Product Requirements Document (PRD)

Financial Services Pricing Calculator

Document Information

Project Name: RatePro - Financial Services Pricing Calculator **Version:** 1.0 **Date:** 2025-11-18 **Author:** Product Team **Status:** Draft

Table of Contents

1. Executive Summary

RatePro is a flexible, multi-factor pricing calculator designed for financial services businesses offering services such as Bookkeeping, Payroll, Year-End Accounting, Tax Preparation, and CFO Services. The system enables dynamic pricing based on multiple configurable factors including business entity type, transaction volume, reporting frequency, and service-specific parameters.

Key Highlights

- **Flexible Pricing Model:** Support for base pricing, volume tiers, entity-type modifiers, and multiple service-specific factors
- **Admin-Configurable:** Add new services and pricing factors without code changes
- **Multi-Factor Calculation:** Combine multiple pricing dimensions (transactions, employees, frequency, complexity, etc.)
- **Quote Generation:** Save, track, and export detailed pricing quotes
- **Extensible Architecture:** Built to scale with growing service offerings and pricing complexity

2. Project Overview

2.1 Problem Statement

Financial services firms need a sophisticated pricing tool that:

- Handles complex, multi-dimensional pricing (not just simple fixed prices)
- Adapts to different business entity types (Sole Proprietor, Partnership, Company, etc.)
- Scales pricing based on volume metrics (transactions, employees, etc.)
- Supports varying service frequencies (monthly, quarterly, yearly reporting)
- Allows easy updates to pricing without developer intervention
- Generates professional quotes for prospects and clients

2.2 Solution

A web-based pricing calculator with:

- Dynamic form generation based on selected service
- Real-time price calculation as users configure options
- Comprehensive admin panel for managing services, factors, and pricing rules
- Quote management system for tracking and converting opportunities
- Flexible data model supporting unlimited services and pricing factors

---

3. Business Objectives

Primary Objectives

1. **Streamline Sales Process:** Enable sales teams to generate accurate quotes in minutes
2. **Pricing Consistency:** Ensure uniform pricing across all sales channels
3. **Operational Efficiency:** Reduce manual quote creation time by 80%
4. **Scalability:** Support business growth with easy addition of new services
5. **Transparency:** Provide clients with clear, itemized pricing breakdowns

Success Metrics

- Quote generation time: < 3 minutes per quote
- Pricing accuracy: 100% (eliminate manual calculation errors)
- Service addition time: < 30 minutes for new service setup
- Quote conversion tracking and analytics

---

4. Target Users

4.1 Primary Users

Sales Team / Account Managers

- Generate quotes for prospects
- Configure service options based on client needs
- Export and send professional quotes

- Track quote status and follow-ups

Self-Service Clients (Future)

- Explore pricing options independently
- Get instant price estimates
- Request formal quotes

4.2 Secondary Users

Admin Users

- Configure services and pricing factors
- Update pricing rules and rates
- Manage business entity types and modifiers
- View pricing analytics and reporting

Finance Team

- Review and approve pricing changes
- Analyze pricing performance
- Generate revenue forecasts based on quote pipeline

---

5. Core Features

5.1 Calculator Features

- Multi-service pricing calculator
- Dynamic form rendering based on service selection
- Real-time price calculation with itemized breakdown
- Support for multiple pricing factors per service
- Factor dependency handling (conditional fields)
- Business entity type selection with pricing modifiers
- Add-on/optional service selection
- Price preview before quote generation

5.2 Quote Management

- Save quotes with unique reference numbers
- Quote status tracking (Draft, Sent, Accepted, Rejected, Expired)
- Customer information capture
- Quote versioning (track changes over time)
- PDF export with professional formatting
- Email delivery integration
- Quote expiration dates

5.3 Admin Panel

- Service catalog management (add/edit/deactivate services)

- Pricing factor configuration per service
- Factor option management (values and prices)
- Factor dependency rules
- Business entity type management
- Add-on management (global and service-specific)
- Pricing history and audit trail
- Bulk pricing updates

5.4 Reporting & Analytics (Future)

- Quote volume and conversion rates
- Popular service combinations
- Average deal size by service
- Pricing trend analysis
- Revenue forecasting

---

6. Data Schema & Architecture

6.1 Database Choice: PostgreSQL

Rationale:

- Strong relational integrity for complex pricing relationships
- JSONB support for flexible factor configurations
- Excellent performance for financial calculations
- Robust transaction support for quote generation
- Superior reporting and analytics capabilities
- Better suited for multi-tenant scenarios (future)

6.2 Core Entities Overview

The schema consists of 12 core tables organized into four functional groups:

1. **Service Configuration:** Services, Pricing Factors, Factor Options, Factor Dependencies
2. **Business Rules:** Business Entity Types, Entity Pricing Modifiers, Add-ons, Service Add-ons
3. **Quote Management:** Customers, Quotes, Quote Line Items
4. **System:** Audit Log (for tracking pricing changes)

6.3 Entity Relationship Diagram

```
erDiagram
    SERVICES ||--o{ PRICING_FACTORS : has
    SERVICES ||--o{ SERVICE_ADDONS : supports
    SERVICES ||--o{ QUOTES : generates

    PRICING_FACTORS ||--o{ FACTOR_OPTIONS : contains
    PRICING_FACTORS ||--o{ FACTOR_DEPENDENCIES : "has dependencies"
    PRICING_FACTORS }o--|| PRICING_FACTORS : "depends on"
    BUSINESS_ENTITY_TYPES ||--o{ ENTITY_PRICING_MODIFIERS : "modifies
pricing"
    SERVICES ||--o{ ENTITY_PRICING_MODIFIERS : "has modifiers"

    ADDONS ||--o{ SERVICE_ADDONS : "available for"

    CUSTOMERS ||--o{ QUOTES : requests
    QUOTES ||--o{ QUOTE_LINE_ITEMS : contains

    SERVICES {
        uuid id PK
        string name
        string description
        decimal base_price
        string category
        boolean is_active
        int display_order
        jsonb metadata
        timestamp created_at
        timestamp updated_at
    }

    PRICING_FACTORS {
        uuid id PK
        uuid service_id FK
        string factor_name
        string factor_type
        boolean is_required
        boolean is_common
        int display_order
        string help_text
        jsonb validation_rules
        timestamp created_at
        timestamp updated_at
    }

    FACTOR_OPTIONS {
        uuid id PK
        uuid factor_id FK
        string option_label
        string option_value
        decimal price_impact
        string price_impact_type
        int min_range
        int max_range
        int display_order
        boolean is_active
        timestamp created_at
        timestamp updated_at
    }

    FACTOR_DEPENDENCIES {
        uuid id PK
        uuid factor_id FK
        uuid depends_on_factor_id FK
        string depends_on_value
        string dependency_type
        jsonb condition_rules
        timestamp created_at
    }

    BUSINESS_ENTITY_TYPES {
        uuid id PK
        string entity_name
        string entity_code
        string description
        boolean is_active
        int display_order
        timestamp created_at
        timestamp updated_at
    }

    ENTITY_PRICING_MODIFIERS {
        uuid id PK
        uuid service_id FK
        uuid entity_type_id FK
        decimal modifier_value
        string modifier_type
        timestamp created_at
        timestamp updated_at
    }

    ADDONS {
        uuid id PK
        string addon_name
        string description
        decimal price
        string price_type
        boolean is_global
        boolean is_active
        timestamp created_at
        timestamp updated_at
    }

    SERVICE_ADDONS {
        uuid id PK
        uuid service_id FK
        uuid addon_id FK
        decimal override_price
        timestamp created_at
    }

    CUSTOMERS {
        uuid id PK
        string company_name
        string contact_name
        string email
        string phone
        text address
        jsonb metadata
        timestamp created_at
        timestamp updated_at
    }

    QUOTES {
        uuid id PK
        string quote_number
        uuid customer_id FK
        uuid service_id FK
        uuid entity_type_id FK
        decimal base_price
        decimal total_price
        string status
        date valid_until
        jsonb selected_factors
        jsonb selected_addons
        jsonb price_breakdown
        text notes
        timestamp created_at
        timestamp updated_at
    }

    QUOTE_LINE_ITEMS {
        uuid id PK
        uuid quote_id FK
        string item_type
        string item_description
        decimal unit_price
        int quantity
        decimal line_total
        jsonb metadata
        int display_order
        timestamp created_at
    }
```

6.4 Detailed Table Definitions

6.4.1 SERVICES Table

Stores all available financial services (Bookkeeping, Payroll, etc.)

```sql
CREATE TABLE services (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(255) NOT NULL UNIQUE,
    description TEXT,
    base_price DECIMAL(10, 2) DEFAULT 0.00,
    category VARCHAR(100),
    is_active BOOLEAN DEFAULT true,
    display_order INTEGER DEFAULT 0,
    metadata JSONB DEFAULT '{}',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_services_active ON services(is_active);
CREATE INDEX idx_services_category ON services(category);
```

Column Descriptions:

- `id`: Unique identifier
- `name`: Service name (e.g., "Bookkeeping", "Payroll")
- `description`: Detailed service description for clients
- `base_price`: Optional base price (can be 0 if fully factor-driven)
- `category`: Group services (e.g., "Accounting", "Tax", "Advisory")
- `is_active`: Enable/disable service without deletion
- `display_order`: Control display sequence in UI
- `metadata`: Flexible JSON for future extensions (icons, colors, etc.)

