Verizon Project Management Tool: System Design & Implementation Plan (Updated)

1.0 Executive Summary

This document outlines the system design for a secure, local-first Project Management (PM) Tool for Verizon. The current spreadsheet-based tracking system is inefficient, does not scale for project managers handling 500+ projects, and lacks necessary KPI reporting for management. The legacy Oracle system it aims to replace is complex and nearing decommissioning.

This new tool solves these problems by providing a high-performance desktop application (runnable via Streamlit) that operates entirely within the Verizon network, adhering to corporate branding guidelines. It requires no new database servers or cloud APIs. It uses local SQLite databases for speed and a secure "Inbox" synchronization model to safely consolidate data to a master database on a shared network drive (simulated as `G_DRIVE`), preventing data corruption.

The system provides role-based dashboards with UI defaults, automated KPI reporting (including monthly scorecards and interval tracking), project dependency management, a project-level contact list (RE, CE, etc.), fields mapped from legacy systems (OPS Numbers, addresses), and an on-device AI assistant. This dramatically reduces administrative overhead, provides instant and accurate reports, supports migration from the legacy Oracle system, and introduces AI-driven insights—all while adhering to security constraints.

2.0 Core Architecture (The "Safe Way")

The application is built in Python using Streamlit for the UI. While initially run via `streamlit run`, it's designed to be packageable into a standalone desktop application (`.exe`) using PyInstaller. This model is secure and requires no server installation beyond Python and necessary libraries.

- **Primary Storage (Local):** Each user runs the app on their local machine. Their working database (`my_projects_USERID.db`) is stored locally (simulated as `LOCAL_DRIVE`). This guarantees speed and eliminates file-locking/corruption issues common with shared network drives.
- **Shared Storage (Network):** A shared network drive (`G_DRIVE`) is used only as a secure "post office" for data synchronization (`SYNC_INBOX`, `ARCHIVE`), central configuration (`config.db`), master databases (`master_users.db`, `master_projects.db`), and AI resources (`AI_Knowledge_Base`, `AI_Feedback`). The application *never* runs a live database directly from the shared drive.

3.0 Security & Access Control

Access is controlled by a mandatory login system via master_users.db. This allows a single application deployment to serve all roles securely.

- **User Database:** master_users.db stores user credentials (hashed passwords) and assigned roles.
- **Role-Based Access Control (RBAC):** The application UI and features adapt based on the logged-in user's role:
  - **Sr. Project Manager (PM):** Manages assigned projects in their local DB, takes KPI snapshots, manages dependencies and project contacts, syncs data, interacts with the AI Assistant.
  - **Principal Engineer:** Read-only view of all master projects (All Projects page), access to reports.
  - **Associate Director (Admin):** "Team View" dashboard, Admin Control Panel (user management, config editor, AI Indexer), Spreadsheet Importer, Sync Inbox Processor, access to all reports.
  - **Director:** High-level portfolio dashboard (Reports page), read-only view of all master projects.
- **Hierarchy:** master_users.db includes a reports_to_id column, enabling automated team-based views and reporting roll-ups.

4.0 Data Synchronization (The "Inbox/Processor" Model)

This core process ensures safe, asynchronous consolidation of data from multiple local databases to the master database.

1. **PM Works Locally:** A PM adds/updates projects, KPIs, dependencies, and contacts in their local my_projects_USERID.db. Changes are marked with a sync_status ('new' or 'updated').
2. **PM Clicks "Sync Data":** The app bundles all items marked 'new' or 'updated' (projects, KPIs, dependencies, contacts) into a unique JSON file (e.g., sync_JaneDoe_20251024_124500.json). This file is copied to the G_DRIVE\SYNC_INBOX\ folder. Local items are marked 'synced'.
3. **Admin Clicks "Process Sync Inbox":** At any time, an Associate Director uses the "Process Sync Inbox" page. Their application instance:
   - Lists JSON files in the SYNC_INBOX.
   - For each file:
     - Reads the JSON bundle.
     - Safely merges changes into master_projects.db using upsert logic (based on ccr_nfid for projects, inserting KPIs/dependencies/contacts). Master IDs are mapped for dependencies.

        - Moves the processed JSON file to `G_DRIVE\ARCHIVE\`.

This one-way, asynchronous flow prevents data corruption and ensures reliability.

5.0 Core Data Model & Features

The database schema uses SQLite and includes tables validated against the legacy Oracle system (`GS.GS_WFM_NF_PROJECTS`, `GS.GS_CCP_CCRS`, `GS.GS_WFM_NF_PRJ_MILESTONES`, etc.) and the "GLS 2025 Scorecard" report.

