Generic Prospecting Platform Transition Plan

Executive Summary

This document proposes evolving FranchiseAI from a single-vertical prospecting tool into a **configurable, generic prospecting platform** where the business domain is defined by seed data and configuration rather than hardcoded logic. The platform would support multiple prospecting use cases -- franchise buyer prospecting, catering client lead generation, hotel conference venue sales, and any future vertical -- through a single codebase driven by **Prospect Profile Configurations**.

This approach is **recommended over a vertical-specific pivot** because it maximizes the value of the existing infrastructure while enabling rapid deployment to new markets with no code changes.

Strategic Direction: Two Approaches Compared

Approach A: Vertical-Specific Pivot (Franchise Buyers Only)

Rewrite domain-specific code to target franchise buyers instead of catering clients. The application remains a single-purpose tool.

Pros:

- Simpler implementation -- just swap domain logic
- Faster initial delivery
- Focused feature set for one audience

Cons:

- Every new vertical requires another code rewrite
- No reuse across business lines
- Locks the platform into one market
- Duplicate effort if the business expands

Approach B: Generic Configurable Platform (RECOMMENDED)

Extract domain logic into configuration files. The platform becomes a prospecting engine that adapts to any vertical through seed data.

Pros:

- New verticals deployed via JSON config, no code changes
- Single codebase to maintain
- Shared improvements benefit all verticals
- Platform becomes a sellable product, not just an internal tool
- Supports multi-tenant / multi-vertical operation

Cons:

- Larger upfront investment in abstraction layer
- Configuration schema requires careful design
- Testing surface increases (must validate each profile)

Side-by-Side Comparison

Factor	Vertical-Specific	Generic Platform
Time to first vertical	Faster	Moderate
Time to second vertical	Same as first (full rewrite)	Config file only
Time to third vertical	Same as first (full rewrite)	Config file only
Codebase maintenance	One codebase per vertical	One shared codebase
Feature parity across verticals	Manual sync required	Automatic
AI prompt management	Hardcoded per vertical	Templated, config-driven
Scoring rules	Hardcoded per vertical	Loaded from profile config
UI labels/terminology	Hardcoded per vertical	Driven by profile terminology map
Data source selection	Hardcoded per vertical	Profile declares which sources to use
Pipeline stages	Hardcoded per vertical	Defined in profile config
SMS/outreach templates	Per-vertical templates	Shared engine, per-profile templates
Scalability	Linear cost per vertical	Near-zero marginal cost
Market positioning	Internal tool	Sellable SaaS platform

Recommendation

Approach B (Generic Platform) is the recommended path. The existing codebase is already ~60% generic -- the infrastructure layer (search orchestration, geocoding, API clients, scoring engine architecture, prospect persistence, pipeline tracking) is domain-agnostic. The domain-specific coupling is concentrated in four areas that can be extracted into configuration with moderate effort.

Current Codebase: Domain Coupling Analysis

An analysis of the existing codebase reveals where domain-specific logic is hardcoded versus what is already generic.

Coupling Assessment

Component	Coupling Level	Details
Data source APIs (Google, OSM, BBB)	LOOSE	Domain-agnostic search and data retrieval

Component	Coupling Level	Details
Scoring rule engine	LOOSE	Supports enable/disable, point adjustment at runtime
Scoring calculations	TIGHT	`calculateCateringPotential()` with catering-specific weights in `BusinessInfo.cpp:27`
AI prompts	TIGHT	"corporate catering" hardcoded in `OpenAIEngine.cpp:289`, `GeminiEngine.cpp`, `AIEngine.cpp`
Search criteria	MEDIUM	Field named `minCateringScore` in `SearchResult.h:105`
BusinessType enum	LOOSE	Generic types (Corporate, Hotel, Warehouse, etc.) in `BusinessInfo.h:14`
Prospect pipeline	LOOSE	Simple string status field, no domain assumptions
UI text	TIGHT	14+ hardcoded "catering" / "Vocelli" references in `FranchiseApp.cpp`
Configuration files	LOOSE	Infrastructure-only (`app_config.json`), no domain config
Franchisee model	MEDIUM	`franchiseName = "Vocelli Pizza"` default in `Franchisee.h:72`

What Stays As-Is (Already Generic)

- Search orchestration (`AISearchService`)
- Multi-source data aggregation pipeline
- Geocoding service (Google + Nominatim fallback)
- OpenStreetMap Overpass API queries
- Google Places API integration
- BBB API integration
- Demographics API integration
- Thread pool and parallel processing
- API response caching (24-hour)
- HTTP client infrastructure (CURL)
- ApiLogicServer REST client
- Authentication service
- Audit logging
- Map visualization (Leaflet.js)
- Wt framework UI widgets

What Needs to Change (4 Concentrated Areas)

1. Scoring Method -- Replace `calculateCateringPotential()` in `BusinessInfo.cpp` with a generic `calculateProspectScore()` that reads rules from the active profile config.

2. AI Prompts -- Extract hardcoded prompt strings from `OpenAIEngine.cpp`, `GeminiEngine.cpp`, and `AIEngine.cpp` into the profile configuration's `aiPrompts` section.