Sample Data:

```
INSERT INTO services (name, description, base_price, category,
display_order) VALUES
('Bookkeeping', 'Monthly bookkeeping services including transaction
categorization and reconciliation', 200.00, 'Accounting', 1),
('Payroll', 'Comprehensive payroll processing and tax filing services',
150.00, 'Payroll', 2),
('Year End Accounting', 'Annual financial statement preparation and
closing', 500.00, 'Accounting', 3),
('Tax Preparation', 'Business tax return preparation and filing', 300.00,
'Tax', 4),
('CFO Services', 'Part-time CFO advisory and strategic financial
planning', 1000.00, 'Advisory', 5);
```

6.4.2 PRICING_FACTORS Table

Defines configurable pricing factors for each service

```
CREATE TABLE pricing_factors (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    service_id UUID REFERENCES services(id) ON DELETE CASCADE,
    factor_name VARCHAR(255) NOT NULL,
    factor_type VARCHAR(50) NOT NULL, -- 'select', 'range', 'numeric',
'boolean'
    is_required BOOLEAN DEFAULT true,
    is_common BOOLEAN DEFAULT false, -- true if applies to all services
    display_order INTEGER DEFAULT 0,
    help_text TEXT,
    validation_rules JSONB DEFAULT '{}',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT unique_service_factor UNIQUE(service_id, factor_name)
);

CREATE INDEX idx_pricing_factors_service ON pricing_factors(service_id);
CREATE INDEX idx_pricing_factors_common ON pricing_factors(is_common);
```

Column Descriptions:

- `service_id`: Links to specific service (NULL if common factor)
- `factor_name`: Factor display name (e.g., "Number of Transactions", "Report Frequency")
- `factor_type`: Data type for UI rendering
  - `select`: Dropdown with predefined options
  - `range`: Numeric range selector (e.g., 0-1000, 1001-1500)
  - `numeric`: Free-form number input
  - `boolean`: Yes/No toggle
- `is_required`: Must be selected to calculate price
- `is_common`: Applies to all services (e.g., Business Entity Type)
- `display_order`: Control field order in forms
- `help_text`: Tooltip or help text for users
- `validation_rules`: JSON rules (min/max values, regex patterns, etc.)

Sample Data:

```
-- Common factor (applies to all services)
INSERT INTO pricing_factors (service_id, factor_name, factor_type,
is_required, is_common, display_order, help_text) VALUES
(NULL, 'Business Entity Type', 'select', true, true, 1, 'Select your
business structure');

-- Bookkeeping-specific factors
INSERT INTO pricing_factors (service_id, factor_name, factor_type,
is_required, is_common, display_order, help_text) VALUES
((SELECT id FROM services WHERE name = 'Bookkeeping'), 'Number of
Transactions', 'range', true, false, 2, 'Average monthly transactions'),
((SELECT id FROM services WHERE name = 'Bookkeeping'), 'Report Frequency',
'select', true, false, 3, 'How often do you need financial reports?'),
((SELECT id FROM services WHERE name = 'Bookkeeping'), 'Accounting
Software', 'select', true, false, 4, 'Which software do you use?');

-- Payroll-specific factors
INSERT INTO pricing_factors (service_id, factor_name, factor_type,
is_required, is_common, display_order, help_text) VALUES
((SELECT id FROM services WHERE name = 'Payroll'), 'Number of Employees',
'range', true, false, 2, 'Total number of employees on payroll'),
((SELECT id FROM services WHERE name = 'Payroll'), 'Payroll Frequency',
'select', true, false, 3, 'How often do you run payroll?'),
((SELECT id FROM services WHERE name = 'Payroll'), 'Multi-State
Operations', 'boolean', false, false, 4, 'Do you have employees in
multiple states?');
```

6.4.3 FACTOR_OPTIONS Table

Stores specific options and their pricing for each factor

```
CREATE TABLE factor_options (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    factor_id UUID REFERENCES pricing_factors(id) ON DELETE CASCADE,
    option_label VARCHAR(255) NOT NULL,
    option_value VARCHAR(255) NOT NULL,
    price_impact DECIMAL(10, 2) DEFAULT 0.00,
    price_impact_type VARCHAR(20) DEFAULT 'fixed', -- 'fixed',
'percentage', 'multiplier'
    min_range INTEGER, -- For range-type factors
    max_range INTEGER, -- For range-type factors
    display_order INTEGER DEFAULT 0,
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_factor_options_factor ON factor_options(factor_id);
CREATE INDEX idx_factor_options_active ON factor_options(is_active);
```

Column Descriptions:

- `factor_id`: Links to pricing factor
- `option_label`: Display text (e.g., "Monthly", "0-1000 transactions")
- `option_value`: Stored value (e.g., "monthly", "0-1000")
- `price_impact`: The price adjustment for this option
- `price_impact_type`: How to apply the price
    - `fixed`: Add/subtract fixed amount (e.g., +$200)
    - `percentage`: Apply percentage change (e.g., +15%)
    - `multiplier`: Multiply current price (e.g., 1.5x)
- `min_range` / `max_range`: For numeric range options
- `display_order`: Control option sequence
- `is_active`: Enable/disable without deletion

Sample Data:

```
-- Business Entity Type options (common factor)
INSERT INTO factor_options (factor_id, option_label, option_value,
price_impact, price_impact_type, display_order) VALUES
```

```sql
((SELECT id FROM pricing_factors WHERE factor_name = 'Business Entity
Type'), 'Sole Proprietor', 'sole_proprietor', 0.00, 'fixed', 1),
((SELECT id FROM pricing_factors WHERE factor_name = 'Business Entity
Type'), 'Partnership', 'partnership', 50.00, 'fixed', 2),
((SELECT id FROM pricing_factors WHERE factor_name = 'Business Entity
Type'), 'LLC', 'llc', 75.00, 'fixed', 3),
((SELECT id FROM pricing_factors WHERE factor_name = 'Business Entity
Type'), 'S-Corp', 's_corp', 100.00, 'fixed', 4),
((SELECT id FROM pricing_factors WHERE factor_name = 'Business Entity
Type'), 'C-Corp', 'c_corp', 150.00, 'fixed', 5),
((SELECT id FROM pricing_factors WHERE factor_name = 'Business Entity
Type'), 'Company', 'company', 200.00, 'fixed', 6);

-- Bookkeeping: Number of Transactions
INSERT INTO factor_options (factor_id, option_label, option_value,
price_impact, price_impact_type, min_range, max_range, display_order)
VALUES
((SELECT id FROM pricing_factors WHERE factor_name = 'Number of
Transactions' AND service_id = (SELECT id FROM services WHERE name =
'Bookkeeping')),
 '0-1000 transactions', '0-1000', 0.00, 'fixed', 0, 1000, 1),
((SELECT id FROM pricing_factors WHERE factor_name = 'Number of
Transactions' AND service_id = (SELECT id FROM services WHERE name =
'Bookkeeping')),
 '1001-1500 transactions', '1001-1500', 150.00, 'fixed', 1001, 1500, 2),
((SELECT id FROM pricing_factors WHERE factor_name = 'Number of
Transactions' AND service_id = (SELECT id FROM services WHERE name =
'Bookkeeping')),
 '1501-2000 transactions', '1501-2000', 300.00, 'fixed', 1501, 2000, 3),
((SELECT id FROM pricing_factors WHERE factor_name = 'Number of
Transactions' AND service_id = (SELECT id FROM services WHERE name =
'Bookkeeping')),
 '2001-3000 transactions', '2001-3000', 500.00, 'fixed', 2001, 3000, 4),
((SELECT id FROM pricing_factors WHERE factor_name = 'Number of
Transactions' AND service_id = (SELECT id FROM services WHERE name =
'Bookkeeping')),
 '3000+ transactions', '3000+', 800.00, 'fixed', 3001, NULL, 5);

-- Bookkeeping: Report Frequency
INSERT INTO factor_options (factor_id, option_label, option_value,
price_impact, price_impact_type, display_order) VALUES
((SELECT id FROM pricing_factors WHERE factor_name = 'Report Frequency'
AND service_id = (SELECT id FROM services WHERE name = 'Bookkeeping')),
 'Yearly', 'yearly', 0.00, 'fixed', 1),
((SELECT id FROM pricing_factors WHERE factor_name = 'Report Frequency'
AND service_id = (SELECT id FROM services WHERE name = 'Bookkeeping')),
 'Half-Yearly', 'half_yearly', 100.00, 'fixed', 2),
((SELECT id FROM pricing_factors WHERE factor_name = 'Report Frequency'
AND service_id = (SELECT id FROM services WHERE name = 'Bookkeeping')),
 'Quarterly', 'quarterly', 200.00, 'fixed', 3),
((SELECT id FROM pricing_factors WHERE factor_name = 'Report Frequency'
AND service_id = (SELECT id FROM services WHERE name = 'Bookkeeping')),
 'Monthly', 'monthly', 350.00, 'fixed', 4);
```

```sql
-- Bookkeeping: Accounting Software
INSERT INTO factor_options (factor_id, option_label, option_value,
price_impact, price_impact_type, display_order) VALUES
((SELECT id FROM pricing_factors WHERE factor_name = 'Accounting Software'
AND service_id = (SELECT id FROM services WHERE name = 'Bookkeeping')),
 'QuickBooks Online', 'qbo', 0.00, 'fixed', 1),
((SELECT id FROM pricing_factors WHERE factor_name = 'Accounting Software'
AND service_id = (SELECT id FROM services WHERE name = 'Bookkeeping')),
 'QuickBooks Desktop', 'qbd', 50.00, 'fixed', 2),
((SELECT id FROM pricing_factors WHERE factor_name = 'Accounting Software'
AND service_id = (SELECT id FROM services WHERE name = 'Bookkeeping')),
 'Xero', 'xero', 0.00, 'fixed', 3),
((SELECT id FROM pricing_factors WHERE factor_name = 'Accounting Software'
AND service_id = (SELECT id FROM services WHERE name = 'Bookkeeping')),
 'Manual/Spreadsheet', 'manual', 200.00, 'fixed', 4);

-- Payroll: Number of Employees
INSERT INTO factor_options (factor_id, option_label, option_value,
price_impact, price_impact_type, min_range, max_range, display_order)
VALUES
((SELECT id FROM pricing_factors WHERE factor_name = 'Number of Employees'
AND service_id = (SELECT id FROM services WHERE name = 'Payroll')),
 '1-10 employees', '1-10', 0.00, 'fixed', 1, 10, 1),
((SELECT id FROM pricing_factors WHERE factor_name = 'Number of Employees'
AND service_id = (SELECT id FROM services WHERE name = 'Payroll')),
 '11-50 employees', '11-50', 200.00, 'fixed', 11, 50, 2),
((SELECT id FROM pricing_factors WHERE factor_name = 'Number of Employees'
AND service_id = (SELECT id FROM services WHERE name = 'Payroll')),
 '51-100 employees', '51-100', 500.00, 'fixed', 51, 100, 3),
((SELECT id FROM pricing_factors WHERE factor_name = 'Number of Employees'
AND service_id = (SELECT id FROM services WHERE name = 'Payroll')),
 '100+ employees', '100+', 1000.00, 'fixed', 101, NULL, 4);

-- Payroll: Payroll Frequency
INSERT INTO factor_options (factor_id, option_label, option_value,
price_impact, price_impact_type, display_order) VALUES
((SELECT id FROM pricing_factors WHERE factor_name = 'Payroll Frequency'
AND service_id = (SELECT id FROM services WHERE name = 'Payroll')),
 'Monthly', 'monthly', 0.00, 'fixed', 1),
((SELECT id FROM pricing_factors WHERE factor_name = 'Payroll Frequency'
AND service_id = (SELECT id FROM services WHERE name = 'Payroll')),
 'Bi-Weekly', 'bi_weekly', 100.00, 'fixed', 2),
((SELECT id FROM pricing_factors WHERE factor_name = 'Payroll Frequency'
AND service_id = (SELECT id FROM services WHERE name = 'Payroll')),
 'Weekly', 'weekly', 200.00, 'fixed', 3);
```

