subatomic

Team Members

Team Member	Role	Scope / Responsibilities	Current Focus Areas
Aaron	Lead AI Engineer	AI architecture and supervision across projects; advanced agent development; workflow integration	Subatomic Atlas (agent enhancements, RBAC, RAG evaluation), Subatomic Nexus/Nucleus (workflow integration), Vantage Financial (memory features, HITL)
Emmanuel	Full-Stack & AI Developer; Data Engineer	Back-end/front-end dev; data pipeline engineering; AI/ML integration; memory management; RAG pipelines	Subatomic Atlas (navigation, regression tests, memory, doc standards, RBAC, latency, notifications), Vantage Financial (DAGs)
Christopher	Full-Stack Developer	Dedicated to Multi-Dimensional Contract Comparison Reviewer	Contract comparison, agent integration, orchestration middleware
Luis	AI Developer (Probation)	Reinforcement Learning, chatbot enhancement; Insights Agent development	Reinforcement Learning for Chat, pattern recognition agent, Insights Analyzer

Task Breakdown

SUBATOMIC ATLAS

Task Name: Support CSV and Various File Types in Core and Sales Agents

Attribute	Description
Assigned To	Aaron
Task Description	Extend the file ingestion pipeline in both the Core and Sales Agents to accept CSV and additional file formats (e.g., Excel, PDF, DOCX). Support parsing, validation, and mapping these files into internal knowledge representation for downstream use.
Purpose / Reason	Users need to ingest diverse types of data sources into the Knowledge Management system, supporting broader use-cases and business requirements. Improves accessibility and system utility.
Expected Output	Core and Sales Agents that can successfully ingest, parse, and process CSV, Excel, PDF, and DOCX files with correct mapping, error handling, and validation. Acceptance: end-to-end ingestion tests pass for all supported types.
Required Inputs	Sample files in each target format, target schema mapping documentation, access to current agent codebases.
Dependencies	Baseline ingestion pipeline must be operational; coordination with Emmanuel on notification system (for ingestion).
Technical Approach	Use file-type detection libraries (e.g., python-magic), integrate parsers (pandas for CSV/XLSX, PyPDF2/pdfplumber for PDF, python-docx for DOCX), extend pipeline for mapping/validation. Unit and integration testing for each format.
Sub-tasks	1. Audit current ingestion pipeline and identify points of extension. 2. Select and integrate libraries/parsers for each file type. 3. Implement validation/error handling. 4. Map parsed data to internal structure. 5. Test with diverse file samples. 6. Update user documentation. 7. Validate integration with notification system.
Estimated Complexity	Medium – Parsers for common formats exist, but error handling and mapping may be tricky.
Potential Challenges	Inconsistent file schemas; large file handling; edge case errors in parsing; mapping data to existing models.
Success Criteria	Files in each supported type are correctly ingested, parsed, and stored; automated tests pass; user can retrieve and utilize ingested knowledge.
Related Tasks	Notification system integration (Emmanuel), documentation standards (Emmanuel), RBAC (Emmanuel/Aaron)
Best Practices	Rely on proven libraries, log errors clearly, cover edge cases, maintain schema mapping standards.
Notes	Coordinate testing with Emmanuel for notification triggers and system integration. Prepare for future expansion to additional formats.

Task Name: Implement HITL with Dynamic Scope for Core and Sales Agents

Attribute	Description
Assigned To	Aaron
Task Description	Design and implement a Human-In-The-Loop (HITL) system for the Core and Sales Agents, enabling human review/approval of agent decisions/tasks. Incorporate a "dynamic scope" mechanism for adjustable review levels based on context (confidence, risk, or user role).
Purpose / Reason	Ensures critical/ambiguous agent actions get human oversight, improving trust, safety, and compliance. Dynamic scope allows balancing automation and control.
Expected Output	Configurable HITL system with: review queue/UI, rules for scope assignment, integration in agent action pipeline; documentation provided. Acceptance: test scenarios validate scope adjustment and review flow.
Required Inputs	Current agent workflow diagrams, user role definitions, examples of decision points, UI wireframes.

Attribute	Description
Dependencies	Agent execution pipeline; RBAC modules; Notification system; User directory
Technical Approach	Middleware pattern for agent action interception, rules engine (e.g., Python `rules`), review UI (React/Flask), asynchronous notifications, audit logging.
Sub-tasks	1. Draft HITL architecture and flow. 2. Implement detection of review-required events. 3. Build/reuse rules engine for scope assignment. 4. Develop review UI. 5. Integrate notification/audit logging. 6. E2E testing of agent-HITL integration. 7. Documentation.
Estimated Complexity	Complex – requires coordination across backend, frontend, and integration with existing flows.
Potential Challenges	Edge cases in review assignment; reviewer availability; latency; scope creep; maintaining auditability.
Success Criteria	HITL can be enabled/disabled; scope adjusts dynamically; positive user feedback on review experience.
Related Tasks	RBAC implementation (Emmanuel); notifications (Emmanuel); auditing/logging.
Best Practices	Modularize interception logic; make configuration adjustable; ensure seamless UX for reviewers.
Notes	Critical for regulated workflows; should tie into compliance reporting where applicable.