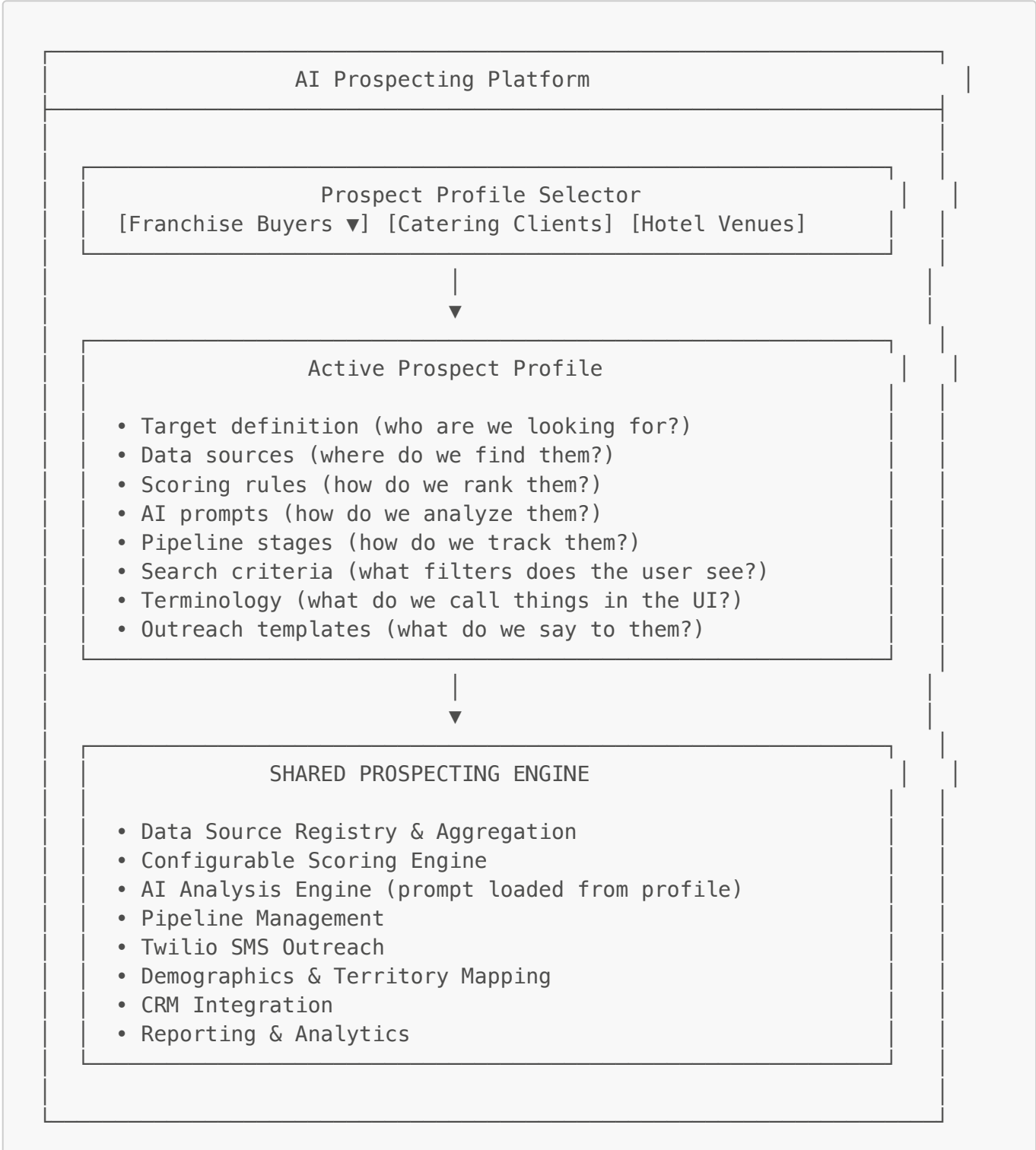
3. Field Names -- Rename `minCateringScore` to `minProspectScore`, `cateringPotentialScore` to `prospectScore` across models.

4. UI Labels -- Replace ~14 hardcoded strings in `FranchiseApp.cpp` and `Sidebar.h` with lookups from the profile's `terminology` map.

Generic Platform Architecture

Core Concept: The Prospect Profile

A **Prospect Profile** is a JSON configuration that defines everything domain-specific about a prospecting use case. The platform loads a profile and adapts its behavior accordingly.



