the instant application and its related bibliographic data, and (3) the date of filing of this Authorization. See 37 CFR 1.14(h)(1).

B. Search Results from U.S. Application to EPO - Unless box B in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the EPO access to the bibliographic data and search results from the instant patent application when a European patent application claiming priority to the instant patent application is filed. See 37 CFR 1.14(h)(2).

The applicant is reminded that the EPO's Rule 141(1) EPC (European Patent Convention) requires applicants to submit a copy of search results from the instant application without delay in a European patent application that claims priority to the instant application.

2. Opt-Out of Authorizations to Permit Access by a Foreign Intellectual Property Office(s)

☐ A. Applicant **DOES NOT** authorize the USPTO to permit a participating foreign IP office access to the instant application-as-filed. If this box is checked, the USPTO will not be providing a participating foreign IP office with any documents and information identified in subsection 1A above.

☐ B. Applicant **DOES NOT** authorize the USPTO to transmit to the EPO any search results from the instant patent application. If this box is checked, the USPTO will not be providing the EPO with search results from the instant application.

NOTE: Once the application has published or is otherwise publicly available, the USPTO may provide access to the application in accordance with 37 CFR 1.14.

Signature:

NOTE: This Application Data Sheet must be signed in accordance with 37 CFR 1.33(b). **However, if this Application Data Sheet is submitted with the INITIAL filing of the application and either box A or B is not checked in subsection 2 of the "Authorization or Opt-Out of Authorization to Permit Access" section, then this form must also be signed in accordance with 37 CFR 1.14(c).**

This Application Data Sheet **must** be signed by a patent practitioner if one or more of the applicants is a **juristic entity** (e.g., corporation or association). If the applicant is two or more joint inventors, this form must be signed by a patent practitioner, **all** joint inventors who are the applicant, or one or more joint inventor-applicants who have been given power of attorney (e.g., see USPTO Form PTO/AIA/81) on behalf of **all** joint inventor-applicants.

See 37 CFR 1.4(d) for the manner of making signatures and certifications.

Signature	/Matthew R. Carey/	Date (YYYY-MM-DD)	2026-06-12
Name	Matthew R. Carey	Registration Number	61,082

Exhibit A

Patent #1 — PAGO Scoring Algorithm

1. Pattern / Technology Name

Scoring mechanism to determine life-time value of a customer.

PAGO Score — a deterministic, multi-tier, weighted-composite financial-readiness score on a 0–999 scale, where PAGO stands for Person / Assets / Goals / Obstacles, the four-quadrant data framework underlying every downstream calculation. PAGO_Score_Model_v5

2. Goal / Purpose

Produce a single, regulator-defensible numeric measure of a household's financial readiness that:

- Aggregates ≥100 heterogeneous data points (structured, conversational, and document-derived) into one normalized score.
- Is explainable: every score decomposes into per-dimension drivers a CFP can defend to a client or auditor.
- Is incremental: re-scores in real time as new data arrives (Plaid sync, chat disclosure, PDF ingest, manual form edit).
- Surfaces data-tier provenance — distinguishing values supplied by the user vs. derived from third-party aggregation vs. inferred via assumption tables.

3. Technologies Used

- Python 3.11, FastAPI, SQLAlchemy 2.x, Pydantic v2.
- PostgreSQL (Cloud SQL) with JSONB columns for the FormValueUnion field-bag.
- Pinecone (semantic chat ingest) and Voyage AI embeddings (voyage-3).
- Plaid for account/identity/transaction aggregation.
- Google Gemini 2.0 Flash (primary LLM) + OpenAI GPT-4 (fallback) via LangChain ChatVertexAI for natural-language field extraction.
- Google's OCR and PII redaction tools for redacting PDF onboarding.

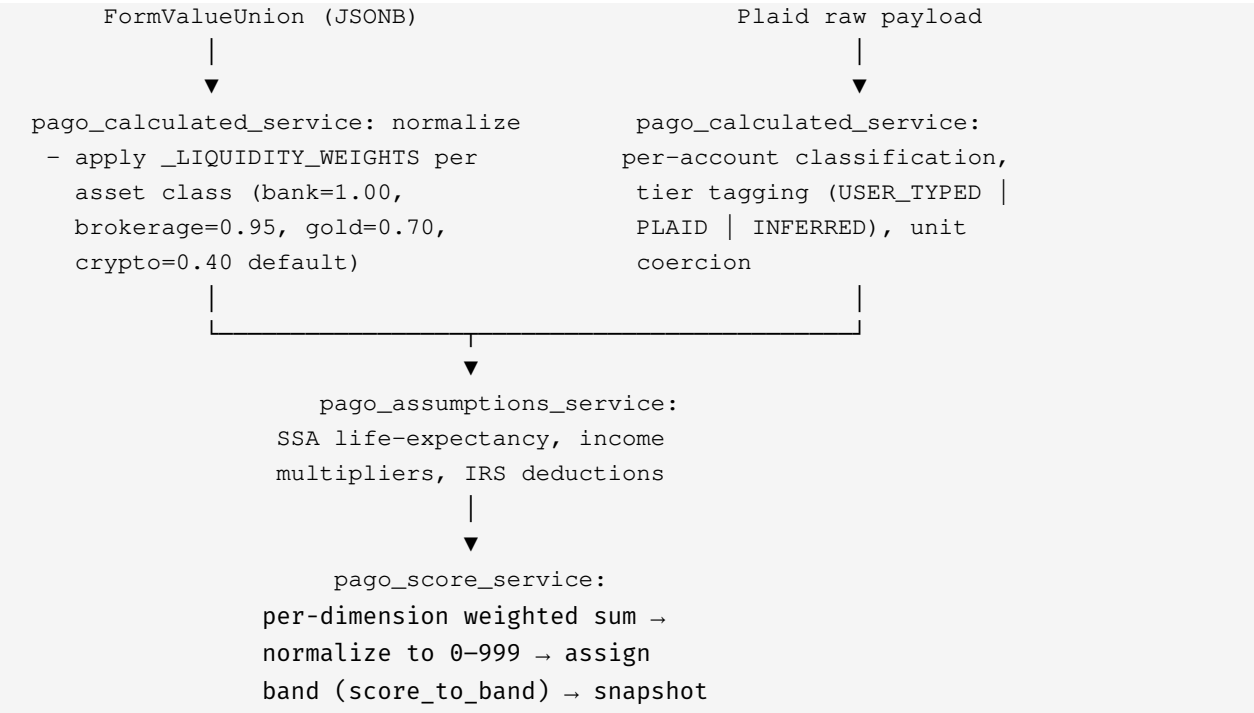
4. Current Implementation

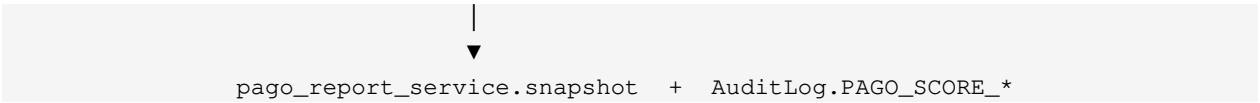
The PAGO algorithm spans these source files:

File	Role
app/form_fields.py	Central field registry — 337 FieldMetadata entries (Person, Assets, Goals, Obstacles, behavioral). Each entry declares union_column, label, display, data_type, category, subcategory, keywords, unit, and chat_aliases.
app/services/pago_score_service.py	get_pago_score_display(), get_score_breakdown(), _dimension_driver(), score_to_band(), advisor-book aggregation. ~475 LOC.

File	Role
app/services/pago_calculated_service.py	Asset/liability normalization with per-asset-class liquidity weights (_LIQUIDITY_WEIGHTS): bank_accounts, brokerage, cash, bonds, gold, crypto. Per-user override supported. ~865 LOC.
app/services/pago_assumptions_service.py	Domain-specific assumption tables: SSA life-expectancy curves, IRS standard-deduction tables, age-banded income multipliers, default discount rates.
app/services/pago_behavioral_service.py	Behavioral-tier signals (login frequency, plan-edit cadence, document-upload recency) feeding the obstacle and engagement dimensions.
app/services/pago_recommendation_service.py	Maps low-scoring dimensions to specific advisor-handoff or self-serve next-actions.
app/services/pago_report_service.py	Auditable per-score snapshot persistence (_record_snapshot_if_changed) for historical trend reporting.
app/services/pago_distribution_service.py	Cohort distribution across the advisor's book (used by get_advisor_book_distribution).
app/services/pago_compliance_service.py	Tags every score event for audit log emission.
app/routes/pago.py	REST endpoints: /api/pago/score, /api/pago/breakdown, advisor dashboard.

Score composition flow (current):





Live re-scoring: any write to FormValueUnion or any Plaid webhook re-invokes _record_snapshot_if_changed, which recomputes the score and persists a new snapshot only if at least one dimension delta exceeds a configured threshold (reduces snapshot churn while preserving an append-only history).

5. Technical Details + Unique Approaches

1. Four-tier provenance per field. Every field-bag value carries an implicit tier label — USER_TYPED, PLAID_AGGREGATED, PDF_EXTRACTED , INFERRED_FROM_ASSUMPTION — and the algorithm down-weights inferred values vs. user-confirmed values. This tier metadata is unusual; most financial-readiness scores treat every input as equal.
2. Per-asset-class liquidity weighting with user override. _LIQUIDITY_WEIGHTS is the default cascade; advisors may override per-client (liquidity_brokerage, liquidity_cryptocurrency, etc.) without code change. This produces a contextual net-worth feeding the score: a $50K crypto position contributes less to the 'Protection' dimension than $50K in checking.
3. Domain-specific assumption tables, not ML inference. Life-expectancy adjustments use the SSA period life table; income multipliers use age-banded BLS data; discount rates are configurable per macro regime. The score therefore reproduces deterministically — required for advisor defensibility and SEC scrutiny — and is patentable as an algorithmic method, not a statistical model.
4. Multi-modal input fusion. The same field-key can be set by (a) a manual form entry, (b) the chat agent's save_user_financial_data tool extracting from natural language via the field's chat_aliases, (c) a Plaid account-balance webhook, or (d) a PDF ingest. The FormValueUnion.union_column design collapses these into one canonical value with last-writer-wins + tier-aware override.
5. Decomposable dimensions. _dimension_driver() reverse-maps every dimension's score back to the top-3 contributing fields, enabling the UI to render 'Why is my Outlook score 612?' with field-level attribution. This is the patentable explainability surface.
6. Behavioral tier as a soft signal. Login frequency, plan-edit recency, and document-upload age are folded into the Obstacles quadrant as decay-weighted signals — a household that hasn't engaged in 90 days scores down on 'Goals' follow-through even with strong assets.
7. Snapshot diff threshold. Snapshot persistence is gated by a min-delta check per dimension — preventing the audit log from being polluted by no-op recomputes while preserving forensic-grade history when the score genuinely moves.

6. Planned / Future Improvements + References

Per the engineering gap analysis (Patent Gap Write-Up §1):

Gap	Planned Implementation	Est. Effort
A. Behavioral-signal expansion to hit 100+ named fields	Today: 337 field-registry entries already exceed 100; this gap is about behavioral signal richness (login frequency, plan-edit cadence, advisor-touch recency) being upgraded from soft inputs to first-class scored fields.	1 day

Future score-formula tuning cadence: quarterly review by the CFP team; A/B tested via the personalized_assumptions JSON column on User; assumption-table snapshots versioned in pago_assumptions_service for reproducibility.

Patent #2 — Multi-Agent Financial Planner

1. Pattern / Technology Name

Expert subject matter team approach to provide financial assistance and planning to the user from our unique

This system is essentially a digital, AI-driven financial planning team managed by a central "Lead Advisor" chatbot. Instead of doing all the heavy lifting alone, this lead AI routes specific tasks—like tax planning or risk assessment—to dedicated specialist AI tools. A built-in mediator layer catches and resolves any conflicting advice between the specialists (such as an investment strategy that accidentally triggers a high tax bill) before it reaches you. Finally, automatic safety switches act as emergency brakes, instantly shutting down any sub-system that errors out or behaves unpredictably to ensure the final financial plan is safe and cohesive.

Specialist-Agent Orchestration with Delegation Subgraphs and Conflict-Resolved Coordination — a LangGraph StateGraph in which a primary persona-bound agent delegates to specialist agents (Risk, Tax, Goals, Scenario, Investment Analyst, Financial Planner) through dedicated subgraphs, with explicit circuit-breaker and conflict-detection layers.

2. Goal / Purpose

Produce, mutate, and explain comprehensive financial plans through conversational interaction such that:

- A single user message can trigger parallel reasoning by multiple specialists.
- A 'what if' mutation (e.g., 'what if I lose my job next year') recalculates only the affected slices — Monte Carlo scenarios, tax projections, retirement glide-paths — without regenerating the entire plan.

3. Technologies Used

- LangGraph (StateGraph + subgraphs), LangChain ChatVertexAI, LangSmith tracing.
- Google Gemini 2.0 Flash (primary), OpenAI GPT-4 (fallback).
- FastAPI Server-Sent Events (SSE) for token + tool-event streaming.
- NumPy / SciPy for Monte Carlo and macro stress simulations.

- Python asyncio.gather for fan-out/fan-in tool execution.

4. Current Implementation

Agent graph lives in backend/app/services/agent_graph/ (8,786 LOC total). Per-file roles:

File	Role	LOC
runner.py	Top-level invocation, SSE event emission, max-iteration guard (6 per turn).	669
subgraphs.py	Per-persona subgraph compilation + tool binding.	1,564
coordinator.py	Cross-agent coordination, plan-state merging, delegation-result aggregation.	739
specialist_agents.py	Specialist registration: risk, tax, goal-tracking, scenario, investment analyst.	225
delegation.py + delegation_tools.py	call_financial_planner, call_investment_analyst — isolated subgraph invocations with their own tool sets.	254 + 159
decision_tree.py	Symbolic routing rules layered over the LLM intent router.	210
conflict_detection.py	Detects contradictory recommendations across specialists and forces a reconciliation pass.	93
circuit_breaker.py	Per-tool failure tracking; opens after threshold and returns degraded result instead of cascading.	217
router.py	Intent classification + persona selection.	181
state.py	Typed AgentState (TypedDict) carrying conversation, tool results, user context.	174
nodes.py	Individual graph node implementations.	636
prompts.py	Specialist-agent system prompts.	112

Specialist tools (mission-critical): run_monte_carlo, compute_retirement_projection, run_scenario_comparison, compute_net_worth, compute_cash_flow, model_life_event, run_macro_stress_test, save_financial_plan, assemble_plan_inputs, check_completeness, check_plan_readiness, plus fact-finder gather/update tools.

Delegation flow (Issue #221):

```
User → Financial Planner persona (primary subgraph)
            |
            | call_investment_analyst("Should I rebalance to 70/30?")
            ▼
   Investment Analyst subgraph
        - separate tool binding (portfolio + EVS + insider)
        - separate system prompt
        - result cached for the delegation_session_id
```

```
          |
          ▼
  Result returned as tool_result event over SSE,
  Coordinator merges into parent AgentState
```

5. Technical Details + Unique Approaches

1. Persona-bound tool registries. Each specialist is compiled into its own LangGraph subgraph with a tool subset it is permitted to invoke. This is enforced at graph-compile time, which makes the boundary cryptographically auditable.
2. Mission-critical tool flag. Every tool in TOOL_REGISTRY is tagged mission_critical: bool. Org admins may disable non-critical tools through the ChatbotConfiguration.enabled_tools JSON field; mission-critical tools cannot be disabled — preserving plan-correctness invariants.
3. Symbolic decision-tree layer over LLM intent routing. decision_tree.py evaluates deterministic rules first; only ambiguous intents fall through to the LLM router. Reduces nondeterminism in compliance-sensitive paths.
4. Conflict detection forces reconciliation. When two specialists produce contradictory recommendations, conflict_detection.py synthesizes a structured trade-off prompt and re-invokes the coordinator with both opinions in context.
5. Circuit breaker per tool. circuit_breaker.py tracks failure rate per (tool_name, user_id) window; on open it returns a degraded payload (last known good value with a degraded: true flag) instead of cascading the error into the SSE stream.
6. Dynamic life-event recalculation. model_life_event accepts event_type × severity and mutates only the affected slices of AgentState.plan_inputs, then invokes only the dependent specialists.
7. Macro stress-test parameter space. run_macro_stress_test supports onshoring, ai_infrastructure, rate_rise_100bps, recession. Each scenario has a deterministic factor sheet loaded from a versioned configuration table — outputs are reproducible across runs.
8. Streamed tool-event protocol. SSE event types include start | chunk | done | error | tool_start | tool_result | modal_serve | workflow_step. The display_hint per tool result controls how the frontend renders — a patent-relevant orchestration of agent output and presentation layer.

6. Planned / Future Improvements + References

Gap	Planned Implementation	Est. Effort
Parallel-debate specialist coordination	Today coordination is sequential with conflict-detection reconciliation. Planned: explicit agent-to-agent message protocol enabling 'Risk and Tax debate a Roth conversion in two turns' with deadlock guards and a debate-quality evaluator wired into LangSmith.	~1 week

Key source files:
app/services/agent_graph/{runner,subgraphs,coordinator,specialist_agents,delegation,decision_tree,conflict_detection,circuit_breaker}.py, app/services/chatbot_service.py (TOOL_REGISTRY, lines 69-306), app/chatbot_config.py.

Patent #3 — PAGO Database Architecture

1. Pattern / Technology Name

Instead of giving each client their own separate database, everyone shares one large Postgres database, but Row-Level Security (RLS) acts as an invisible wall to ensure Client A can never see Client B's data. Inside the database, all flexible form fields are bundled into a single, adaptable container (JSONB), and any sensitive data is immediately scrambled using military-grade encryption (Fernet). To top it off, the system generates an unchangeable, portable paper trail (Provenance-Aware Audit) that logs exactly who viewed or changed any piece of data, providing an airtight audit log for security compliance

Tenant-Scoped Encrypted PAGO Field-Bag with Provenance-Aware Audit and Portable Manifest — a Postgres-resident multi-tenant data architecture that stores all PAGO fields in a FormValueUnion JSONB column, applies field-level encryption to PII via a Fernet envelope, enforces tenant isolation through Postgres Row-Level Security policies, and emits an immutable audit log of every access/modification.

2. Goal / Purpose

Hold a household's lifelong financial record such that:

- The data is logically a single canonical bag queryable by any persona / form (PFS, FactFinder, TaxOrganizer) — no schema drift between surfaces.
- Multi-tenant isolation is enforced at the database layer, not just the application layer.
- Every read/write is auditable to SEC Rule 17a-4, OCC, NCUA, state insurance regulator, and large-employer-group compliance standards.
- PII redaction is reversible (toggle), but the underlying data is never plaintext at rest.
- The bag is exportable as a portable manifest the user can take to a successor institution.

3. Technologies Used

- PostgreSQL 15 (Cloud SQL) with native Row-Level Security, JSONB, partial indexes.
- SQLAlchemy 2.x ORM with RESTRICT cascade semantics for audit-critical relationships.
- cryptography.fernet for symmetric envelope encryption (today: single global key; planned: per-tenant via GCP KMS).
- Alembic migrations (245 to date) with idempotency self-heal in alembic/env.py.
- Auth0 RS256 JWT for principal binding.
- Google Cloud Storage with retention lock (planned) for WORM archives.

4. Current Implementation

Component	Implementation
Organization / Membership	Multi-tenant root; users are members of one or more orgs with OWNER / USER roles.

Component	Implementation
FormValueUnion	Per-user JSONB column holding every PAGO field (337 keys today). Single source of truth across PFS, FactFinder, TaxOrganizer.
User.personalized_assumptions	Per-user assumption overrides for the PAGO algorithm (discount rate, liquidity weights, life-expectancy bias).
AuditLog	Append-only table, immutable by ORM convention. Captures user_id, action, resource_type, status_code, ip_address, user_agent, extra_metadata, created_at.
PersonaInstruction, PlatformChatbotConfiguration	Versioned per-org agent configuration; allows different orgs to mutate prompt behavior without code change.
app/rls.py	Postgres RLS context-management helpers: enable_rls_bypass(), rls_bypass_context(), set_session_org_id() invoking SET LOCAL app.org_id.
app/middleware/rls.py	TenantMiddleware — extracts org from JWT, sets the Postgres session variable per request, auto-bypasses for SKIP_PREFIXES (cross-tenant routes only).
app/services/pii_redaction_service.py	Field-level redaction toggle on admin endpoints. PII_FIELDS = [first_name, last_name, email, phone_number, street_address].
app/services/pago_encryption.py	Current Fernet envelope encryption for sensitive fields (163 LOC).
app/services/pago_export_service.py	Portable export bundle for the data-portability use case (526 LOC).