Task Name: Enable Multi-dimensional Wiki Management (Role-based)

Attribute	Description
Assigned To	Aaron
Task Description	Enable role-based and multi-dimensional access, editing, and organization of Wikis in Subatomic Atlas—so users see/edit content as appropriate for their role and context (department, permissions, etc.).
Purpose / Reason	Allows tailored knowledge distribution, prevents information overload, enforces access controls.
Expected Output	Wiki system supporting multiple "dimensions" (roles/departments), with permissions and structured navigation. Passes positive and negative access tests.
Required Inputs	Current Wiki DB/schema, RBAC framework, user role mappings, usage requirements from stakeholders.
Dependencies	RBAC implementation (Emmanuel); current Wiki infrastructure.
Technical Approach	Extend Wiki data model to support flags/tags for roles/departments; apply permissions at query/UI layer; partner with Emmanuel for RBAC modules.
Sub-tasks	1. Analyze existing Wiki model. 2. Define "dimensions" and tagging taxonomy. 3. Integrate with RBAC. 4. Adjust UI for filtering/search by dimension. 5. Test for permission leaks. 6. Document configuration.
Estimated Complexity	Medium – complexity depends on current data model flexibility.
Potential Challenges	Permission edge cases; user onboarding/migration; data migration.
Success Criteria	Only authorized users can access/edit appropriate sections; tested by QA/users.
Related Tasks	RBAC implementation, documentation generation, UI work (Emmanuel)
Best Practices	Fine-grained permissions, clear role assignment, logged permission checks.
Notes	Considerability for future expansion (e.g., tags, departments, projects).

Task Name: Navigation Support on Streaming Pages

Attribute	Description
Assigned To	Emmanuel
Task Description	Implement seamless navigation support (back/forward, deep linking) on pages with streaming content (e.g., live agent responses, document viewers). Ensure navigation events don't interrupt streams or cause UI/UX issues.
Purpose / Reason	Improves usability and accessibility, allowing users to navigate easily during/after data streaming.
Expected Output	Streaming page navigation works smoothly—no dropped streams, correct browser/app history updates, passes usability tests.
Required Inputs	Existing streaming UI code, page routing components, use cases for streaming navigation.
Dependencies	Current streaming implementation; knowledge of all affected pages.
Technical Approach	Use React Router (or app framework's router), implement route-preserving streaming state; apply Suspense/Context API if React.

Attribute	Description
Sub-tasks	1. Audit affected pages/components. 2. Refactor streaming logic for state persistence. 3. Implement navigation support. 4. Test with long-running streams. 5. Document approach for future streams.
Estimated Complexity	Medium – streaming state management can be tricky.
Potential Challenges	Preserving stream state across navigation; handling user re-entry to streaming routes.
Success Criteria	All navigation operations work without dropped data or errors.
Related Tasks	Regression testing (Emmanuel), UI styling validation
Best Practices	Decouple stream state from route, keep user informed of state changes.
Notes	Ensure accessibility for all users/devices.

Task Name: Validate Global Styling Across All Pages

Attribute	Description
Assigned To	Emmanuel
Task Description	Audit all Subatomic Atlas front-end pages/components to ensure global styles (typography, colors, layout) have been applied uniformly. Identify and fix deviations.
Purpose / Reason	Ensures consistent look and feel, brand integrity, and UI/UX standards.
Expected Output	No visual inconsistencies remain; all pages adhere to style guide. Acceptance: UI review checklist passes.
Required Inputs	Current style guide/specification, access to all UI components/pages for review.
Dependencies	Existing global styles in place; access to design documentation.
Technical Approach	Use automated tools (e.g., Storybook, Chromatic), manual review, code search for overrides.
Sub-tasks	1. List all views/components. 2. Automated scan for CSS overrides. 3. Manual visual review. 4. Fix style deviations. 5. Validate with stakeholders. 6. Update style guide if needed.
Estimated Complexity	Medium – depending on app size.
Potential Challenges	Legacy components; conflicting style overrides; browser compatibility.
Success Criteria	No reported styling inconsistencies after release.
Related Tasks	Navigation on streaming pages, regression tests.
Best Practices	Centralize styling with design tokens, avoid inline overrides.
Notes	Consider accessibility (WCAG) compliance during review.

Task Name: Improve Memory Report and Management (Episodic, Procedural, Semantic Types)

Attribute	Description
Assigned To	Emmanuel
Task Description	Refactor memory subsystem to categorize and report data as episodic, procedural, and semantic types; update management/visualization features to leverage these distinctions for better insights and control.
Purpose / Reason	Fine-grained memory categorization enhances transparency, debugging, and knowledge retrieval, supporting advanced AI behaviors.
Expected Output	Memory subsystem with clear distinctions and reporting for all types, updated management UI/API, documented usage.
Required Inputs	Current memory implementation, definitions/criteria for each memory type, user stories requiring memory management.
Dependencies	Existing memory system, RBAC if access matters, documentation update.
Technical Approach	Extend memory schemas, implement type tagging/classification; refactor retrieval/storage logic; enhance reports/UI.
Sub-tasks	1. Analyze existing memory types. 2. Draft/refine schema changes. 3. Update storage/retrieval logic. 4. Upgrade reporting tool. 5. Update UI/endpoint docs. 6. Validate by querying/testing.
	Medium – Requires design and data migration.

Attribute	Description
Estimated Complexity	
Potential Challenges	Backward compatibility; understanding implicit memory usage patterns.
Success Criteria	All memories properly classified and reported; acceptance tests pass for examples of each type.
Related Tasks	Documentation generation, RAG regression tests.
Best Practices	Document memory type criteria; log classification for traceability.
Notes	Lay groundwork for memory-based agent improvements.

Task Name: Generate Standard Documentation Format for RAG Ingestion & Retrieval

Attribute	Description
Assigned To	Emmanuel
Task Description	Develop and enforce a standardized documentation format/spec (likely Markdown/JSON) for content ingestible by Retrieval Augmented Generation (RAG) pipelines—including metadata tags for effective retrieval. Automate creation/conversion tools.
Purpose / Reason	Consistent documentation structure enables reliable ingestion, richer retrieval, and content governance.
Expected Output	Specified documentation format (template + schema), validator/converter tool, docs for users/authors; passes ingestion/retrieval tests.
Required Inputs	Current RAG ingestion code, existing content samples, search feature requirements.
Dependencies	RAG pipeline; schema/metadata requirements from AI agents.
Technical Approach	Current content analysis, template design, build linter/validator, code samples for automation.
Sub-tasks	1. Define required fields/metadata. 2. Design template (Markdown/JSON). 3. Build validation/conversion tool. 4. Update ingestion code for strict conformance. 5. Train users/authors. 6. Test with sample docs.
Estimated Complexity	Medium – depends on content diversity.
Potential Challenges	Legacy document conversion; user adoption; schema drift.
Success Criteria	100% ingestion conformance; no ingestion errors; users can convert legacy docs.
Related Tasks	RAG regression testing, notification of ingestion.
Best Practices	Version schema; build CI check for docs pre-ingestion.
Notes	Include example templates, changelog for spec evolution.