- **`master_users.db`:**
    - `users`: Stores user credentials, roles, and reporting hierarchy.
- **`master_projects.db`:**
    - `programs`: High-level initiatives (e.g., '5G Rollout').
    - `project_types`: Categories (e.g., 'BAR', 'Circuit', 'BAU Rev', 'Decom').
    - `projects`: Core project table. Includes standard fields (`name`, `ccr_nfid`, `program_id`, `project_type_id`, `pm_id`, `status`, `phase`, `notes`) **plus fields validated against legacy systems:**
        - **Legacy Oracle Fields:** `nfid`, `customer`, `clli`, `rft_date`, `system_type`, `current_queue`, `site_address` (TEXT).
        - **Scorecard Reporting Fields:** `project_start_date` (DATE), `project_complete_date` (DATE). These are critical for calculating intervals.
    - `project_dependencies`: Links projects (`project_id`, `depends_on_project_id`). Replaces complex legacy relationships.
    - `project_contacts`: **(New)** Stores project team members.
        - `contact_id` (PK), `project_id` (FK), `contact_name` (TEXT), `contact_role` (TEXT, e.g., "RE", "CE", "SME"), `contact_email` (TEXT).
    - `work_packages`: Tasks, milestones, deliverables (`project_id`, `name`, `due_date`, `status`). Schema exists, UI pending.
    - `kpi_snapshots`: Historical metrics (`project_id`, `snapshot_date`, `budget_status`, `schedule_status`, `on_time_percent`).
    - `ai_knowledge_base`: Stores text chunks and embeddings from indexed documents.
    - `ai_feedback`: Logs user feedback on AI responses.
- **`config.db`:**
    - `config_settings`: Key-value store for business logic (SLA formulas, KPI thresholds, AI model name), editable via the Admin Panel.
- **`my_projects_USERID.db` (Local):**
    - Mirrors relevant master tables (`projects`, `kpi_snapshots`, `project_dependencies`, `work_packages`, `project_contacts`) but includes `sync_status` and uses local IDs,

mapping to master_project_id after sync.

6.0 User Interface (UI) & Key Workflows (Modular App Structure)

The application uses Streamlit in a multi-page app structure located in the app/ directory, promoting modularity and team collaboration. Core logic resides in the src/vtrack/ package.

- **app/Home.py:** Login page.
- **app/sidebar.py:** Manages sidebar display and logout.
- **app/pages/ Directory:** Contains individual .py files for each major view/feature:
    - **1_My_Dashboard.py:** PM's home screen. Features:
        - Editable st.data_editor for local projects (including site_address).
        - Expander for taking KPI Snapshots.
        - Expander for managing Project Dependencies.
        - *(Future: Will add links to the new Project Details page)*.
    - **2_New_Project.py:** Form for creating new project stubs (including site_address).
    - **3_Sync_Data.py:** Displays pending changes (projects, KPIs, dependencies, contacts) and triggers sync to inbox.
    - **4_Team_Dashboard.py:** Admin/Director view of master projects with filtering.
    - **5_Import_Data.py:** Admin tool for bulk importing projects from spreadsheets (.xlsx, .csv), will include site_address.
    - **6_Process_Sync_Inbox.py:** Admin tool to process JSON sync files.
    - **7_Admin_Panel.py:** Admin interface with tabs for:
        - User Management (Create users).
        - SLA & KPI Configuration (Edit config.db).
        - AI Knowledge Base Indexer.
    - **8_Reports.py:** Role-based reporting dashboards. Will be expanded in Phase 6.
    - **9_All_Projects.py:** Read-only view of all master projects for authorized roles.
    - **10_AI_Assistant.py:** Chatbot interface for querying the knowledge base, with feedback mechanism.
    - **11_Project_Details.py (Future - Phase 5):** Page for managing work_packages (milestones/tasks) and project_contacts (team) for a specific project.

6.1 Branding

The application UI will incorporate Verizon branding elements:

- **Logo:** The official Verizon logo will be displayed prominently, likely at the top of the sidebar on all pages.
- **Color Scheme:** A custom Streamlit theme will be implemented using a .streamlit/config.toml file to apply Verizon's primary colors (Red, Black, White/Gray) to

UI elements like headers, buttons, and backgrounds, ensuring visual consistency with corporate standards.

6.2 UI Templates & Defaults

To improve user efficiency, the application will incorporate template-like features:

- **Role-Based Defaults:** Forms and views will be tailored to the user's role. For example, the 'PM' field might be pre-filled when a PM creates a new project. Required fields might differ based on role or context.
- **Future Enhancements:** Consideration will be given to adding features allowing users to save custom filter sets on dashboards or define reusable templates for creating projects with standard milestones based on `project_type`. This would require extending the database schema to store user preferences.

6.3 Advanced Reporting Dashboards (Phase 6)

The "GLS 2025 Scorecard" and "UT Market Update" show the need for advanced reporting. These will be added to the `app/pages/8_Reports.py` page:

- **Monthly Scorecard:** A dynamic report, similar to the "GLS 2025 Scorecard," that allows selecting a time range (e.g., YTD) and displays:
  - Monthly counts of "Projects Closed" and "Projects Met RFT."
  - Grouping by `project_type` (e.g., "BAU Rev," "Decom," "BAU Non-Rev").
  - Calculation of "Average Interval" (in business days) from `project_start_date` to `project_complete_date` or `rft_date`.
  - "In-flight" project counts (projects active during a given month).
- **Visual Program Tracker:** A report inspired by the "UT Market Update" PDF.
  - User selects a `program` (e.g., "UT Market Update").
  - The report displays a table of all projects in that program, grouped by `clli` (Site).
  - Key fields like `current_queue` (Equipment Status), `system_type` (Equipment Type), and `site_address` will be shown, providing a live "PDF-style" status view.