Prospect Profile Configuration Schema

```
{
  "profileId": "franchise-buyer-prospecting",
  "profileName": "Franchise Buyer Prospecting",
  "version": "1.0",
  "description": "Find potential franchise investors and buyers",

  "target": {
    "type": "INDIVIDUAL",
    "label": "Candidate",
    "labelPlural": "Candidates",
    "icon": "user-tie"
  },

  "dataSources": {
    "primary": ["linkedin", "demographics"],
    "secondary": ["google_places", "franchise_brokers"],
    "optional": ["bbb", "osm"],
    "sourceConfig": {
      "linkedin": {
        "searchType": "people",
        "defaultFilters": {
          "titles": ["Executive", "Director", "VP", "Owner"],
          "minExperience": 10
        }
      },
      "google_places": {
        "searchType": "business",
        "categories": ["franchise_consultant", "business_broker"]
      }
    }
  },

  "searchCriteria": {
    "fields": [
      { "id": "location", "type": "geo", "label": "Location", "required":
true },
      { "id": "radius", "type": "range", "label": "Search Radius",
"default": 25, "unit": "miles" },
      { "id": "titles", "type": "multi_select", "label": "Job Titles",
        "options": ["Executive", "Director", "VP", "Owner", "Manager"] },
      { "id": "industries", "type": "multi_select", "label": "Industries",
        "options": ["Food Service", "Retail", "Hospitality", "Management",
"Finance"] },
      { "id": "minExperience", "type": "number", "label": "Min Years
Experience", "default": 5 },
      { "id": "netWorthTier", "type": "select", "label": "Est. Net Worth",
        "options": ["$100K+", "$250K+", "$500K+", "$1M+"] },
      { "id": "recentCareerChange", "type": "boolean", "label": "Recent
Career Transition" },
```

```
      { "id": "hasManagementExp", "type": "boolean", "label": "Management
Experience" },
      { "id": "isBusinessOwner", "type": "boolean", "label": "Previous
Business Owner" }
    ]
  },

  "scoringRules": [
    { "id": "mgmt_exp", "name": "Management Experience", "points": 15,
"type": "bonus", "enabled": true },
    { "id": "biz_ownership", "name": "Previous Business Owner", "points":
20, "type": "bonus", "enabled": true },
    { "id": "food_retail_bg", "name": "Food/Retail Background", "points":
12, "type": "bonus", "enabled": true },
    { "id": "career_transition", "name": "Recent Career Change", "points":
15, "type": "bonus", "enabled": true },
    { "id": "high_net_worth", "name": "High Net Worth Tier", "points": 18,
"type": "bonus", "enabled": true },
    { "id": "linkedin_complete", "name": "Complete LinkedIn Profile",
"points": 8, "type": "bonus", "enabled": true },
    { "id": "verified_contact", "name": "Verified Contact Info", "points":
10, "type": "bonus", "enabled": true },
    { "id": "no_linkedin", "name": "Missing LinkedIn Profile", "points":
-15, "type": "penalty", "enabled": true },
    { "id": "no_contact", "name": "No Direct Contact Method", "points":
-10, "type": "penalty", "enabled": true },
    { "id": "unemployed_risk", "name": "Currently Unemployed", "points":
-5, "type": "penalty", "enabled": true }
  ],

  "scoringWeights": {
    "categories": [
      { "id": "financial_readiness", "label": "Financial Readiness",
"weight": 0.30 },
      { "id": "professional_fit", "label": "Professional Fit", "weight":
0.25 },
      { "id": "entrepreneurial", "label": "Entrepreneurial Indicators",
"weight": 0.20 },
      { "id": "geographic_fit", "label": "Geographic Fit", "weight": 0.15
},
      { "id": "timing_signals", "label": "Timing Signals", "weight": 0.10
}
    ]
  },

  "aiPrompts": {
    "systemPrompt": "You are a franchise development analyst specializing
in identifying and qualifying potential franchise investors. Analyze
candidates for their potential to successfully invest in and operate a
franchise business.",
    "analysisPrompt": "Analyze the following candidate profile for
franchise investment potential.\n\nCANDIDATE
PROFILE:\n{profile_json}\n\nFRANCHISE REQUIREMENTS:\n- Initial investment:
$150,000 - $400,000\n- Liquid capital required: $100,000 minimum\n- Net
```

```
worth required: $300,000 minimum\n- Time commitment: Full-time owner-
operator preferred\n\nProvide analysis in JSON format with: prospectScore
(0-100), fitAssessment, keyStrengths[], potentialConcerns[],
recommendedApproach, timingAssessment.",
    "marketPrompt": "Analyze this geographic area for franchise expansion
opportunity. Consider population growth, household income, competitor
density, and commercial real estate availability.",
    "outreachPrompt": "Generate a personalized outreach message for this
franchise buyer candidate based on their profile. Tone: professional,
warm, entrepreneurial.",
    "smsPersonalizationPrompt": "Personalize this SMS template for the
candidate based on their profile. Keep under 160 characters when possible.
Never include financial guarantees."
  },

  "pipeline": {
    "stages": [
      { "id": "lead", "label": "Lead", "order": 1 },
      { "id": "qualified", "label": "Qualified", "order": 2 },
      { "id": "discovery_day", "label": "Discovery Day", "order": 3 },
      { "id": "fdd_sent", "label": "FDD Sent", "order": 4 },
      { "id": "applied", "label": "Applied", "order": 5 },
      { "id": "approved", "label": "Approved", "order": 6 },
      { "id": "funded", "label": "Funded", "order": 7 }
    ],
    "defaultStage": "lead",
    "convertedStage": "funded",
    "staleAfterDays": 30
  },

  "terminology": {
    "prospect": "Candidate",
    "prospectPlural": "Candidates",
    "score": "Investor Score",
    "search": "Candidate Search",
    "pipeline": "Franchise Pipeline",
    "convert": "Fund",
    "converted": "Funded",
    "dashboard": "Candidate Dashboard",
    "addAction": "Add to Pipeline",
    "analyzeAction": "Analyze Fit"
  },

  "outreach": {
    "smsEnabled": true,
    "emailEnabled": true,
    "inMailEnabled": true,
    "defaultChannel": "sms"
  }
}
```