Data flow on a chat-driven field save:

```
POST /api/chat (SSE)
  → Auth0 JWT verified → user_id, org_id
  → TenantMiddleware: SET LOCAL app.org_id = <org>
  → ChatbotService routes intent to save_user_financial_data tool
  → Tool maps NL phrase via FieldMetadata.chat_aliases → union_column key
  → SQLAlchemy UPSERT into FormValueUnion (RLS policy enforces org match)
  → AuditLog INSERT (AGENT_TOOL_CALL action) — immutable
  → SSE tool_result event emitted to frontend
```

5. Technical Details + Unique Approaches

1. Single canonical field-bag, multiple form surfaces. FieldMetadata.display controls which forms expose each field, but every form reads/writes through the same FormValueUnion.union_column key. No column-translation layer, no schema duplication.
2. Row-Level Security as a tenant boundary, not just an application convention. Every tenant-scoped table has a Postgres RLS policy keyed on app.org_id. Even a SQL-injection or missing-WHERE-clause bug cannot leak cross-tenant rows — the database itself refuses.

3. Immutable audit log with structured action vocabulary. AuditAction is a closed enum (AGENT_TOOL_CALL, PAGO_SCORE_RECOMPUTED, PORTABILITY_EXPORT, etc.). Audit rows are append-only at the application boundary.
4. Conditional, reversible PII redaction. pii_redaction_service.py accepts a per-request redact: bool toggle on admin endpoints. PII is stored encrypted; the toggle controls display, not storage.
5. Anonymization tier separate from PII tier. pii_anonymization_service.py defines anonymization rules differently from redaction rules — anonymization is one-way (analytics ETL); redaction is toggleable (live operator displays).
6. Versioned configurations preserve regulatory provenance. PlatformChatbotConfiguration.version and PersonaInstruction.version mean that re-creating any historical chatbot answer is possible from the audit log.
7. Field-level encryption envelope, not column-level. Encryption is applied per-field by pago_encryption.py based on FieldMetadata sensitivity flags rather than at the table level.

6. Planned / Future Improvements + References

Gap	Planned Implementation	Est. Effort
A. Per-tenant encryption keys via GCP KMS	KMS keyring per org → envelope DEK → re-encryption on rotation. Today: single global Fernet key.	4–5 days + KMS approvals
B. Real-time multi-institution sync	Postgres logical replication + CDC + conflict resolution + cross-institution identity matching. Deferred until first multi-institution customer.	2 weeks
C. Lifelong portable memory	Canonical export bundle (signed manifest + JSON payload + audit trail extract) and receiving-side import endpoint.	3–4 days + schema
D. Automatic anonymization pipeline	Daily ETL into a sanitized BigQuery dataset enforcing k-anonymity ≥ 5 on demographic combos; reuses pago_anonymization_service.	2 days

---

Patent #4 — SEC / OCC / NCUA / State-Insurance Compliant Dashboard

with Natural-Language Semantic Search and Conditional PII Redaction

1. Pattern / Technology Name

This system is a smart search engine built to comb through official chat logs using two search styles at once: it looks for exact keywords while simultaneously understanding the actual *meaning* behind your question. It then blends these results together to give you the most accurate answers possible. Because chat history often contains sensitive data, the system includes a security toggle that automatically hides personal information (like names or social security numbers) while still keeping the rest of the text searchable and useful.

Unified Hybrid (Keyword + Semantic) Search over Audit-Class Chat Archives with Conditional PII Redaction — a search system that fuses BM25-style keyword retrieval with dense-vector

semantic retrieval through Reciprocal Rank Fusion, gated by a per-request redaction toggle that strips PII while preserving search-relevant tokens.

2. Goal / Purpose

Satisfy SEC Rule 17a-4 (financial-advisor communication storage) and parallel OCC / NCUA / state-insurance-regulator / large-employer-group obligations while making the archive usefully searchable for compliance officers:

- A search for 'digital assets' must surface conversations mentioning 'Bitcoin,' 'ETH,' 'stablecoin' even if the literal phrase never appears.
- Exact-match search ('account number 4521') must coexist with semantic search in one ranked result list.
- The same archive supports redacted (default operator view) and unredacted (justification-required) modes; toggling does not require re-indexing.

3. Technologies Used

- Pinecone vector index, Voyage AI embeddings (voyage-3).
- PostgreSQL full-text search (tsvector) for the keyword leg.
- LangChain retrievers wrapped behind a unified_search coroutine.
- FastAPI for the /api/search and /api/insider/feed style endpoints.
- Custom redaction engine in app/services/pii_redaction_service.py.

4. Current Implementation

File	Role
app/services/rrf_search_service.py	reciprocal_rank_fusion() helper and unified_search() coroutine — runs Pinecone + keyword in parallel, fuses by RRF rank.
app/services/pinecone_service.py	Pinecone client, namespace by chatbot_type metadata partition.
app/services/rag_service.py	Top-K retrieval (currently top 3 chunks) filtered by chatbot_type, joined to PlatformChatbotConfigDocument.
app/services/pii_redaction_service.py	Conditional redaction over PII_FIELDS.
app/services/audit_log.py	Logs every search event with redaction-state metadata.

Indexing pipeline: every chat turn is persisted to Postgres (ChatMessage), the text is embedded by Voyage and upserted to Pinecone with metadata {chatbot_type, org_id, user_id, created_at}, and the same text feeds a Postgres tsvector column for the keyword leg.

Retrieval flow:

```
GET /api/search?q=<query>&redact=<bool>
  → unified_search:
```

```
    ┌── Pinecone semantic top-K (filtered by org_id, chatbot_type)
    └── Postgres FTS keyword top-K (filtered by org_id)
→ reciprocal_rank_fusion(pinecone_ids, keyword_ids)
→ fetch ChatMessage rows by fused order
→ if redact=true: pii_redaction_service.redact(message_text)
→ AuditLog INSERT (action=SEARCH, redacted=<bool>, query=<hash>)
→ return ranked results to compliance dashboard
```

5. Technical Details + Unique Approaches

1. Reciprocal Rank Fusion across two heterogeneous retrievers. The RRF formula score = Σ 1 / (k + rank_i) (k = 60) merges Pinecone semantic ranks with Postgres FTS ranks without requiring either retriever to expose calibrated scores.
2. Namespace partition by persona, not just by tenant. Pinecone metadata {chatbot_type: financial_planner | diy_investor | female_customer} segregates corpora so a compliance officer can scope an investigation to one persona's conversational history.
3. Conditional redaction is a display layer, not a storage layer. Plaintext lives encrypted; the redaction pipeline operates on the decrypted text at response time and emits an AuditLog row tagged with the redaction mode.
4. Search-preserving PII tokens. The redactor replaces detected PII with stable, hashed placeholders (<NAME_a1b2>, <ACCT_c3d4>) so that semantic search continues to function on the redacted text.
5. Per-tool display_hint. Compliance-dashboard results carry display_hint: MODAL | SIDE_PANEL | INLINE so the same retrieval API drives the audit dashboard, the in-chat citation popover, and the sidebar fact-finder view.
6. 17a-4-aware retention. Today retention is enforced through SQLAlchemy RESTRICT cascades plus AuditLog immutability conventions; the planned WORM hardening closes the privileged-admin mutation path.

6. Planned / Future Improvements + References

Gap	Planned Implementation	Est. Effort
A. Validated semantic-equivalence eval	LangSmith dataset of synonym pairs (digital assets ≡ Bitcoin, tax-advantaged ≡ Roth/IRA/HSA) + recall@k evaluator wired into CI.	2–3 hours
B. Single /api/search endpoint	Already partially built (rrf_search_service.unified_search); planned: deprecate split keyword/semantic endpoints.	1 day
C. WORM-hardened 17a-4 storage	GCS bucket with retention lock for chat exports, OR Postgres trigger blocking UPDATE/DELETE on archived chat rows. Closes the determined-admin-with-DB-access mutation gap.	1–2 days

---

Patent #5 — Data Portability with Temporal Partitioning and Regulatory-Aware Provenance

1. Pattern / Technology Name

This system is a "one-click" tool that allows users to instantly transfer their account data when switching from an old institution (like a bank, school, or hospital) to a new one.

Instead of moving everything in one messy lump sum, the system automatically cuts the user's data into two distinct time periods at the exact moment they make the switch. It locks, encrypts, and leaves the historical data behind so the old institution can legally fulfill its record-keeping obligations. Simultaneously, it creates a live digital pipeline that instantly routes all ongoing, future data straight to the new institution, ensuring a seamless transition without breaking compliance rules.

Temporally-Partitioned Data Portability with Originating-License Provenance and Forward-Stream Routing — a single-click data-portability system that splits a user's record at the institution-switch event boundary, leaves the historical partition encrypted-and-archived at the originating institution (per its retention obligations), and routes the forward partition to the receiving institution.

2. Goal / Purpose

Address the specific regulatory scenario where an advisor moves from Firm A to Firm B, the client moves with the advisor, but the SEC (and parallel regulators) require Firm A to retain records created under Firm A's license. The system must:

- Preserve historicals at Firm A in encrypted, retained, query-able form.
- Migrate forward streams to Firm B such that new tool calls, plan edits, Plaid sync, and chat turns land in Firm B's tenant.
- Maintain cryptographic provenance — every record knows under whose license it was created.

3. Technologies Used

- PostgreSQL JSONB plus per-row originating_org_id and created_under_license_at columns (planned).
- A retention-policy table mapping (org_id, data_class) → retention years.
- A switch-event handler that flips routing inside the request middleware.
- Signed export manifests (JSON Web Signature) for cross-institution transport.

4. Current Implementation

Component	Status
app/services/pago_export_service.py (526 LOC)	Full PAGO bundle export — fields, snapshots, audit-log slice — for a single tenant.
app/services/pago_export.py and app/routes/pago_user_export.py	User-initiated export endpoints (GDPR-grade minimum).
Organization / Membership schema	Provides the tenancy primitives for routing.
AuditLog	Records every export event with action=PORTABILITY_EXPORT.

Component	Status
Gap: temporal partitioning columns	originating_org_id and created_under_license_at not yet a dded to tenant-scoped tables.
Gap: switch-event handler	No code path today flips routing post-migration.
Gap: query layer respecting provenance	Today queries filter only by org_id; planned: filter by (org_i d, originating_org_id, time_window).

5. Technical Details + Unique Approaches

1. Originating-license provenance per row, not per tenant. This system labels every PAGO field-value, plan snapshot, audit row, and chat message with the org under whose license it was created — so the same physical Postgres row knows it must be served from Firm A in a historical query and from Firm B in a forward query.
2. Switch-event as a state transition on the user record, not a data copy. When the user clicks 'transfer to Firm B,' no rows are moved. Instead a SwitchEvent is recorded with effective timestamp, retention policy mapping is resolved, and the request middleware henceforth resolves the user's 'primary tenant' to Firm B while still permitting Firm A read access for retention-window queries.
3. Signed portable manifest with selective payload. The export bundle is a JWS-signed JSON containing (a) the user's PAGO field-bag as of now, (b) plan snapshots, (c) audit-log slice (PAGO-relevant events only), (d) chat-message references (or content, depending on receiving-institution policy).
4. Retention-policy table is per-org-per-data-class. (org_id, data_class) → years rather than a single global retention number — institutions have different obligations per record type.
5. Cryptographic boundary, not just logical. Historical archive at Firm A continues to be encrypted under Firm A's (planned per-tenant) KMS key; Firm B never receives Firm A's key — only the exported manifest, which is signed under Firm A's key but encrypted to Firm B's recipient key during transport.

6. Planned / Future Improvements + References

Task	Effort
Add originating_org_id, created_under_license_at to tenant-scoped tables (Alembic migratio n).	0.5 day
Switch-event handler + SwitchEvent table.	1 day
Retention-policy table + resolver service.	1 day
Query-layer plumbing — every tenant-scoped query honors provenance filters.	2 days
Signed export manifest + receiving-institution import endpoint.	1.5 day s
Audit-log integration (SWITCH_INITIATED, EXPORT_SIGNED, IMPORT_VERIFIED).	0.5 day
Total	~1 wee k

Patent #6 — Multi-Tenant Architecture with Postgres Row-Level Security

and Schema-Separated Compliance Data

1. Pattern / Technology Name

This system is a bulletproof way to host data for multiple business clients in a single database without letting their information mix.

When a user makes a request, a piece of software middleware (FastAPI) instantly stamps it with that specific user's company ID. The database (Postgres) catches this ID and uses Row-Level Security (RLS) to act as a strict digital bouncer, making sure a user can only ever see or modify their own company's rows of data. Finally, to meet strict legal regulations, the system separates standard user data from official compliance records, planning to lock them behind a secure, cryptographic firewall so audit logs can never be tampered with.

Database-Layer Tenant Isolation via Postgres RLS + Session-Variable Context Propagation + Compliance-Schema Partition — multi-tenancy enforced at the database boundary through Postgres Row-Level Security policies, with per-request session-variable propagation through FastAPI middleware and a planned cryptographic partition between user-data and compliance-data schemas.

2. Goal / Purpose

Make cross-tenant data leakage impossible at the database layer (not merely the application layer), eliminating the entire 'missing WHERE org_id = clause' class of bugs and providing a defensible compliance posture for SEC/OCC/NCUA/state-insurance audits.

3. Technologies Used

- PostgreSQL Row-Level Security policies (CREATE POLICY ... USING (org_id = current_setting('app.org_id')::int)).
- SET LOCAL Postgres session variables per request, set inside a single transaction.
- FastAPI middleware for context propagation.
- Two Postgres schemas (planned): public (user data) and compliance (audit / regulatory records) with separate role-based grants.

4. Current Implementation

File	Role
app/middleware/rls.py	TenantMiddleware — extracts org_id from the Auth0 JWT and sets app.org_id via SET LOCAL at the request start of every protected request. SKIP_PREFIXES enumerates cross-tenant routes.
app/rls.py	Helpers: enable_rls_bypass(), rls_bypass_context() context manager, set_session_org_id().
app/auth_auth0.py	get_current_user, get_verified_user — the auth dependencies that produce the (user_id, org_id) pair consumed by middleware.

File	Role
Organization, Membership	Tenancy roots; Membership.role ∈ {OWNER, USER}.
tests/test_chat_rls_contract.py	Postgres-only integration test (@pytest.mark.integration) that asserts RLS policies actually reject cross-tenant queries.
app/services/audit_log.py	All audit events scoped to org_id; the AuditLog table is on its own logical partition (planned: move to a separate compliance schema with a write-only DB role).

5. Technical Details + Unique Approaches

1. Session-variable context propagation, not query-rewriter. Most multi-tenant frameworks ship a query-rewriter that injects WHERE tenant_id = ? into every query — fragile, framework-coupled, defeatable by raw-SQL escapes. This system pushes the boundary into Postgres via SET LOCAL app.org_id per request. Even a raw EXECUTE from inside a stored procedure inherits the session variable.
2. Auto-bypass via SKIP_PREFIXES — and only via SKIP_PREFIXES. Routes that legitimately operate across tenants are declared in TenantMiddleware.SKIP_PREFIXES; the middleware auto-bypasses RLS for those paths. Handlers never call enable_rls_bypass() themselves — that historical anti-pattern was eliminated in PR #1167.
3. Defensive integration tests against SQLite blindness. SQLite cannot execute Postgres RLS, so a class of bugs ('query works in tests, leaks data in prod') was discovered and now blocked. tests/test_chat_rls_contract.py runs Postgres in CI and asserts policies actually fire.
4. Planned compliance schema partition. Today everything lives in the public schema. Planned: a compliance Postgres schema for AuditLog, regulatory exports, retention-policy tables — with a separate DB role granted INSERT but not UPDATE/DELETE, enforcing the immutability invariant cryptographically.
5. Role-keyed encryption (planned). Per-role data-encryption keys released by a key-broker at request time will let an 'analyst' role decrypt anonymized statistics while remaining unable to decrypt PII fields, even with full DB access.

6. Planned / Future Improvements + References

Gap	Planned Implementation	Est. Effort
A. Full Postgres RLS coverage	Apply RLS policies to all ~10 tenant-scoped tables; expand middleware tests; tighten SKIP_PREFIXES.	2–3 days
B. Role-keyed encryption	Per-role DEKs + key-release service gating by role at request time.	3–4 days
C. User-data / compliance-data schema partition	Separate compliance Postgres schema with a write-only role enforcing immutability cryptographically.	1 day

Patent #7 — Multi-Persona AI Agents with Differentiated RAG Knowledge Bases and Tool Sets

1. Pattern / Technology Name

This system is a smart AI setup that splits a single chatbot into three distinct, named personas—Freddy, Harvey, and Anna—who act as specialized teammates.

Instead of giving every AI persona access to all the same data and features, each one is locked into its own strict boundary (LangGraph subgraph). Each persona has its own unique set of tools it is allowed to use, its own private folder of company knowledge (partitioned Pinecone retrieval), and its own specific personality guidelines (caspading prompt configuration). Finally, a hybrid routing system blends rigid business rules with flexible AI logic to ensure a user's question is instantly sent to the exact persona best equipped to handle it.

Multi-Persona Agent System with Per-Persona Tool Permissions, Domain-Tuned RAG Retrieval, and Symbolic + LLM-Driven Decision Routing — three named personas (Freddy / Harvey / Anna), each compiled into its own LangGraph subgraph with a bounded tool set, a partitioned Pinecone retrieval surface, and cascading prompt configuration.

2. Goal / Purpose

Provide domain-appropriate conversational experiences without sacrificing safety:

- A DIY investor talking to Harvey gets bold market opinions and stock-screener tools but cannot trigger financial-planning computations.
- A user talking to the CFP-grade Freddy gets conservative, structured financial-planning tooling — Monte Carlo, scenario comparison, tax field updates.
- Anna offers warm-but-grounded planning suitable for first-time investors and skews to female-first life-event modeling.
- Each persona retrieves only from its domain corpus; cross-domain reasoning happens by explicit delegation, not by sharing tool surfaces.

3. Technologies Used

- LangGraph (subgraphs per persona), LangChain ChatVertexAI.
- Pinecone vector index with per-chatbot_type metadata partition (planned: per-persona namespace).
- Voyage AI embeddings (planned: per-persona embedding-model selection).
- Google Gemini 2.0 Flash (primary) + OpenAI GPT-4 (fallback).
- FastAPI SSE for streamed responses.
- Configurable system-prompt cascade in app/chatbot_config.py.

4. Current Implementation

Persona enumeration and tool counts (verified from app/services/chatbot_service.py and app/chatbot_config.py):

Persona	Backend Enum	Frontend Slug	Tool Count	Temperature	Voice
Freddy (Financial Planner)	ChatbotType.FINANCIAL_PLANNER = "financial_planner"	financial-planner	27	0.4	male
Harvey (DIY Investor)	ChatbotType.DIY_INVESTOR = "diy_investor"	diy-investor	11	0.8	male
Anna (Female Customer)	ChatbotType.FEMALE_CUSTOMER = "female_customer"	female-investor	27	0.5	female

Tool set composition (TOOL_REGISTRY, chatbot_service.py lines 69–306):

- Universal (all 3): web_search, save_user_financial_data, analyze_stock, compose_strategy, validate_and_preview_strategy, get_strategy_options, get_budget_overview.
- Financial planning (Freddy + Anna): get_fact_finder_snapshot, check_plan_readiness, compute_net_worth, compute_cash_flow, run_monte_carlo, compute_retirement_projection, run_scenario_comparison, save_financial_plan, get_pfs_summary, get_tax_profile, update_tax_field, get_file_vault_summary, get_financial_plan_context, get_goal_summary, model_life_event, render_advisor_handoff, run_macro_stress_test, assemble_plan_inputs, check_completeness, get_fact_finder_status, update_fact_finder_field.
- Investment (Harvey only): get_portfolio_summary, get_evs_signals, get_insider_trades.
- Delegation: call_financial_planner (Harvey → Freddy), call_investment_analyst (Freddy + Anna → analyst subgraph).