7.0 Artificial Intelligence (AI) Sub-System

Implemented using Hugging Face models running locally for security.

- **Local-First AI:** Uses `sentence-transformers` for embedding documents and queries. Model name is configurable in `config.db`. Embeddings run on the user's machine (or Admin's machine for indexing).
- **AI Knowledge Base:**
  - **Indexing (Admin):** The "Run AI Indexer" button in the Admin Panel scans `.pdf`, `.docx`, `.txt` files in `G_DRIVE\AI_Knowledge_Base\`. It chunks the text, generates embeddings using the configured model, and stores chunks + embeddings in `master_projects.db`. Uses `faiss-cpu` for efficient similarity search.

    - **Searching (All Users):** The AI Assistant page embeds user queries and performs a similarity search against the indexed embeddings using FAISS. Relevant text chunks are retrieved and presented.
- **AI Assistant Window (`10_AI_Assistant.py`):** Provides a chatbot interface (`st.chat_input`, message history) for querying the knowledge base.
- **AI Learning/Feedback:** Users can rate chatbot responses (👍/👎). Feedback (prompt, response, rating) is stored in the `ai_feedback` table in `master_projects.db`. This data can be used for future model fine-tuning (manual process for now).
- **AI Hints:** *(Not yet implemented)* The plan included proactive hints; this would be a future enhancement.

8.0 Full Technical Stack

- **Programming Language:** Python (v3.10+)
- **Database:** SQLite 3 (via Python's `sqlite3` library).
- **User Interface (UI):** Streamlit (multi-page app structure).
- **Data Handling:** Pandas (for spreadsheet import, data manipulation).
- **AI Engine:**
    - Embeddings: Hugging Face `sentence-transformers` library (model configurable).
    - Vector Search: `faiss-cpu`.
    - Document Loading: `pypdf`, `python-docx`.
    - Core ML: `torch`, `transformers` (as dependencies).
- **Packaging (Future):** PyInstaller (to bundle into `.exe`).
- **Project Structure:** Modular layout (`src/vtrack` package, `app/` UI directory, `scripts/` utilities).

9.0 Implementation Roadmap & Status

- **Phase 1 (MVP - Core Tool): COMPLETE**
    - Login system, Local/Master DB schema, Basic PM dashboard, Sync-to-Inbox model, Spreadsheet Importer, Inbox Processor, Modular project structure.
- **Phase 2 (Reporting & Admin): COMPLETE**
    - KPI Snapshots table and local creation, Basic Reports tab, Admin Control Panel (User Mgmt, Config Editor).
- **Phase 3 (AI Assistant): COMPLETE**
    - AI libraries, Knowledge Base tables & functions, AI Indexer, AI Assistant chatbot UI, AI Feedback mechanism.
- **Phase 4 (Legacy Fields & Dependencies): COMPLETE**
    - Added legacy fields (`nfid`, `clli`, etc.) to `projects` table.
    - Updated UIs (`My Dashboard`, `New Project`) with new fields.
    - Added `project_dependencies` table and UI.
    - Updated Sync & Processor for dependencies.

- **Next Step: Phase 5 (Project Details Page: Milestones & Team):**
  - Add `site_address` (TEXT) field to the `projects` table schema in `database.py`.
  - Add the `project_contacts` table to the schema in `database.py`.
  - Update all create/update UIs (`1_My_Dashboard`, `2_New_Project`, `importer.py`) to include the new `site_address` field.
  - Create `app/pages/11_Project_Details.py`.
  - Add navigation links from `My Dashboard` to the details page.
  - Build UI on the details page with two tabs/editors:
    1. An editor to manage `work_packages` (Milestones) for the project.
    2. An editor to manage `project_contacts` (Team) for the project.
  - Update Sync & Processor to handle `work_packages` and `project_contacts`.
- **Future: Phase 6 (Advanced Reporting & Data Model):**
  - Add `project_start_date` and `project_complete_date` to the `projects` table schema in `database.py`.
  - Update all create/update UIs (`1_My_Dashboard`, `2_New_Project`, `importer.py`) to include these new date fields.
  - Update Sync & Processor to handle the new date fields.
  - Enhance `src/vtrack/reports.py` and `app/pages/8_Reports.py` to add the "Monthly Scorecard" and "Visual Program Tracker" dashboards as defined in section 6.3.
- **Future: Phase 7 (Branding & UI Defaults):**
  - Create `.streamlit/config.toml` file with Verizon color theme.
  - Add Verizon logo image to project and display in `app/sidebar.py`.
  - Implement role-based default values in forms (e.g., pre-fill PM name).

10.0 Packaging & Deployment (Future)

Once development stabilizes, the application will be packaged using PyInstaller into a single executable (`.exe`) for easy distribution within Verizon. This will bundle the Python interpreter, all required libraries (including AI models if feasible, or provide instructions for separate download), and the application code.

1.