Task Name: Implement RBAC Across All Existing Modules

Attribute	Description
Assigned To	Emmanuel (in partnership with Aaron for agent-related parts)
Task Description	Design and implement full Role-Based Access Control (RBAC) for all modules, ensuring proper permissions, role management, and integration with user identity/authentication.
Purpose / Reason	Necessary for compliance, security, and tailored user experiences; supports other features (Wikis, HITL, Doc Ingestion)
Expected Output	Comprehensive RBAC implementation, passes penetration and misuse tests; documentation for roles/permissions; admin UI.
Required Inputs	List of user roles and permissions, user directory/SSO provider access, code for all modules
Dependencies	Authentication system; module list; requirements from PM/security
Technical Approach	Apply or extend access middleware, use RBAC libraries/frameworks (e.g. CASL for JS, Flask-Principal), test and document.
Sub-tasks	1. Inventory modules/entry points. 2. Define permission matrix. 3. Implement/extend RBAC middleware. 4. Update UI for permission control. 5. Testing (incl. negative tests). 6. Documentation.
Estimated Complexity	Complex – wide scope and risk if permissiveness too high.
Potential Challenges	Hidden codepaths; legacy code lacking clear authentication; change management.
Success Criteria	Zero unauthorized access in security testing; users have appropriate access only.

Attribute	Description
Related Tasks	Multi-dimensional Wiki management, HITL, all user-facing features.
Best Practices	Test with least-privilege; log permission failures; document all permission rules.
Notes	Prioritize critical modules; coordinate with Aaron for agent-integrated RBAC.

Task Name: Regression Testing for RAG Retrieval (Aligned With Supervised Evaluation)

Attribute	Description
Assigned To	Emmanuel (Aaron supervises for AI evaluation)
Task Description	Design and execute comprehensive regression tests on mechanisms for Retrieval Augmented Generation (RAG), focusing on retrieval accuracy, context usage, and system robustness under updates. Integrate with Aaron's supervised evaluation criteria.
Purpose / Reason	Ensures reliability and correctness of RAG after code/data/model changes. Early bug detection.
Expected Output	Regression test suite, test result reports, bug tickets for issues; reproducible and automated.
Required Inputs	Current RAG code, test dataset, documented retrieval requirements & criteria from Aaron
Dependencies	RAG retrieval mechanism; Aaron's evaluation framework; access to deployed environment.
Technical Approach	Implement automated regression suite (pytest/robot/Jest); use representative data, cover edge cases, assertions on context extraction quality.
Sub-tasks	1. Gather test cases/scenarios. 2. Implement automated test scripts. 3. Run and baseline tests. 4. Validate with Aaron. 5. Report/triage bugs. 6. Integrate in CI/CD.
Estimated Complexity	Medium – effort scales with scope/coverage.
Potential Challenges	Changing data/schema; ambiguous evaluation criteria; false positives/negatives.
Success Criteria	All regression tests pass after new changes; test suite covers >90% of typical scenarios.
Related Tasks	RAG documentation, HITL, notification of ingestion.
Best Practices	Version test cases; automate test runs; peer review assertions and coverage.
Notes	Coordinate test plan with Aaron to capture AI-specific nuances.

Task Name: Document Plan for Scaling Vector Store During Document Ingestion

Attribute	Description
Assigned To	Emmanuel
Task Description	Develop a plan and technical design for horizontally scaling the vector store used for document ingestion (sharding, replication, failover, etc.), including strategies for seamless integration and migration.
Purpose / Reason	Prepare for scale; maintain ingestion performance and reliability as data/documents grow.
Expected Output	Documented scalability plan (architecture, migration, risks), review with team; update backlog based on recommendations.
Required Inputs	Current vector store implementation, ingestion traffic data, performance benchmarks.
Dependencies	Underlying storage tech (e.g., Pinecone, Weaviate, self-hosted), understanding doc workloads
Technical Approach	Analyze scale points, propose sharding/partitioning scheme, plan for redundancy and monitoring. Include rollback/fallback approach.
Sub-tasks	1. Analyze current bottlenecks. 2. Research scaling patterns for current vector DB. 3. Draft proposed architecture. 4. Outline migration steps. 5. Identify monitoring/alerting needs. 6. Team review.
Estimated Complexity	Medium – mostly design but high impact.
Potential Challenges	Data migration downtime, consistency, vendor limitations.
Success Criteria	Plan approved, risk/impact understood, ready for implementation as needed.
Related Tasks	Ingestion pipeline, benchmarking, notifications.
Best Practices	Favor modular, vendor-agnostic design; update as data volumes grow.
Notes	Consider growth projections and multi-region design if needed.