Example Profile: Catering Client Prospecting

```
{
  "profileId": "franchise-catering-clients",
  "profileName": "Catering Client Prospecting",
  "version": "1.0",
  "description": "Find businesses with high catering potential for
franchise locations",

  "target": {
    "type": "BUSINESS",
    "label": "Prospect",
    "labelPlural": "Prospects",
    "icon": "building"
  },

  "dataSources": {
    "primary": ["google_places", "osm"],
    "secondary": ["bbb", "demographics"],
    "optional": ["linkedin"],
    "sourceConfig": {
      "google_places": {
        "searchType": "business",
        "categories": ["corporate_office", "hotel", "conference_center",
"medical_facility"]
      },
      "osm": {
        "searchType": "poi",
        "tags": ["office", "hotel", "hospital", "university"]
      }
    }
  },

  "searchCriteria": {
    "fields": [
      { "id": "location", "type": "geo", "label": "Location", "required":
true },
      { "id": "radius", "type": "range", "label": "Search Radius",
"default": 10, "unit": "miles" },
      { "id": "businessTypes", "type": "multi_select", "label": "Business
Types",
        "options": ["Corporate Office", "Hotel", "Conference Center",
"Medical Facility",
                    "Educational Institution", "Government Office",
"Manufacturing", "Warehouse"] },
      { "id": "minEmployees", "type": "number", "label": "Min Employees",
"default": 50 },
      { "id": "maxEmployees", "type": "number", "label": "Max Employees"
},
      { "id": "minProspectScore", "type": "number", "label": "Min Catering
Score", "default": 0 }
    ]
  },
```

```
  "scoringRules": [
    { "id": "high_employees", "name": "500+ Employees", "points": 25,
"type": "bonus", "enabled": true },
    { "id": "conf_room", "name": "Has Conference Room", "points": 15,
"type": "bonus", "enabled": true },
    { "id": "event_space", "name": "Has Event Space", "points": 15,
"type": "bonus", "enabled": true },
    { "id": "bbb_accredited", "name": "BBB Accredited", "points": 10,
"type": "bonus", "enabled": true },
    { "id": "high_rating", "name": "4.0+ Google Rating", "points": 8,
"type": "bonus", "enabled": true },
    { "id": "verified", "name": "Verified Business", "points": 5, "type":
"bonus", "enabled": true },
    { "id": "no_address", "name": "Missing Address", "points": -10,
"type": "penalty", "enabled": true },
    { "id": "no_phone", "name": "Missing Phone", "points": -5, "type":
"penalty", "enabled": true },
    { "id": "closed", "name": "Business Closed", "points": -25, "type":
"penalty", "enabled": true }
  ],

  "scoringWeights": {
    "categories": [
      { "id": "employee_count", "label": "Employee Count", "weight": 0.30
},
      { "id": "business_type", "label": "Business Type Fit", "weight":
0.25 },
      { "id": "facilities", "label": "Facility Features", "weight": 0.20
},
      { "id": "proximity", "label": "Proximity", "weight": 0.15 },
      { "id": "reputation", "label": "Reputation & Ratings", "weight":
0.10 }
    ]
  },

  "aiPrompts": {
    "systemPrompt": "You are an expert business analyst specializing in
corporate catering market analysis. Analyze businesses for their potential
as catering clients.",
    "analysisPrompt": "Analyze the following business for corporate
catering potential.\n\nBUSINESS PROFILE:\n{profile_json}\n\nConsider
employee count, business type, meeting frequency, event hosting
capability, and proximity to the franchise location.\n\nProvide analysis
in JSON format with: prospectScore (0-100), fitAssessment, keyStrengths[],
potentialConcerns[], recommendedApproach.",
    "marketPrompt": "Analyze this geographic area for corporate catering
business opportunities in the food service industry.",
    "outreachPrompt": "Generate a personalized catering pitch for this
business based on their profile.",
    "smsPersonalizationPrompt": "Personalize this SMS template for the
business prospect. Keep under 160 characters. Focus on catering benefits."
  },

  "pipeline": {
```

```
    "stages": [
      { "id": "new", "label": "New", "order": 1 },
      { "id": "contacted", "label": "Contacted", "order": 2 },
      { "id": "quoted", "label": "Quoted", "order": 3 },
      { "id": "tasting", "label": "Tasting Scheduled", "order": 4 },
      { "id": "contracted", "label": "Contracted", "order": 5 }
    ],
    "defaultStage": "new",
    "convertedStage": "contracted",
    "staleAfterDays": 14
  },

  "terminology": {
    "prospect": "Prospect",
    "prospectPlural": "Prospects",
    "score": "Catering Score",
    "search": "Prospect Search",
    "pipeline": "Sales Pipeline",
    "convert": "Contract",
    "converted": "Contracted",
    "dashboard": "Prospect Dashboard",
    "addAction": "Add to Prospects",
    "analyzeAction": "Analyze Potential"
  },

  "outreach": {
    "smsEnabled": true,
    "emailEnabled": true,
    "inMailEnabled": false,
    "defaultChannel": "email"
  }
}
```

