Agentic Task Force Canvas – Full Feature Backlog, Technologies & Design Principles

1. Core Design Principles

- Low-code/No-code First: Abstract complexity behind intuitive UIs and dropdowns.
- Reusability: Modular agent templates and connectors should be reusable across workflows.
- Transparency: Every agent and routing decision must be observable and auditable.
- Human-in-the-Loop: Manual overrides and feedback are integral to trust and correction.
- Semantic Context Awareness: Use structured and unstructured data (KG + embeddings) for accurate routing.
- Multi-Tenancy and Security by Design: Tenant isolation, RBAC, and per-tenant data encryption.
- Open and Extendable: Enable integration with different LLMs, APIs, data sources.

2. Build Backlog with Suggested Technologies

Feature/Component	Suggested Technologies
Canvas UI (drag & drop agents)	React, React Flow
Agent Builder Panel (tools, memory, LLM)	React, Zustand, TailwindCSS
Reusable Agent Templates	JSON templates, Redux or Zustand
Routing Connector + Inspector Panel	React Flow Extensions, Formik, SVG
Live Routing Visualization	Color-coded paths, conditional render logic
GraphQL Knowledge Interface	Hasura, Apollo Server, Neo4j GraphQL
Cypher + Advanced Queries	Neo4j driver, GraphQL @cypher extensions
Temporal Execution Engine	Temporal Python SDK
LangGraph DAG Orchestration	LangGraph, LangChain DAG builder
Backend Service Layer	FastAPI, Uvicorn, SQLAlchemy
Database (Workflows, Logs)	PostgreSQL
Memory Store for Agents	Qdrant or Weaviate
Knowledge Graph Backend	Neo4j or AWS Neptune
Authentication + RBAC	OAuth2, Auth0, FastAPI Users
Multi-Tenancy Layer	Temporal Namespaces, DB schema partitioning
Audit Logging + Feedback	PostgreSQL, structured JSON log pipeline
ETL from Procurement Systems	Airbyte, dbt, Fivetran
Agent Memory & RAG Context	Embeddings (OpenAI/SBERT) + Qdrant
Execution Log Dashboard	WebSockets, React charts

Learning Feedback Capture	LLM prompt correction forms, feedback classifier