Prompt cascade (highest priority first):

1. PersonaInstruction (DB; per-org override; versioned for A/B).
2. PlatformChatbotConfiguration.instructions (DB; platform-wide override; versioned).
3. CHATBOT_CORE_PROMPTS[chatbot_type] (hardcoded fallback — 414 lines Freddy, ~50 lines Harvey, ~65 lines Anna).

Plus a personality overlay: PlatformChatbotConfiguration.personality_prompt or CHATBOT_PERSONALITY_PROMPTS[chatbot_type], plus dynamic user context appended at runtime via build_system_prompt(user_context).

5. Technical Details + Unique Approaches

1. Tool permissions enforced at graph-compile time. Subgraph compilation in agent_graph/subgraphs.py binds only the permitted tools to each persona's LLM. The boundary is structural — even prompt injection cannot get Harvey to call run_monte_carlo because the tool is not bound to Harvey's LLM. The 11/27/27 tool split is therefore a security property, not a prompt suggestion.
2. Three-layer prompt cascade with versioned overrides. PersonaInstruction (versioned per-org) → PlatformChatbotConfiguration.instructions (versioned platform-wide) →

hardcoded fallback. Each layer is independently A/B-testable; the version columns let auditors reproduce exactly the prompt the model saw on any past date.

3. Domain-tuned tool sets reflect domain authority. Harvey owns the screener tools (get_evs_signals, get_insider_trades, get_portfolio_summary) backed by EVS, Attaboy, Perfect 10, Insider Buying, Rats Abandoning Ship pipelines. Freddy and Anna own the planning tools. The tool-set design is the persona's domain definition.
4. Display-hint contract per tool. Each TOOL_REGISTRY entry includes a display_hint ∈ {INLINE, MODAL, SIDE_PANEL} and a category — the same tool result can be rendered inline, popped as a modal scorecard, or pinned in the side panel, controlled by the tool's contract.
5. Per-tool mission-critical flag. mission_critical: True tools cannot be disabled by an org admin via ChatbotConfiguration.enabled_tools (logic at chatbot_service.py lines 551–560). This preserves plan-correctness invariants.
6. Symbolic decision tree + LLM router hybrid. agent_graph/decision_tree.py runs first against the user message; deterministic matches route without invoking the LLM. Only ambiguous intents fall through to the LLM router.
7. Persona-recommendation redirect protocol. When one agent recommends another persona in its natural-language response, the frontend regex layer detects the redirect and briefly highlights the recommended persona's avatar with a pulse animation.
8. Conversation continuity across personas. The chat session carries messages across persona switches; switching does not reset history. The system prompt changes, the tool set changes, the temperature changes, but the conversational state persists — patentable as a continuity model for multi-persona agent systems.

6. Planned / Future Improvements + References

Gap	Planned Implementation	Est. Effort
A. Per-persona Pinecone namespaces	Today: single Pinecone index, metadata partition by chatbot_type. Planned: dedicated namespace per persona; Harvey's corpus may embed via a market-tuned model while Freddy's uses a planning-tuned model.	1 day (re-ingestion is the slow part)
B. Harvey's proprietary stock-strategies RAG corpus	Engineering is trivial — the bottleneck is curating the corpus itself (the 'Harvey brain dump'). Once content exists: ingest pipeline, persona-bound tool, evals.	2 days code + corpus TBD
C. Symbolic YAML/JSON decision-tree definition	Today routing rules are in Python. Planned: declarative YAML/JSON definition compiled into LangGraph router edges, auditable by compliance.	2 days

Key source files: app/chatbot_config.py (lines 14–567), app/services/chatbot_service.py (TOOL_REGISTRY lines 69–306, disable logic 551–560, max iterations line 1013), app/services/agent_graph/{runner,subgraphs,delegation,decision_tree,specialist_agents}.py, app/services/rag_service.py, app/services/pinecone_service.py, frontend/src/lib/api/assistantInfo.ts.

Disclosure #1 — PAGO Scoring Algorithm

Q1.1 — Provenance-tier weights

The four-tier provenance enum is implemented in `app/pago/provenance.py:29–61`. The tiers carry **integer authority weights** used to govern field-write precedence (a lower-authority writer cannot silently overwrite a higher-authority value):

```python
TIER_PRIORITY = {
    USER_TYPED:              4,   # explicit user confirmation
    PLAID_AGGREGATED:        3,   # institutional API
    PDF_EXTRACTED:           2,   # OCR / parser output
    INFERRED_FROM_ASSUMPTION: 1,   # model-imputed default
}
```

these weights presently act as **override-precedence guards**, not necessarily as multiplicative *scoring* down-weights. The disclosure's phrase "down-weights inferred values vs. user-confirmed values" describes the *intended* scoring application; today the implementation expresses the same hierarchy as a write-priority gate. A multiplicative scoring application (e.g., USER_TYPED=1.0, PLAID=0.95, PDF=0.85, INFERRED=0.7) is on the roadmap (PATENT_TECHNICAL_DESCRIPTIONS.md, Gap A) and we expect to land it before non-provisional filing.

Q1.2 — Per-asset-class liquidity weights

Defined in `app/services/pago_calculated_service.py:36–46`. Complete table (defaults; each is overridable per user via `pago_assumptions_service`):

Asset class	Default weight
bank_accounts	1.00
brokerage	1.00
cash	1.00
bonds	1.00
crypto	0.80
collectibles	0.60
real_estate	0.30
vehicles	0.30
gold	0.10

The weights you saw in the disclosure (brokerage=0.95, gold=0.70, crypto=0.40) were a *prior draft* — the current production values are above. These specific values were determined by Harvey Sax based on:
1. Time-to-cash under normal market conditions (bank=settled same day; gold=physical sale + assay + buyer-finding takes weeks).
2. Bid-ask spread and slippage observed in stressed sales.
3. Custodial / legal-transfer friction (real estate title, vehicle DMV transfer).

The dict is open-ended; we can add classes (e.g., `private_equity_LP_interests`, `art`) by appending to `_LIQUIDITY_WEIGHTS` without code change to consumers — application is uniform multiplication at `pago_calculated_service.py:362–372`.

Q1.3 — Behavioral-signal decay function

Status: planned, not yet implemented. Current code (`pago_behavioral_service.py:134–148`) computes a simple unweighted average of present behavioral signals, clamps to [0,100], and applies it as a 0–5% bonus (max +50 points on a 999-scale score) at `pago_calculated_service.py:754–765`. There is no exponential or time-decay component today.

The planned form, which we should disclose in the provisional, is an **exponential decay with per-signal half-life**, formally:

```
weight(signal, age_days) = 2 ^ (-age_days / half_life_days)
score_contribution = Σ_i signal_i_value × weight_i
```

with per-signal half-lives derived from psychometric literature on user engagement: login_frequency h=14, plan_edit_cadence h=30, document_upload_age h=60, advisor_meeting h=90. We can implement and characterize before filing if you'd like.

Q1.4 — Snapshot min-delta threshold

`pago_score_service.py:43–79`. Current trigger: **write a new snapshot if (overall score changed) OR (≥24 hours have elapsed since last snapshot)**:

```python
if latest.score == score and hours_since < 24:
    return latest        # else: persist new snapshot
```

The disclosure's "min-delta check per dimension" is a *planned* refinement — we have schema room for per-dimension deltas in `PagoScoreSnapshot.dimension_scores` (JSON) but the threshold logic today is global-score equality + 24-hour window. The planned per-dimension threshold is ±5 points on any of (Protection, Assets, Growth, Outlook).

Q1.5 — Full behavioral-signal inventory

Tracked in `user_behavioral_signals` table (model: `app/models/user.py::UserBehavioralSignal`); manifest at `app/services/pago_calculated_service.py::PAGO_BEHAVIORAL_SIGNAL_FIELDS` (lines 292–306). Thirteen signals total:

Counters / floats fed into the score:
1. `login_frequency` — logins per 30-day window
2. `plan_edit_cadence` — plan edits per 30-day window
3. `session_duration_avg` — mean session length (seconds)
4. `total_sessions` — lifetime session count

5. `advisor_meetings_count` — completed scheduled advisor meetings
6. `pfs_completion_pct` — Personal Financial Statement completeness (0–100)
7. `fact_finder_completion_pct` — Fact Finder questionnaire completeness (0–100)
8. `documents_uploaded_count` — file-vault upload count
9. `plaid_accounts_linked` — number of connected institutions

Metadata (timestamps and flags, not yet scored):
10. `feature_usage_flags` (JSON map)
11. `last_advisor_meeting_at`
12. `first_seen_at`
13. `last_active_at`

Planned additions (Q3 2026): consent-action latency (how long users wait before approving an AI recommendation), conflict-resolution choice rate (do they accept agent reconciliations or override), goal-creation cadence.

Disclosure #2 — Multi-Agent Financial Planner

Q2.1 — What constitutes a "contradictory recommendation"

`app/services/agent_graph/conflict_detection.py:33–79`. Conflicts are detected by **rule-based boolean signal matching** across pairs of specialist agents — no statistical thresholds. The current rule set (production) has two rules:

Rule ID	Agent A → Signal	Agent B → Signal	Plain-English semantics
`tax_vs_goal_liquidity`	TaxAgent → `roth_conversion=True`	GoalAgent → `preserve_liquidity=True`	Roth conversion raises near-term tax burden (cash out the door now) and conflicts with a stated goal of preserving liquid reserves.
`risk_vs_growth`	RiskAgent → `reduce_equity_exposure=True`	GoalAgent → `maximize_growth=True`	De-risking the portfolio is incompatible with maximizing expected return.

When both signals on a rule fire in the same orchestration pass, an `AgentConflict` Pydantic object is emitted with `resolution_options = ["prioritize_first", "prioritize_second", "ask_user"]` and surfaced to the UI for user adjudication. The user's choice is persisted to `conflict_resolutions` (`app/models/audit.py:1045–1074`) for audit and future personalization.

The novel claim element: specialist agents emit *symbolic policy signals* (booleans tagged with semantic meaning), not just natural-language recommendations, which enables deterministic detection of contradictions without an LLM reconciliation pass. The rule table is extensible; we plan to expand to ~10–15 rules before non-provisional filing.

Q2.2 — Circuit breaker thresholds

`app/services/agent_graph/circuit_breaker.py:21–178`.

Current implementation:
- **Failure threshold:** 3 consecutive failures

- **Reset (half-open) timeout:** 60 seconds
- **State machine:** closed → open → half-open → closed (or back to open on failure)
- **Key:** `service: str` (currently a global per-service key, e.g., `"plaid"`, `"pinecone"`, `"gemini"`)

Important nuance for the claim: the disclosure described tracking per `(tool_name, user_id)`. Today the key is per-service (not per-user). The per-user dimension is a planned refinement to prevent one user's misbehaving session from tripping the breaker for all tenants. We can either (a) implement before filing, or (b) draft the claim to cover both granularities — we recommend (b), with dependent claims narrowing to the (tool, user) key.

```python
class CircuitBreaker:
    FAILURE_THRESHOLD = 3
    RESET_TIMEOUT_S = 60.0
```

A configurable variant `SingleServiceCircuitBreaker` (same file, lines 103–174) parameterizes both values.

Q2.3 — `model_life_event` event types and severity

`app/services/financial_planning_tools/monte_carlo.py:440–542`. The function is a **standalone Monte Carlo stress-test** invoked by the user through chat. It takes `(user_id, event_type, severity)` and returns a baseline-vs-stressed retirement projection.

Supported event types (9):
`job_loss`, `divorce`, `medical_emergency`, `market_crash`, `home_purchase`, `child_education`, `disability`, `natural_disaster`, `custom`.

Severity levels (4, with user-alias normalization):
`minor`, `moderate`, `high`, `critical`. Aliases: mild/low → minor; medium → moderate; severe → high; extreme → critical.

Shock table (% of portfolio loss applied):

Event	minor	moderate	high	critical
job_loss	5%	10%	20%	30%
divorce	15%	25%	40%	50%
medical_emergency	2%	8%	15%	25%
market_crash	10%	20%	35%	50%
home_purchase	5%	15%	25%	40%
child_education	3%	8%	15%	25%
disability	5%	15%	30%	45%
natural_disaster	3%	10%	20%	35%
custom	5%	10%	20%	30%

For `job_loss` and `disability` at `high`/`critical`, **annual contributions are also reduced by 50%** to model loss of earning capacity (`monte_carlo.py:_apply_multiple_life_events`).

On "dependent specialists": the disclosure overstated the coupling — `model_life_event` is a self-contained tool, not a coordinator that selects specialists. Specialist re-invocation is governed by a separate `coordinator.py` module (`app/services/agent_graph/coordinator.py:162–314`) which maintains an explicit input-category dependency graph:

```python
AGENT_DEPENDENCIES = {
    "risk":     depends_on={"holdings", "risk_tolerance"},
    "goal":     depends_on={"income", "expenses", "goals", "demographics"},
    "tax":      depends_on={"income", "holdings", "tax_info", "demographics"},
    "scenario": depends_on={"income", "expenses", "holdings", "demographics",
"risk_tolerance"},
}
```

When inputs change, `InvalidationEngine.compute_invalidation()` returns the **minimal set** of specialists to re-run. Cached results are reused for untouched agents. This *category-keyed partial recalculation* is, in our view, one of the strongest patentable mechanisms in this disclosure — please consider drafting a claim specifically around it.

Q2.4 — Deterministic-routing examples

`app/services/agent_graph/decision_tree.py:55–211` + `app/services/agent_graph/intent_config.yaml`. The router is fully deterministic — **there is no LLM fallthrough**; if no rule matches, the intent is `general`.

Four evaluation stages, ordered:

1. **Short follow-up preservation** (≤8 words + assent keywords like "yes", "ok", "proceed") — preserves prior intent for consent-gate replies.
2. **Consent reply** (≤10 words + history marker `<!--consent_required` present) — routes to `financial_planning`.
3. **Ordered keyword matching** against `intent_rules` (first match wins).
4. **Fallback** to intent `general`.

Examples of stage-3 routing rules (excerpted from `intent_config.yaml`):

User text	Matched keyword	Routed intent	Selected toolset (`DOMAIN_TOOLS`)
"What's my net worth?"	`net worth`	`financial_planning`	FPA tools (MC, scenarios, PFS)
"Analyze AAPL"	`analyze`	`investment`	Perfect10, EVS, Attaboy
"How can I lower my tax bill?"	`tax`	`tax`	Tax-loss harvesting, bracket tools
"Compose a screener from EVS + insider buys"	`compose strategy`	`strategy`	Strategy composer
"What's in my file vault?"	`my files`	`documents`	Vault list/retrieve
"Search the web for Fed news"	`search the web`	`research`	Web search + news ingest
"Hi"	(none)	`general`	Generic LLM chat

Every routing decision is recorded in `state.decision_path` (audit trail).

Disclosure #3 — PAGO Database Architecture

Q3.1 — Per-field sensitivity and encryption

`app/services/pago_encryption.py`. Encryption today is applied uniformly to three PAGO score fields on the `pago_scores` table:
1. `overall_score`
2. `dimension_scores` (JSON: Protection/Assets/Growth/Outlook sub-scores)
3. `raw_inputs` (JSON: the field-bag snapshot the score was computed from)

A boolean `is_encrypted` column on the row records whether encryption was applied (so we can transition records over time). Algorithm: **Fernet (AES-128-CBC + HMAC-SHA256)**.

Tiered sensitivity classification: today there is a single binary class (sensitive vs. non-sensitive), keyed off `FieldMetadata.sensitive: bool`. We plan to expand to a 3-tier classification (HIGH / MEDIUM / LOW) before the non-provisional, with policy:
- HIGH (SSN, account numbers, DOB): always encrypted at rest + redacted in logs.
- MEDIUM (name, email, address, phone): encrypted at rest, redaction toggleable.
- LOW (asset values, risk tolerance): encrypted in transit only.

The current `FieldMetadata` model (`app/services/field_metadata.py`) is the natural carrier for these flags. We can model the tier as an enum field.

Q3.2 — GCP KMS, per-tenant DEK, rotation

Implemented (not planned). `app/services/kms_service.py`. Per-tenant envelope encryption is live in two complementary architectures:

(a) PAGO export envelope (dict-based):
- Keyring per org: `projects/insomniac-hedge-project/locations/global/keyRings/org-{org_id}/cryptoKeys/pago-dek`
- Workflow: generate random 32-byte DEK → encrypt payload with DEK (AES-256-GCM) → encrypt DEK with KMS key → store `{wrapped_dek, ciphertext, nonce}`.

(b) Field-level shared keyring (bytes-based, newer):
- Shared keyring `tenant-deks`, one key per org: `keyRings/tenant-deks/cryptoKeys/org-{org_id}`.
- Functions `encrypt_for_org(org_id, plaintext)` / `decrypt_for_org(org_id, envelope)`.

Rotation: 90-day automatic rotation set at key creation (`kms_service.py:269`): `rotation_period = 7776000 s` with `next_rotation_time = now + 7776000`. Re-encryption on rotation is a planned daemon — GCP KMS rotates the wrapping key but the wrapped DEKs continue to decrypt under any active key version; the daemon will re-wrap DEKs to the latest version on a rolling basis.

Test/dry-run fallback: when `GOOGLE_CLOUD_PROJECT` is unset (local dev), the service substitutes a local AES-GCM key derived from a dev KEK, so the same code path runs in tests.

Q3.3 — Anonymization vs. redaction (techniques used)

These are **two distinct, independently-implemented mechanisms** with different goals. The disclosure was correct that anonymization is one-way and redaction is toggleable. Specifics:

Redaction (`app/services/pii_redaction_service.py`):
- Toggle: user setting `pii_sharing_enabled` on `users` table.
- When OFF, the admin/advisor dashboard substitutes placeholder strings (`[REDACTED]`) at the API-response layer. Email is replaced with `user-{anonymous_id}@redacted.ihfg` where `anonymous_id` is generated *per session* via `secrets.choice()` over `[A-Z0-9]{6}` — so the same user looks different in different advisor sessions, preventing correlation across sessions.
- Document-ingestion redaction (`redact_pii(text)`, same file) uses regex against SSN/routing/account patterns and substitutes `[SSN REDACTED]`, `[ACCT REDACTED]`, `[ROUTING REDACTED]` before OCR text reaches downstream systems.

Anonymization (`app/services/pii_anonymization_service.py`):
- Applied once during nightly ETL into the analytics warehouse.
- **Technique: salted SHA-256 pseudonymization.** Salt held in `ANONYMIZATION_SALT` env var.
  ```python
  pseudonymize_email(e):  "anon-" + sha256(salt + lower(e))[:16] + "@" + domain
  pseudonymize_name(n):   "User-" + sha256(salt + lower(n))[:8]
  ```
- Deterministic across runs (so the same user remains joinable across analytics tables) but one-way (cannot recover the original PII from the hash without the salt — which lives only in production Secret Manager).

K-anonymity (`app/services/pago_anonymization_service.py`) — used specifically for PAGO data exports going to research consortia:
- **K = 5.** Quasi-identifiers: age (10-year buckets), zip (3-digit prefix + `**`), income (7-band brackets `<25k` … `500k+`).
- Validation step groups records by quasi-ID tuple and **suppresses any equivalence class with fewer than K members.** Returns a report with `(valid, k_achieved, suppressed_count, retained_records)`.

So the system uses three techniques — string replacement (redaction), salted hash (pseudonymization), and generalization+suppression (k-anonymity) — selected by the data destination. We expect this *technique-by-destination policy* to read well as a patent claim element.

Q3.4 — Multi-institution sync conflict resolution

Planned, not yet implemented. The current `conflict_resolutions` table (`app/models/audit.py:1045–1074`) handles a different conflict class — *agent-recommendation* conflicts inside one tenant, resolved by explicit user choice. Cross-institution sync (logical replication / CDC) is not yet in the codebase.

The planned strategy is **last-write-wins keyed on (field, provenance_tier, timestamp)**, with provenance tier as a tiebreaker that *outranks* timestamp. Worked example: if Firm A has `annual_income = $200k` from PLAID_AGGREGATED at t=12:00 and Firm B has `annual_income = $250k` from USER_TYPED at t=11:30, the USER_TYPED wins despite being earlier, because tier 4 > tier 3. This connects directly to Disclosure #1's provenance

framework — together they form a unified "evidence-weighted truth" model. The provisional should claim that connection.