Example Profile: Hotel Conference Venue Clients

```
{
  "profileId": "hotel-conference-venues",
  "profileName": "Conference Venue Client Prospecting",
  "version": "1.0",
  "description": "Find businesses and organizations that need conference
and event venue services",

  "target": {
    "type": "BUSINESS",
    "label": "Lead",
    "labelPlural": "Leads",
    "icon": "calendar-event"
  },

  "dataSources": {
    "primary": ["google_places", "osm"],
```

```
    "secondary": ["demographics", "eventbrite"],
    "optional": ["linkedin", "bbb"],
    "sourceConfig": {
      "google_places": {
        "searchType": "business",
        "categories": ["corporate_office", "tech_company",
"financial_services",
                       "nonprofit", "association", "training_center"]
      },
      "linkedin": {
        "searchType": "companies",
        "defaultFilters": {
          "minEmployees": 100,
          "industries": ["Technology", "Finance", "Consulting", "Pharma"]
        }
      }
    }
  },

  "searchCriteria": {
    "fields": [
      { "id": "location", "type": "geo", "label": "Location", "required":
true },
      { "id": "radius", "type": "range", "label": "Search Radius",
"default": 30, "unit": "miles" },
      { "id": "businessTypes", "type": "multi_select", "label":
"Organization Types",
        "options": ["Corporate HQ", "Tech Company", "Financial Services",
"Consulting Firm",
                    "Pharmaceutical", "Nonprofit / Association",
"Government Agency", "University"] },
      { "id": "minEmployees", "type": "number", "label": "Min Employees",
"default": 100 },
      { "id": "hasEventHistory", "type": "boolean", "label": "Known Event
History" },
      { "id": "annualBudget", "type": "select", "label": "Est. Event
Budget",
        "options": ["$10K+", "$50K+", "$100K+", "$500K+"] }
    ]
  },

  "scoringRules": [
    { "id": "large_org", "name": "500+ Employees", "points": 20, "type":
"bonus", "enabled": true },
    { "id": "event_history", "name": "Known Event History", "points": 25,
"type": "bonus", "enabled": true },
    { "id": "corporate_acct", "name": "Corporate Account Program",
"points": 15, "type": "bonus", "enabled": true },
    { "id": "high_revenue", "name": "High Annual Revenue", "points": 18,
"type": "bonus", "enabled": true },
    { "id": "recurring_events", "name": "Recurring Event Pattern",
"points": 20, "type": "bonus", "enabled": true },
    { "id": "proximity", "name": "Within 10 Miles", "points": 10, "type":
"bonus", "enabled": true },
```

```
    { "id": "no_contact", "name": "No Event Planner Contact", "points":
-10, "type": "penalty", "enabled": true },
    { "id": "small_org", "name": "Under 25 Employees", "points": -15,
"type": "penalty", "enabled": true }
  ],

  "scoringWeights": {
    "categories": [
      { "id": "event_potential", "label": "Event Hosting Potential",
"weight": 0.30 },
      { "id": "org_size", "label": "Organization Size & Budget", "weight":
0.25 },
      { "id": "booking_frequency", "label": "Booking Frequency Signals",
"weight": 0.20 },
      { "id": "proximity", "label": "Geographic Proximity", "weight": 0.15
},
      { "id": "relationship", "label": "Relationship Potential", "weight":
0.10 }
    ]
  },

  "aiPrompts": {
    "systemPrompt": "You are a hospitality industry analyst specializing
in conference and event venue sales. Analyze organizations for their
potential as conference venue clients.",
    "analysisPrompt": "Analyze the following organization for conference
venue client potential.\n\nORGANIZATION
PROFILE:\n{profile_json}\n\nConsider organization size, industry, event
hosting patterns, budget indicators, and proximity to our
venue.\n\nProvide analysis in JSON format with: prospectScore (0-100),
fitAssessment, keyStrengths[], potentialConcerns[], recommendedApproach.",
    "marketPrompt": "Analyze this geographic area for conference and event
venue demand. Consider corporate density, industry mix, and competing
venues.",
    "outreachPrompt": "Generate a personalized venue pitch for this
organization based on their profile and event needs.",
    "smsPersonalizationPrompt": "Personalize this SMS for the venue
prospect. Keep under 160 characters. Focus on venue capabilities."
  },

  "pipeline": {
    "stages": [
      { "id": "identified", "label": "Identified", "order": 1 },
      { "id": "contacted", "label": "Contacted", "order": 2 },
      { "id": "site_visit", "label": "Site Visit", "order": 3 },
      { "id": "proposal", "label": "Proposal Sent", "order": 4 },
      { "id": "contracted", "label": "Contracted", "order": 5 }
    ],
    "defaultStage": "identified",
    "convertedStage": "contracted",
    "staleAfterDays": 21
  },

  "terminology": {
```

```
    "prospect": "Lead",
    "prospectPlural": "Leads",
    "score": "Venue Fit Score",
    "search": "Lead Search",
    "pipeline": "Venue Sales Pipeline",
    "convert": "Contract",
    "converted": "Contracted",
    "dashboard": "Venue Sales Dashboard",
    "addAction": "Add to Leads",
    "analyzeAction": "Analyze Fit"
  },

  "outreach": {
    "smsEnabled": true,
    "emailEnabled": true,
    "inMailEnabled": false,
    "defaultChannel": "email"
  }
}
```

Platform Implementation Architecture

Data Source Registry

A plugin-style registry that allows the platform to activate data sources based on the active profile.

```
// src/services/DataSourceRegistry.h
class DataSourceRegistry {
public:
    // Register available data sources at startup
    void registerSource(const std::string& id,
std::shared_ptr<IDataSourceAPI> source);

    // Load sources for the active prospect profile
    std::vector<std::shared_ptr<IDataSourceAPI>> getSourcesForProfile(
        const ProspectProfile& profile);

    // Get a specific source
    std::shared_ptr<IDataSourceAPI> getSource(const std::string& id);

    // Check availability
    bool isSourceAvailable(const std::string& id);
    bool isSourceConfigured(const std::string& id);

private:
    std::map<std::string, std::shared_ptr<IDataSourceAPI>> sources_;
};

// Common interface for all data sources
class IDataSourceAPI {
```

```
public:
    virtual ~IDataSourceAPI() = default;
    virtual std::string getId() const = 0;
    virtual std::string getName() const = 0;
    virtual bool isConfigured() const = 0;
    virtual bool requiresApiKey() const = 0;

    virtual std::vector<ProspectResult> search(
        const SearchQuery& query,
        const json& sourceConfig) = 0;

    virtual ProspectResult enrich(
        const ProspectResult& basic,
        const json& sourceConfig) = 0;
};
```

Source Registration at Startup:

```
void FranchiseApp::initializeDataSources() {
    auto& registry = DataSourceRegistry::instance();

    // Always available (free)
    registry.registerSource("osm", std::make_shared<OpenStreetMapAPI>());

    // Available when API keys are configured
    if (config.hasGoogleApiKey())
        registry.registerSource("google_places",
std::make_shared<GooglePlacesAPI>(config));

    if (config.hasBBBApiKey())
        registry.registerSource("bbb", std::make_shared<BBBAPI>(config));

    if (config.hasDemographicsKey())
        registry.registerSource("demographics",
std::make_shared<DemographicsAPI>(config));

    // New sources for franchise buyer prospecting
    if (config.hasLinkedInCredentials())
        registry.registerSource("linkedin", std::make_shared<LinkedInAPI>
(config));

    if (config.hasFranchiseBrokerKey())
        registry.registerSource("franchise_brokers",
std::make_shared<FranchiseBrokerAPI>(config));

    if (config.hasEventbriteKey())
        registry.registerSource("eventbrite",
std::make_shared<EventbriteAPI>(config));
}
```