6.4.4 FACTOR_DEPENDENCIES Table

Manages conditional factor visibility and validation

```sql
CREATE TABLE factor_dependencies (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    factor_id UUID REFERENCES pricing_factors(id) ON DELETE CASCADE,
    depends_on_factor_id UUID REFERENCES pricing_factors(id) ON DELETE
CASCADE,
    depends_on_value VARCHAR(255), -- The value that triggers this
dependency
    dependency_type VARCHAR(50) DEFAULT 'show_if', -- 'show_if',
'hide_if', 'require_if'
    condition_rules JSONB DEFAULT '{}', -- Complex conditions (AND/OR
logic)
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT no_self_dependency CHECK (factor_id !=
depends_on_factor_id)
);

CREATE INDEX idx_factor_dependencies_factor ON
factor_dependencies(factor_id);
CREATE INDEX idx_factor_dependencies_depends_on ON
factor_dependencies(depends_on_factor_id);
```

Column Descriptions:

- `factor_id`: The dependent factor (the one that appears/hides)
- `depends_on_factor_id`: The factor it depends on
- `depends_on_value`: The specific value that triggers the dependency
- `dependency_type`: How the dependency behaves
  - `show_if`: Only show factor if condition is met
  - `hide_if`: Hide factor if condition is met
  - `require_if`: Make factor required if condition is met
- `condition_rules`: JSON for complex conditions (future enhancement)

Sample Data:

```sql
-- Example: Show "Multi-State Operations" only if employees > 10
INSERT INTO factor_dependencies (factor_id, depends_on_factor_id,
depends_on_value, dependency_type) VALUES
((SELECT id FROM pricing_factors WHERE factor_name = 'Multi-State
Operations' AND service_id = (SELECT id FROM services WHERE name =
'Payroll')),
 (SELECT id FROM pricing_factors WHERE factor_name = 'Number of Employees'
AND service_id = (SELECT id FROM services WHERE name = 'Payroll')),
 '11-50', 'show_if');

-- Note: Would need multiple rows for "11-50", "51-100", "100+" or use
condition_rules for range logic
```

6.4.5 BUSINESS_ENTITY_TYPES Table

Defines available business entity types

```
CREATE TABLE business_entity_types (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    entity_name VARCHAR(100) NOT NULL UNIQUE,
    entity_code VARCHAR(50) NOT NULL UNIQUE,
    description TEXT,
    is_active BOOLEAN DEFAULT true,
    display_order INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_entity_types_active ON business_entity_types(is_active);
```

Sample Data:

```
INSERT INTO business_entity_types (entity_name, entity_code, description,
display_order) VALUES
('Sole Proprietor', 'sole_proprietor', 'Single-owner unincorporated
business', 1),
('Partnership', 'partnership', 'Business owned by two or more partners',
2),
('LLC', 'llc', 'Limited Liability Company', 3),
('S-Corporation', 's_corp', 'Small business corporation with pass-through
taxation', 4),
('C-Corporation', 'c_corp', 'Standard corporation with double taxation',
5),
('Company', 'company', 'General company designation', 6),
('Non-Profit', 'non_profit', '501(c)(3) or other non-profit organization',
7);
```

6.4.6 ENTITY_PRICING_MODIFIERS Table

Service-specific pricing adjustments for entity types

```
CREATE TABLE entity_pricing_modifiers (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    service_id UUID REFERENCES services(id) ON DELETE CASCADE,
    entity_type_id UUID REFERENCES business_entity_types(id) ON DELETE
CASCADE,
    modifier_value DECIMAL(10, 2) NOT NULL,
    modifier_type VARCHAR(20) DEFAULT 'fixed', -- 'fixed', 'percentage',
'multiplier'
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```sql
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT unique_service_entity UNIQUE(service_id, entity_type_id)
);

CREATE INDEX idx_entity_modifiers_service ON
entity_pricing_modifiers(service_id);
CREATE INDEX idx_entity_modifiers_entity ON
entity_pricing_modifiers(entity_type_id);
```

Column Descriptions:

- `service_id`: The service this modifier applies to
- `entity_type_id`: The entity type
- `modifier_value`: The adjustment amount
- `modifier_type`: How to apply
  - `fixed`: Add fixed amount
  - `percentage`: Add percentage of base price
  - `multiplier`: Multiply total by this value

Sample Data:

```sql
-- Bookkeeping entity modifiers
INSERT INTO entity_pricing_modifiers (service_id, entity_type_id,
modifier_value, modifier_type) VALUES
((SELECT id FROM services WHERE name = 'Bookkeeping'), (SELECT id FROM
business_entity_types WHERE entity_code = 'sole_proprietor'), 0.00,
'multiplier'),
((SELECT id FROM services WHERE name = 'Bookkeeping'), (SELECT id FROM
business_entity_types WHERE entity_code = 'partnership'), 1.15,
'multiplier'),
((SELECT id FROM services WHERE name = 'Bookkeeping'), (SELECT id FROM
business_entity_types WHERE entity_code = 'llc'), 1.20, 'multiplier'),
((SELECT id FROM services WHERE name = 'Bookkeeping'), (SELECT id FROM
business_entity_types WHERE entity_code = 's_corp'), 1.30, 'multiplier'),
((SELECT id FROM services WHERE name = 'Bookkeeping'), (SELECT id FROM
business_entity_types WHERE entity_code = 'c_corp'), 1.50, 'multiplier'),
((SELECT id FROM services WHERE name = 'Bookkeeping'), (SELECT id FROM
business_entity_types WHERE entity_code = 'company'), 1.50, 'multiplier');
```

6.4.7 ADDONS Table

Optional add-on services or features

```sql
CREATE TABLE addons (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    addon_name VARCHAR(255) NOT NULL UNIQUE,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    price_type VARCHAR(20) DEFAULT 'fixed', -- 'fixed', 'percentage'
    is_global BOOLEAN DEFAULT false, -- true if available for all services
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_addons_global ON addons(is_global);
CREATE INDEX idx_addons_active ON addons(is_active);
```

Sample Data:

```
INSERT INTO addons (addon_name, description, price, price_type, is_global)
VALUES
('Rush Service (48-hour turnaround)', 'Expedited processing with 48-hour
delivery', 200.00, 'fixed', true),
('Priority Support', 'Dedicated account manager and priority phone
support', 150.00, 'fixed', true),
('Multi-Currency Support', 'Handle transactions in multiple currencies',
100.00, 'fixed', false),
('Industry-Specific Reporting', 'Custom reports for specific industries
(construction, e-commerce, etc.)', 175.00, 'fixed', false),
('Cloud Storage Integration', 'Automatic backup to Dropbox/Google Drive',
50.00, 'fixed', true);
```

6.4.8 SERVICE_ADDONS Table

Links add-ons to specific services (for non-global add-ons)

```
CREATE TABLE service_addons (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    service_id UUID REFERENCES services(id) ON DELETE CASCADE,
    addon_id UUID REFERENCES addons(id) ON DELETE CASCADE,
    override_price DECIMAL(10, 2), -- Optional: Override the default addon
price for this service
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT unique_service_addon UNIQUE(service_id, addon_id)
);

CREATE INDEX idx_service_addons_service ON service_addons(service_id);
CREATE INDEX idx_service_addons_addon ON service_addons(addon_id);
```

Sample Data:

```sql
-- Make "Multi-Currency Support" available for Bookkeeping
INSERT INTO service_addons (service_id, addon_id) VALUES
((SELECT id FROM services WHERE name = 'Bookkeeping'), (SELECT id FROM
addons WHERE addon_name = 'Multi-Currency Support'));

-- Make "Industry-Specific Reporting" available for Bookkeeping with
custom price
INSERT INTO service_addons (service_id, addon_id, override_price) VALUES
((SELECT id FROM services WHERE name = 'Bookkeeping'), (SELECT id FROM
addons WHERE addon_name = 'Industry-Specific Reporting'), 200.00);
```

6.4.9 CUSTOMERS Table

Client/prospect information

```sql
CREATE TABLE customers (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    company_name VARCHAR(255),
    contact_name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    phone VARCHAR(50),
    address TEXT,
    metadata JSONB DEFAULT '{}', -- Additional custom fields
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_customers_email ON customers(email);
CREATE INDEX idx_customers_company ON customers(company_name);
```