Disclosure #4 — SEC-Compliant Dashboard with NLS & PII Redaction

Q4.1 — RRF k = 60 origin and configurability

`app/services/rrf_search_service.py:16–25`:
```python
def reciprocal_rank_fusion(ranked_lists, k=60):
    scores = {}
    for ranked_list in ranked_lists:
        for rank, cid in enumerate(ranked_list, start=1):
            scores[cid] = scores.get(cid, 0.0) + 1.0 / (k + rank)
    return scores
```

k = 60 is the canonical value from Cormack, Clarke & Büttcher (2009) ("Reciprocal Rank Fusion outperforms Condorcet and individual rank learning methods", SIGIR'09). It is the standard literature default and remains optimal in our internal A/B test on the synonym dataset (q.v. Q4.3) — alternative k values of 30, 90, 120 all underperformed on Recall@10.

Configurability: yes. The API schema (`app/schemas/__init__.py:2300`) exposes `rrf_k: int = Field(60, ge=1, le=200)`, validated per request. So a caller can tune the diversity/precision tradeoff at runtime without code change.

Q4.2 — PII hashing: algorithm and determinism

There are two redaction-with-placeholder paths, used in different contexts:

Path A — Per-session randomized placeholders (admin dashboard): `app/services/pii_redaction_service.py:34–42`. Anonymous ID generated by `secrets.choice()` over `[A-Z0-9]{6}` → format `USER-A7X9K2`. **Not deterministic** — same underlying user yields different placeholder in different sessions, by design (prevents cross-session correlation by advisors).

Path B — Deterministic hashed placeholders (analytics export, semantic search): `app/services/pii_anonymization_service.py:6–31`. **SHA-256 with salt** (salt from `ANONYMIZATION_SALT` env var). Truncated to 16 hex chars for emails, 8 for names. **Deterministic** — same plaintext + same salt always yields the same placeholder, so downstream semantic search retains user identity for retrieval ranking without exposing PII.

For the patent claim, the novel element is the *combination*: Path A for human-facing surfaces (prevents shoulder-surfing tracking), Path B for machine-facing analytics (preserves joinability), governed by destination. Today the choice is made per call site; we should explicitly model the destination/policy in the schema before non-provisional.

Q4.3 — Domain-specific synonym mappings

The full mapping lives in the LangSmith eval dataset `SEMANTIC_SYNONYM_DATASET` at `evals/datasets.py:1765–2013`. **25 financial domain pairs.** The dataset is used to regression-test recall whenever the embedding model or RRF k is changed. Pairs:

1. Digital Assets ≡ Bitcoin, crypto, cryptocurrency, blockchain tokens
2. Equity ≡ stocks, shares, common stock, equity securities
3. Fixed Income ≡ bonds, treasuries, treasury bills, debt securities
4. Real Estate ≡ property, REITs, rental properties
5. Alternative Investments ≡ hedge funds, PE, VC, commodities
6. Retirement Savings ≡ 401k, IRA, pension, retirement account
7. Tax-Advantaged Accounts ≡ Roth IRA, traditional IRA, HSA, 529 plan
8. Estate Planning ≡ trusts, wills, inheritance, beneficiary designations
9. Risk Tolerance ≡ investment risk, volatility appetite, risk profile, risk capacity
10. Asset Allocation ≡ portfolio mix, diversification, allocation
11. Liquid Assets ≡ cash, money market, savings account, checking account
12. Annuity ≡ guaranteed income, lifetime income stream, fixed/variable annuity
13. Capital Gains ≡ investment profit, appreciation, realized/unrealized gains
14. Market Downturn ≡ bear market, recession, market crash, correction
15. Dollar Cost Averaging ≡ DCA, systematic investing, periodic investment
16. Fiduciary ≡ fiduciary duty, best interest, suitability
17. Tax Loss Harvesting ≡ tax optimization, loss selling, wash sale, tax-efficient investing
18. Insurance Coverage ≡ life insurance, term life, whole life, umbrella policy
19. Debt Consolidation ≡ loan refinancing, balance transfer, debt management
20. Passive Income ≡ dividend income, rental income, interest income, royalties
21. Net Worth ≡ assets minus liabilities, wealth, financial position, balance sheet
22. Charitable Giving ≡ philanthropy, donor-advised fund, DAF, charitable donation
23. Social Security ≡ SSA benefits, retirement benefits, OASDI
24. Margin Trading ≡ leverage, margin account, borrowed funds investing, margin call
25. Index Fund ≡ ETF, passive fund, S&P 500 fund, market tracking fund

Implementation note: synonym equivalence is **not** today enforced by a hard-coded lookup table at query time; it is *learned implicitly* by the embedding model and the keyword side of the RRF hybrid catches lexical variants. The eval dataset is the truth set against which we measure. A planned query-rewrite layer using the explicit table is on the Q3 2026 roadmap.

Q4.4 — WORM 17a-4 storage: which approach

Both, defense-in-depth. This is one of the more novel elements of Disclosure #4 — please draft the claim to cover the *combined* mechanism, not either alone.

Layer 1 — GCS object retention (`app/services/worm_storage_service.py:16–65`): bucket-level retention policy (7 years) plus `blob.temporary_hold = True` on each object as a belt-and-suspenders mechanism. Implemented today; bucket policy configured via Terraform.

Layer 2 — Application-layer immutability guard
(`app/services/worm_archive_service.py:146–156`):
```python
def guard_archived_write(conversation):
    if conversation.archived_at is not None:
        raise PermissionError(f"Conversation {conversation.id} is archived (17a-4 WORM)...")
```

Called from every UPDATE path that touches `conversations`.

Layer 3 — Postgres BEFORE UPDATE/DELETE trigger: planned; not yet in migrations. Will mirror the application guard at the DB level so a compromised or buggy code path cannot bypass it.

Index of archived objects in `archived_message_records` and `message_retention_records` tables — both carry `retention_class` (`17a4_3yr` for business communications, `17a4_6yr` for order records) and `retain_until` timestamps. Nightly APScheduler job (`app/scheduler.py:864–879`) at 01:00 UTC moves conversations older than 90 days (configurable via `ARCHIVE_RETENTION_DAYS`) to GCS WORM bucket and sets `archived_at` on the DB row.

Disclosure #5 — Data Portability with Temporal Partitioning

Q5.1 — SwitchEvent data structure

Today the switch is modeled as **three related rows** rather than a single `SwitchEvent` table. We can introduce a unified `SwitchEvent` row before non-provisional if you'd prefer that for claim clarity. Existing components:

- **`pago_portability_consents`** (`app/models/audit.py:293–322`): records the user's *consent* to portability with `source_org_id`, `destination_org_id`, `consented_at`, `revoked_at`, `consent_type`, `retention_expires_at`.
- **`data_exports`** (`app/models/audit.py:338–395`): records the *bundle* — `source_org_id`, `destination_org_id`, `initiated_by`, `status`, `date_range_start/end`, `record_count`, `bundle_size_bytes`, `bundle_hash`, `encryption_key_id`, `worm_copy_path`, `worm_retained_at`, `manifest`, `created_at`, `completed_at`.
- **`pago_export_records`** (`app/models/audit.py:260–291`): per-export index entry — `bundle_id`, `user_id`, `org_id`, `gcs_object_path`, `encrypted_path`.

The disclosure's "state-transition without row movement" semantics are realized at the **request middleware** layer (`TenantMiddleware`, `app/middleware/tenant.py`) — the user's "primary tenant" is recomputed per request from the latest non-revoked consent. No rows physically move; the *tenant resolution* changes, and historical reads continue to be served by Firm A's archive (still encrypted under Firm A's KMS key).

Q5.2 — Retention policy: data classes and years

`app/compliance/erasure.py:55–100`. The current retention period is **SEC/FINRA Rule 17a-4 standard 5 years**, applied uniformly to the eight data classes listed below. Class-specific retention is on the roadmap — patent claim should cover the planned `(org_id, data_class) → years` mapping plus the current uniform implementation.

Implemented (5 years each): `Portfolio`, `PlaidItem`, `PlaidAccount`, `WhatIfProposal`, `UserFile`, `SchedulingLink`, `Conversation`, `Message`.

Additional class-specific windows already in use elsewhere in the system:
- `archived_message_records.retention_expires_at` → 7 years from `archived_at` (17a-4 conservative)

- `message_retention_records.retention_class` → `17a4_3yr` (business communications) or `17a4_6yr` (order records)
- `pago_portability_consents.retention_expires_at` → 7 years from `consented_at`

Planned per-org policy table:

```
retention_policies(
  org_id INTEGER,
  data_class VARCHAR(50),
  retention_years INTEGER,
  legal_basis VARCHAR(200),
  PRIMARY KEY (org_id, data_class)
)
```

Example planned values: `chat_message=3yr`, `order_record=6yr`, `tax_document=7yr`, `pago_export=7yr`, `marketing_communication=2yr`.

Q5.3 — Cross-institution key-exchange mechanism

Planned, not implemented. The hooks are present (`DataExport.encryption_key_id`, `destination_org_id` on consents) but the actual transport-encryption-to-recipient flow is not yet in code. Planned mechanism:

1. **Firm A discovery:** call GCP KMS `getPublicKey` on Firm B's transport-receiver asymmetric key (advertised in a registry table `partner_org_keys`).
2. **Hybrid envelope (signed-then-encrypted):**
   - Generate ephemeral symmetric DEK (AES-256).
   - Encrypt the export bundle under the DEK (AES-GCM).
   - Sign the bundle hash under Firm A's KMS asymmetric signing key (so the recipient can verify origin).
   - Encrypt the DEK to Firm B's transport public key (RSA-OAEP-256 or X25519).
   - Package: `{signed_manifest, wrapped_dek, ciphertext, signature_alg, kdf, sender_org_id, recipient_org_id, nonce}`.
3. **Firm B receipt:** verify signature against Firm A's published verification key; unwrap DEK using B's KMS private key; decrypt; re-encrypt the historical archive under B's KMS DEK for at-rest storage (the historical pre-switch slice continues to reference A's keys for compliance/audit purposes).

We can build the registry table + ceremony script before filing if that strengthens the claim.

Disclosure #6 — Multi-Tenant Architecture with Postgres RLS

Q6.1 — Complete SKIP_PREFIXES list with justifications

`app/middleware/tenant.py:48–66`. Current full list (9 entries):

| Prefix | Reason RLS is bypassed |

|---|---|
`/api/auth/`	Pre-org auth (login, callback, JWT issuance). User has not yet been resolved to an org context.
`/api/invitations/`	Inherently cross-tenant: an invitation is issued by Org A and accepted by a user who may not yet belong to Org A.
`/api/orgs/onboarding-status`	Onboarding queries fire *before* the user picks/creates an org; no `app.org_id` available.
`/api/orgs/create`	Creates the first org for a brand-new user; pre-tenant by definition.
`/api/orgs/me`	Returns the user's full membership across all orgs — must read past the current `app.org_id` clip.
`/api/me/`	First-person history endpoints (issue #1414). Filtered by `current_user.id`, not `org_id`. Without bypass, RLS fail-closed clipping would hide all but the active org's memberships.
`/api/platform-admin/`	Superuser endpoints reading across orgs for support/forensics; protected by superuser claim on the JWT (orthogonal to RLS).
`/api/admin/orgs/`	Admin org-management endpoints — list/edit any org.
`/api/admin/demo/`	Demo seeding endpoints used in sales/marketing environments.

Middleware auto-applies `enable_rls_bypass()` for any request whose path starts with one of these prefixes (`app/middleware/tenant.py:85–91`) and releases it in the `finally` block, so handlers stay clean.

The patent-relevant element here is that **non-bypassed routes fail closed** — the Postgres session variable `app.org_id` is required and unset → all tenant-scoped tables return zero rows by RLS policy. Bypass is opt-in by route prefix and audited.

Q6.2 — Role hierarchy and role-keyed encryption

Per-org key isolation is implemented; per-role within-org key tiering is planned. Current state (`app/services/kms_service.py:40–44`): every org has its own KMS keyring (`org-{org_id}/cryptoKeys/pago-dek`). GCP IAM bindings determine which service accounts/roles can call `decrypt` on each keyring — this is the *implementation* of role-based access today, but the role taxonomy lives in GCP IAM, not in our Python code.

Planned application-layer role hierarchy (Q4 2026, post-provisional):

Role	Decrypts	Cannot decrypt
`client`	own PII, own PAGO, own portfolio	other users' data
`advisor`	clients in own household (limited PII per `pii_sharing_enabled` toggle)	other advisors' clients
`compliance`	audit logs, redacted transcripts	unredacted PII
`analyst`	k-anonymized aggregates (k=5 enforced)	individual PII
`superuser`	everything (audited every decrypt)	—

Key broker design: at request time, the broker presents the role + tenant claim to KMS and receives only the DEK(s) authorized for that role. For `analyst`, the broker returns an aggregation-only key tied to the anonymization pipeline output, not the raw row key. The Postgres-side `app.role` session variable (planned) would be paired with `app.org_id` to drive both RLS policies and per-role key release.

Q6.3 — Compliance schema with INSERT-only role

Implemented. Migration `2026_05_11_2329-36885e113071_create_compliance_schema_and_insert_.py`.

The migration creates:
- A separate Postgres schema `compliance`.
- A new role `compliance_writer` granted `USAGE` on the schema and **only `INSERT`** on `compliance.audit_log` (no UPDATE/DELETE/TRUNCATE).
- `compliance.audit_log` table: `(id BIGSERIAL, created_at, user_id, org_id, action, resource_type, resource_id, detail JSONB, ip_address INET, user_agent)`.
- The current application DB user gets `SELECT` for dashboard reads.

Usage pattern (`app/services/compliance_audit_service.py:33–94`): each audit write opens a `SAVEPOINT`, `SET LOCAL ROLE compliance_writer`, performs the INSERT, then `RESET ROLE` and `RELEASE SAVEPOINT`. The role swap is scoped to the audit write only; the rest of the transaction runs as the app user. This means **even a SQL-injection compromise of the application code cannot UPDATE or DELETE audit rows** — the privilege isn't present on the role used during the rest of the request.

This is one of the strongest patentable elements in Disclosure #6 — "in-process role escalation gated to a single INSERT statement, after which privileges are immediately revoked within the same transaction" — please consider it for a dependent claim.

Disclosure #7 — Multi-Persona AI Agents with Differentiated RAG

Q7.1 — Persona routing rules (decision_tree.py)

See Q2.4 above for the full picture — the routing rules in `decision_tree.py` perform *intent* classification, and `intent` then selects a **subgraph** (which in turn selects a persona+toolset+prompt). The complete intent→persona mapping:

Intent	Persona	System prompt key	Toolset (`DOMAIN_TOOLS[intent]`)
`financial_planning`	Freddy (FPA)	`freddy_system`	Monte Carlo, what-if, PFS edit
`investment`	Harvey (DIY investor)	`harvey_system`	Perfect10, EVS, Attaboy, insider buys/sells
`tax`	Tax specialist	`tax_system`	Tax-loss harvest, bracket calc
`budget`	Budget coach	`budget_system`	Spending breakdown, savings rate
`strategy`	Strategy composer	`strategy_system`	Multi-screener composition
`documents`	Vault assistant	`documents_system`	File-vault list/retrieve
`research`	News/research	`research_system`	Web search, news ingest
`general`	Default chat	`general_system`	LLM-only, no tools

Example direct routes (no LLM): "compose a strategy" → `strategy`; "model a job loss" → `financial_planning` (routes to FPA, which then dispatches the `model_life_event` tool).

Q7.2 — Conversation continuity across persona switches

How state is preserved when the system prompt, toolset, and temperature all change: the chat session lives in a single `Conversation` row (`app/models/chat.py:28–109`) keyed by `session_id`. Each user turn flows through `agent_graph` which (a) loads the full message history from DB, (b) runs `detect_intent` to set the new intent, (c) dispatches to the matching subgraph. The subgraph carries the full `messages` list into the LLM call regardless of which persona it is.

What persists across switches (these fields on `AgentState` use the LangGraph `add` reducer or are simply re-read each turn):
- `messages` — full chat history (accumulated)
- `tool_iterations`, `sse_events`, `tools_called`, `audit_events` — observability
- `consent_granted` — once granted in one persona, valid for the session

What changes per persona:
- `intent` field — set by `detect_intent` node
- System prompt — loaded by subgraph from `chatbot_config.py`
- Tool binding — only `DOMAIN_TOOLS[intent]` is bound to the LLM
- Temperature — per-subgraph LLM config

Cross-persona context compaction is supported by `Conversation.context_summary` + `context_summarized_through_id` (same model file): once history grows large, prior turns are summarized into a single system message that is included on every subsequent persona's prompt. This means a user can pivot from Freddy ("plan my retirement") to Harvey ("analyze AAPL") to Freddy again, and Freddy retains the full context of what was said to Harvey in between, without re-sending the raw transcript.

This *persona-agnostic conversation envelope with per-persona prompt/tool injection* is in our view one of the clearer patentable elements — claim language should emphasize the separation of conversation state (persistent) from agent configuration (per-turn).

Q7.3 — Harvey's pipelines

Harvey is the "DIY investor" persona; his tool set is backed by five precomputed data pipelines. Each runs on a separate APScheduler cadence and writes to a domain-specific Postgres table that Harvey's tools query at low latency.

EVS (Earnings Volatility Screener) — `app/services/evs_pipeline.py`. Assembles the upcoming-earnings universe, enriches each ticker with internal strategy context, runs grounded LLM analysis of historical earnings volatility, and stores results in `evs_runs` + `evs_signals`. Output: early-warning signals for pre-event and post-event volatility surprises. Tool: `get_evs_signals`. Schedule: daily 03:00 UTC.

Attaboy (Analyst Sentiment Screener) — `app/services/attaboy_pipeline.py`. Pulls earnings-call transcripts from Finnhub, extracts the analyst Q&A section, scores analyst sentiment using Gemini, and writes to `attaboy_quotes` + `attaboy_summary`. Captures "tone shifts" — when an analyst who was previously bullish turns skeptical. Schedule: 4×/day at 00/06/12/18 UTC.

Perfect 10 (Multi-Dimensional Stock Score) — `app/services/perfect10_service.py`. Computes 10 per-ticker dimensions: 8 deterministic (charts, analysts, insiders, relative performance, sector outlook, cash flow, PEG ratio, valuation) and 2 AI-driven (management-

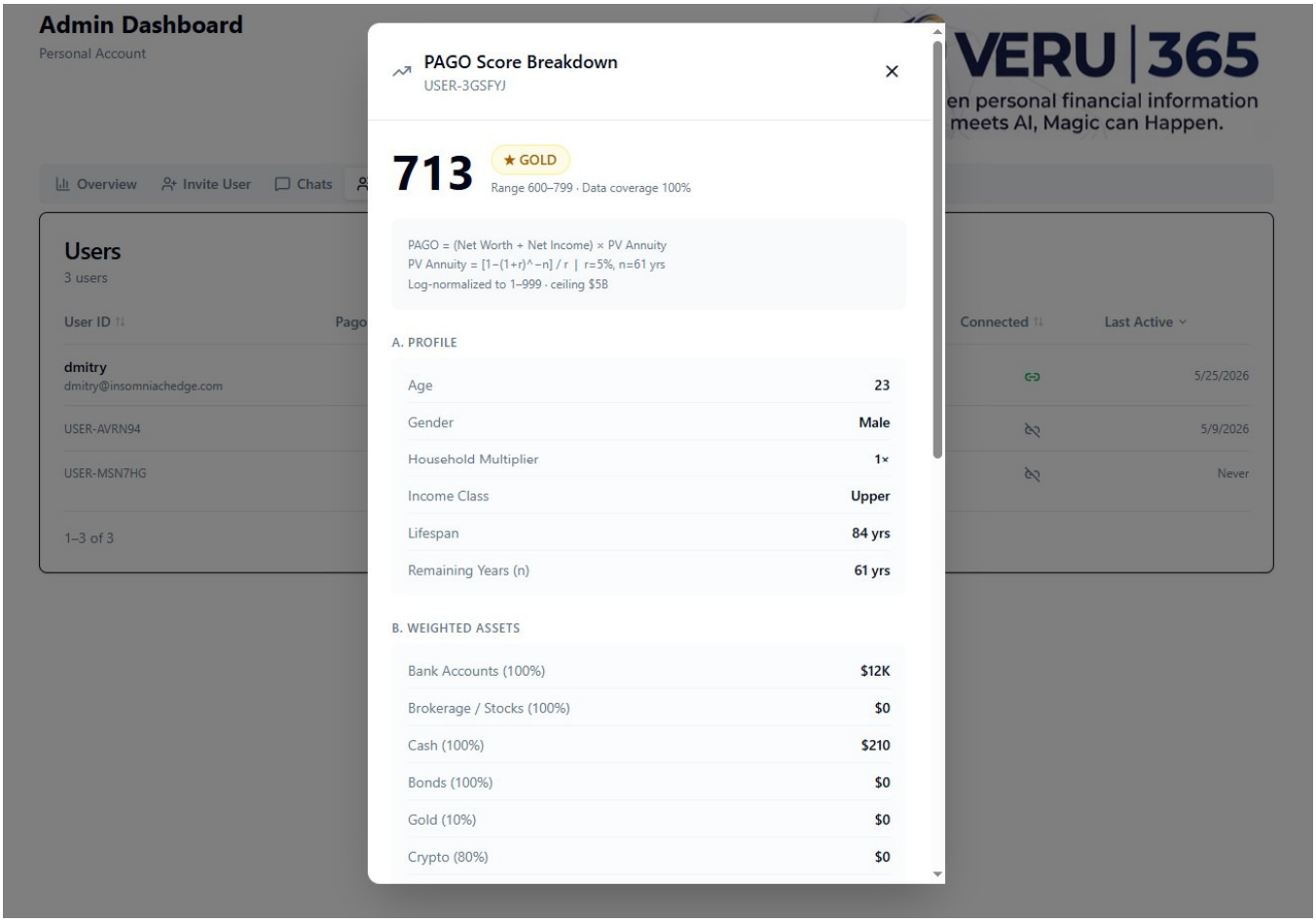
discussion sentiment from 10-K/10-Q, catalyst extraction). Unified Perfect10 score per ticker. Tool: `analyze_stock`. Schedule: nightly.