Task Name: Benchmark & Reduce Latency in Existing Agents

Attribute	Description
Assigned To	Emmanuel
Task Description	Profile agent workflows to identify sources of latency, benchmark current performance, and optimize or remove unnecessary model/LLM calls to reduce overall agent response times.
Purpose / Reason	Improves user experience, increases system throughput, reduces cloud/Ops costs.
Expected Output	Benchmark report, optimized agent code, measurable reduction in response times; supporting documentation.
Required Inputs	Access to agent workflows/source code, current performance logs, list of LLM/model APIs in use.
Dependencies	Operational agents; performance test environment.
Technical Approach	Use profiling tools (cProfile, py-spy, browser perf tools), identify bottlenecks; refactor to batch/process in parallel or cache, cut unneeded API calls.
Sub-tasks	1. Profile agent execution end-to-end. 2. Identify slow operations. 3. Refactor/cut superfluous calls. 4. Implement caching/parallelism where sensible. 5. Run before/after benchmarks. 6. Document improvements.
Estimated Complexity	Medium – some analysis, some code refactor.
Potential Challenges	Dependencies on external APIs; risk of breaking downstream logic; accidental feature regression.
Success Criteria	Quantitative reduction in average/max response time; stable ops post-deployment.
Related Tasks	Benchmarking for scaling vector store; agent migration to Deep Agents.
Best Practices	Isolate optimizations for validation; track pre/post metrics for evidence.
Notes	Share learnings with the broader team for similar optimizations elsewhere.

Task Name: Reduce User Session Time and Implement MFA

Attribute	Description
Assigned To	Emmanuel
Task Description	Modify user authentication/session management to shorten session durations for improved security; add Multi-Factor Authentication (MFA) for all logins.
Purpose / Reason	Meets security best practices and compliance; mitigates risk of user credential compromise.
Expected Output	System with reduced session time and required MFA for all users; passes penetration and usability tests.
Required Inputs	Current auth/session code, access to MFA provider/API, security requirements (e.g., SOC2).
Dependencies	User auth system, RBAC, notification system (for MFA).
Technical Approach	Integrate with commercial MFA API (Authy, Okta, etc.), set session duration/timeouts in server code, update front-end for MFA flow.
Sub-tasks	1. Research/select MFA provider. 2. Integrate provider API into login flow. 3. Update session cookie/token settings. 4. Adjust UI for MFA. 5. Usability/security testing. 6. Documentation update.
Estimated Complexity	Medium – standard libraries exist but config/testing is critical.
Potential Challenges	MFA provider downtime; user friction or opt-outs; mobile authentication support.
Success Criteria	100% MFA enforcement and shorter session times on all user logins; zero bypasses.
Related Tasks	RBAC rollout, notification integration.
Best Practices	Provide fallback for lost MFA devices, audit/log all auth attempts.
Notes	Communicate change to users in advance, prepare support resources.

Task Name: Migrate All Agents to Deep Agents Module (LangGraph)

Attribute	Description
Assigned To	Emmanuel
Task Description	Refactor and transition all current agents to utilize the Deep Agents module from LangGraph for improved, maintainable agent orchestration.
Purpose / Reason	Consolidates architecture, improves orchestration, leverages latest LangGraph features, and reduces long-term maintenance.

Attribute	Description
Expected Output	All agents operational via Deep Agents, passes regression/stability tests; legacy code retired.
Required Inputs	Agent source code, Deep Agents module/SDK/API docs, test scripts.
Dependencies	Stable Deep Agents module; regression test suite.
Technical Approach	Audit current agents/features, design mapping to new module; implement incrementally, with test/validation at each stage.
Sub-tasks	1. Analyze agents to be migrated. 2. Plan migration sequence. 3. Refactor code for compatibility. 4. Implement & test with Deep Agents. 5. Roll out incrementally. 6. Retire/clean legacy agents. 7. Update documentation.
Estimated Complexity	Complex – agent-specific logic may make migration nontrivial.
Potential Challenges	Ensuring feature parity, regression risk; missing features in new module.
Success Criteria	100% of agents running via Deep Agents; equal or better stability/perf.
Related Tasks	Regression tests, benchmarking, doc generation.
Best Practices	Test in isolation, maintain rollout plan for rollback if issues emerge.
Notes	Identify and prioritize high-impact agents first for migration.

Task Name: Dynamic Connection to Notification System for Scheduled File Ingestions

Attribute	Description
Assigned To	Emmanuel
Task Description	Implement a dynamic, real-time integration between file ingestion workflows and the notification system, triggering updates/alerts to stakeholders on schedule or upon ingestion events.
Purpose / Reason	Keeps users/teams informed on ingestion status, supports transparency and coordination.
Expected Output	Notifications (email, in-app, etc.) triggered correctly for scheduled and ad hoc ingestion events; configuration options for notification recipients.
Required Inputs	Ingestion pipeline/event hooks, notification system API, stakeholder communication prefs.
Dependencies	Ingestion system, RBAC (for notification recipients), notification infra.
Technical Approach	Event-driven integration (webhooks or pub/sub), message templating, update notification rules/config.
Sub-tasks	1. Map ingestion points for event emission. 2. Integrate event triggers with notifications. 3. Implement recipient targeting logic. 4. Test events in sandbox. 5. Document notification options.
Estimated Complexity	Medium – system integration with some configurability.
Potential Challenges	Race conditions (duplicate/truncated events), noisy notifications, access control.
Success Criteria	Timely, accurate notifications for all ingestion schedules; no excess/duplicate alerts.
Related Tasks	Ingestion pipeline, RBAC, documentation.
Best Practices	Debounce/throttle notifications; make recipient targeting flexible.
Notes	Initial rollout with key teams, expand recipients as needed.

REINFORCEMENT LEARNING FOR CHAT WITH ATLAS

Task Name: Configure and Store RLHF Dataset

Attribute	Description
Assigned To	Luis
Task Description	Set up a secure, version-controlled data repository to store Reinforcement Learning from Human Feedback (RLHF) datasets for training/enhancing Chat with Atlas; enforce data privacy and safe access.
Purpose / Reason	Enables RL-based improvement of chatbot responses; ensures safe, recoverable data for current/future experiments.
Expected Output	Secure data store (e.g., S3, Databricks, GCP Bucket) with role-based access, versioning, clear documentation of data schema and use protocols.
Required Inputs	Initial RLHF data (annotation/export), infrastructure credentials, team data policies.
Dependencies	Access to infrastructure; data privacy compliance; system for dataset import/export.