Sample Data:

```sql
INSERT INTO customers (company_name, contact_name, email, phone, address)
VALUES
('Acme Retail Inc.', 'John Smith', 'john.smith@acmeretail.com', '555-
0100', '123 Main St, New York, NY 10001'),
('Tech Startup LLC', 'Jane Doe', 'jane@techstartup.com', '555-0200', '456
Innovation Dr, San Francisco, CA 94105'),
('Family Restaurant Partners', 'Mike Johnson',
'mike@familyrestaurant.com', '555-0300', '789 Food Ave, Chicago, IL
60601');
```

6.4.10 QUOTES Table

Generated price quotes

```sql
CREATE TABLE quotes (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    quote_number VARCHAR(50) NOT NULL UNIQUE,
    customer_id UUID REFERENCES customers(id) ON DELETE SET NULL,
    service_id UUID REFERENCES services(id) ON DELETE SET NULL,
    entity_type_id UUID REFERENCES business_entity_types(id) ON DELETE SET
NULL,
    base_price DECIMAL(10, 2) NOT NULL,
    total_price DECIMAL(10, 2) NOT NULL,
    status VARCHAR(50) DEFAULT 'draft', -- 'draft', 'sent', 'accepted',
'rejected', 'expired'
    valid_until DATE,
    selected_factors JSONB DEFAULT '{}', -- Store all selected factor
values
    selected_addons JSONB DEFAULT '[]', -- Store selected addon IDs
    price_breakdown JSONB DEFAULT '{}', -- Itemized calculation details
    notes TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_quotes_customer ON quotes(customer_id);
CREATE INDEX idx_quotes_service ON quotes(service_id);
CREATE INDEX idx_quotes_status ON quotes(status);
CREATE INDEX idx_quotes_number ON quotes(quote_number);
CREATE INDEX idx_quotes_created ON quotes(created_at DESC);
```

Column Descriptions:

- `quote_number`: Human-readable unique ID (e.g., "Q-2025-001")
- `selected_factors`: JSON object storing factor selections

  ```json
  {
    "transaction_count": "1001-1500",
    "report_frequency": "monthly",
    "accounting_software": "qbo"
  }
  ```

- `selected_addons`: JSON array of addon IDs

  ```json
  ["uuid-addon-1", "uuid-addon-2"]
  ```

- `price_breakdown`: Detailed calculation for transparency

  ```json
  {
    "base_price": 200.00,
  ```

```
    "entity_modifier": 1.20,
    "after_entity": 240.00,
    "factors": [
      {"name": "Transactions (1001-1500)", "amount": 150.00},
      {"name": "Monthly Reports", "amount": 350.00}
    ],
    "subtotal": 740.00,
    "addons": [
      {"name": "Rush Service", "amount": 200.00}
    ],
    "total": 940.00
  }
```

Sample Data:

```
INSERT INTO quotes (quote_number, customer_id, service_id, entity_type_id,
base_price, total_price, status, valid_until, selected_factors,
selected_addons, price_breakdown, notes) VALUES
('Q-2025-001',
 (SELECT id FROM customers WHERE email = 'john.smith@acmeretail.com'),
 (SELECT id FROM services WHERE name = 'Bookkeeping'),
 (SELECT id FROM business_entity_types WHERE entity_code = 'c_corp'),
 200.00,
 940.00,
 'sent',
 '2025-12-31',
 '{"transaction_count": "1001-1500", "report_frequency": "monthly",
"accounting_software": "qbo"}',
 '[]',
 '{"base_price": 200.00, "entity_modifier": 1.5, "after_entity": 300.00,
"factors": [{"name": "Transactions (1001-1500)", "amount": 150.00},
{"name": "Monthly Reports", "amount": 350.00}], "subtotal": 800.00,
"addons": [{"name": "Rush Service", "amount": 200.00}], "total":
1000.00}',
 'Client requested rush turnaround for year-end');
```

6.4.11 QUOTE_LINE_ITEMS Table

Detailed line items for each quote (alternative to JSONB storage)

```
CREATE TABLE quote_line_items (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    quote_id UUID REFERENCES quotes(id) ON DELETE CASCADE,
    item_type VARCHAR(50) NOT NULL, -- 'base', 'factor',
'entity_modifier', 'addon'
    item_description TEXT NOT NULL,
    unit_price DECIMAL(10, 2) NOT NULL,
    quantity INTEGER DEFAULT 1,
    line_total DECIMAL(10, 2) NOT NULL,
    metadata JSONB DEFAULT '{}',
    display_order INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_quote_items_quote ON quote_line_items(quote_id);
CREATE INDEX idx_quote_items_type ON quote_line_items(item_type);
```

Sample Data:

```
INSERT INTO quote_line_items (quote_id, item_type, item_description,
unit_price, quantity, line_total, display_order) VALUES
((SELECT id FROM quotes WHERE quote_number = 'Q-2025-001'), 'base',
'Bookkeeping - Base Service', 200.00, 1, 200.00, 1),
((SELECT id FROM quotes WHERE quote_number = 'Q-2025-001'),
'entity_modifier', 'C-Corporation Premium (1.5x multiplier)', 100.00, 1,
100.00, 2),
((SELECT id FROM quotes WHERE quote_number = 'Q-2025-001'), 'factor',
'Transaction Volume: 1001-1500', 150.00, 1, 150.00, 3),
((SELECT id FROM quotes WHERE quote_number = 'Q-2025-001'), 'factor',
'Monthly Reporting', 350.00, 1, 350.00, 4),
((SELECT id FROM quotes WHERE quote_number = 'Q-2025-001'), 'addon', 'Rush
Service (48-hour)', 200.00, 1, 200.00, 5);
```

6.5 Database Indexes Summary

Key indexes for performance optimization:

```
-- Services
CREATE INDEX idx_services_active ON services(is_active);
CREATE INDEX idx_services_category ON services(category);

-- Pricing Factors
CREATE INDEX idx_pricing_factors_service ON pricing_factors(service_id);
CREATE INDEX idx_pricing_factors_common ON pricing_factors(is_common);

-- Factor Options
CREATE INDEX idx_factor_options_factor ON factor_options(factor_id);
CREATE INDEX idx_factor_options_active ON factor_options(is_active);

-- Factor Dependencies
CREATE INDEX idx_factor_dependencies_factor ON
factor_dependencies(factor_id);
CREATE INDEX idx_factor_dependencies_depends_on ON
factor_dependencies(depends_on_factor_id);

-- Entity Types & Modifiers
```

```sql
CREATE INDEX idx_entity_types_active ON business_entity_types(is_active);
CREATE INDEX idx_entity_modifiers_service ON
entity_pricing_modifiers(service_id);
CREATE INDEX idx_entity_modifiers_entity ON
entity_pricing_modifiers(entity_type_id);

-- Add-ons
CREATE INDEX idx_addons_global ON addons(is_global);
CREATE INDEX idx_addons_active ON addons(is_active);
CREATE INDEX idx_service_addons_service ON service_addons(service_id);
CREATE INDEX idx_service_addons_addon ON service_addons(addon_id);

-- Quotes
CREATE INDEX idx_quotes_customer ON quotes(customer_id);
CREATE INDEX idx_quotes_service ON quotes(service_id);
CREATE INDEX idx_quotes_status ON quotes(status);
CREATE INDEX idx_quotes_number ON quotes(quote_number);
CREATE INDEX idx_quotes_created ON quotes(created_at DESC);

-- Quote Line Items
CREATE INDEX idx_quote_items_quote ON quote_line_items(quote_id);
CREATE INDEX idx_quote_items_type ON quote_line_items(item_type);

-- Customers
CREATE INDEX idx_customers_email ON customers(email);
CREATE INDEX idx_customers_company ON customers(company_name);
```

6.6 Schema Design Principles

1. Flexibility First

- New services can be added via admin UI without schema changes
- Pricing factors are fully configurable per service
- Support for both common and service-specific factors

2. Normalization

- Avoid data duplication
- Use foreign keys to maintain referential integrity
- Separate concerns (services, factors, options, quotes)

3. Extensibility

- JSONB columns (`metadata`, `validation_rules`, `condition_rules`) allow future enhancements without migrations
- Soft deletes via `is_active` flags preserve historical data
- Versioning support through timestamps

4. Performance

- Strategic indexes on frequently queried columns

- UUID for distributed systems and security
- Optimized for read-heavy quote generation workload

5. Audit Trail

- All tables include `created_at` and `updated_at`
- Quote history preserved even if pricing changes
- Consider adding separate `audit_log` table for compliance

---

7. Pricing Calculation Logic

7.1 Calculation Algorithm

The pricing engine follows this multi-step process:

Step 1: Initialize with Base Price

```
price = service.base_price
```

Step 2: Apply Entity Type Modifier

```
entity_modifier = get_entity_modifier(service_id, entity_type_id)

if entity_modifier.modifier_type == 'multiplier':
    price = price * entity_modifier.modifier_value
else if entity_modifier.modifier_type == 'fixed':
    price = price + entity_modifier.modifier_value
else if entity_modifier.modifier_type == 'percentage':
    price = price * (1 + entity_modifier.modifier_value / 100)
```

Step 3: Add Factor-Based Pricing

```
for each selected_factor in user_selections:
    factor_option = get_factor_option(factor_id, selected_value)

    if factor_option.price_impact_type == 'fixed':
        price = price + factor_option.price_impact
    else if factor_option.price_impact_type == 'percentage':
        price = price * (1 + factor_option.price_impact / 100)
    else if factor_option.price_impact_type == 'multiplier':
        price = price * factor_option.price_impact
```

Step 4: Add Optional Add-ons

```
for each addon in selected_addons:
    if addon.price_type == 'fixed':
        price = price + addon.price
    else if addon.price_type == 'percentage':
        price = price * (1 + addon.price / 100)
```

Step 5: Return Final Price

```
return round(price, 2)
```

7.2 Calculation Example

Scenario: Bookkeeping for C-Corp with 1200 transactions, monthly reports

Input:

- Service: Bookkeeping (base_price = $200)
- Entity Type: C-Corp (multiplier = 1.5x)
- Transactions: 1001-1500 (fixed +$150)
- Report Frequency: Monthly (fixed +$350)
- Accounting Software: QuickBooks Online (fixed +$0)
- Add-on: Rush Service (fixed +$200)