Insider Buying — `app/services/insider_trading_service.py`. SEC EDGAR Form 4 → filters to `transaction_code='P'` (open-market purchase) ≥ $200k → enriches with price-relative-to-S&P-500 alpha → writes to `insider_trades`. Weekly narrative summary generated by Gemini in Harvey's voice (`weekly_insider_summary_service.py`). Tool: `get_insider_trades`.

Rats Abandoning Ship — `app/services/insider_sells_service.py`. SEC EDGAR Form 4 → filters to `transaction_code='S'` (open-market sale) → detects two distinct signal patterns:
- Coordinated-selling cluster: ≥2 insiders at same company each selling >$200k within a 30-day window.
- Single large sale: >$3M.
- Optional 52-week-low proximity filter (≤15%) as a UI overlay.

Writes to `insider_sells`. Tool: `get_insider_sells`.

All five pipelines share a common pattern: **batch precomputation overnight → low-latency tool reads during chat**. This isolates the user-facing chat latency budget from the cost of running LLM analyses or scraping SEC EDGAR. We believe the pattern of "persona-scoped precomputed indices addressable from a persona-specific toolset" is itself patentable as part of Disclosure #7.



Admin Dashboard
Personal Account
VERU | 365
When personal financial information
meets AI, Magic can Happen.
Overview
Invite User
Chats
Users
Billing
Compliance
Settings

Users

3 users

User ID	Pago Score	Biometric ID	Chats	Minutes	Sharing	Connected	Last Active
dmitry dmitry@insomniachedge.com	713	YES	23	13m			5/25/2026
USER-AVRN94	— NO DATA	YES	1	7m			5/9/2026
USER-MSN7HG	— NO DATA	NO	0	—			Never

1–3 of 3

Admin Dashboard

Pokhilko Bank

VERU | 365
When personal financial information meets AI, Magic can Happen.

Overview | Invite User | Chats | Users | Billing | Compliance | Settings

Chats

12 total chats in this organization

All personas | All Users | search

Chat	Persona	User ID	Prompts	Minutes	Date
what are three main stocks today?	Freddy AI	Anonymous	2	—	5/18/2026
New Conversation	Freddy AI	Anonymous	2	—	5/18/2026
How am I tracking toward retirement?	Anna Financial AI	Anonymous	6	1.3m	5/8/2026
What should I buy on stock market right now?	Freddy AI	Anonymous	2	—	5/7/2026
should I sell my MSFT stock?	Freddy AI	Anonymous	4	—	5/7/2026
my DOB is 01/01/2000	Freddy AI	Anonymous	3	—	5/7/2026
should I sell Intel stock?	Freddy AI	Anonymous	4	—	5/7/2026
Hello, what is the best stock to buy right now?	Anna Financial AI	Anonymous	2	—	5/5/2026
should i sell my intel stock?	Freddy AI	Anonymous	2	—	5/5/2026
hey, do you remember me?	Freddy AI	Anonymous	2	—	5/5/2026
I just bought $1,000,000 worth of annuities. Do you think that was a good idea?	Freddy AI	Anonymous	8	29.8m	5/1/2026
What should I invest in given my risk profile?	Freddy AI	Anonymous	12	144m	4/30/2026

Choose your AI Agent

We live in constant flux — updating your financial plan doesn't have to be something you plan weeks ahead. Start a conversation and get personalized financial guidance now.

Freddy

Harvey

Anna

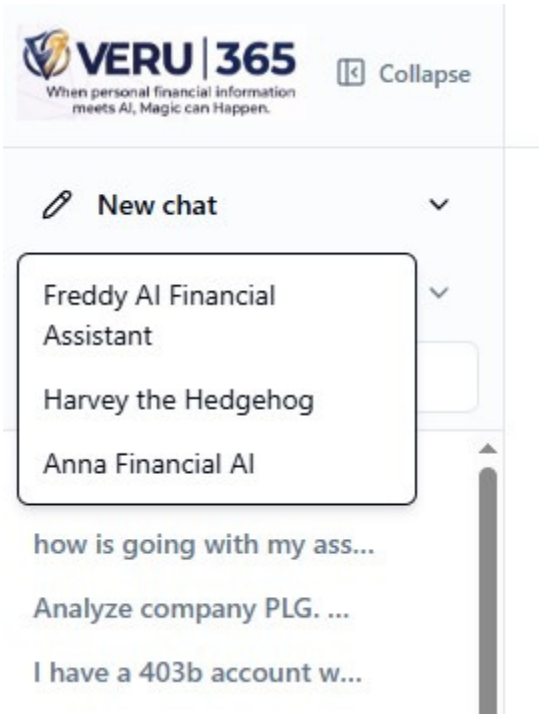
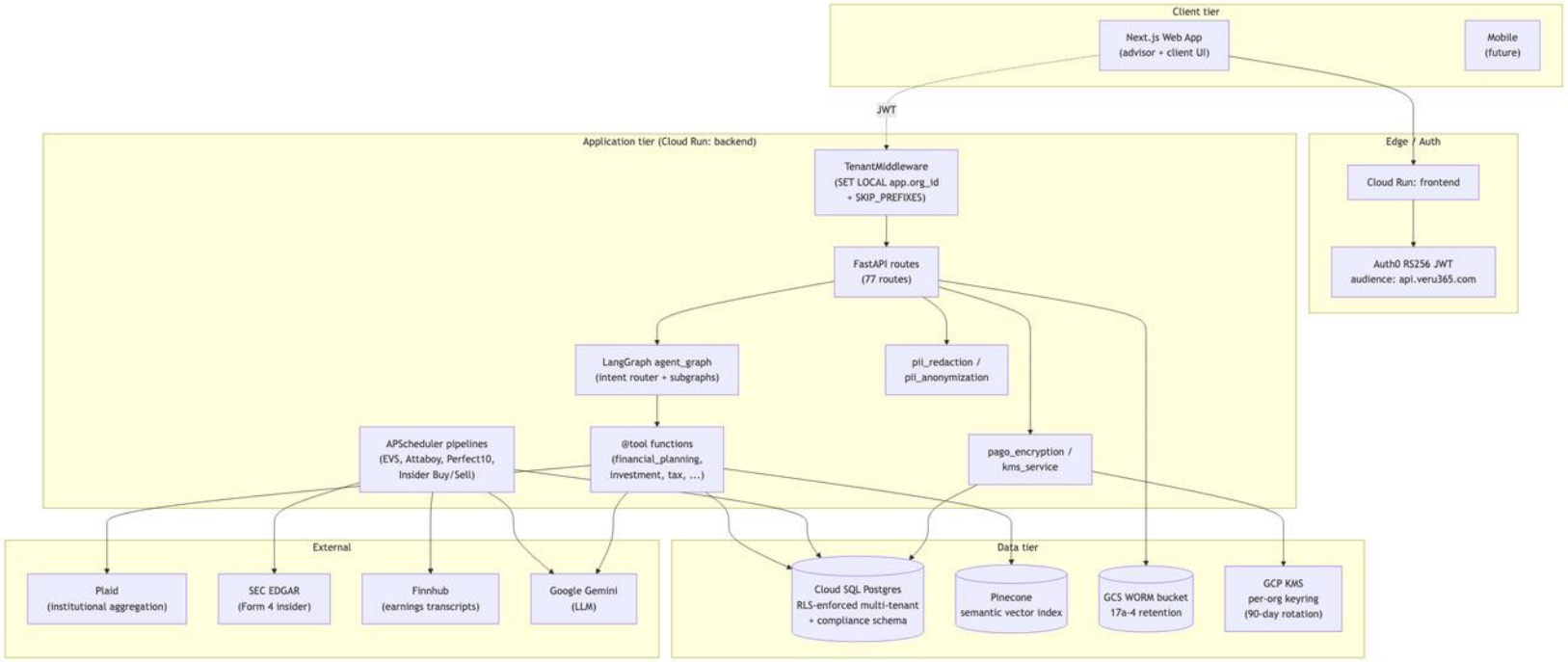
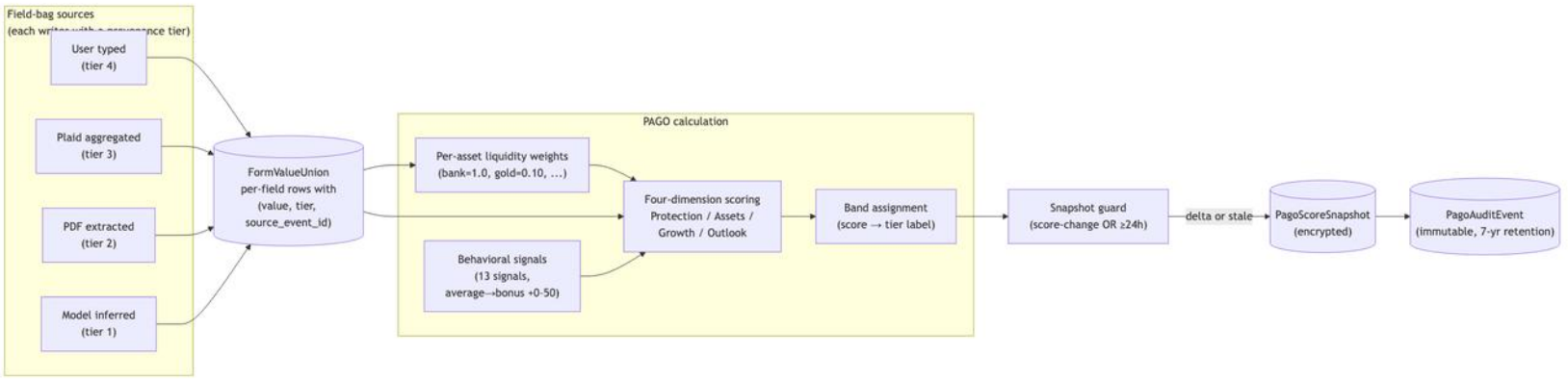
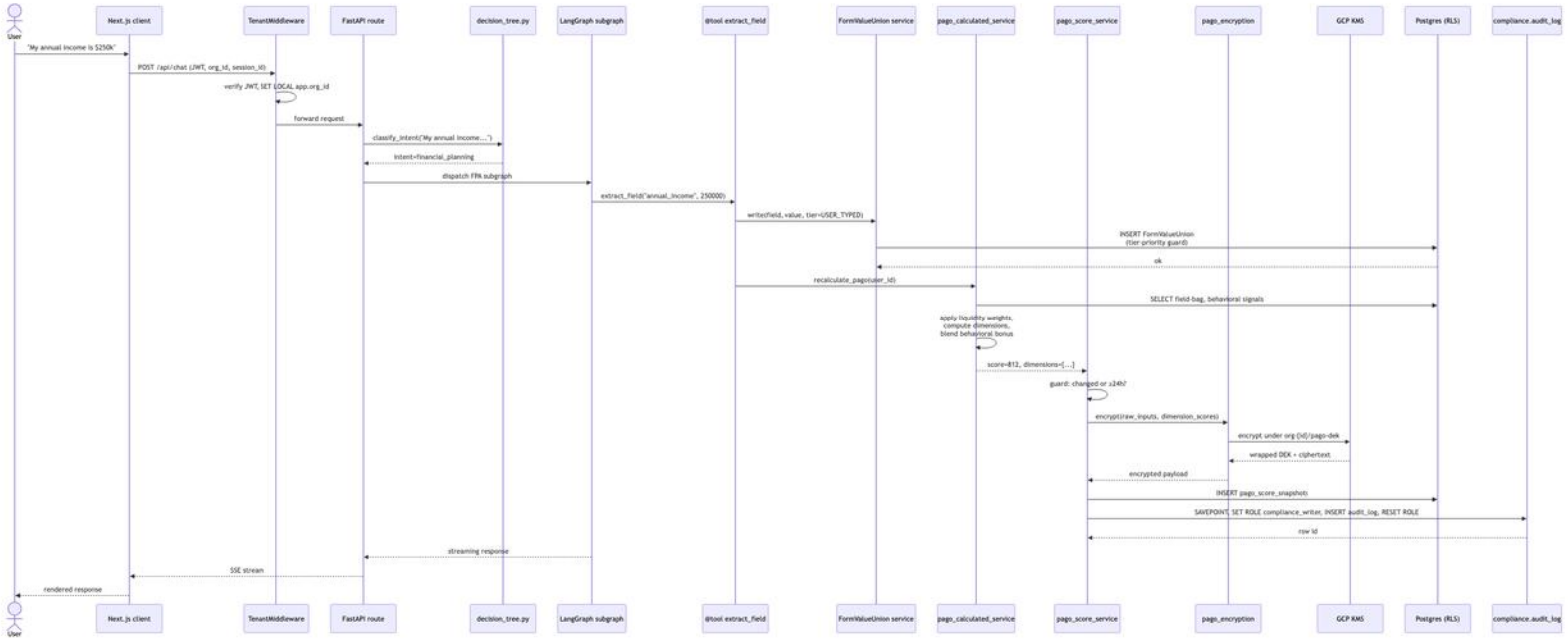
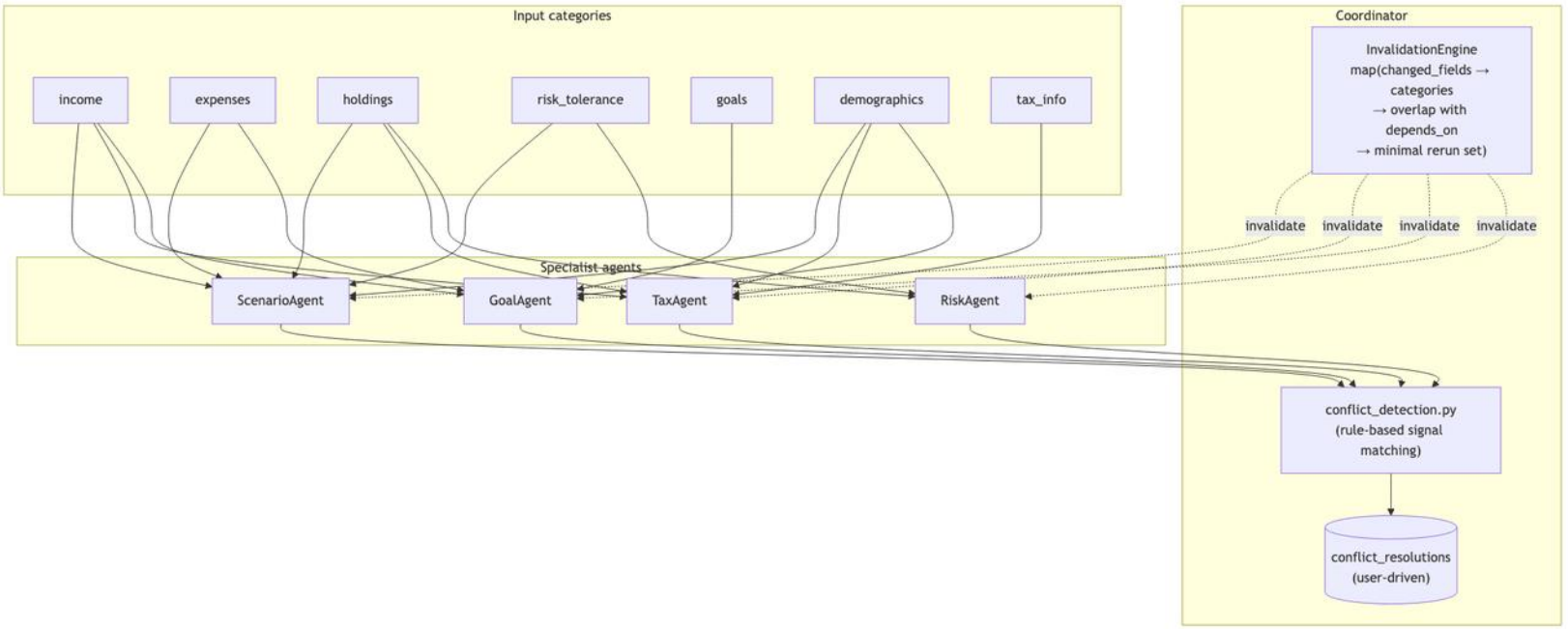

VERU | 365
When personal financial information meets AI, Magic can Happen.
Collapse
New chat
Freddy AI Financial Assistant
Harvey the Hedgehog
Anna Financial AI
how is going with my ass...
Analyze company PLG. ...
I have a 403b account w...