Configurable Scoring Engine

Replace the hardcoded `calculateCateringPotential()` with a profile-driven scoring method.

```
// src/services/ScoringEngine.h (refactored)
class ScoringEngine {
public:
    // Load scoring rules from the active prospect profile
    void loadProfile(const ProspectProfile& profile);

    // Generic scoring -- applies rules from the loaded profile
    int calculateProspectScore(const ProspectResult& prospect);

    // Weighted category scoring
    double calculateWeightedScore(const ProspectResult& prospect,
                                  const std::vector<ScoringCategory>&
categories);

    // Rule management (runtime adjustments)
    void setRuleEnabled(const std::string& ruleId, bool enabled);
    void setRulePoints(const std::string& ruleId, int points);
    void resetToProfileDefaults();

    // Get current rules for UI display
    std::vector<ScoringRule> getRules() const;

private:
    std::vector<ScoringRule> rules_;
    std::vector<ScoringCategory> categories_;
    std::string activeProfileId_;
};
```

Templated AI Prompts

```
// src/services/AIPromptManager.h
class AIPromptManager {
public:
    void loadProfile(const ProspectProfile& profile);

    // Get prompts with variable substitution
    std::string getSystemPrompt() const;
    std::string getAnalysisPrompt(const ProspectResult& prospect) const;
    std::string getMarketPrompt(const GeoLocation& location) const;
    std::string getOutreachPrompt(const ProspectResult& prospect) const;
    std::string getSMSPersonalizationPrompt(const std::string&
templateText,
                                            const ProspectResult&
prospect) const;

private:
```

```
    ProspectProfile profile_;

    // Replace {variable} placeholders with actual values
    std::string renderTemplate(const std::string& tmpl,
                               const std::map<std::string, std::string>&
vars) const;
};
```

Dynamic UI Labels

```
// src/services/TerminologyManager.h
class TerminologyManager {
public:
    void loadProfile(const ProspectProfile& profile);

    // UI label lookups
    Wt::WString prospect() const;        // "Candidate" or "Prospect" or
"Lead"
    Wt::WString prospectPlural() const;   // "Candidates" or "Prospects"
or "Leads"
    Wt::WString score() const;            // "Investor Score" or "Catering
Score"
    Wt::WString search() const;           // "Candidate Search" or
"Prospect Search"
    Wt::WString pipeline() const;         // "Franchise Pipeline" or
"Sales Pipeline"
    Wt::WString addAction() const;        // "Add to Pipeline" or "Add to
Prospects"
    Wt::WString dashboard() const;        // "Candidate Dashboard" or
"Prospect Dashboard"

    // Generic lookup
    Wt::WString get(const std::string& key) const;

private:
    std::map<std::string, std::string> terms_;
};
```

Profile Loader and Manager

```
// src/services/ProspectProfileManager.h
class ProspectProfileManager {
public:
    // Load all available profiles from config directory
    void loadProfiles(const std::string& configDir);

    // Get available profiles
    std::vector<ProspectProfileSummary> getAvailableProfiles() const;
```

```
    // Activate a profile (reconfigures entire platform)
    void activateProfile(const std::string& profileId);

    // Get active profile
    const ProspectProfile& getActiveProfile() const;

    // Apply profile to all subsystems
    void applyToScoringEngine(ScoringEngine& engine);
    void applyToAIPromptManager(AIPromptManager& prompts);
    void applyToTerminology(TerminologyManager& terms);
    void applyToSearchCriteria(SearchCriteriaBuilder& builder);
    void applyToPipeline(PipelineManager& pipeline);

private:
    std::map<std::string, ProspectProfile> profiles_;
    std::string activeProfileId_;
};
```

Generic Data Model

Replace `BusinessInfo` / `CandidateProfile` with a unified `ProspectRecord` that handles both business and individual targets.

```
// src/models/ProspectRecord.h
struct ProspectRecord {
    // Universal fields
    std::string id;
    std::string profileId;              // Which prospect profile this
belongs to
    ProspectTargetType targetType;       // BUSINESS or INDIVIDUAL

    // Identity (used by both types)
    std::string name;                    // Business name or person name
    std::string description;
    ContactInfo contact;                 // phone, email, website
    Address address;
    GeoLocation location;

    // Business-specific fields (populated when targetType == BUSINESS)
    std::string businessCategory;
    int employeeCount;
    double annualRevenue;
    double googleRating;
    int googleReviewCount;
    bool bbbAccredited;
    bool hasConferenceRoom;
    bool hasEventSpace;

    // Individual-specific fields (populated when targetType ==
INDIVIDUAL)
    std::string currentTitle;
```

```
    std::string currentEmployer;
    int yearsExperience;
    std::string linkedInUrl;
    std::vector<std::string> industryBackground;
    bool hasManagementExperience;
    bool hasBusinessOwnership;
    std::string netWorthTier;
    std::string educationLevel;

    // Universal scoring and analysis
    int prospectScore;                    // 0-100 (generic, not "catering
score")
    double relevanceScore;
    double aiConfidenceScore;
    std::string aiSummary;
    std::vector<std::string> keyStrengths;
    std::vector<std::string> concerns;
    std::vector<std::string> recommendedActions;
    std::string matchReason;

    // Pipeline tracking
    std::string pipelineStage;
    std::string assignedTo;
    std::string nextAction;
    std::string notes;
    bool isContacted;
    bool isConverted;
    bool doNotContact;

    // Metadata
    DataSource source;
    std::string createdAt;
    std::string updatedAt;
    bool isActive;

    // Extended attributes (profile-specific fields stored as key-value)
    std::map<std::string, std::string> extendedAttributes;
};
```