Attribute	Description
Technical Approach	Provision cloud storage with RBAC/encryption, set up versioning/snapshot, enforce IAM policies, document process.
Sub-tasks	1. Choose/provision cloud storage. 2. Implement RBAC & encryption. 3. Upload & version baseline dataset. 4. Write docs on access/protocols. 5. Test data access by roles. 6. Periodic backup script.
Estimated Complexity	Medium – data pipeline/infra setup, but low code.
Potential Challenges	Security misconfigurations; data versioning errors; cost overage.
Success Criteria	Dataset can be safely accessed/updated by intended users; audit/logs confirm security.
Related Tasks	Pattern recognition agent (uses data), overall RL pipeline.
Best Practices	Encrypt at rest/in transit, automate audit logs, restrict data egress.
Notes	Validate permissions with Aaron before go-live.

Task Name: Implement Atlas Agent Insights Analyzer

Attribute	Description
Assigned To	Luis
Task Description	Develop an Insights Analyzer agent/component to evaluate and report on the impact of RL enhancements in the current Chat with Atlas; compare pre/post RLHF implementation metrics.
Purpose / Reason	Quantifies the business/UX impact of improvements, guides further optimization.
Expected Output	Analytics dashboard/report comparing conversation quality, user satisfaction, and agent performance before and after RLHF application.
Required Inputs	Historical and current chat transcripts, RLHF experiment logs, performance KPIs.
Dependencies	Dataset stored/setup complete, access to chat logs, KPIs defined.
Technical Approach	Data aggregation pipeline, metrics computation (e.g., BLEU, ROUGE, engagement), dashboard (e.g., React, Tableau, or Jupyter if early), automate regular reporting.
Sub-tasks	1. Define impact metrics. 2. Aggregate/test data pipelines. 3. Build insights computation scripts. 4. Create report/dashboard. 5. Document findings. 6. Plan ongoing analysis cadence.
Estimated Complexity	Medium – analytics-focused, collaboration on metric selection.
Potential Challenges	Data integrity; ambiguous metrics; insufficient data split for A/B.
Success Criteria	Clear, actionable report produced; team agrees on interpretation of results.
Related Tasks	RLHF dataset setup, pattern recognition agent.
Best Practices	Use blinded review; verify statistical significance where feasible.
Notes	Have Aaron review methodology for technical validity.

Task Name: Pattern Recognition Agent Based on Integrated Data Source

Attribute	Description
Assigned To	Luis (testing supervised by Aaron)
Task Description	Design and implement a pattern recognition agent leveraging ingested data sources to surface trends, anomalies, or actionable signals for the Atlas chat experience; Aaron to supervise and approve test design/results.
Purpose / Reason	Drives proactive insights and enables smarter, more adaptive chatbot behaviors.
Expected Output	Pattern recognition agent, code repo, sample analysis results, test suite signed off by Aaron.
Required Inputs	Integrated data source (from RLHF dataset or other ingestion), historical chat data, problem definitions/goals.
Dependencies	Data sources available, Aaron available for test review, infra for running/hosting.
Technical Approach	Use clustering (e.g., K-Means, DBSCAN), sequence analysis, or transformer-based models to detect patterns/anomalies; expose results via API/report.
Sub-tasks	1. Meet with Aaron for criteria. 2. Prepare datasets. 3. Prototype algorithms.

Attribute	Description
	4. Build/validate agent. 5. Analyze/interpret output. 6. Testing and documentation. 7. Aaron signs off on evaluation.
Estimated Complexity	Complex – model exploration, iterative.
Potential Challenges	Data volume/quality, false positives, model overfitting/generalizability.
Success Criteria	Patterns match expectations, useful signals delivered; reviewed and approved by Aaron.
Related Tasks	RLHF dataset, Insights Analyzer, agent orchestration.
Best Practices	Build incremental, start with simple patterns, validate results with real-world feedback.
Notes	Use explainability tools as feasible—surface "why" as well as "what".

SUBATOMIC NEXUS

Task Name: Integrate Full Workflow in UI

Attribute	Description
Assigned To	Aaron
Task Description	Build UI/workflow in Subatomic Nexus that guides users through the entire data engineering process, from data source connection to data processing and output, with status and error handling.
Purpose / Reason	Delivers a user-friendly, end-to-end solution; reduces user friction and error.
Expected Output	Intuitive UI supporting all workflow stages, tested with "happy" and edge-case paths, demo-ready.
Required Inputs	Workflow definitions, UI design mocks/specs, APIs for workflow steps, feedback from beta users.
Dependencies	Stable backend/workflow API, design input, authentication/authorization infra.
Technical Approach	Use React or similar for front-end, integrate workflow state with backend API, visually indicate status/errors, log all workflow steps.
Sub-tasks	1. Confirm E2E workflow definition. 2. Draft UI design. 3. Implement stepwise user flow. 4. Integrate error/status feedback. 5. User testing. 6. Iterative improvements. 7. Final demo.
Estimated Complexity	Medium – full-stack coordination but standardized elements.
Potential Challenges	Workflow step consistency, error propagation/display, real-time status sync.
Success Criteria	Users can complete full workflow in UI; feedback indicates clarity and usability.
Related Tasks	Workflow integration in Subatomic Nucleus, error handling patterns.
Best Practices	Use stepper/progress indicator; surface clear help/errors; test all workflow branches.
Notes	Reuse UI components where possible for maintainability.