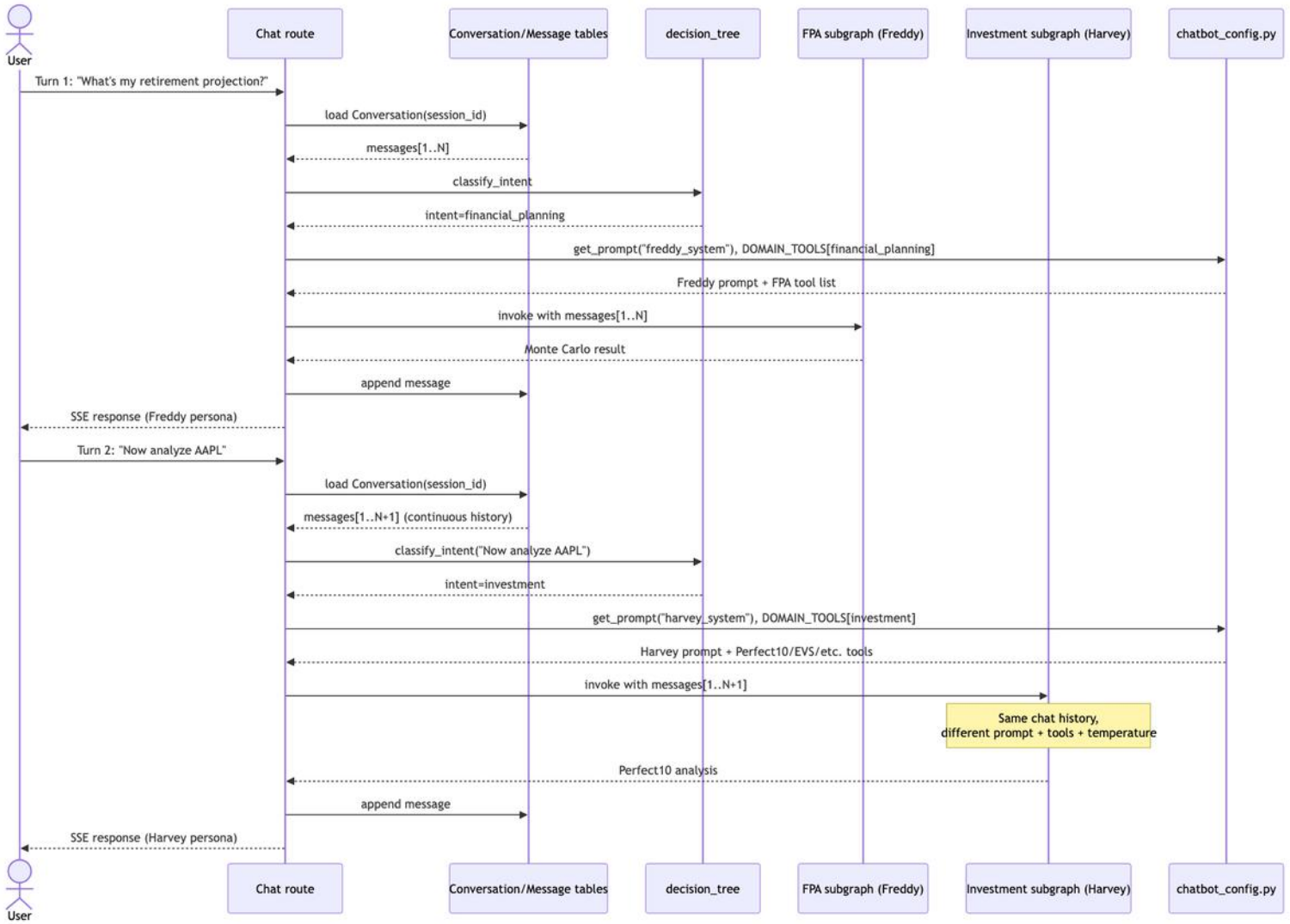
Exhibit B

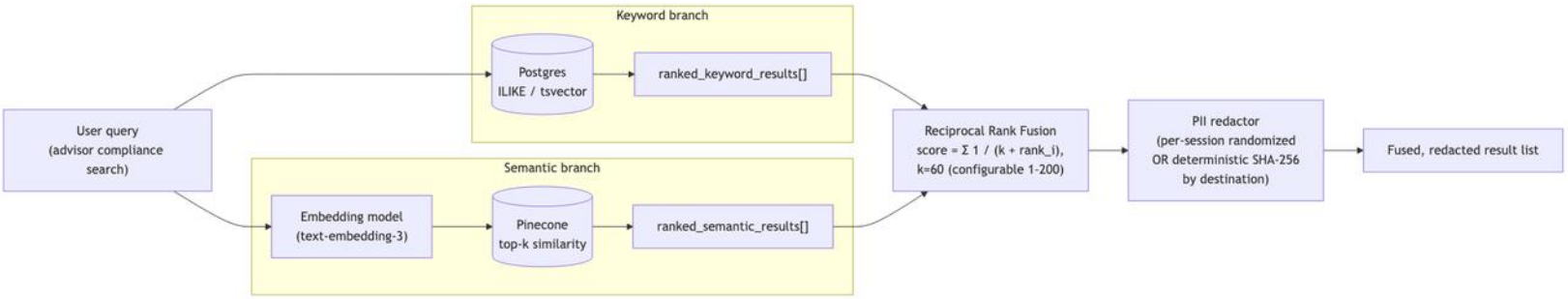

Client tier
Next.js Web App
(advisor + client UI)
Mobile
(future)
JWT
Application tier (Cloud Run: backend)
TenantMiddleware
(SET LOCAL app.org_id
+ SKIP_PREFIXES)
FastAPI routes
(77 routes)
LangGraph agent_graph
(intent router + subgraphs)
pii_redaction /
pii_anonymization
APScheduler pipelines
(EVS, Attaboy, Perfect10,
Insider Buy/Sell)
@tool functions
(financial_planning,
investment, tax, ...)
pago_encryption /
kms_service
Edge / Auth
Cloud Run: frontend
Auth0 RS256 JWT
audience: api.veru365.com
External
Plaid
(institutional aggregation)
SEC EDGAR
(Form 4 insider)
Finnhub
(earnings transcripts)
Google Gemini
(LLM)
Data tier
Cloud SQL Postgres
RLS-enforced multi-tenant
+ compliance schema
Pinecone
semantic vector index
GCS WORM bucket
17a-4 retention
GCP KMS
per-org keyring
(90-day rotation)

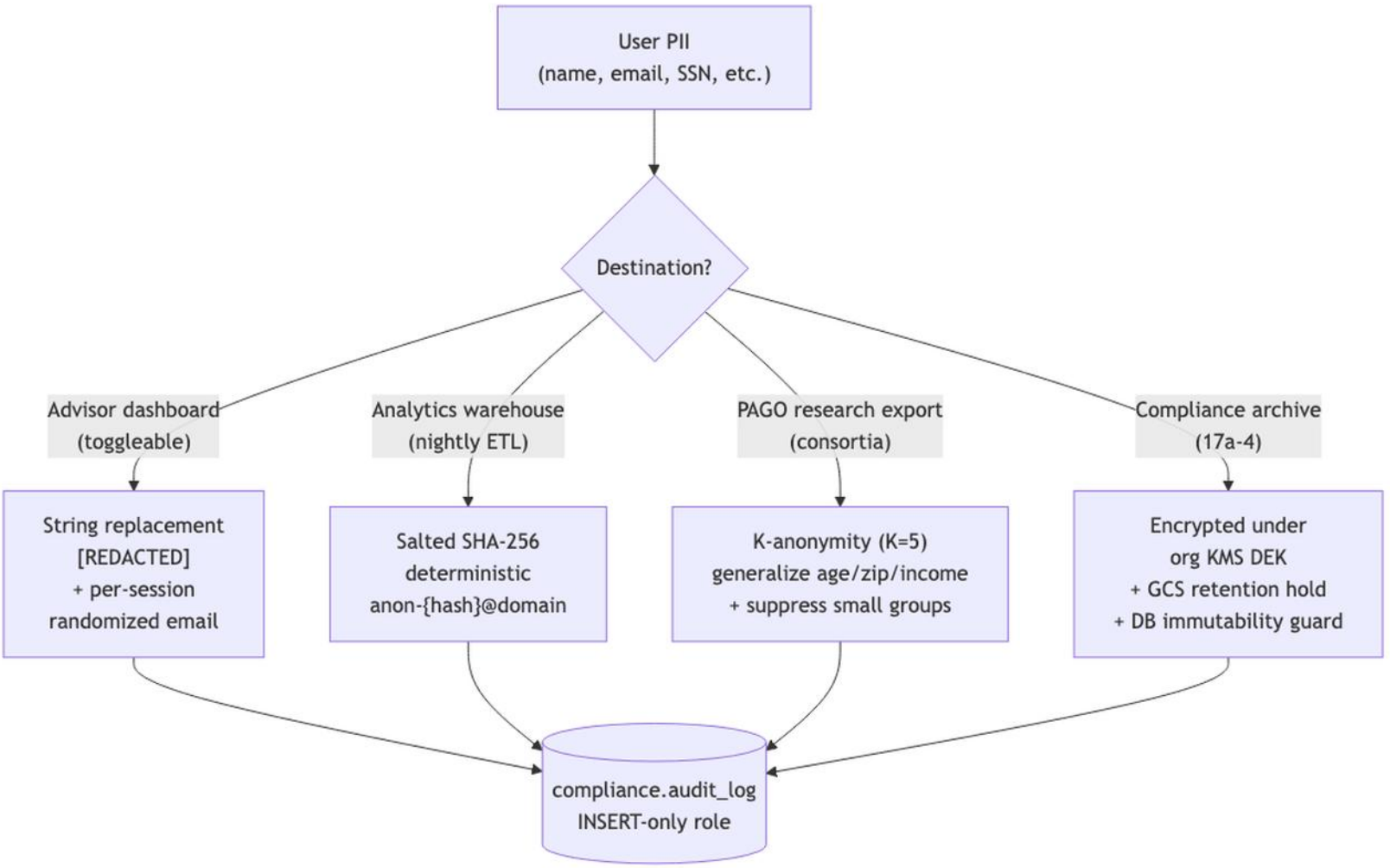

Field-bag sources
User typed
(tier 4)
Plaid aggregated
(tier 3)
PDF extracted
(tier 2)
Model inferred
(tier 1)
FormValueUnion
per-field rows with
(value, tier,
source_event_id)
PAGO calculation
Per-asset liquidity weights
(bank=1.0, gold=0.10, ...)
Behavioral signals
(13 signals,
average→bonus +0-50)
Four-dimension scoring
Protection / Assets /
Growth / Outlook
Band assignment
(score → tier label)
Snapshot guard
(score-change OR ≥24h)
delta or stale
PagoScoreSnapshot
(encrypted)
PagoAuditEvent
(immutable, 7-yr retention)

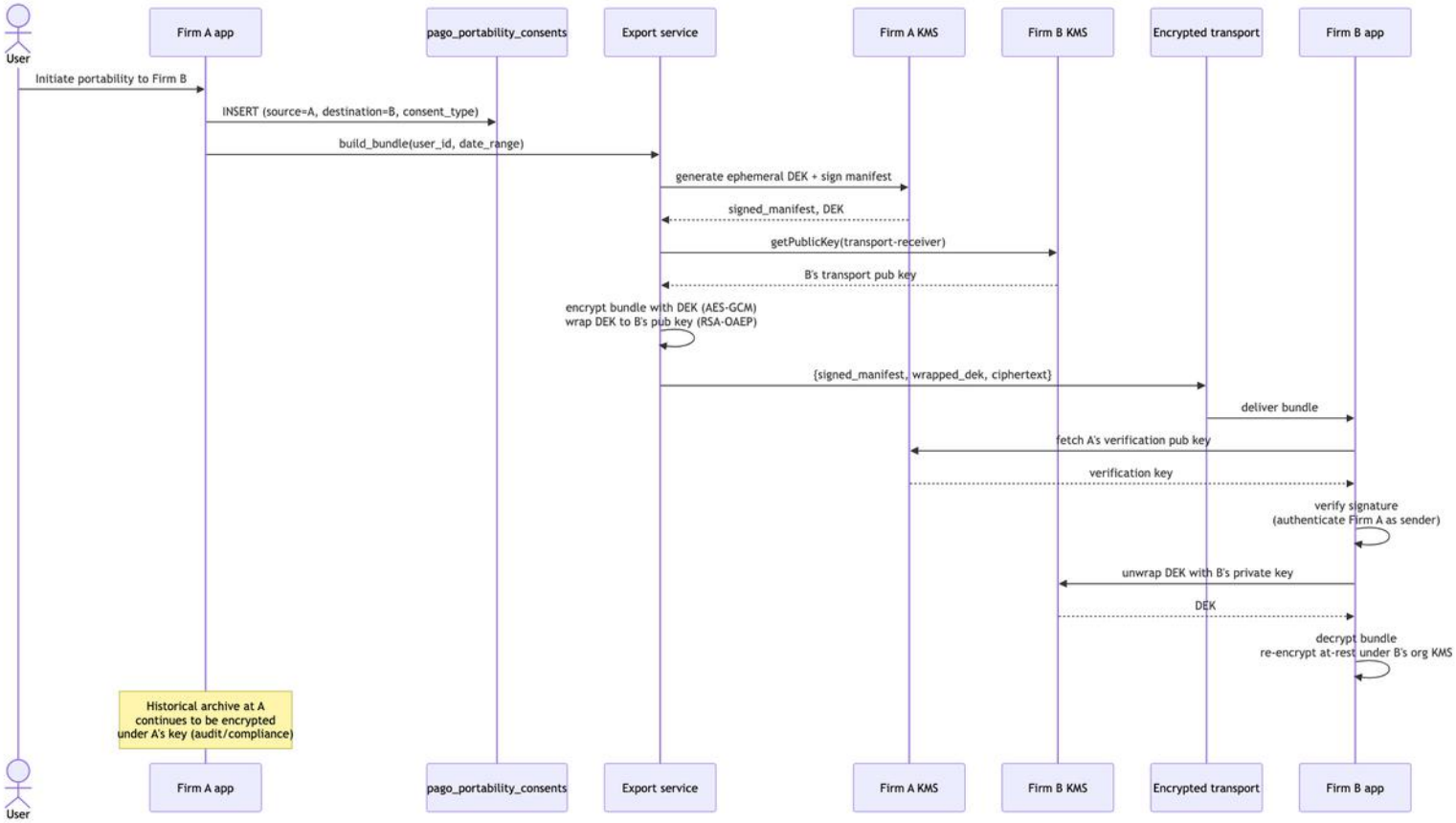

User
Next.js client
TenantMiddleware
FastAPI route
decision_tree.py
LangGraph subgraph
@tool extract_field
FormValueUnion service
pago_calculated_service
pago_score_service
pago_encryption
GCP KMS
Postgres (RLS)
compliance.audit_log
"My annual income is $250k"
POST /api/chat (JWT, org_id, session_id)
verify JWT, SET LOCAL app.org_id
forward request
classify_intent("My annual income...")
intent=financial_planning
dispatch FPA subgraph
extract_field("annual_income", 250000)
write(field, value, tier=USER_TYPED)
INSERT FormValueUnion
(tier-priority guard)
ok
recalculate_pago(user_id)
SELECT field-bag, behavioral signals
apply liquidity weights,
compute dimensions,
blend behavioral bonus
score=812, dimensions={...}
guard: changed or >24h?
encrypt(raw_inputs, dimension_scores)
encrypt under org-{id}/pago-dek
wrapped DEK + ciphertext
encrypted payload
INSERT pago_score_snapshots
SAVEPOINT, SET ROLE compliance_writer, INSERT audit_log, RESET ROLE
row id
streaming response
SSE stream
rendered response

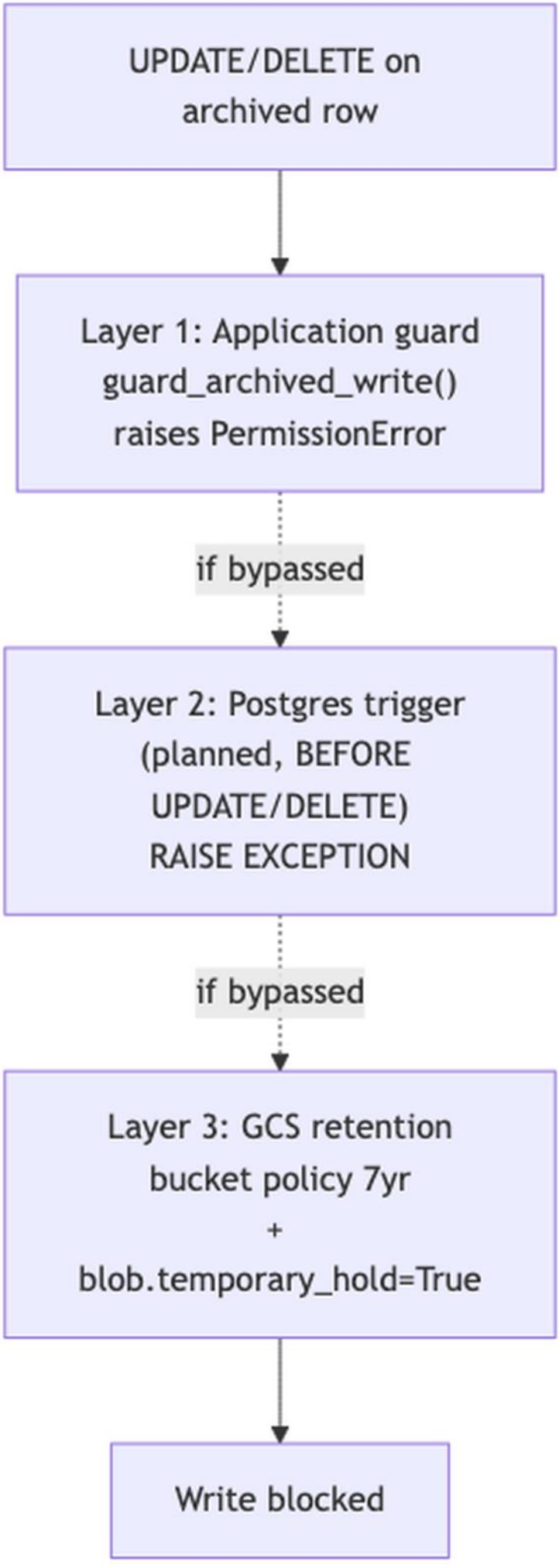

Input categories
income
expenses
holdings
risk_tolerance
goals
demographics
tax_info
Specialist agents
ScenarioAgent
GoalAgent
TaxAgent
RiskAgent
Coordinator
InvalidationEngine
map(changed_fields → categories
→ overlap with depends_on
→ minimal rerun set)
invalidate
invalidate
invalidate
invalidate
conflict_detection.py
(rule-based signal matching)
conflict_resolutions
(user-driven)


User
Chat route
Conversation/Message tables
decision_tree
FPA subgraph (Freddy)
Investment subgraph (Harvey)
chatbot_config.py
Turn 1: "What's my retirement projection?"
load Conversation(session_id)
messages[1..N]
classify_intent
intent=financial_planning
get_prompt("freddy_system"), DOMAIN_TOOLS[financial_planning]
Freddy prompt + FPA tool list
invoke with messages[1..N]
Monte Carlo result
append message
SSE response (Freddy persona)
Turn 2: "Now analyze AAPL"
load Conversation(session_id)
messages[1..N+1] (continuous history)
classify_intent("Now analyze AAPL")
intent=investment
get_prompt("harvey_system"), DOMAIN_TOOLS[investment]
Harvey prompt + Perfect10/EVS/etc. tools
invoke with messages[1..N+1]
Same chat history,
different prompt + tools + temperature
Perfect10 analysis
append message
SSE response (Harvey persona)
Chat route
Conversation/Message tables
decision_tree
FPA subgraph (Freddy)
Investment subgraph (Harvey)
chatbot_config.py
User


Keyword branch
User query
(advisor compliance
search)
Postgres
ILIKE / tsvector
ranked_keyword_results[]
Semantic branch
Embedding model
(text-embedding-3)
Pinecone
top-k similarity
ranked_semantic_results[]
Reciprocal Rank Fusion
score = Σ 1 / (k + rank_i),
k=60 (configurable 1-200)
PII redactor
(per-session randomized
OR deterministic SHA-256
by destination)
Fused, redacted result list


User PII
(name, email, SSN, etc.)
Destination?
Advisor dashboard
(toggleable)
Analytics warehouse
(nightly ETL)
PAGO research export
(consortia)
Compliance archive
(17a-4)
String replacement
[REDACTED]
+ per-session
randomized email
Salted SHA-256
deterministic
anon-{hash}@domain
K-anonymity (K=5)
generalize age/zip/income
+ suppress small groups
Encrypted under
org KMS DEK
+ GCS retention hold
+ DB immutability guard
compliance.audit_log
INSERT-only role


User
Firm A app
pago_portability_consents
Export service
Firm A KMS
Firm B KMS
Encrypted transport
Firm B app
Initiate portability to Firm B
INSERT (source=A, destination=B, consent_type)
build_bundle(user_id, date_range)
generate ephemeral DEK + sign manifest
signed_manifest, DEK
getPublicKey(transport-receiver)
B's transport pub key
encrypt bundle with DEK (AES-GCM)
wrap DEK to B's pub key (RSA-OAEP)
{signed_manifest, wrapped_dek, ciphertext}
deliver bundle
fetch A's verification pub key
verification key
verify signature
(authenticate Firm A as sender)
unwrap DEK with B's private key
DEK
decrypt bundle
re-encrypt at-rest under B's org KMS
Historical archive at A
continues to be encrypted
under A's key (audit/compliance)
Firm A app
pago_portability_consents
Export service
Firm A KMS
Firm B KMS
Encrypted transport
Firm B app
User


UPDATE/DELETE on archived row
Layer 1: Application guard guard_archived_write() raises PermissionError
if bypassed
Layer 2: Postgres trigger (planned, BEFORE UPDATE/DELETE) RAISE EXCEPTION
if bypassed
Layer 3: GCS retention bucket policy 7yr + blob.temporary_hold=True
Write blocked

