Milestone 1 — Discovery & Technical Design (Full Automation)

Generated from Milestone.md

Milestone 1 — Discovery & Technical Design (Full Automation)

Objective (Milestone 1)

Deliver a **client-approval-ready automation plan and architecture** for a *fully automated* system that:

1. Connects **Dynamics 365 (ERP/CRM)** → **Power BI** for analysis and dashboards
2. Produces an **ICP scoring + segmentation** output (A/B/Win-Back/Strategic)
3. Sends segmented targets into an **automation layer (custom app + database)**
4. Integrates **Apollo.io** for enrichment + outreach sequencing
5. Uses **AI (ChatGPT)** to generate outbound messaging at scale

---

Deliverable 1 — Review of Current StrikeZone Workflow

Strikezone’s BDaaS workflow (as provided in Strikezone Process v1/v2) runs in phases:

1. **Governance & Engagement Setup**
   - NDA/MSA/DPA/SOW in place
   - Secure data access established
2. **ERP Data Extraction & High-Value Customer Identification**
   - Extract customer, orders, products, payment data (36 months)
   - Identify **top 20% customers by gross margin**
   - Analyze order behavior, product mix, payment behavior
3. **ICP Development & Look-Alike Identification**
   - Reverse-engineer traits of best customers (firmographic + behavioral + financial + operational)
   - Identify look-alikes via external sources
4. **Account Planning & Enrichment**
   - Enrich company + decision-makers (Apollo)
5. **Outreach Execution**
   - Build cadences, execute multi-touch outreach, handle replies

Where time is currently spent manually:
- Extracting/cleaning ERP data
- Repeating the same analysis each engagement
- Enriching accounts/contacts
- Writing personalized outbound messages
- Tracking campaign outcomes across tools

Deliverable 2 — Identification of Automatable Components

Below are the key parts that can be automated end-to-end in a “full automation” build:

A) Dynamics 365 → Power BI (Data & Dashboards)
- Scheduled refresh (daily/weekly)
- Standard dashboards:
  - Top 20% customers by gross margin (36 months)
  - Margin trend, AOV, order frequency
  - Product mix breadth/category penetration
  - Geo + industry distribution

B) ICP Scoring + Segmentation
- Automated score calculation aligned to Strikezone scoring model:
  - Financial Fit (25)
  - Behavioral Fit (25)
  - Operational Fit (20)
  - Firmographic Fit (20)
  - Strategic Fit (10)
- Automated tiers:
  - **A-Tier:** 85–100
  - **B-Tier:** 70–84
  - **C-Tier:** 50–69
  - **Low fit:** 0–49

C) Enrichment (Apollo.io)
- Automated company and contact enrichment via Apollo API
- Deduping + normalization (company name, domains, titles)
- Persistent storage to avoid re-enrichment costs

D) Messaging (AI)
- Generate message sets per segment (Win-Back vs Strategic Prospect)
- Personalize using enrichment + ERP insights
- Create variants for A/B testing

E) Outreach Execution + Tracking
- Automatically push approved contacts/messages into Apollo sequences
- Capture outcomes (opens, clicks, replies, meetings booked)
- Feed performance metrics back into reporting

Deliverable 3 — High-Level System Architecture / Flow Diagram

Architecture (3 layers)

1. **Data/Analytics Layer**
   - Dynamics 365 → Power BI Dataset + Dashboards

2. **Automation & AI Layer (Core of “Full Automation”)**
   - Custom Web App + Backend API (React + Node.js)
   - Database (PostgreSQL)
   - Integrations:
     - Apollo API (enrichment + sequencing)
     - AI API (ChatGPT)

3. **Execution Layer**
   - Apollo.io sequences/cadences
   - Response tracking + meeting outcomes

Custom App (what it is, and why)

The **Custom App** is the “automation brain” between Power BI and Apollo. Power BI is excellent for analytics, but it is not designed to:
- call enrichment APIs (Apollo)
- generate/store AI messages
- manage approvals/audit logs
- orchestrate workflow states (Ready → Enriched → Message Drafted → Approved → Sent)

So we add a lightweight application that operationalizes the workflow.

