**Total Cost Control (TCC) App Handover Document**

**1. Introduction**

**1.1 Purpose**

This document provides a comprehensive handover of the **Total Cost Control (TCC) app** to a professional development firm specializing in **Microsoft Field Service (Dynamics 365)**. It outlines the existing architecture, functionalities, database structure, and roadmap for integrating TCC into Microsoft’s ecosystem.

**1.2 Target Audience**

* **Microsoft Field Service Developers**
* **System Integrators for Dynamics 365**
* **Project Managers overseeing the transition**
* **Cloud & Infrastructure Engineers (Azure, Power Automate)**

**2. Existing TCC System Overview**

**2.1 Technology Stack**

|  |  |
| --- | --- |
| **Component** | **Technology** |
| **Backend** | Flask (Python) |
| **Frontend** | HTML, Jinja, JavaScript (AJAX, Fetch API) |
| **Database** | SQLite (planned migration to PostgreSQL or Dataverse) |
| **API** | RESTful Endpoints |
| **Hosting** | On-premise or cloud (planned Azure migration) |
| **Authentication** | Session-based (Flask-Session) |

**2.2 Main Functionalities**

The TCC app is a **construction project management tool** focused on:

* **Labor & Equipment Management**
* **Activity Codes for Work Orders**
* **Daily Cost & Work Reports**
* **Project Progress Tracking**
* **Exporting Reports (CSV, SQL)**
* **Mobile-Friendly Data Entry**

**3. Database & Data Structure**

**3.1 Database Schema Overview**

TCC currently uses **SQLite**, with planned migration to **Microsoft Dataverse**. Key tables include:

|  |  |
| --- | --- |
| Table | Description |
| **workers** | Stores employee data |
| **equipment** | Tracks construction equipment |
| **projects** | Manages project metadata |
| **activity\_codes** | Defines work types & cost codes |
| **entries\_workers** | Tracks worker hours per project |
| **entries\_equipment** | Logs equipment usage per project |
| **daily\_report\_statuses** | Stores daily work logs |
| **subcontractors** | Manages third-party contractors |
| **materials** | Tracks material usage |
| **tab\_progress** | Stores progress status for UI workflow |

**3.2 Data Export**

**Exports Provided:**

* **SQL Dump:** TCC\_export.sql (Full database structure & data)
* **CSV Files:** All key tables exported to /mnt/data/sqlite\_exports/

**4. Mapping to Microsoft Field Service**

|  |  |
| --- | --- |
| TCC Data | Microsoft Field Service Equivalent |
| **Workers** | **Technicians & Resources** |
| **Equipment** | **Assets in Field Service** |
| **Projects** | **Work Orders & Jobs** |
| **Activity Codes** | **Work Order Types** |
| **Daily Reports** | **Service Tasks & Inspections** |

**4.1 Field Service Integrations Needed**

* **Power Automate** for scheduling workers & jobs
* **Dataverse (CDS)** for database migration
* **Mobile Access** via **Field Service App**
* **Power BI Dashboards** for cost tracking
* **Azure Logic Apps** for automation

**5. Transition Plan & Next Steps**

**5.1 Integration Approach**

The recommended approach is **Hybrid Integration**, allowing TCC to continue running while synchronizing with **Microsoft Field Service**.

|  |  |  |  |
| --- | --- | --- | --- |
| Approach | Description | Pros | Cons |
| **Full Migration** | Move everything to Dataverse & Field Service | Future-proof | High cost & training needed |
| **Hybrid Integration** | Keep TCC, sync workers & projects | Less disruption, fast deployment | Requires ongoing sync maintenance |
| **API-Based** | Convert Flask API to Azure Functions | Scalable & secure | Requires custom API setup |

**5.2 Immediate Action Plan**

1. **Set up a workshop** with Microsoft Field Service integrators.
2. **Define Field Mapping** between TCC and Dataverse.
3. **Migrate workers & equipment first** (low-risk data sync).
4. **Build Power Automate Flows** for automation.
5. **Test data sync in a sandbox environment**.
6. **Deploy in phases to avoid disruptions**.

**6. Deliverables for the New Development Team**

**6.1 Provided Files**

* **TCC Codebase** (Flask, JavaScript, Jinja, SQLAlchemy)
* **Database Dumps (SQL, CSV)**
* **API Reference Document** (Endpoints, Authentication, Data Flows)
* **Field Mapping Guide** (TCC → Field Service)
* **Deployment Instructions** (Current vs. Target Environment)