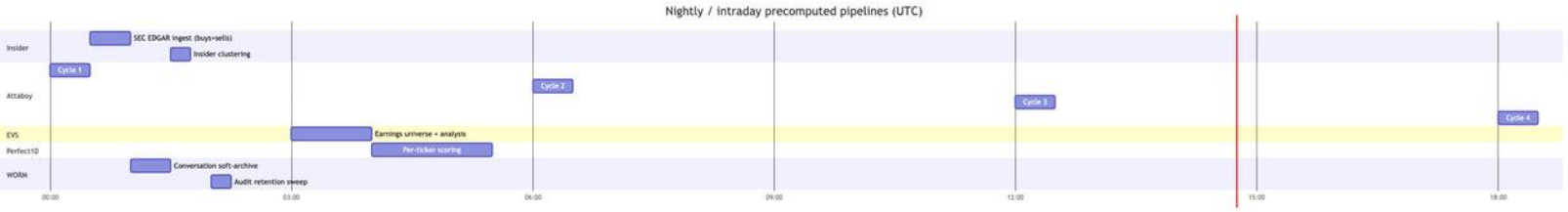
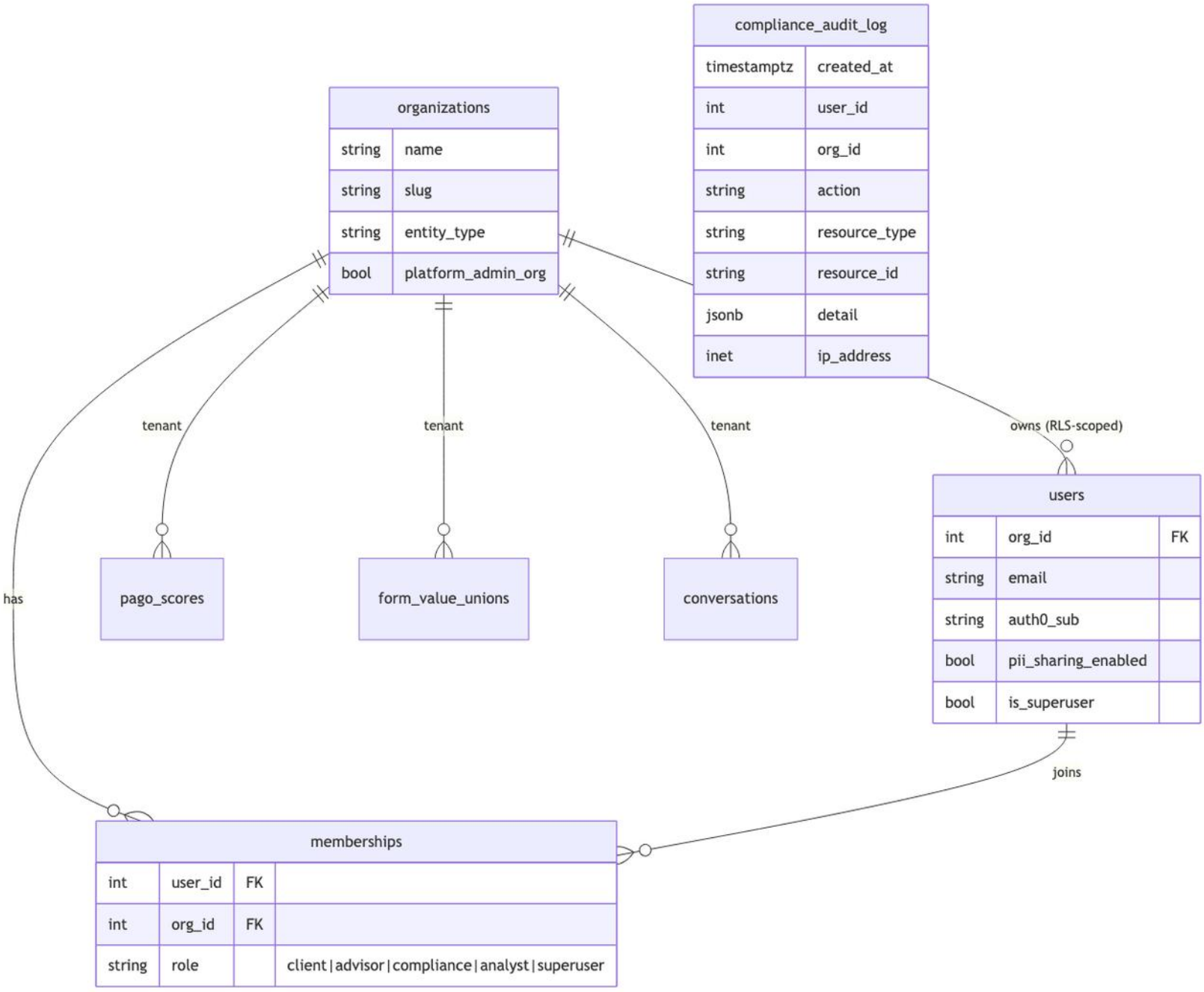
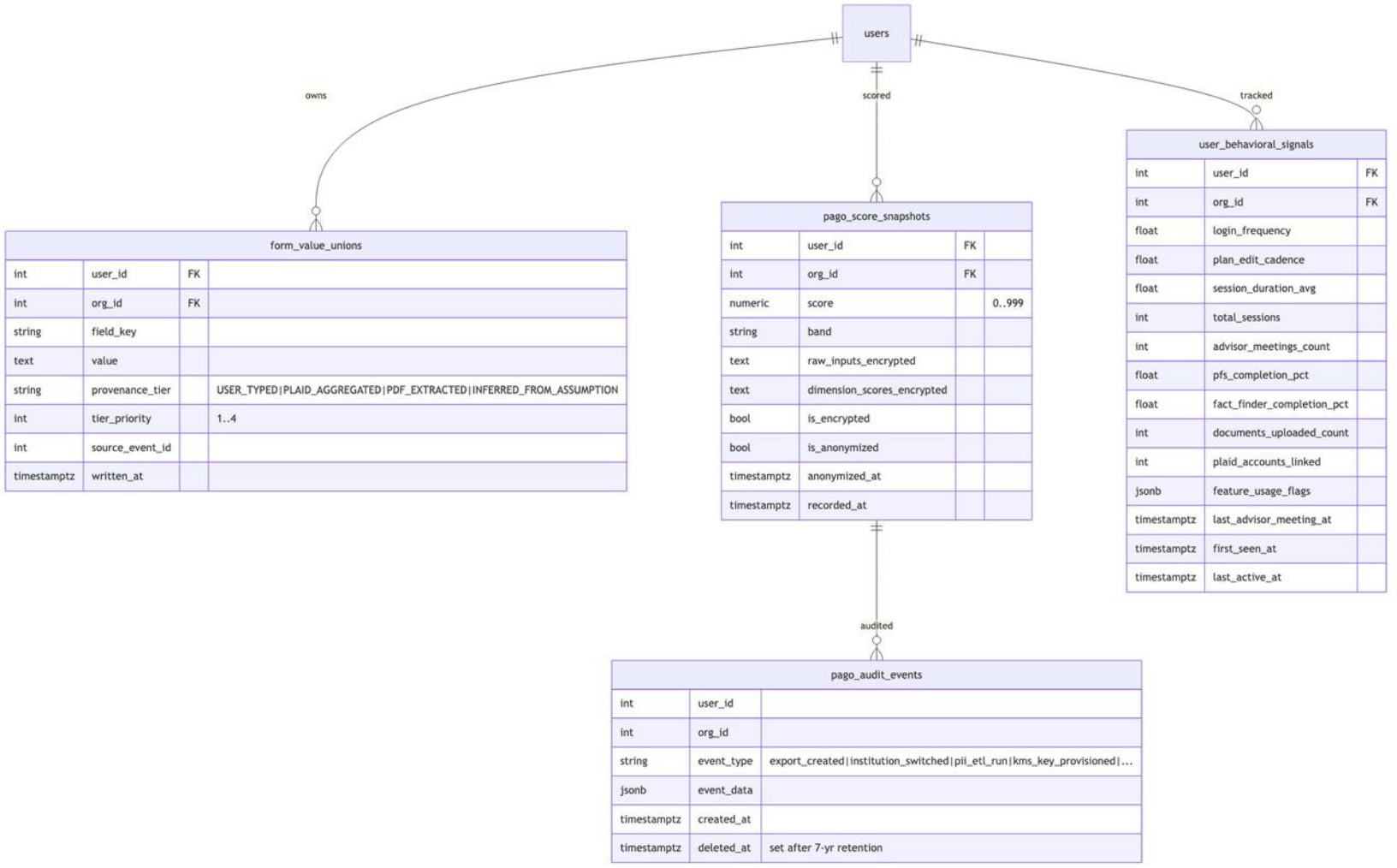
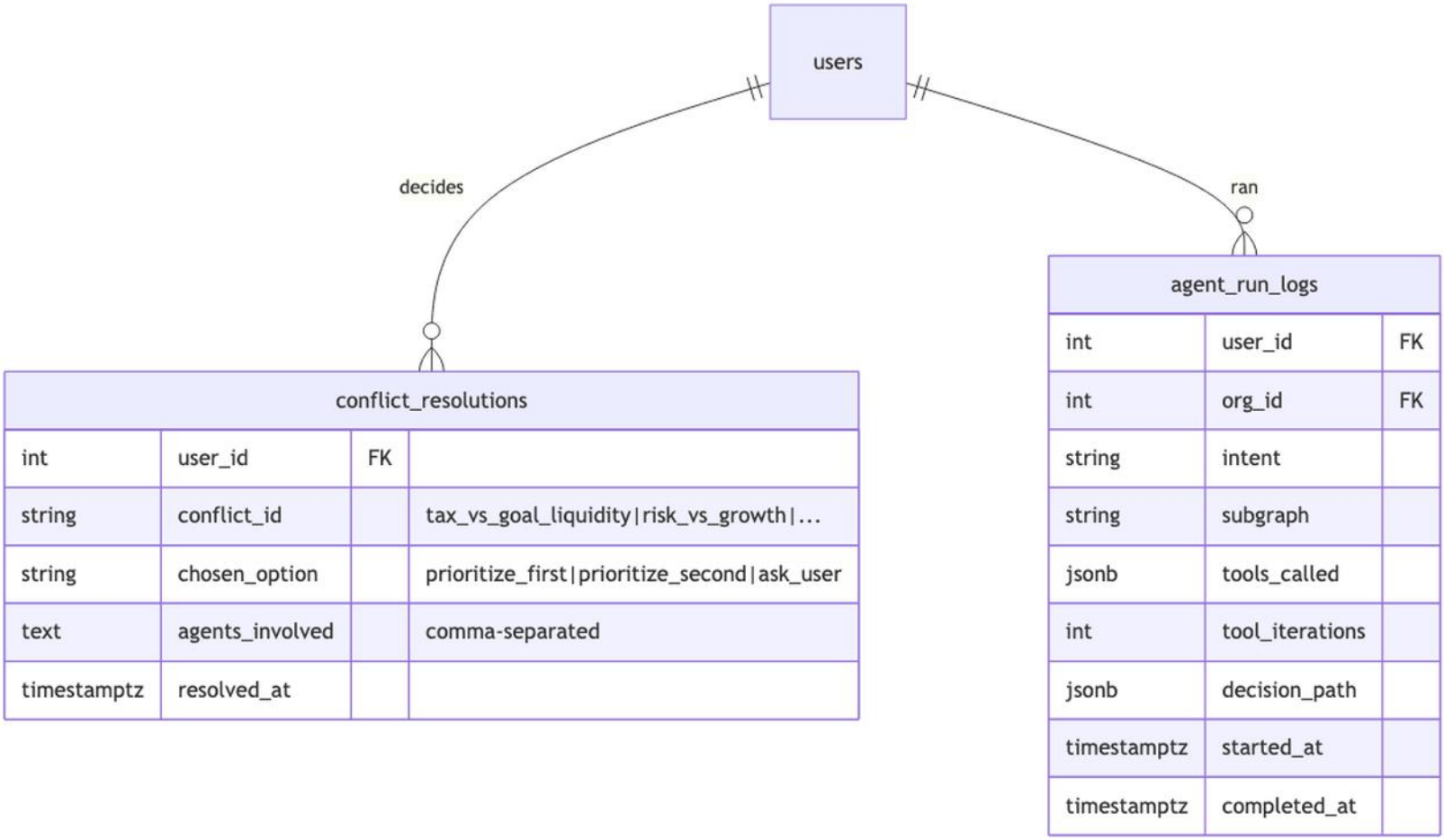
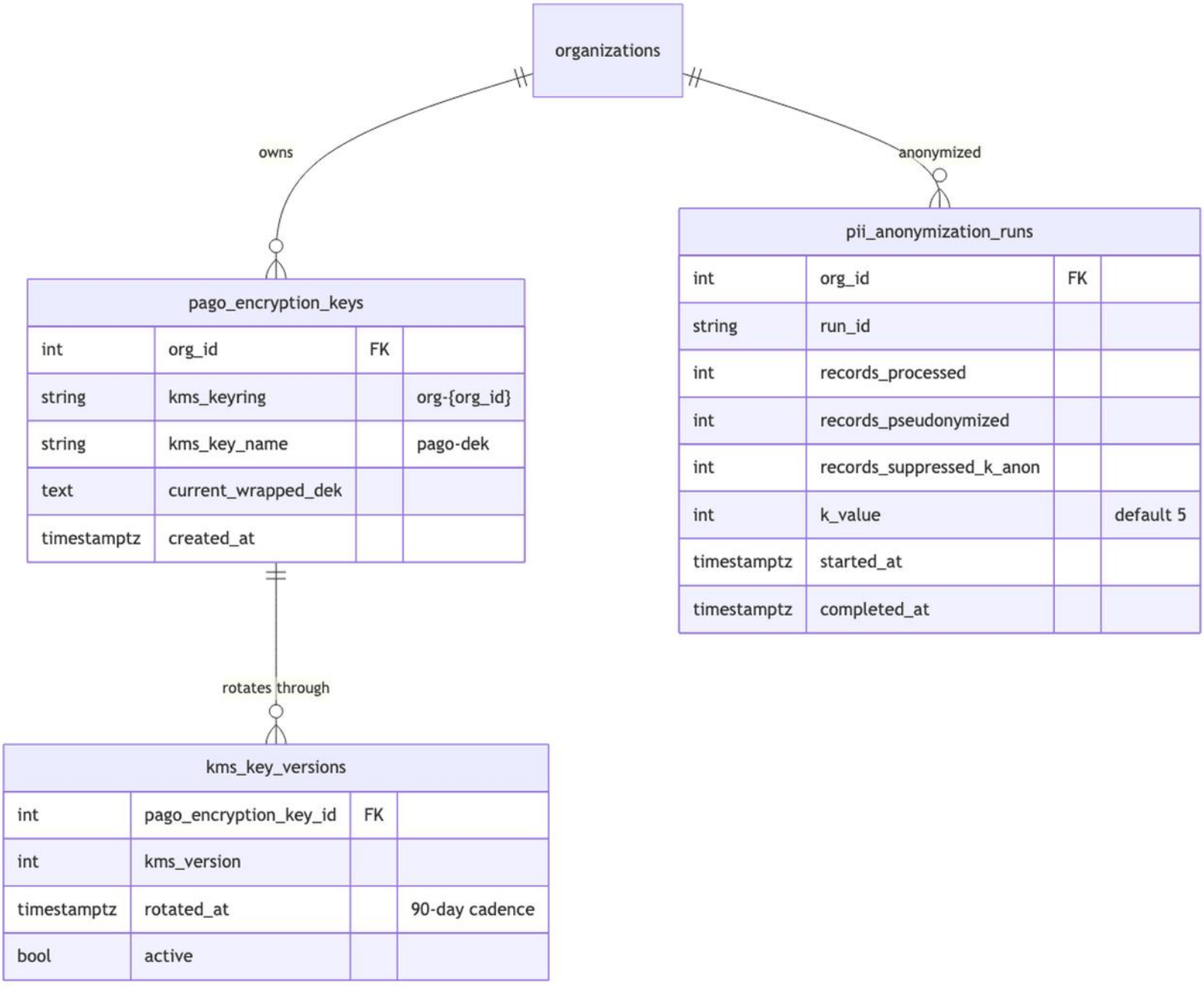

Nightly / intraday precomputed pipelines (UTC)
Insider
SEC EDGAR ingest (buys+sells)
Insider clustering
Cycle 1
Attaboy
Cycle 2
Cycle 3
Cycle 4
EVS
Earnings universe + analysis
Perfect10
Per-ticker scoring
WORM
Conversation soft-archive
Audit retention sweep
00:00
03:00
06:00
09:00
12:00
15:00
18:00