SUBATOMIC NUCLEUS

Task Name: Integrate Full Workflow in UI

Attribute	Description
Assigned To	Aaron
Task Description	Build UI to support the full workflow for AI Co-Worker Team Generation: from team configuration, capability choice, tool assignment, to team deployment/activation.
Purpose / Reason	Streamlines user experience, provides transparency, and reduces team formation errors.
Expected Output	UI supports all Nucleus workflow steps with guidance, error handling, permissions integration. Beta users can form AI teams successfully.
Required Inputs	Team generation workflow, available tools catalog, UX specs, auth/RBAC info.
Dependencies	Stable workflow engine, tool catalog, RBAC.
Technical Approach	Modular UI (React), guided wizard/stepper, integrate with workflow and RBAC APIs, mock with real world/team samples.
Sub-tasks	1. Confirm workflow stages. 2. UI wireframe/mockup. 3. Implement configuration/selection. 4. Integrate with workflow/tool APIs.

Attribute	Description
	5. Handle errors/perms. 6. Beta feedback/testing. 7. Finalize for launch.
Estimated Complexity	Medium – mirrors Nexus but adds more branching.
Potential Challenges	Handling workflow exceptions; tool assignment edge cases.
Success Criteria	End-users can successfully create/deploy AI teams; workflow tracked/logged for support.
Related Tasks	Workflow UI in Nexus, dynamic tool assignment.
Best Practices	Inline progress, save/resume draft, confirm each step.
Notes	Sync workflow logic with backend to minimize translation bugs.

Task Name: Scale and Improve AI Co-Worker Tool Generation with Dynamic Tool Calling Agentic Pattern

Attribute	Description
Assigned To	Aaron
Task Description	Refactor Nucleus to support dynamic tool assignment and invocation using agentic patterns—AI co-workers automatically select/invoke appropriate tools depending on user goal and team configuration, enabling scale and flexibility.
Purpose / Reason	Increases platform flexibility, supports complex use-cases, underpins future growth of AI team toolsets.
Expected Output	Nucleus supports agentic tool calling: dynamic selection/invocation, logging, and fallback; tests confirm correct tool assignment/execution.
Required Inputs	Catalog of available tools, agentic pattern references (LangChain, etc.), current team-gen code, use-case samples.
Dependencies	Up-to-date tools catalog, agent orchestration logic, RBAC (for permissioned tools).
Technical Approach	Apply AI agent pattern (intent parsing, tool selection), refactor orchestration code, model dynamic capabilities.
Sub-tasks	1. Design/refine agentic tool selection logic. 2. Refactor co-worker orchestration. 3. Integrate tool catalog APIs. 4. Implement tool selection/execution logging. 5. Test multiple team configurations. 6. Update UI/docs. 7. Gather feedback, iterate.
Estimated Complexity	Complex – dynamic dispatch, ongoing refactor, new logic.
Potential Challenges	Tool permissioning, model ambiguity, error handling in tool failures.
Success Criteria	Dynamic tool assignment works seamlessly; demonstrated with real user workflows.
Related Tasks	Nucleus workflow UI, Deep Agents migration.
Best Practices	Modularize agent logic; provide clear fallback/exception flows; document invariant behaviors.
Notes	Coordinate tool metadata schemas with Emmanuel if shared with other modules.

MULTI-DIMENSIONAL CONTRACT COMPARISON REVIEWER

Task Name: Add Agent for Client Qualification ("Less Restrictive" Terms & Conditions)

Attribute	Description
Assigned To	Christopher
Task Description	Design and implement a specialized agent that analyzes contracts to identify and qualify clients based on detection of less restrictive terms and conditions (T&Cs). Integrate with overall contract review flow.
Purpose / Reason	Streamlines and automates client vetting, enabling better/faster risk assessment.
Expected Output	Agent code fully integrated, documented, and validated with real-world contract samples; accuracy metrics reported.
Required Inputs	Contract samples with annotated T&Cs, legal criteria for "less restrictive", integration specs, access to comparison pipeline.
Dependencies	Current comparison reviewer framework, access to contract DB, annotated training data.
Technical Approach	Use NLP models (e.g., GPT-4.x), rule-based analysis for T&C extraction/classification, integrate with reviewer UI/backend.
Sub-tasks	1. Gather legal/classification criteria. 2. Prepare annotated contract data. 3. Prototype extraction/classification. 4. Integrate with comparison backend.

Attribute	Description
	5. Validate outputs/test cases. 6. Document feature/add user guide.
Estimated Complexity	Medium – relies on clear T&C definitions, tested pipelines.
Potential Challenges	Defining/agreeing on "less restrictive", ambiguous contract language, model calibration.
Success Criteria	Agent correctly flags clients per criteria; passes legal/user evaluation.
Related Tasks	Comparison reviewer orchestration, Deep Agent integration.
Best Practices	Collaborate with legal SME, build explainable output, unit test extensively with edge cases.
Notes	Feedback loop with SME/legal required for continuous improvement.

Task Name: Implement Deep Agent (LangChain) with GPT-4.1 Orchestration and Planning Middleware

Attribute	Description
Assigned To	Christopher
Task Description	Integrate a Deep Agent (LangChain) with GPT-4.1 as the top-level planner/orchestrator, using all internal sub-agents. Middleware should enable "Planning Mode" for structured/stepwise case handling.
Purpose / Reason	Centralizes advanced orchestration, supports transparency and scalability as more agents are added.
Expected Output	Orchestrator runs all sub-agents per plan, logs all steps, user can view/modify execution plan.
Required Inputs	Current agent code, LangChain Deep Agent docs, list of internal sub-agents, workflow specs.
Dependencies	All sub-agents available; LangChain/GPT-4.1 access; contract comp backend ready.
Technical Approach	Use LangChain agent composition, define planner/step logic in GPT-4.1 prompts/middleware, log execution path, integrate into Review UI.
Sub-tasks	1. Inventory sub-agents and roles. 2. Design planning middleware logic. 3. Integrate GPT-4.1 planner. 4. Refactor for stepwise orchestration. 5. Test with complex cases. 6. Expose/edit plan in UI. 7. Document usage and maintenance.
Estimated Complexity	Complex – AI orchestration, real-time planning, integration overhead.
Potential Challenges	Step consistency, sub-agent misalignment, GPT-4.1 prompt design.
Success Criteria	All agents orchestrated successfully in stepwise fashion; execution is traceable and auditable.
Related Tasks	Client qualification agent, reviewer UI integration.
Best Practices	Log all plan steps, provide fallback in error states, modularize for future sub-agent expansion.
Notes	Coordinate with backend team for smooth integration; document for future handoffs.