Database Schema (Generic)

```
-- Replaces both saved_prospects and candidate_profiles with a unified
table
CREATE TABLE prospect_records (
    id UUID PRIMARY KEY,
    profile_id VARCHAR(100) NOT NULL,          -- Which prospect profile
config
    store_location_id UUID REFERENCES store_locations(id),
    target_type VARCHAR(20) NOT NULL,          -- BUSINESS or INDIVIDUAL

    -- Identity
```

```
    name VARCHAR(300) NOT NULL,
    description TEXT,
    phone VARCHAR(20),
    email VARCHAR(255),
    website VARCHAR(500),
    linkedin_url VARCHAR(500),
    street VARCHAR(300),
    city VARCHAR(100),
    state VARCHAR(50),
    zip_code VARCHAR(20),
    latitude DOUBLE PRECISION,
    longitude DOUBLE PRECISION,

    -- Business fields
    business_category VARCHAR(100),
    employee_count INTEGER,
    annual_revenue DOUBLE PRECISION,
    google_rating DOUBLE PRECISION,
    google_review_count INTEGER,
    bbb_accredited BOOLEAN,
    has_conference_room BOOLEAN,
    has_event_space BOOLEAN,

    -- Individual fields
    current_title VARCHAR(200),
    current_employer VARCHAR(200),
    years_experience INTEGER,
    industry_background JSONB,
    has_management_experience BOOLEAN,
    has_business_ownership BOOLEAN,
    net_worth_tier VARCHAR(50),
    education_level VARCHAR(100),

    -- Scoring
    prospect_score INTEGER DEFAULT 0,
    relevance_score DOUBLE PRECISION,
    ai_confidence_score DOUBLE PRECISION,
    ai_summary TEXT,
    key_strengths JSONB,
    concerns JSONB,
    recommended_actions JSONB,
    match_reason TEXT,

    -- Pipeline
    pipeline_stage VARCHAR(50),
    assigned_to VARCHAR(100),
    next_action TEXT,
    next_action_date DATE,
    notes TEXT,
    is_contacted BOOLEAN DEFAULT FALSE,
    is_converted BOOLEAN DEFAULT FALSE,
    do_not_contact BOOLEAN DEFAULT FALSE,

    -- Metadata
```

```
    data_source VARCHAR(50),
    distance_miles DOUBLE PRECISION,
    extended_attributes JSONB,                -- Profile-specific extra
fields
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW(),
    is_active BOOLEAN DEFAULT TRUE
);

CREATE INDEX idx_prospect_records_profile ON prospect_records(profile_id);
CREATE INDEX idx_prospect_records_store ON
prospect_records(store_location_id);
CREATE INDEX idx_prospect_records_stage ON
prospect_records(pipeline_stage);
CREATE INDEX idx_prospect_records_score ON prospect_records(prospect_score
DESC);
CREATE INDEX idx_prospect_records_location ON prospect_records(latitude,
longitude);

-- Prospect profile configurations stored in DB for multi-tenant support
CREATE TABLE prospect_profiles (
    id VARCHAR(100) PRIMARY KEY,
    name VARCHAR(200) NOT NULL,
    description TEXT,
    version VARCHAR(20),
    config JSONB NOT NULL,                    -- Full profile JSON
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);

-- Interactions remain generic (work across all profiles)
CREATE TABLE prospect_interactions (
    id UUID PRIMARY KEY,
    prospect_id UUID REFERENCES prospect_records(id),
    profile_id VARCHAR(100),
    interaction_type VARCHAR(50),
    interaction_date TIMESTAMP,
    channel VARCHAR(20),                      -- sms, email, phone,
inmail, in_person
    notes TEXT,
    outcome VARCHAR(100),
    created_by VARCHAR(100),
    created_at TIMESTAMP DEFAULT NOW()
);
```

Refactoring Effort Summary

What Changes

Area	Scope	Files Affected

Area	Scope	Files Affected
ProspectProfile config model + loader	New classes	3-4 new files
Rename catering-specific fields	~20 renames	`BusinessInfo.h/cpp`, `SearchResult.h`, `AISearchService.cpp`
Extract AI prompts to config	Move strings	`AIEngine.cpp`, `OpenAIEngine.cpp`, `GeminiEngine.cpp`
Generic `calculateProspectScore()`	Replace method	`BusinessInfo.cpp`, `ScoringEngine.cpp`
Config-driven UI labels	Replace ~14 strings	`FranchiseApp.cpp`, `Sidebar.h`
Data source registry	New abstraction	1 new file + modify `AISearchService.cpp`
Profile selector UI	New dropdown	`FranchiseApp.cpp` or Settings page
Seed data files	3 JSON configs	`config/profiles/` directory

What Stays Untouched

- Search orchestration (`AISearchService` -- structure stays, just reads from profile)
- All existing API client classes (Google, OSM, BBB, Demographics)
- Geocoding service
- Thread pool and parallel processing
- HTTP/CURL infrastructure
- ApiLogicServer REST client
- Authentication service
- Audit logging
- Map visualization (Leaflet.js)
- Wt framework widgets (structure stays, labels become dynamic)
- Twilio SMS service (shared across all profiles)
- CRM integration layer

Updated Implementation Roadmap

The generic platform approach front-loads the abstraction work but accelerates all subsequent vertical deployments.

Timeline Overview

```
Month 1–2          Month 3–4          Month 5–6          Month 7–8
Month 9–10
```

```
┌─────────────────┐  ┌─────────────────┐  ┌─────────────────┐  ┌─────────────────┐
│                 │  │                 │  │                 │  │                 │
Phase 0:          Phase 1:          Phase 2:          Phase 3:
Phase 4:
Platform          First Vertical    New Data          Outreach &
Additional
Abstraction       (Franchise        Sources           Engagement
Verticals
                  Buyers)           (LinkedIn)
```

Phase 0: Platform Abstraction (Weeks 1-8) -- NEW

Week	Tasks	Deliverables
1-2	ProspectProfile schema design, ProspectRecord model	JSON schema, unified data model
3-4	ProspectProfileManager, DataSourceRegistry, TerminologyManager	Profile loading, source plugin system, dynamic labels
5-6	ScoringEngine refactor, AIPromptManager	Config-driven scoring, templated prompts
7-8	UI dynamic labels, profile selector, database migration	Generic UI, profile switching, `prospect_records` table