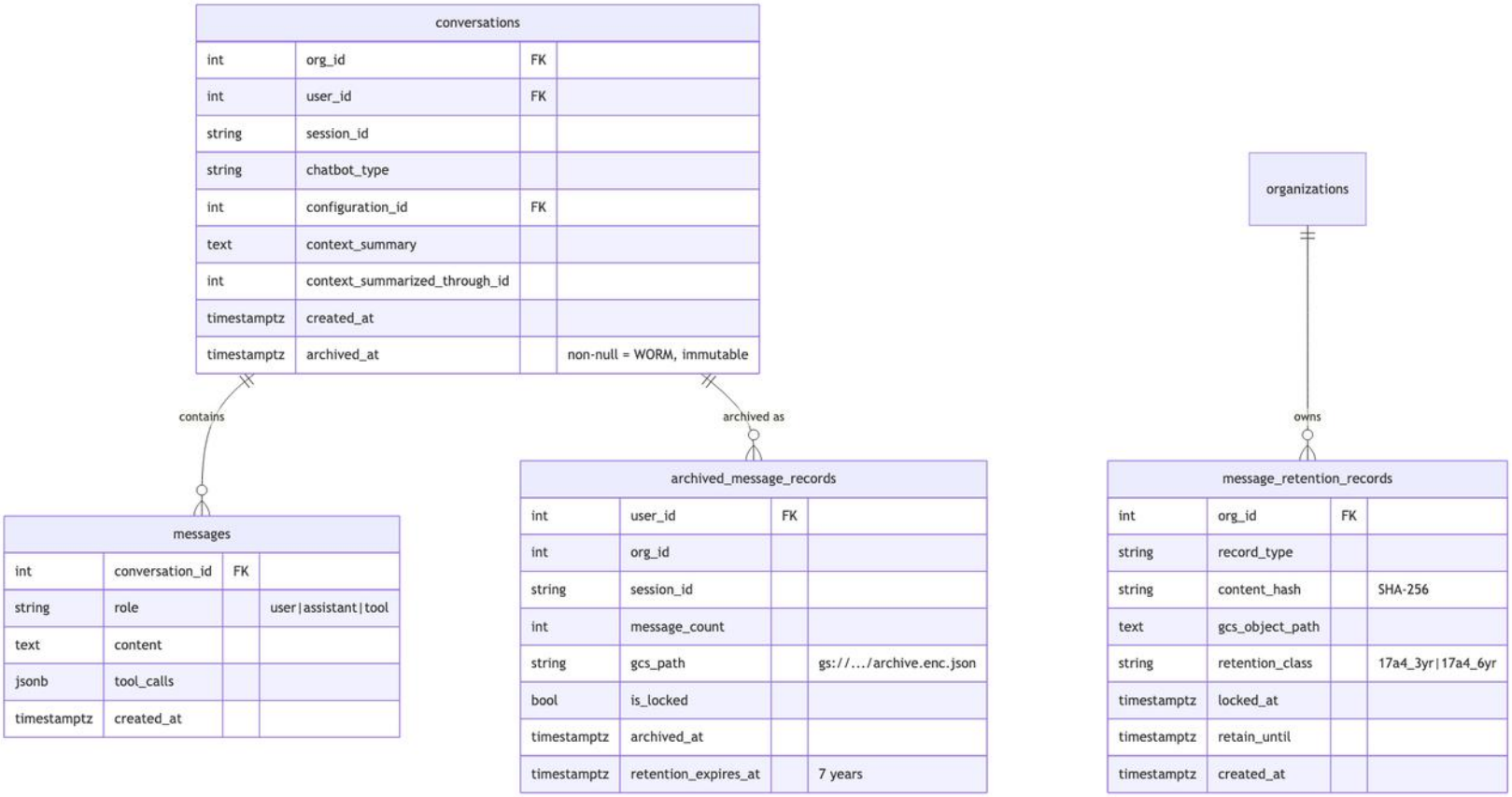
Exhibit C

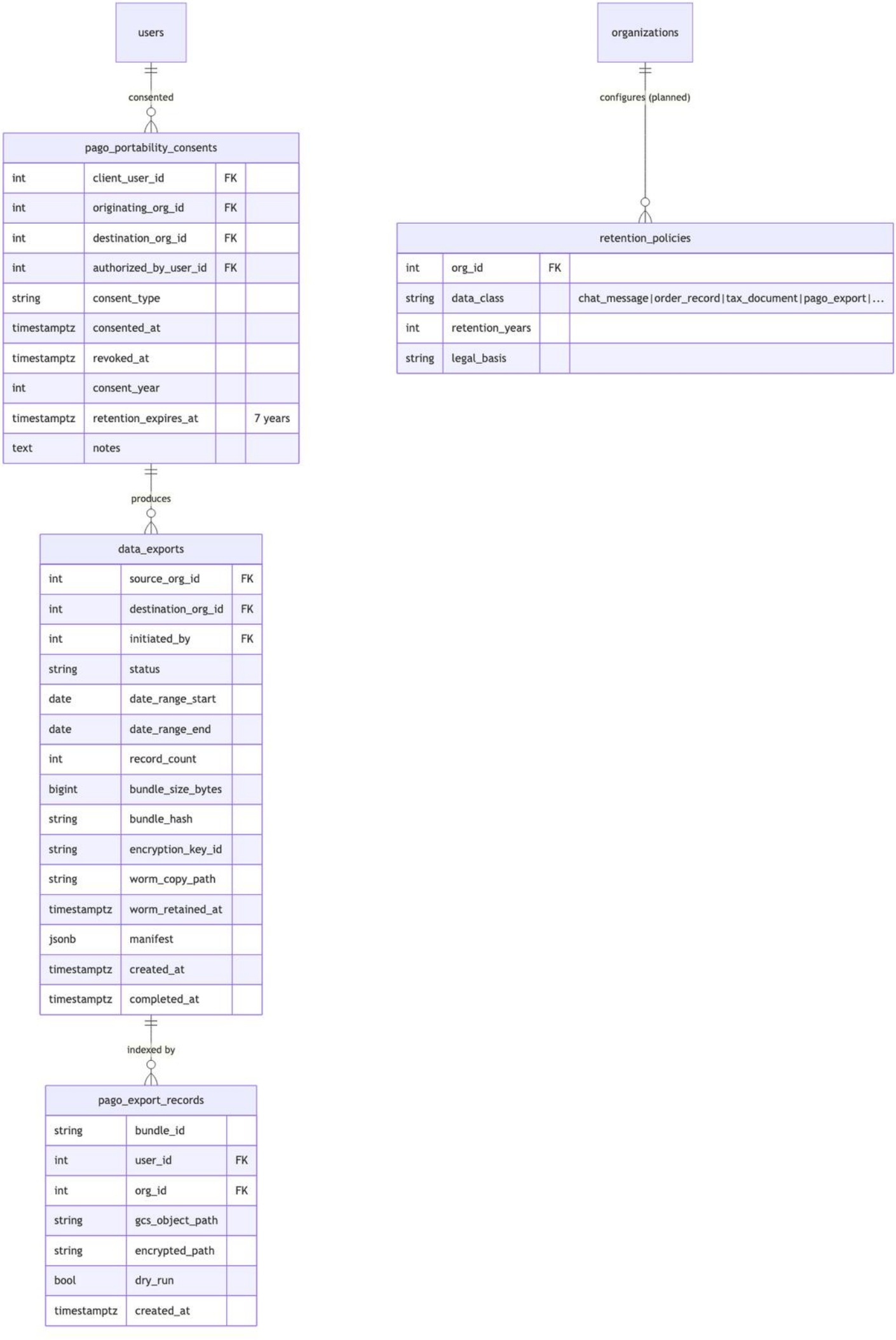

compliance_audit_log
timestamptz created_at
int user_id
int org_id
string action
string resource_type
string resource_id
jsonb detail
inet ip_address
organizations
string name
string slug
string entity_type
bool platform_admin_org
tenant
tenant
tenant
owns (RLS-scoped)
has
pago_scores
form_value_unions
conversations
users
int org_id FK
string email
string auth0_sub
bool pii_sharing_enabled
bool is_superuser
joins
memberships
int user_id FK
int org_id FK
string role client | advisor | compliance | analyst | superuser

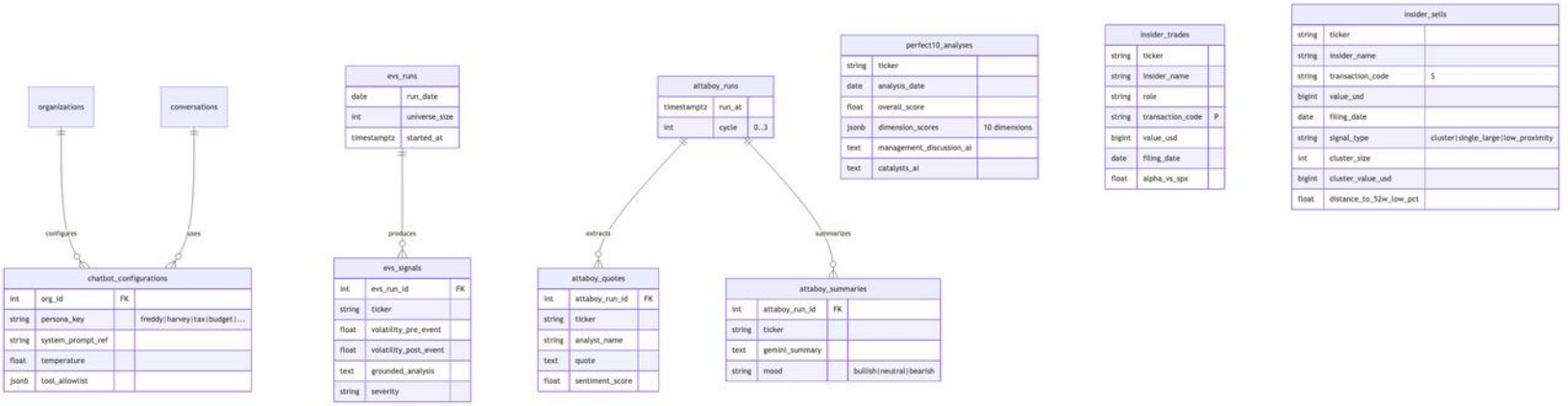

users
owns
scored
tracked
form_value_unions
int user_id FK
int org_id FK
string field_key
text value
string provenance_tier USER_TYPED|PLAID_AGGREGATED|PDF_EXTRACTED|INFERRED_FROM_ASSUMPTION
int tier_priority 1..4
int source_event_id
timestamptz written_at
pago_score_snapshots
int user_id FK
int org_id FK
numeric score 0..999
string band
text raw_inputs_encrypted
text dimension_scores_encrypted
bool is_encrypted
bool is_anonymized
timestamptz anonymized_at
timestamptz recorded_at
user_behavioral_signals
int user_id FK
int org_id FK
float login_frequency
float plan_edit_cadence
float session_duration_avg
int total_sessions
int advisor_meetings_count
float pfs_completion_pct
float fact_finder_completion_pct
int documents_uploaded_count
int plaid_accounts_linked
jsonb feature_usage_flags
timestamptz last_advisor_meeting_at
timestamptz first_seen_at
timestamptz last_active_at
audited
pago_audit_events
int user_id
int org_id
string event_type export_created|institution_switched|pii_etl_run|kms_key_provisioned|...
jsonb event_data
timestamptz created_at
timestamptz deleted_at set after 7-yr retention

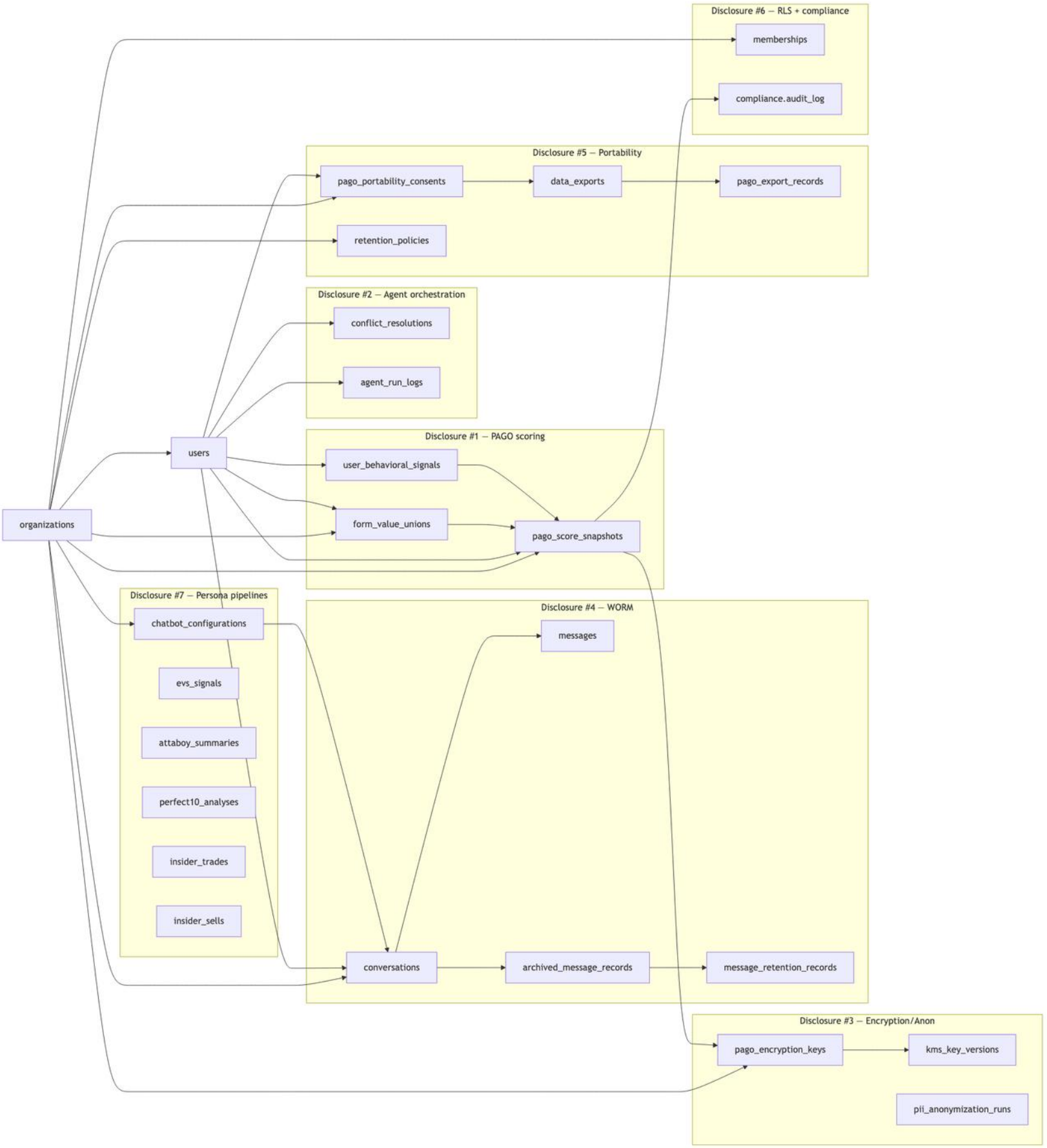

users
decides
ran
conflict_resolutions
int user_id FK
string conflict_id tax_vs_goal_liquidity|risk_vs_growth|...
string chosen_option prioritize_first|prioritize_second|ask_user
text agents_involved comma-separated
timestamptz resolved_at
agent_run_logs
int user_id FK
int org_id FK
string intent
string subgraph
jsonb tools_called
int tool_iterations
jsonb decision_path
timestamptz started_at
timestamptz completed_at


organizations
owns
anonymized
pago_encryption_keys
int org_id FK
string kms_keyring org-{org_id}
string kms_key_name pago-dek
text current_wrapped_dek
timestamptz created_at
pii_anonymization_runs
int org_id FK
string run_id
int records_processed
int records_pseudonymized
int records_suppressed_k_anon
int k_value default 5
timestamptz started_at
timestamptz completed_at
rotates through
kms_key_versions
int pago_encryption_key_id FK
int kms_version
timestamptz rotated_at 90-day cadence
bool active


conversations
int org_id FK
int user_id FK
string session_id
string chatbot_type
int configuration_id FK
text context_summary
int context_summarized_through_id
timestamptz created_at
timestamptz archived_at non-null = WORM, immutable
contains
archived as
messages
int conversation_id FK
string role user|assistant|tool
text content
jsonb tool_calls
timestamptz created_at
archived_message_records
int user_id FK
int org_id
string session_id
int message_count
string gcs_path gs://.../archive.enc.json
bool is_locked
timestamptz archived_at
timestamptz retention_expires_at 7 years
organizations
owns
message_retention_records
int org_id FK
string record_type
string content_hash SHA-256
text gcs_object_path
string retention_class 17a4_3yr|17a4_6yr
timestamptz locked_at
timestamptz retain_until
timestamptz created_at


users
consented
pago_portability_consents
int client_user_id FK
int originating_org_id FK
int destination_org_id FK
int authorized_by_user_id FK
string consent_type
timestamptz consented_at
timestamptz revoked_at
int consent_year
timestamptz retention_expires_at 7 years
text notes
produces
data_exports
int source_org_id FK
int destination_org_id FK
int initiated_by FK
string status
date date_range_start
date date_range_end
int record_count
bigint bundle_size_bytes
string bundle_hash
string encryption_key_id
string worm_copy_path
timestamptz worm_retained_at
jsonb manifest
timestamptz created_at
timestamptz completed_at
indexed by
pago_export_records
string bundle_id
int user_id FK
int org_id FK
string gcs_object_path
string encrypted_path
bool dry_run
timestamptz created_at
organizations
configures (planned)
retention_policies
int org_id FK
string data_class chat_message|order_record|tax_document|pago_export|...
int retention_years
string legal_basis


organizations
conversations
configures
uses
chatbot_configurations
int org_id FK
string persona_key freddy|harvey|tax|budget|...
string system_prompt_ref
float temperature
jsonb tool_allowlist
evs_runs
date run_date
int universe_size
timestamptz started_at
produces
evs_signals
int evs_run_id FK
string ticker
float volatility_pre_event
float volatility_post_event
text grounded_analysis
string severity
attaboy_runs
timestamptz run_at
int cycle 0...3
extracts
summarizes
attaboy_quotes
int attaboy_run_id FK
string ticker
string analyst_name
text quote
float sentiment_score
attaboy_summaries
int attaboy_run_id FK
string ticker
text gemini_summary
string mood bullish|neutral|bearish
perfect10_analyses
string ticker
date analysis_date
float overall_score
jsonb dimension_scores 10 dimensions
text management_discussion_ai
text catalysts_ai
insider_trades
string ticker
string insider_name
string role
string transaction_code P
bigint value_usd
date filing_date
float alpha_vs_spx
insider_sells
string ticker
string insider_name
string transaction_code S
bigint value_usd
date filing_date
string signal_type cluster|single_large|low_proximity
int cluster_size
bigint cluster_value_usd
float distance_to_52w_low_pct


Disclosure #6 — RLS + compliance
memberships
compliance.audit_log
Disclosure #5 — Portability
pago_portability_consents
data_exports
pago_export_records
retention_policies
Disclosure #2 — Agent orchestration
conflict_resolutions
agent_run_logs
Disclosure #1 — PAGO scoring
users
user_behavioral_signals
organizations
form_value_unions
pago_score_snapshots
Disclosure #7 — Persona pipelines
chatbot_configurations
Disclosure #4 — WORM
messages
evs_signals
attaboy_summaries
perfect10_analyses
insider_trades
insider_sells
conversations
archived_message_records
message_retention_records
Disclosure #3 — Encryption/Anon
pago_encryption_keys
kms_key_versions
pii_anonymization_runs

UNITED STATES PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
64/089,185	**06/12/2026 03:25:21 PM Z ET**	**34280/70000P**

Title of Invention

MULTI-TENANT FINANCIAL PLANNING PLATFORM WITH AGENT ORCHESTRATION AND PROVENANCE-AWARE DATA PROCESSING

Application Information

APPLICATION TYPE	Utility - Provisional Application under 35 USC 111(b)	PATENT #	-
CONFIRMATION #	7077	FILED BY	Jacquelin Garcia
PATENT CENTER #	77366834	FILING DATE	-
CUSTOMER #	4743	FIRST NAMED INVENTOR	Dmitry Pokhilko
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	Matthew Carey

Documents

TOTAL DOCUMENTS: 9

DOCUMENT		PAGES	DESCRIPTION	SIZE (KB)
Transmittal -- Provisional Application PTO SB-16.pdf		2	Provisional Cover Sheet (SB16)	91 KB
Warning: This is not a USPTO supplied Provisional Cover Sheet fillable form. Data in the form cannot be automatically loaded to other USPTO systems.				
34280-70000P - Veru365 Provisional Patent Application_1.pdf		39	-	149 KB
34280-70000P - Veru365 Provisional Patent Application_1-SPEC.pdf	(1-33)	33	Specification	132 KB
34280-70000P - Veru365	(34-38)	5	Claims	26 KB

Provisional Patent Application_1-CLM.pdf				
34280-70000P - Veru365 Provisional Patent Application_1-ABST.pdf	(39-39)	1	Abstract	16 KB
34280-70000P Figures.pdf		7	Drawings-only black and white line drawings	156 KB
Application Data Sheet ADS - PTO AIA-14.pdf		6	Application Data Sheet	62 KB

Warning: This is not a USPTO supplied ADS fillable form. Data in the form cannot be automatically loaded to other USPTO systems.

34280-70000P Exhibit A.pdf	36	Appendix to the specification	780 KB
34280-70000P Exhibit B.pdf	10	Appendix to the specification	736 KB

Warning: The attached file contains comments or annotations. Comments are not recommended and may cause processing problems with the document.

34280-70000P Exhibit C.pdf	8	Appendix to the specification	810 KB

Warning: The attached file contains comments or annotations. Comments are not recommended and may cause processing problems with the document.

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
Transmittal -- Provisional Application PTO SB-16.pdf	72A9326A0A9D8C1CFB109972F1C6DE2961C3E8A53C2A71F28E9A864E5B95CE3F74B391303494778A1A891A878BFE12C2A7A92D45C4390C457C5DB80275BC6E1A
34280-70000P - Veru365 Provisional Patent Application_1.pdf	A92984439B1E38B9E7310DF501CE11B1254D880A818F6EC7C28BCF34B12CAC6A7A6AB55CBAA13CEF72661D660FF310756038B96232436AEB31D53607C397BD15
34280-70000P - Veru365 Provisional Patent Application_1-SPEC.pdf	BE78E84A34DBC14037DB7F3566453A7C514A5FF1EE6819974FAFD717017120F5FA82A9BD84B09E1768400EB562D626320A2872BCEF1F1BC380DF68733F8ABC06
34280-70000P - Veru365 Provisional Patent Application_1-CLM.pdf	F37BAE8F4DBF5B0F6A3FB58EE6F0A2728ABEA922783036D009327D1F904E839BC7B9450085805E701B6512EB3F268591F73866FDF9160A392EE67B0AA41B0FCE
34280-70000P - Veru365 Provisional Patent Application_1-ABST.pdf	5EFCA6B8E1953C8E76D5A5112D86C524F05CDE2C3DFF7465EAC63273A5CE5F019AB96830308B207DE2ACF5D71340259BDFB5DAD360382FF2ACD6A37A86488B72

34280-70000P Figures.pdf	7E3C735A59E5514A49AE0BEFC44D5201D2F8078C67A6C209EC33458295DD69DF06D70C3A09097060A0CB471E39033DC49712230E92D0B75362C2DBC7694F35ED
Application Data Sheet ADS - PTO AIA-14.pdf	6F5927F95698892B1A3DA56750DCEF3D972B003EF0EC57A073893B7D1B5BBC23365321DD3198471A592AFF9FE445484AAADA522ECB5E49D2213FE05456ABC108
34280-70000P Exhibit A.pdf	CE36A8B799C96533893A58B9574BAF6E40507CDEED1D1F86BAF2D45D13A51F2F363C6D56211618002C8E52716B3EAA48CA2B041D5F06CCFFE0AACAA3DB51D864
34280-70000P Exhibit B.pdf	BE0F63897E0712BD7B6BF7437DAB34EE78A03B736EBBD156FAC40EF6416C62FE9458644B8E9188BDB5DC12402C4554A8347F29059DD411B7D09265586BF462D4
34280-70000P Exhibit C.pdf	E3C9A8F995CF5299351F2900196932BCC947CE8698969C19700756474E04C78861E984C995627AFEC6936BB13D2943B09DF02D8225D5E941CB6C1AA19A7677A6

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.