VANTAGE FINANCIAL

Task Name: Deploy Automatic DAGs (Airflow) to Azure

Attribute	Description
Assigned To	Emmanuel
Task Description	Automate deployment of Airflow Directed Acyclic Graphs (DAGs) into Azure (likely using Azure Data Factory/Airflow integration). Include CI/CD pipeline triggers, error handling, and monitoring.
Purpose / Reason	Enables scalable, automated data engineering for client; reduces manual error and operational overhead.
Expected Output	DAGs successfully deployed and running in Azure; documented CI/CD pipeline, monitoring alerts in place.
Required Inputs	Airflow DAG code, Azure subscription/credentials, deployment architecture, monitoring tool access.
Dependencies	Stable Airflow environment, Azure setup, access to code repo.
Technical Approach	Use Azure DevOps (or similar) for pipeline, write deploy scripts, integrate DAG validation and monitoring.
Sub-tasks	1. Confirm Azure/Airflow environment. 2. Adapt DAGs to Azure environment. 3. Set up CI/CD workflow. 4. Implement DAG health monitoring. 5. Test deploy pipeline. 6. Write deployment/runbook docs.
	Medium – some cloud/infra lift, standard Airflow patterns.

Attribute	Description
Estimated Complexity	
Potential Challenges	Azure perm/access issues, DAG cross-dependencies, monitoring/alerting noise.
Success Criteria	100% automated deploy/redeploy; no missing/failed DAGs post-pipeline run.
Related Tasks	Agenda-related features (Aaron), client-specific memory integration.
Best Practices	Immutable infra where possible, alert on pipeline failure, periodic validation jobs.
Notes	Schedule routine audits for pipeline and DAG failures.

Task Name: Client-Specific Memories in Agenda Compilation

Attribute	Description
Assigned To	Aaron
Task Description	Integrate support for client-specific knowledge/memory in the agenda compilation system, enabling context-aware agenda proposals and recommendations for Vantage Financial.
Purpose / Reason	Personalizes automated agenda building, improves relevance and client value.
Expected Output	Agenda compiler accesses/utilizes client memories; user sees accurate, tailored output in end-to-end testing.
Required Inputs	Client memory schemas, current agenda compilation flows, test client data.
Dependencies	Memory subsystem, agenda compiler, access/permission management.
Technical Approach	Extend agenda compiler logic for memory access; test queries with real/simulated clients; integrate w/ permission checks.
Sub-tasks	1. Review agenda and memory system. 2. Define client memory API/integration. 3. Implement in compilation pipeline. 4. Test with simulated clients. 5. Validate user-facing accuracy. 6. Document feature.
Estimated Complexity	Medium – depends on memory integration complexity.
Potential Challenges	Memory schema mismatches, permission edge cases, interface stability.
Success Criteria	Agendas produced are more accurate/relevant for specific clients; passes user eval.
Related Tasks	Enhanced HITL for visual agendas, DAG deployment (Emmanuel).
Best Practices	Modularize for future clients, log memory access for debugging.
Notes	Coordinate memory schema unification with Emmanuel if applicable.

Task Name: Enhanced HITL for Visual Agendas

Attribute	Description
Assigned To	Aaron
Task Description	Develop Human-In-The-Loop (HITL) mechanism for visual agenda generation, enabling human users to review, modify, or approve compiled agendas before finalization and delivery.
Purpose / Reason	Ensures accountability, reduces errors, increases trust for high-stakes financial outputs.
Expected Output	Visual agenda HITL system implemented, with clear review/approval UI, audit logs, and configuration options.
Required Inputs	Visual agenda samples, review process requirements, current agenda system.
Dependencies	Agenda pipeline complete, suitable UI component available, access controls.
Technical Approach	Middleware interception of agenda outputs, React UI for review, assign tasks/notifications to reviewers, log all review actions.
Sub-tasks	1. Analyze visual agenda output pipeline. 2. Design review/approval workflow. 3. Build/extend review UI. 4. Integrate review logic. 5. Log all actions/audits. 6. End-to-end testing. 7. User training/docs.
Estimated Complexity	Complex – cross-team/IU integration needed, user education.
Potential Challenges	Reviewer availability; feature creep; audit log completeness.
Success Criteria	Users can reliably review/approve agendas; logs capture all actions; no unapproved outputs.

Attribute	Description
Related Tasks	Client memory integration, DAG deployment.
Best Practices	Modularize reviewer UI, build in reviewer delegation/escalation, automate notifications.
Notes	Consider regulatory requirements for audit trail retention.

Task Name: Scheduled & On-Demand Agenda Compilation

Attribute	Description
Assigned To	Aaron
Task Description	Implement both scheduled and user-triggered ("on-demand") generation of agendas; ensure result consistency, status reporting, and error recovery.
Purpose / Reason	Supports automated workflows and ad hoc requests, improves flexibility for Vantage Financial.
Expected Output	System can create agendas per schedule, or when users press "compile", with process tracking and usable error messages.
Required Inputs	Scheduling requirements, UI/UX flow, agenda API, user stories.
Dependencies	Agenda compiler feature set, notification system, RBAC.
Technical Approach	Use job/task queue for scheduling, API endpoints for triggers, status notification hooks, logging for failure cases.
Sub-tasks	1. Define schedule/triggers. 2. Implement job queue processing. 3. Build on-demand UI/API. 4. Integrate notifications. 5. Monitor/surface errors. 6. QA for schedule edge cases. 7. User documentation.
Estimated Complexity	Medium – workflow automation with notifications.
Potential Challenges	Job queue reliability, duplicate agenda runs, time zone support.
Success Criteria	Agendas compile as expected in both modes, no missed/double runs, status clear to users.
Related Tasks	Visual agenda HITL, DAG automation.
Best Practices	Debounce jobs where needed, track all runs for auditing.
Notes	Schedule periodic revalidation of trigger/schedule reliability.