Calculation:

```
1. Base Price: $200.00

2. Entity Modifier (C-Corp 1.5x multiplier):
   $200.00 × 1.5 = $300.00

3. Factor: Transactions (1001-1500):
   $300.00 + $150.00 = $450.00

4. Factor: Monthly Reports:
   $450.00 + $350.00 = $800.00

5. Factor: QuickBooks Online:
   $800.00 + $0.00 = $800.00

6. Add-on: Rush Service:
   $800.00 + $200.00 = $1,000.00

TOTAL: $1,000.00
```

Price Breakdown Stored in Quote:

```json
{
  "base_price": 200.00,
  "entity_type": "C-Corp",
  "entity_modifier": 1.5,
  "price_after_entity": 300.00,
  "factors": [
    {"name": "Transactions (1001-1500)", "impact": 150.00,
"running_total": 450.00},
    {"name": "Monthly Reports", "impact": 350.00, "running_total":
800.00},
    {"name": "QuickBooks Online", "impact": 0.00, "running_total": 800.00}
  ],
  "subtotal": 800.00,
  "addons": [
    {"name": "Rush Service (48-hour)", "price": 200.00}
  ],
  "total": 1000.00
}
```

7.3 Edge Cases & Validation

1. Missing Required Factors

- Validate all required factors are selected before calculation
- Return error with list of missing factors

2. Invalid Factor Values

- Validate selected values exist in `factor_options`
- Check range boundaries for numeric inputs

3. Factor Dependencies

- Evaluate dependencies before showing/requiring factors
- Hide dependent factors if conditions not met
- Clear dependent selections if parent changes

4. Circular Dependencies

- Prevent during admin configuration
- Implement validation at database level (`no_self_dependency` constraint)

5. Inactive Options

- Filter out inactive services, factors, options, and add-ons
- Preserve inactive data in historical quotes

6. Pricing Changes

- Quotes store snapshot of pricing at time of generation
- Recalculation uses current pricing rules

- Version tracking for pricing rule changes

7.4 Pseudo-code Implementation

```
function calculatePrice(serviceId, entityTypeId, selectedFactors,
selectedAddons) {
    // Step 1: Get base price
    const service = await Service.findById(serviceId);
    let price = service.base_price;
    const breakdown = {
        base_price: price,
        entity_type: null,
        entity_modifier: 1.0,
        price_after_entity: price,
        factors: [],
        subtotal: price,
        addons: [],
        total: 0
    };

    // Step 2: Apply entity type modifier
    const entityModifier = await EntityPricingModifier.findOne({
        service_id: serviceId,
        entity_type_id: entityTypeId
    });

    if (entityModifier) {
        breakdown.entity_type = await
BusinessEntityType.findById(entityTypeId).entity_name;
        breakdown.entity_modifier = entityModifier.modifier_value;

        switch(entityModifier.modifier_type) {
            case 'multiplier':
                price = price * entityModifier.modifier_value;
                break;
            case 'fixed':
                price = price + entityModifier.modifier_value;
                break;
            case 'percentage':
                price = price * (1 + entityModifier.modifier_value / 100);
                break;
        }

        breakdown.price_after_entity = price;
    }

    // Step 3: Apply factor-based pricing
    for (const [factorId, selectedValue] of
Object.entries(selectedFactors)) {
        const factorOption = await FactorOption.findOne({
            factor_id: factorId,
            option_value: selectedValue,
            is_active: true
        });

        if (!factorOption) {
            throw new Error(`Invalid factor selection: ${factorId} =
${selectedValue}`);
        }

        const factor = await PricingFactor.findById(factorId);
        let impact = 0;

        switch(factorOption.price_impact_type) {
            case 'fixed':
                price += factorOption.price_impact;
                impact = factorOption.price_impact;
                break;
            case 'percentage':
                const percentageImpact = price *
(factorOption.price_impact / 100);
                price += percentageImpact;
                impact = percentageImpact;
                break;
            case 'multiplier':
                const originalPrice = price;
                price *= factorOption.price_impact;
                impact = price - originalPrice;
                break;
        }

        breakdown.factors.push({
            name: `${factor.factor_name}: ${factorOption.option_label}`,
            impact: impact,
            running_total: price
        });
    }

    breakdown.subtotal = price;

    // Step 4: Apply add-ons
    for (const addonId of selectedAddons) {
        const addon = await Addon.findById(addonId);

        if (!addon || !addon.is_active) {
            throw new Error(`Invalid or inactive addon: ${addonId}`);
        }

        let addonPrice = addon.price;

        // Check for service-specific override
        const serviceAddon = await ServiceAddon.findOne({
            service_id: serviceId,
            addon_id: addonId
        });
        if (serviceAddon && serviceAddon.override_price) {
            addonPrice = serviceAddon.override_price;
        }

        if (addon.price_type === 'percentage') {
            addonPrice = price * (addon.price / 100);
        }

        price += addonPrice;

        breakdown.addons.push({
            name: addon.addon_name,
            price: addonPrice
        });
    }

    breakdown.total = Math.round(price * 100) / 100; // Round to 2
decimals

    return {
        total_price: breakdown.total,
        breakdown: breakdown
    };
}
```

---

8. Functional Requirements

8.1 Calculator Interface

FR-1: Service Selection

- User selects a service from dropdown or card grid
- Display service description and base price (if applicable)
- Only show active services

FR-2: Dynamic Form Generation

- Load pricing factors specific to selected service
- Display common factors (e.g., Business Entity Type) for all services
- Render appropriate input types (dropdown, radio, numeric input) based on factor_type
- Show help text/tooltips for each factor

FR-3: Factor Dependency Handling

- Show/hide factors based on dependency rules
- Re-validate when dependent factor values change
- Clear selections when dependency conditions no longer met

FR-4: Real-Time Price Calculation

- Calculate price as user makes selections

- Display running total
- Show itemized breakdown (expandable/collapsible)

FR-5: Add-on Selection

- Display available add-ons (global + service-specific)
- Allow multiple add-on selections
- Update price in real-time

FR-6: Validation

- Validate all required factors are selected
- Show clear error messages for missing or invalid inputs
- Prevent quote generation until validation passes

FR-7: Price Breakdown Display

```
Base Price:                     $200.00
Business Entity (C-Corp 1.5x):  $100.00
─────────────────────────────────────
Subtotal:                       $300.00

Pricing Factors:
  • Transactions (1001-1500):   $150.00
  • Monthly Reports:            $350.00
  • QuickBooks Online:            $0.00
─────────────────────────────────────
Service Subtotal:               $800.00

Add-ons:
  • Rush Service (48-hour):     $200.00
─────────────────────────────────────
TOTAL:                        $1,000.00
```

8.2 Quote Generation

FR-8: Customer Information Capture

- Form fields: Company Name, Contact Name, Email, Phone, Address
- Validate email format
- Optional: Check for existing customer

FR-9: Quote Creation

- Generate unique quote number (e.g., Q-2025-001)
- Set default expiration date (e.g., 30 days from creation)
- Store complete price breakdown
- Store all selected factors and add-ons (for recalculation)
- Initial status: "Draft"

FR-10: Quote Preview

- Professional quote layout with:
  - Your company branding
  - Quote number and date
  - Customer information
  - Service description
  - Itemized pricing breakdown
  - Total amount
  - Validity period
  - Terms and conditions

FR-11: Quote Actions

- Save as Draft
- Mark as Sent (updates status, records timestamp)
- Export to PDF
- Send via Email
- Copy shareable link (future)

8.3 Quote Management

FR-12: Quote List View

- Display all quotes in table/card view
- Columns: Quote #, Customer, Service, Total, Status, Date, Actions
- Filter by: Status, Date Range, Service, Customer
- Sort by: Date, Amount, Status
- Search by: Quote #, Customer Name

FR-13: Quote Detail View

- View complete quote information
- See all line items and breakdown
- View customer details
- Status history (when created, sent, accepted, etc.)
- Notes/comments

FR-14: Quote Status Management

- Update status: Draft → Sent → Accepted/Rejected
- Mark as Expired (automated based on valid_until date)
- Reopen/duplicate expired quotes

FR-15: Quote Editing

- Edit draft quotes
- Create new version of sent quotes (preserves original)
- Update customer information
- Add internal notes

8.4 Admin Panel

FR-16: Service Management

- CRUD operations on services
- Set base price, category, description
- Activate/deactivate services
- Reorder services (display_order)

FR-17: Pricing Factor Management

- Add new factors to services
- Configure factor properties:
  - Name, type, required/optional
  - Help text, validation rules
  - Common vs service-specific
- Reorder factors
- Deactivate factors (preserve in old quotes)

FR-18: Factor Option Management

- Add/edit/delete options for each factor
- Set option label, value, price impact
- Configure impact type (fixed/percentage/multiplier)
- For range-type factors: set min/max values
- Activate/deactivate options

FR-19: Factor Dependency Configuration

- Define when factors should show/hide
- Set dependency conditions
- Test dependencies before saving

FR-20: Entity Type Management

- CRUD operations on business entity types
- Set entity-specific pricing modifiers per service
- Configure modifier type (fixed/percentage/multiplier)

FR-21: Add-on Management

- Create global and service-specific add-ons
- Set pricing (fixed or percentage)
- Link add-ons to services
- Set service-specific override prices

FR-22: Pricing Preview/Test

- "Test Calculator" mode in admin
- Validate pricing logic before publishing
- See how changes affect example quotes

FR-23: Audit Trail

- Log all pricing changes (who, what, when)
- Version history for services and pricing rules
- Ability to view historical pricing

8.5 Reporting & Analytics (MVP - Basic)

FR-24: Quote Analytics Dashboard

- Total quotes generated (by period)
- Quote conversion rate
- Average quote value
- Quote distribution by service
- Quote status breakdown (pie chart)

FR-25: Service Performance

- Most quoted services
- Most selected add-ons
- Average price by service
- Common factor combinations

---

9. Technical Architecture

9.1 Technology Stack

Frontend:

- **Framework:** React 18+ with TypeScript
- **State Management:** React Context API + Reducer (or Redux Toolkit for complex state)
- **UI Components:** Material-UI (MUI) or Ant Design
- **Form Management:** React Hook Form + Yup validation
- **HTTP Client:** Axios
- **PDF Generation:** jsPDF or React-PDF
- **Charts/Graphs:** Recharts or Chart.js

Backend:

- **Runtime:** Node.js 18+
- **Framework:** Express.js
- **Language:** TypeScript
- **ORM:** Prisma or TypeORM
- **Validation:** Joi or Zod
- **Authentication:** JWT (future - Phase 2)

Database:

- **Primary:** PostgreSQL 15+
- **Connection Pooling:** pg-pool or Prisma connection pool

DevOps & Infrastructure:

- **Containerization:** Docker & Docker Compose
- **API Documentation:** Swagger/OpenAPI
- **Testing:** Jest + Supertest (backend), React Testing Library (frontend)
- **Linting:** ESLint + Prettier
- **Version Control:** Git

Deployment (Future):

- **Frontend:** Vercel, Netlify, or AWS S3 + CloudFront
- **Backend:** AWS ECS, DigitalOcean, or Heroku
- **Database:** AWS RDS PostgreSQL or managed PostgreSQL service

9.2 System Architecture

```
┌──────────────────────────────────────────────────────────────┐
│                        Client Layer                           │
│  ┌────────────────┐  ┌────────────────┐  ┌──────────────┐   │
│  │   Calculator   │  │   Quote Mgmt   │  │  Admin Panel │   │
│  │   Interface    │  │   Interface    │  │   Interface  │   │
│  └────────────────┘  └────────────────┘  └──────────────┘   │
│                     React Application                         │
└──────────────────────────────────────────────────────────────┘
                             │
                       HTTP/REST API
                             │
┌──────────────────────────────────────────────────────────────┐
│                      Application Layer                        │
│  ┌───────────────────────────────────────────────────────┐   │
│  │                 Express.js REST API                    │   │
│  │  ┌─────────────┐  ┌─────────────┐  ┌──────────────┐  │   │
│  │  │   Service   │  │    Quote    │  │    Admin     │  │   │
│  │  │ Controllers │  │ Controllers │  │ Controllers  │  │   │
│  │  └─────────────┘  └─────────────┘  └──────────────┘  │   │
│  │  ┌─────────────┐  ┌─────────────┐  ┌──────────────┐  │   │
│  │  │   Pricing   │  │    Quote    │  │ Config Mgmt  │  │   │
│  │  │   Engine    │  │   Service   │  │   Service    │  │   │
│  │  └─────────────┘  └─────────────┘  └──────────────┘  │   │
│  └───────────────────────────────────────────────────────┘   │
│                    Node.js + TypeScript                        │
└──────────────────────────────────────────────────────────────┘
                             │
                         ORM Layer
                             │
┌──────────────────────────────────────────────────────────────┐
│                         Data Layer                            │
│                    PostgreSQL Database                         │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐       │
│  │  Services    │  │  Quotes      │  │  Customers   │       │
│  │  Factors     │  │  Line Items  │  │  Entity Types│       │
│  │  Options     │  │  Add-ons     │  │  Modifiers   │       │
│  └────────────┘  └────────────┘  └────────────┘    │
└──────────────────────────────────────────────────────┘
```

9.3 Application Components

Backend Services:

1. **Pricing Engine Service**
    - Core price calculation logic
    - Factor evaluation
    - Dependency resolution
    - Validation
2. **Quote Management Service**
    - Quote CRUD operations
    - Quote number generation
    - Status management
    - PDF export
3. **Configuration Management Service**
    - Service/factor/option CRUD
    - Dependency management
    - Validation rules
4. **Customer Service**
    - Customer CRUD
    - Duplicate detection

Frontend Components:

1. **Calculator Module**
    - ServiceSelector
    - DynamicForm (renders factors based on service)
    - PriceBreakdown
    - AddOnSelector
    - QuotePreview
2. **Quote Management Module**
    - QuoteList
    - QuoteDetail
    - QuoteFilter
    - StatusManager
3. **Admin Module**

    - ServiceManager
    - FactorManager
    - OptionManager
    - DependencyBuilder
    - EntityTypeManager
    - AddOnManager

9.4 API Design Principles

- **RESTful:** Follow REST conventions
- **Versioning:** Use `/api/v1` prefix
- **Stateless:** No server-side sessions (use JWT for auth in future)
- **JSON:** All requests/responses in JSON
- **Error Handling:** Consistent error response format
- **Pagination:** For list endpoints
- **Filtering:** Query parameters for filtering
- **Validation:** Request validation at API layer

---

10. API Specifications

10.1 API Endpoints Overview

Base URL: `http://localhost:3000/api/v1`

Endpoint Categories:

1. Services API (`/services`)
2. Pricing Calculator API (`/calculator`)
3. Quotes API (`/quotes`)
4. Customers API (`/customers`)
5. Admin Configuration API (`/admin`)

10.2 Services API

GET /api/v1/services

Get all active services

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "name": "Bookkeeping",
      "description": "Monthly bookkeeping services...",
      "base_price": 200.00,
      "category": "Accounting",
      "display_order": 1
    }
  ]
}
```

GET /api/v1/services/:id

Get service details with pricing factors

Response:

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "name": "Bookkeeping",
    "description": "...",
    "base_price": 200.00,
    "pricing_factors": [
      {
        "id": "uuid",
        "factor_name": "Number of Transactions",
        "factor_type": "range",
        "is_required": true,
        "help_text": "Average monthly transactions",
        "options": [
          {
            "id": "uuid",
            "option_label": "0-1000 transactions",
            "option_value": "0-1000",
            "price_impact": 0.00,
            "price_impact_type": "fixed",
            "min_range": 0,
            "max_range": 1000
          }
        ]
      }
    ],
    "available_addons": [
      {
        "id": "uuid",
        "addon_name": "Rush Service",
        "price": 200.00,
        "price_type": "fixed"
      }
    ]
  }
}
```

10.3 Pricing Calculator API

POST /api/v1/calculator/calculate

Calculate price for given selections

Request:

```json
{
  "service_id": "uuid",
  "entity_type_id": "uuid",
  "selected_factors": {
    "factor_uuid_1": "selected_value_1",
    "factor_uuid_2": "selected_value_2"
  },
  "selected_addons": ["addon_uuid_1", "addon_uuid_2"]
}
```

Response:

```json
{
  "success": true,
  "data": {
    "total_price": 1000.00,
    "breakdown": {
      "base_price": 200.00,
      "entity_type": "C-Corp",
      "entity_modifier": 1.5,
      "price_after_entity": 300.00,
      "factors": [
        {
          "name": "Transactions (1001-1500)",
          "impact": 150.00,
          "running_total": 450.00
        }
      ],
      "subtotal": 800.00,
      "addons": [
        {
          "name": "Rush Service",
          "price": 200.00
        }
      ],
      "total": 1000.00
    }
  }
}
```

GET /api/v1/calculator/entity-types

Get all business entity types

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "entity_name": "Sole Proprietor",
      "entity_code": "sole_proprietor",
      "description": "Single-owner unincorporated business"
    }
  ]
}
```

10.4 Quotes API

POST /api/v1/quotes

Create a new quote

Request:

```
{
  "customer": {
    "company_name": "Acme Inc.",
    "contact_name": "John Smith",
    "email": "john@acme.com",
    "phone": "555-0100",
    "address": "123 Main St"
  },
  "service_id": "uuid",
  "entity_type_id": "uuid",
  "selected_factors": {...},
  "selected_addons": [...],
  "notes": "Client requested rush turnaround",
  "valid_until": "2025-12-31"
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "quote_number": "Q-2025-001",
    "customer_id": "uuid",
    "service_id": "uuid",
    "total_price": 1000.00,
    "status": "draft",
    "valid_until": "2025-12-31",
    "price_breakdown": {...},
    "created_at": "2025-11-18T10:00:00Z"
  }
}
```

GET /api/v1/quotes

Get all quotes with filters

Query Parameters:

- `status`: filter by status (draft, sent, accepted, rejected, expired)
- `service_id`: filter by service
- `customer_id`: filter by customer
- `from_date`: filter by creation date (start)
- `to_date`: filter by creation date (end)
- `page`: pagination page number (default: 1)
- `limit`: items per page (default: 20)

Response:

```
{
  "success": true,
  "data": {
    "quotes": [...],
    "pagination": {
      "page": 1,
      "limit": 20,
      "total": 45,
      "total_pages": 3
    }
  }
}
```

GET /api/v1/quotes/:id

Get quote details

PUT /api/v1/quotes/:id

Update quote (draft only)

PATCH /api/v1/quotes/:id/status

Update quote status

Request:

```
{
  "status": "sent"
}
```

GET /api/v1/quotes/:id/pdf

Generate PDF export (returns PDF file)

10.5 Customers API

POST /api/v1/customers

Create customer

GET /api/v1/customers

List customers (with search/pagination)

GET /api/v1/customers/:id

Get customer details (includes quote history)

PUT /api/v1/customers/:id

Update customer

10.6 Admin Configuration API

Services Management

POST /api/v1/admin/services Create new service

PUT /api/v1/admin/services/:id Update service

DELETE /api/v1/admin/services/:id Deactivate service (soft delete)

Pricing Factors Management

POST /api/v1/admin/services/:serviceId/factors Add pricing factor to service

Request:

```
{
  "factor_name": "Report Frequency",
  "factor_type": "select",
  "is_required": true,
  "help_text": "How often do you need reports?",
  "display_order": 3
}
```

PUT /api/v1/admin/factors/:id Update factor

DELETE /api/v1/admin/factors/:id Deactivate factor

Factor Options Management

POST /api/v1/admin/factors/:factorId/options Add option to factor

Request:

```
{
  "option_label": "Monthly",
  "option_value": "monthly",
  "price_impact": 350.00,
  "price_impact_type": "fixed",
  "display_order": 4
}
```

PUT /api/v1/admin/factor-options/:id Update option

DELETE /api/v1/admin/factor-options/:id Deactivate option

Factor Dependencies

POST /api/v1/admin/factors/:factorId/dependencies Create dependency rule

Request:

```
{
  "depends_on_factor_id": "uuid",
  "depends_on_value": "11-50",
  "dependency_type": "show_if"
}
```

Entity Types & Modifiers

POST /api/v1/admin/entity-types Create business entity type

POST /api/v1/admin/services/:serviceId/entity-modifiers Set entity pricing modifier for service

Request:

```
{
  "entity_type_id": "uuid",
  "modifier_value": 1.5,
  "modifier_type": "multiplier"
}
```

Add-ons Management

POST /api/v1/admin/addons Create add-on

Request:

```json
{
  "addon_name": "Rush Service",
  "description": "48-hour turnaround",
  "price": 200.00,
  "price_type": "fixed",
  "is_global": true
}
```

POST /api/v1/admin/services/:serviceId/addons Link add-on to service (for non-global add-ons)

10.7 Error Response Format

Standard Error Response:

```json
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Required factors missing",
    "details": [
      {
        "field": "transaction_count",
        "message": "Transaction count is required"
      }
    ]
  }
}
```

HTTP Status Codes:

- 200: Success
- 201: Created
- 400: Bad Request (validation error)
- 404: Not Found
- 409: Conflict (e.g., duplicate)
- 500: Internal Server Error

11. User Workflows

11.1 Sales Team: Generate Quote

Scenario: Sales rep creates quote for prospect

Steps:

1. Navigate to Calculator page
2. Select service: "Bookkeeping"
3. Form dynamically loads with factors:
   - Business Entity Type → Select "C-Corp"
   - Number of Transactions → Select "1001-1500"
   - Report Frequency → Select "Monthly"
   - Accounting Software → Select "QuickBooks Online"
4. Price updates in real-time: **$800.00**
5. Add optional add-on: "Rush Service (+$200)"
6. Final price: **$1,000.00**
7. Review itemized breakdown
8. Click "Generate Quote"
9. Enter customer information:
   - Company: Acme Inc.
   - Contact: John Smith
   - Email: john@acme.com
   - Phone: 555-0100
10. Add notes: "Client needs year-end closing by Dec 15"
11. Set expiration: 30 days
12. Preview quote
13. Actions:
    - Save as Draft (for later)
    - Send via Email (marks as "Sent")
    - Export PDF (download for manual sending)

11.2 Sales Team: Manage Quotes

Scenario: Track and follow up on quotes

Steps:

1. Navigate to Quotes page
2. View quote list (table view)
3. Filter by:
   - Status: "Sent" (to find pending quotes)
   - Date: Last 30 days
4. See columns: Quote #, Customer, Service, Amount, Status, Date
5. Click on quote to view details
6. Actions available:
   - Update status (Accepted/Rejected)
   - Resend email

    - Download PDF
    - Duplicate (create new version)
    - Add notes
7. Mark quote as "Accepted" when client confirms
8. Analytics: View conversion rate dashboard

11.3 Admin: Add New Service

Scenario: Admin adds "Tax Preparation" service

Steps:

1. Navigate to Admin Panel → Services
2. Click "Add New Service"
3. Fill form:
    - Name: "Tax Preparation"
    - Description: "Business tax return preparation..."
    - Base Price: $300
    - Category: "Tax"
4. Save service
5. Navigate to "Configure Pricing Factors"
6. Add Factor 1:
    - Name: "Number of Tax Forms"
    - Type: Range
    - Required: Yes
    - Help Text: "Estimated number of forms needed"
7. Add Options for Factor 1:
    - "1-5 forms" → +$0
    - "6-10 forms" → +$200
    - "11+ forms" → +$500
8. Add Factor 2:
    - Name: "Multi-State Filing"
    - Type: Boolean
    - Required: No
9. Add Options for Factor 2:
    - "No" → +$0
    - "Yes" → +$300
10. Set Entity Type Modifiers:
    - Sole Proprietor → 1.0x (base)
    - Partnership → 1.2x
    - C-Corp → 1.5x
11. Link global add-ons (Rush Service, Priority Support)
12. Test in calculator preview
13. Activate service

11.4 Admin: Update Pricing

Scenario: Increase pricing due to inflation

Steps:

1. Navigate to Admin Panel → Services
2. Select "Bookkeeping"
3. Update base price: $200 → $250
4. Navigate to Pricing Factors
5. Select "Monthly Reports" option
6. Update price: $350 → $400
7. Save changes
8. System logs pricing change with timestamp
9. New quotes use updated pricing
10. Old quotes preserve original pricing

11.5 Admin: Configure Factor Dependencies

Scenario: Show "Multi-State Operations" only for businesses with 50+ employees

Steps:

1. Navigate to Admin Panel → Factor Dependencies
2. Select Service: "Payroll"
3. Select Dependent Factor: "Multi-State Operations"
4. Configure Rule:
    - Depends On: "Number of Employees"
    - Condition: "show_if"
    - Values: ["51-100", "100+"]
5. Save dependency
6. Test in calculator:
    - Select 1-10 employees → "Multi-State" hidden
    - Select 51-100 employees → "Multi-State" appears

---

12. Admin Configuration

12.1 Admin Panel Structure

Navigation Menu:

```
Admin Panel
├── Dashboard
│   └── System overview, recent changes
├── Services
│   ├── All Services
│   ├── Add New Service
│   └── Service Categories
├── Pricing Configuration
│   ├── Pricing Factors
│   ├── Factor Options
│   └── Factor Dependencies
├── Business Entity Types
```

```
│   ├── Entity Types List
│   └── Entity Pricing Modifiers
├── Add-ons
│   ├── All Add-ons
│   ├── Add New Add-on
│   └── Service Add-on Links
├── Pricing History
│   └── Audit Log
└── Settings
    └── System Configuration
```

12.2 Service Configuration Workflow

Adding a New Service:

1. **Basic Information**

    - Service name
    - Description (rich text)
    - Category (dropdown or create new)
    - Base price
    - Display order

2. **Pricing Factors**

    - Add multiple factors
    - For each factor:
        - Factor name
        - Factor type (select/range/numeric/boolean)
        - Required or optional
        - Help text/tooltip
        - Display order
    - Reorder factors (drag-and-drop)

3. **Factor Options**

    - For each factor, add options
    - For each option:
        - Label (what users see)
        - Value (stored in DB)
        - Price impact (amount)
        - Impact type (fixed/percentage/multiplier)
        - For ranges: min/max values
    - Reorder options

4. **Entity Type Pricing**

    - Select which entity types apply
    - For each entity type:
        - Set modifier value

        - Set modifier type
    - Preview impact on sample quote

5. **Add-ons**

    - Select from global add-ons
    - Create service-specific add-ons
    - Set override pricing if needed

6. **Dependencies (Optional)**

    - Define conditional logic
    - Test dependency rules

7. **Preview & Test**

    - Test calculator with sample inputs
    - Verify pricing calculations
    - Check all factor combinations

8. **Activate**

    - Set service as active
    - Publish to calculator

12.3 Bulk Operations

Bulk Pricing Updates:

- Select multiple services
- Apply percentage increase/decrease
- Preview impact before confirming
- Log all changes

Import/Export:

- Export service configurations to JSON
- Import configurations (for multi-environment setup)
- Backup pricing rules

12.4 Validation & Safeguards

Configuration Validation:

- Prevent circular dependencies
- Validate pricing logic before activation
- Warn if required factors have no options
- Check for orphaned options

Change Confirmation:

- Require confirmation for price changes

- Show impact summary (e.g., "This will affect 23 active quotes")
- Option to notify sales team of pricing changes

Version Control:

- Track all configuration changes
- Who made the change
- What was changed
- When it was changed
- Option to rollback (future)

---

13. MVP Scope

13.1 Phase 1: MVP Features (Weeks 1-8)

Core Calculator (Weeks 1-3)

- ✅ Service selection interface
- ✅ Dynamic form rendering based on service
- ✅ Real-time price calculation
- ✅ Itemized price breakdown display
- ✅ Add-on selection
- ✅ Form validation
- ✅ Basic responsive UI

Quote Generation (Weeks 3-4)

- ✅ Customer information capture
- ✅ Quote creation and storage
- ✅ Quote number generation
- ✅ Basic quote preview
- ✅ Save as draft functionality
- ✅ Simple PDF export

Quote Management (Week 4-5)

- ✅ Quote list view
- ✅ Quote detail view
- ✅ Status management (draft/sent/accepted/rejected)
- ✅ Basic filtering (by status, date)
- ✅ Search by quote number or customer name

Admin Panel - Basic (Weeks 5-7)

- ✅ Service CRUD
- ✅ Pricing factor management
- ✅ Factor option management
- ✅ Entity type configuration
- ✅ Entity pricing modifiers

- ✅ Global add-on management
- ✅ Basic validation

Database & Backend (Weeks 1-6)

- ✅ PostgreSQL schema implementation
- ✅ All migrations
- ✅ REST API endpoints
- ✅ Pricing calculation engine
- ✅ ORM integration (Prisma/TypeORM)

Infrastructure (Weeks 6-8)

- ✅ Docker setup
- ✅ Environment configuration
- ✅ API documentation (Swagger)
- ✅ Basic error handling
- ✅ Logging

13.2 Out of Scope for MVP

Deferred to Phase 2:

- ❌ User authentication & authorization
- ❌ Role-based access control
- ❌ Email integration (SMTP)
- ❌ Advanced PDF customization
- ❌ Quote approval workflows
- ❌ Advanced analytics dashboard
- ❌ Factor dependencies (start with simple logic)
- ❌ Multi-currency support
- ❌ Internationalization (i18n)
- ❌ Client self-service portal
- ❌ Online quote acceptance/e-signature
- ❌ Payment integration
- ❌ CRM integration
- ❌ Automated quote expiration emails
- ❌ A/B testing different pricing models
- ❌ Mobile app

Technical Debt Acceptable in MVP:

- Limited test coverage (focus on critical paths)
- Basic UI/UX (refine in Phase 2)
- Manual deployment (CI/CD in Phase 2)
- Single-tenant architecture (multi-tenant in Phase 3)

---

14. Future Enhancements

14.1 Phase 2: Enhanced Features (Weeks 9-16)

Authentication & Authorization

- User login (JWT-based)
- Role-based access control:
    - Admin (full access)
    - Sales Manager (manage all quotes)
    - Sales Rep (manage own quotes)
    - Viewer (read-only)
- User management interface

Advanced Quote Management

- Quote approval workflows
- Quote versioning (track changes)
- Quote comparison tool
- Bulk quote operations
- Quote templates
- Email integration:
    - Send quotes via email
    - Email templates
    - Tracking (open/click rates)

Factor Dependencies

- Full implementation of dependency engine
- Complex conditions (AND/OR logic)
- Dependency testing interface
- Visual dependency mapper

Enhanced Reporting

- Advanced analytics dashboard
- Revenue forecasting
- Win/loss analysis
- Popular configuration analytics
- Pricing optimization insights
- Custom report builder

PDF Enhancements

- Customizable templates
- Company branding (logo, colors)
- Multiple template options
- Attach terms & conditions
- Digital signatures

Notifications

- Quote expiration alerts

- Status change notifications
- Pricing update alerts
- Activity feed

14.2 Phase 3: Advanced Capabilities

Client Self-Service Portal

- Public calculator (limited features)
- Client quote requests
- Quote acceptance online
- Client dashboard
- Historical quotes

E-Signature Integration

- DocuSign/HelloSign integration
- Quote acceptance workflow
- Signed contract storage

Payment Integration

- Stripe/PayPal integration
- Deposit collection
- Payment plans
- Invoice generation

CRM Integration

- Salesforce connector
- HubSpot integration
- Automatic contact sync
- Opportunity tracking

Advanced Pricing

- Dynamic pricing (based on demand, seasonality)
- Customer-specific pricing
- Contract-based pricing
- Volume discount automation
- Promotional pricing campaigns

Multi-Tenant Architecture

- Support multiple companies
- Isolated data
- Company-specific branding
- Per-tenant configuration

Mobile Application

- iOS/Android apps

- Offline calculator
- Push notifications
- Mobile quote generation

14.3 Phase 4: Enterprise Features

API Access

- Public API for partners
- Webhooks
- API rate limiting
- Developer portal

Advanced Analytics

- Machine learning price optimization
- Predictive quote conversion
- Customer lifetime value analysis
- Churn prediction

Compliance & Audit

- SOC 2 compliance
- GDPR compliance
- Advanced audit trails
- Data retention policies
- Automated backups

White-Label Solution

- Fully customizable branding
- Custom domain support
- Reseller program

---

15. Appendices

Appendix A: Example Pricing Calculations

Example 1: Simple Bookkeeping Quote

Input:

- Service: Bookkeeping
- Entity: Sole Proprietor
- Transactions: 0-1000
- Reports: Yearly
- Software: QuickBooks Online
- Add-ons: None

Calculation:

```
Base Price: $200
Entity Modifier (Sole Proprietor 1.0x): $200
Transactions (0-1000): +$0 = $200
Reports (Yearly): +$0 = $200
Software (QBO): +$0 = $200
Add-ons: $0
TOTAL: $200
```

Example 2: Complex Bookkeeping Quote

Input:

- Service: Bookkeeping
- Entity: C-Corp
- Transactions: 2001-3000
- Reports: Monthly
- Software: Manual/Spreadsheet
- Add-ons: Rush Service, Priority Support

Calculation:

```
Base Price: $200
Entity Modifier (C-Corp 1.5x): $200 × 1.5 = $300
Transactions (2001-3000): $300 + $500 = $800
Reports (Monthly): $800 + $350 = $1,150
Software (Manual): $1,150 + $200 = $1,350
Add-ons:
  - Rush Service: +$200
  - Priority Support: +$150
TOTAL: $1,700
```

Example 3: Payroll Service

Input:

- Service: Payroll
- Entity: LLC
- Employees: 11-50
- Payroll Frequency: Bi-Weekly
- Multi-State: Yes
- Add-ons: None

Calculation:

```
Base Price: $150
Entity Modifier (LLC 1.2x): $150 × 1.2 = $180
Employees (11-50): $180 + $200 = $380
```

```
Frequency (Bi-Weekly): $380 + $100 = $480
Multi-State (Yes): $480 + $150 = $630
TOTAL: $630
```

Appendix B: Database Migration Strategy

Migration Approach:

1. Use migration tool (Prisma Migrate or TypeORM Migrations)
2. Version control all migrations
3. Never modify existing migrations (create new ones)
4. Test migrations on staging before production
5. Include rollback scripts

Initial Migration:

```sql
-- migrations/001_initial_schema.sql
-- Creates all tables defined in Section 6.4
```

Sample Future Migration:

```sql
-- migrations/002_add_service_tags.sql
ALTER TABLE services ADD COLUMN tags JSONB DEFAULT '[]';
CREATE INDEX idx_services_tags ON services USING GIN (tags);
```

Appendix C: Sample API Request/Response

Full Quote Generation Flow:

1. **Get Service Details**

```
GET /api/v1/services/550e8400-e29b-41d4-a716-446655440000
```

2. **Calculate Price**

```
POST /api/v1/calculator/calculate
Content-Type: application/json

{
  "service_id": "550e8400-e29b-41d4-a716-446655440000",
  "entity_type_id": "660e8400-e29b-41d4-a716-446655440000",
  "selected_factors": {
    "770e8400-e29b-41d4-a716-446655440000": "1001-1500",
    "880e8400-e29b-41d4-a716-446655440000": "monthly"
  },
  "selected_addons": ["990e8400-e29b-41d4-a716-446655440000"]
}
```

3. **Create Quote**

```
POST /api/v1/quotes
Content-Type: application/json

{
  "customer": {
    "company_name": "Acme Inc.",
    "contact_name": "John Smith",
    "email": "john@acme.com",
    "phone": "555-0100"
  },
  "service_id": "550e8400-e29b-41d4-a716-446655440000",
  "entity_type_id": "660e8400-e29b-41d4-a716-446655440000",
  "selected_factors": {...},
  "selected_addons": [...],
  "total_price": 1000.00,
  "price_breakdown": {...}
}
```

Appendix D: UI Component Mockups (Text Description)

Calculator Page Layout:

```
┌─────────────────────────────────────────────────────┐
│ RatePro - Financial Services Pricing Calculator     │
├─────────────────────────────────────────────────────┤
│                                                     │
│  Step 1: Select Service                             │
│  [Dropdown: Choose a service...]                    │
│                                                     │
│  Step 2: Business Information                       │
│  [Dropdown: Business Entity Type]                   │
│                                                     │
│  Step 3: Service Configuration                      │
│  ┌─────────────────┐  ┌─────────────────────┐       │
│  │                 │  │ Price Breakdown     │       │
│  │ [Factor 1]      │  │ ─────────────────── │       │
│  │ [Factor 2]      │  │ Base Price:    $200 │       │
│  │ [Factor 3]      │  │ Entity (1.5x): $100 │       │
│  │                 │  │ Factors:       $500 │       │
│  │ Add-ons:        │  │ Add-ons:       $200 │       │
│  │ ☐ Rush Service  │  │ ─────────────────── │       │
│  │ ☐ Priority Supp │  │ TOTAL:       $1,000 │       │
│  │                 │  │                     │       │
│  │ [Generate Quote]│  │  [View Details ▼]      │  │
│  └─────────────────┘  └────────────────────────┘  │
└───────────────────────────────────────────────────┘
```

Admin - Service Configuration:

```
┌───────────────────────────────────────────────────┐
│ Admin Panel > Services > Edit: Bookkeeping        │
├───────────────────────────────────────────────────┤
│                                                   │
│  Basic Information                               │
│  Name: [Bookkeeping                         ]    │
│  Description: [_____________________________ ]  │
│  Base Price: [$200.00        ]                  │
│  Category: [Accounting ▼]                       │
│                                                   │
│  Pricing Factors                     [+ Add Factor]│
│  ┌──────────────────────────────────────────────┐ │
│  │ ≡ Number of Transactions          [Edit]     │ │
│  │   Type: Range | Required: Yes                │ │
│  │   Options: 0-1000 ($0), 1001-1500 ($150)...  │ │
│  ├──────────────────────────────────────────────┤ │
│  │ ≡ Report Frequency                [Edit]     │ │
│  │   Type: Select | Required: Yes               │ │
│  │   Options: Yearly ($0), Monthly ($350)...    │ │
│  └──────────────────────────────────────────────┘ │
│                                                   │
│  [Save Changes]  [Preview Calculator]  [Cancel]  │
└───────────────────────────────────────────────────┘
```

Appendix E: Glossary

Terms:

- **Factor:** A pricing variable that affects the final price (e.g., transaction volume, report frequency)
- **Factor Option:** A specific value/choice for a factor (e.g., "Monthly" for Report Frequency)
- **Entity Type:** Business structure (Sole Proprietor, LLC, Corporation, etc.)
- **Entity Modifier:** Pricing adjustment based on entity type
- **Add-on:** Optional service or feature that can be added to a quote
- **Price Impact:** The amount a factor option or add-on changes the price
- **Impact Type:** How price impact is applied (fixed amount, percentage, multiplier)
- **Common Factor:** A factor that applies to all services (e.g., Entity Type)
- **Service-Specific Factor:** A factor unique to one service (e.g., transaction volume for bookkeeping)
- **Factor Dependency:** A rule that shows/hides factors based on other selections
- **Quote:** A generated price estimate for a customer
- **Quote Line Item:** Individual component of a quote's price breakdown

---

Document Change Log

Version	Date	Author	Changes
1.0	2025-11-18	Product Team	Initial PRD creation with complete schema design

---

End of Product Requirements Document