Phase 1: First Vertical -- Franchise Buyers (Weeks 9-16)

Week	Tasks	Deliverables
9-10	Franchise buyer profile config, search criteria	`franchise_buyer_prospecting.json`, UI filters
11-12	Pipeline stages, candidate card design	7-stage pipeline, investor-focused cards
13-14	AI prompts for investor analysis	Investor scoring, fit assessment, outreach recs
15-16	Territory management, investment calculator	Territory CRUD, ROI projections

Phase 2: New Data Sources (Weeks 17-24)

Week	Tasks	Deliverables
17-18	LinkedIn API setup, OAuth flow	`LinkedInAPI` class, token management
19-20	LinkedIn profile search, result mapping	Search API, `ProspectRecord` mapping
21-22	Career transition detection, profile enrichment	Algorithm, caching
23-24	Franchise broker API, Eventbrite API	Additional source plugins

Phase 3: Outreach & Engagement (Weeks 25-32)

Week	Tasks	Deliverables
25-26	Twilio SMS integration, pipeline hooks	`TwilioSMSService`, event-driven messaging
27-28	Marketing campaigns, grooming sequences	Campaign engine, nurture drips
29-30	AI message personalization, CRM integration	GPT-powered templates, Salesforce/HubSpot sync
31-32	SMS compliance, analytics, settings UI	TCPA/10DLC, SMS dashboard

Phase 4: Additional Verticals (Weeks 33-40)

Week	Tasks	Deliverables
33-34	Catering client profile config (restore original use case)	`franchise-catering-clients.json` -- **config only, no code**
35-36	Hotel conference venue profile config	`hotel-conference-venues.json` -- **config only, no code**
37-38	Per-profile reporting dashboards	Profile-aware analytics
39-40	Multi-profile testing, polish, documentation	Cross-profile validation, user guide

Note the key difference: **Phase 4 requires no new code**. New verticals are deployed by writing a JSON config file and loading it into the platform.

Risk Assessment (Generic Platform Additions)

Risk	Probability	Impact	Mitigation
Over-engineering the abstraction	Medium	Medium	Start with 3 concrete profiles to validate schema before generalizing further
Profile config becomes too complex	Medium	Low	Provide good defaults, validation, and a profile editor UI
Performance overhead from config lookups	Low	Low	Cache parsed profiles in memory, benchmark early
Edge cases across target types	Medium	Medium	Unified `ProspectRecord` with `extendedAttributes` for profile-specific fields
LinkedIn API access denied	Medium	High	Alternative data sources, broker partnerships

Risk	Probability	Impact	Mitigation
TCPA SMS compliance violation	Low	Critical	Legal review, strict opt-in enforcement, automated opt-out
High API costs across verticals	Medium	Medium	Aggressive caching, per-profile API budgets
User confusion with multiple profiles	Low	Medium	Clear profile selector, per-profile onboarding

Success Metrics (Platform Level)

Metric	Target	Measurement
Time to deploy new vertical	< 1 day	Config file creation to working instance
Code changes for new vertical	Zero	No source code modifications required
Cross-vertical feature parity	100%	All platform features available to all profiles
Prospect Discovery Rate	50+ qualified/month per profile	New qualified prospects
AI Scoring Accuracy	80%+ across all profiles	Scored 80+ that convert
Pipeline Velocity	Profile-dependent	Lead to conversion time
SMS Delivery Rate	> 95%	Across all profile outreach
Platform Uptime	99.5%	Shared infrastructure reliability

Appendix

A. Profile Configuration Directory Structure

```
config/
├── profiles/
│   ├── franchise-buyer-prospecting.json
│   ├── franchise-catering-clients.json
│   ├── hotel-conference-venues.json
│   └── _template.json                    // Template for creating new
profiles
├── app_config.json                        // Infrastructure config (API
keys, endpoints)
└── app_config.sample.json
```

B. Adding a New Vertical (Checklist)

1. Copy `config/profiles/_template.json` to `config/profiles/your-profile-id.json`
2. Define target type (BUSINESS or INDIVIDUAL)
3. Configure data sources (primary, secondary, optional)
4. Define search criteria fields
5. Create scoring rules and category weights
6. Write AI prompts (system, analysis, market, outreach)
7. Define pipeline stages
8. Set terminology map
9. Load profile via Settings UI or restart application
10. Test search, scoring, and pipeline flow
11. Configure SMS templates and grooming sequences

No code changes. No redeployment. No rebuild.

C. Existing Franchise Buyer Transition Plan

The original franchise buyer targeting plan (covering LinkedIn integration, investor analysis, territory management, Twilio SMS outreach, and detailed implementation specifications) is preserved in this repository's git history and remains the reference specification for the franchise buyer prospecting vertical. All of its content -- data models, API integrations, AI prompts, UI wireframes, SMS campaigns, grooming sequences, compliance requirements, and cost estimates -- is fully compatible with the generic platform architecture described in this document. The franchise buyer profile configuration above implements that plan as a profile config.

Document History

Version	Date	Author	Changes
1.0	2026-02-03	Claude AI	Initial franchise buyer transition plan
1.1	2026-02-04	Claude AI	Added Twilio SMS outbound communication section
2.0	2026-02-04	Claude AI	Restructured as generic prospecting platform plan; added platform architecture, ProspectProfile config schema, data source registry, three example vertical configs, codebase coupling analysis, unified data model, and updated roadmap

---

This document outlines the strategic transition plan for evolving FranchiseAI into a generic, configuration-driven prospecting platform. The franchise buyer vertical serves as the first implementation, with catering clients and hotel conference venues following as config-only deployments. Implementation details may be adjusted based on technical discoveries and business requirements during development.