Task Prioritization & Sequencing

Critical Path Tasks:
- Implement RBAC Across All Existing Modules (Emmanuel): Needed for security and supports several other features (Wiki, HITL, notification targeting).
- Migrate All Agents to Deep Agents Module (Emmanuel): Architectural consolidation needed for support/feature expansion.
- Support CSV & Diverse File Types in Core & Sales Agents (Aaron): Expands data sources for knowledge system—core platform feature.
- Configure and Store RLHF Dataset (Luis): Needed as basis for RL and pattern agent efforts.
- Integrate Full Workflow in UI for Nexus/Nucleus (Aaron): Needed for complete E2E product demo/functionality.

High Priority Tasks:
- HITL Implementation for Agents (Aaron), Visual Agenda HITL (Aaron): Enhances trust/compliance.
- Navigation on Streaming Pages, Global Styling, Regression Tests (Emmanuel): Improves stability and user experience.
- Pattern Recognition Agent (Luis), Insights Analyzer (Luis): Drives direct improvement of chatbot systems.
- DAG Deployment and Client Memory in Agenda Compilation (Aaron & Emmanuel): Core for Vantage Financial deliverables.

Can Be Parallelized:
- UI styling/global styling and navigation support (Emmanuel).
- Documentation generation and reporting/memory management improvement (Emmanuel).
- Pattern Recognition and Insights Analyzer (Luis; can use mock data at first).
- Contract comparison reviewer tasks (Christopher—isolated codebase).

Suggested Task Order:

1. **Implement RBAC Across Modules** (Emmanuel)
2. **Migrate All Agents to Deep Agents** (Emmanuel)
3. **Support CSV & New File Types in Core/Sales Agents** (Aaron)
4. **Configure & Store RLHF Dataset** (Luis)
5. **Integrate Full Workflow UI (Nexus & Nucleus)** (Aaron)
6. **Implement HITL (Agents, Visual Agendas)** (Aaron)
7. **Enable Navigation on Streaming Pages** (Emmanuel)
8. **Validate Global Styling on All Pages** (Emmanuel)
9. **Regression Testing for RAG Retrieval** (Emmanuel)
10. **Benchmark & Reduce Latency in Agents** (Emmanuel)
11. **Improve Memory Report & Management** (Emmanuel)
12. **Generate Standard Documentation Format for RAG** (Emmanuel)
13. **Pattern Recognition Agent** (Luis; supervised by Aaron)
14. **Implement Atlas Agent Insights Analyzer** (Luis)
15. **Dynamic Notification Connection for Ingestion** (Emmanuel)

16. **Deploy DAGs to Azure** (Emmanuel)
17. **Client-Specific Memories in Agenda Compilation** (Aaron)
18. **Scheduled and On-Demand Agenda Compilation** (Aaron)
19. **Enhanced HITL for Visual Agendas** (Aaron)
20. **Nucleus: Scale & Improve Tool Generation with Dynamic Agentic Pattern** (Aaron)
21. **Add Client Qualification Agent (Contract Reviewer)** (Christopher)
22. **Deep Agent (LangChain + GPT-4.1) Orchestration for Comparison Reviewer** (Christopher)
23. **Document Plan for Scaling Vector Store** (Emmanuel)
24. **Reduce User Session Time, Implement MFA** (Emmanuel)

Summary & Recommendations

Key Insights:

- **RBAC and Deep Agent migration** are top architectural priorities, blocking several downstream features requiring fine-grained access control and orchestrated workflows.
- **Aaron** is spread across several projects—prioritize based on features that unblock teammates or deliver on immediate client needs.
- **Luis** should focus on completing foundational RLHF setup before moving onto analytics and agent builds—ensuring Aaron can supervise/validate as needed.
- **Emmanuel's workload** is heavy on Subatomic Atlas UI/backend and Vantage Financial infra; ensure tasks are spread to match capacity and readiness.
- **Christopher's project** is siloed and can progress in parallel, minimizing interdependencies.

Resource Considerations:

- **Potential bottlenecks**: RBAC/agent migration (Emmanuel), agent review/supervision (Aaron)
- **Onboarding/support**: Luis will need guidance from Aaron for RL/pattern work; ensure regular check-ins.
- **Stagger deep/complex integrations** (e.g., HITL, LangGraph migrations) to reduce risk of system-wide regression.

Risk Areas:

- **RBAC rollout**: Permission errors can block features or cause security risk—test thoroughly!
- **Cross-module migrations** (LangGraph, RBAC): May reveal hidden dependencies or legacy codepaths.
- **RL/pattern agents**: Data quality and interpretability may hamper results; allocate time for troubleshooting/iteration.
- **User-facing features**: UI/UX regressions (styling, navigation) may draw end-user complaints.

Recommendations:

- Confirm priorities at team meeting, clarifying feature dependencies and milestones.
- Pair complex backend architectural tasks (RBAC, Deep Agents) with thorough peer reviews and comprehensive testing.
- Ensure documentation and knowledge transfer are included as deliverables for modularization and future maintainability.
- Schedule regular checkpoints for Luis's work to assure alignment and knowledge transfer with Aaron.
- Use early prototypes/mocks for UI-heavy work to gather quick feedback, minimizing rework cycles.