Recommended stack (simple + scalable):
- **Frontend:** React (or Next.js) for a clean internal UI
- **Backend:** Node.js (Express or NestJS) for APIs and orchestration
- **Database:** PostgreSQL for accounts, contacts, ICP scores, messages, campaign status, logs

Why React + Node.js is a good fit here:
- Fast to build and iterate (important for early phases and changing sales workflows)
- Great ecosystem for APIs + integrations (Apollo/OpenAI) and background jobs
- Easy to hire for, and maintain long-term
- Works well on Azure (App Service) and supports enterprise auth (Microsoft Entra ID)

Core modules in the Custom App:
1. **Import/Sync Module**: ingest segmented accounts from Power BI (CSV export or API)
2. **Enrichment Module**: call Apollo API to enrich companies + contacts
3. **AI Messaging Module**: call ChatGPT to generate emails/LinkedIn/call scripts
4. **Approval Workflow**: human review, edit, approve, and audit trail
5. **Campaign Push Module**: create/update Apollo sequences and enroll contacts
6. **Tracking Module**: pull outcomes from Apollo (reply/meeting) and update status

Data Flow Diagram (end-to-end)

```text
(1) Dynamics 365 ──► (2) Power BI ──► (3) Segmented Target Export
      │              │               │
      │ (scheduled)  │ (dashboards)  │ (A/B/WinBack/Strategic)
      ▼              ▼               ▼
  ERP Tables      ICP metrics &     Automation Layer (Custom App)
  (Accounts,      scoring inputs    ├─► Apollo Enrichment
  Orders, etc.)                     ┌─┴─► AI Messaging Generation
                                    │ └─► Approval + Audit Log
                                    ▼
                                  Apollo Sequences
                                    │
                                    ▼
                                  Outreach Results
                                    │
                                    ▼
                                  Performance Reporting
                                  (Power BI optional)
```

```

---

## Deliverable 4 — Technical Approach & Assumptions

### Technical Approach (recommended for full automation)

**Step 1 — Connect & Model ERP data in Power BI**
- Use Power BI native connector / Dataverse connector (depending on D365 module)
- Build a clean model (Accounts, Orders, Order Lines, Products, Payments)
- Create measures for gross margin, AOV, frequency, category breadth

**Step 2 — Define outputs needed from Power BI**
- Export (or API feed) of:
  - account_id, name, domain (if available), location
  - revenue/margin metrics
  - ICP score + tier
  - key traits used for personalization

**Step 3 — Automation layer (Custom App)**
- Backend service to:
  - ingest Power BI exports
  - call Apollo enrichment
  - call AI to generate messaging
  - push records into Apollo sequences
  - store everything in DB (traceability)

**Custom App implementation detail (React + Node.js):**

- **React UI (internal tool)** screens:
  - Accounts list (tier, ICP score, enrichment status, owner)
  - Account detail (ERP insights + enriched firmographics + contacts)
  - Message drafts (email/LinkedIn/call script) with approve/edit
  - Sequence enrollment view (which cadence, which contacts)
  - Reporting view (basic funnel + handoff to Power BI dashboards)

- **Node.js API** responsibilities:
  - Secure auth (Azure AD / Microsoft Entra ID recommended)
  - Data model CRUD (accounts, contacts, messages, sequences, events)
  - Background jobs/queues for enrichment + AI generation (so UI stays fast)
  - Rate limiting + retries for Apollo/OpenAI APIs
  - Webhooks/endpoints to receive delivery/reply events (where available)

- **Database tables (high-level):**
  - accounts, account_metrics, icp_scores
  - contacts, contact_roles
  - enrichment_snapshots (Apollo results)
  - message_templates, message_drafts, approvals
  - sequences, enrollments
  - events (sent/open/click/reply/meeting)

**Step 4 — Close the loop**
- Pull Apollo outcomes (reply, meeting booked)
```

- Update status in DB
- Optional: push performance KPIs back to Power BI

Deliverable 5 — Suggestions on Tools

Required (recommended)
- **Power BI**: analysis + dashboards, scheduled refresh
- **Dynamics 365 connector**: native connector/Dataverse
- **Apollo.io**: enrichment + sequences (API plan recommended)
- **AI**: **OpenAI ChatGPT API**
- **Database**: PostgreSQL for storing enriched data + messages + status
- **Custom App stack**: React (or Next.js) + Node.js (Express/NestJS)
- **Hosting**: Azure App Service (aligns with Microsoft ecosystem)

Recommended Azure add-ons (for reliability)
- **Azure Key Vault**: store API keys/secrets (Apollo/OpenAI)
- **Azure Storage / Blob**: store exports, attachments, logs if needed
- **Azure Functions or WebJobs**: scheduled/background processing (optional)
- **Application Insights**: monitoring + error tracking

File: Milestone.pdf