PRODUCT REQUIREMENTS DOCUMENT

InvoiceForge Pro

Desktop Invoice Generator

Version 1.0.0

January 8, 2025

Document Status: APPROVED

Document Owner	Product Team
Target Release	Q1 2025
Platform	Windows 10/11 (64-bit)
Technology Stack	Electron + React + Node.js
Last Updated	January 8, 2025

TABLE OF CONTENTS

1. Executive Summary

InvoiceForge Pro is a standalone Windows desktop application designed to instantly generate professional, customizable invoices from simple text input. The application eliminates the complexity of traditional invoicing software by providing an intuitive text-based interface that parses natural language and structured data to produce publication-ready invoice documents.

> **Key Value Proposition**
>
> Transform plain text into professionally formatted invoices in under 10 seconds, with zero learning curve and complete offline functionality.

Core Capabilities

- Parse structured and semi-structured text input to extract invoice data
- Generate professional .docx and .pdf invoice documents
- Support multiple invoice templates with full customization
- Maintain client and service databases locally
- Operate completely offline with no cloud dependencies
- Export to multiple formats including Word, PDF, and HTML

Success Metrics

Metric	Target	Priority
Invoice generation time	< 10 seconds	P0
Application startup time	< 3 seconds	P0
Text parsing accuracy	> 95%	P0
User task completion rate	> 90%	P1
Installer size	< 150 MB	P2

2. Problem Statement

Current Pain Points

Freelancers, small business owners, and independent contractors face significant friction when creating professional invoices:

1. Existing solutions require extensive setup and configuration
2. Cloud-based tools have subscription costs and require internet connectivity
3. Manual document creation is time-consuming and error-prone
4. Template-based solutions lack flexibility for custom line items

5. Most invoicing software has steep learning curves
6. Data privacy concerns with cloud-hosted financial information

Market Gap

There is no widely available solution that combines:

- Instant text-to-invoice conversion
- Professional output quality
- Complete offline functionality
- Zero configuration requirements
- One-time purchase (no subscription)

Opportunity

InvoiceForge Pro addresses this gap by providing a lightweight, fast, offline-capable application that transforms plain text into professional invoices instantly. The target market includes 59 million freelancers in the US alone, plus millions of small business owners globally.

3. Product Vision & Goals

Vision Statement

> **Product Vision**
>
> To become the fastest, simplest way to create professional invoices—enabling anyone to generate publication-ready financial documents from plain text in seconds, completely offline.

Strategic Goals

Goal	Description	Success Criteria
G1	Minimize time-to-invoice	< 10 seconds from text input to document
G2	Zero learning curve	New users create first invoice without documentation
G3	Professional output quality	Output indistinguishable from designer-created invoices
G4	Complete data privacy	100% offline operation, zero data transmission
G5	Cross-format compatibility	Export to DOCX, PDF, HTML with full fidelity

Scope Boundaries

In Scope (v1.0):

- Text-based invoice generation
- Template management system
- Client/service database
- Multi-format export
- Invoice history and search
- Basic customization options

Out of Scope (v1.0):

- Online payment integration
- Multi-user/team features
- Cloud sync or backup
- Mobile companion app
- Accounting software integration
- Email sending functionality

4. Target Users & Personas

Primary Persona: The Freelance Developer

Name	Alex Chen
Role	Independent Software Developer
Age	32
Technical Level	High - comfortable with text-based interfaces
Pain Points	Hates context-switching to invoicing tools, wants to stay in text/code mode
Goals	Generate invoices as fast as possible, maintain professional appearance
Invoicing Frequency	2-5 invoices per week

Secondary Persona: The Small Business Owner

Name	Maria Santos
Role	Owner, Graphic Design Studio
Age	45
Technical Level	Medium - uses standard business software
Pain Points	Subscription fatigue, wants simple one-time purchase tools
Goals	Professional invoices that match her brand, easy customization
Invoicing Frequency	10-20 invoices per month

Tertiary Persona: The Privacy-Conscious Consultant

Name	David Park
Role	Security Consultant
Age	38
Technical Level	Very High - security expert
Pain Points	Cannot use cloud tools due to client confidentiality requirements
Goals	100% offline solution, no data leaves his machine
Invoicing Frequency	5-10 invoices per month

5. Functional Requirements

5.1 Text Input & Parsing (FR-100)

The application shall accept text input and intelligently parse it to extract invoice data.

ID	Requirement	Priority	Status
FR-101	Accept multi-line text input via main text area	**P0**	Planned
FR-102	Parse key-value pairs (e.g., "Client: Acme Corp")	**P0**	Planned
FR-103	Parse tabular data for line items (description, qty, rate)	**P0**	Planned
FR-104	Support CSV-style input for bulk line items	**P1**	Planned
FR-105	Auto-detect and parse dates in multiple formats	**P1**	Planned
FR-106	Support natural language input (e.g., "30 hours at $50")	**P2**	Planned
FR-107	Validate parsed data and highlight errors/warnings	**P0**	Planned
FR-108	Support file drag-and-drop for .txt input	**P2**	Planned

Supported Input Formats

```
# Format 1: Key-Value Pairs
Invoice: INV-2025-001
Client: Delta Service Company
Date: 2025-01-15
Due: Net 30

# Format 2: Line Items (Tab or Pipe Separated)
Development Work | 40 hours | $50/hr
Design Services | 20 hours | $75/hr
Hosting | 1 month | $29

# Format 3: CSV Style
"Web Development","40","50"
"API Integration","20","60"

# Format 4: Natural Language
Bill Acme Corp for 40 hours of development at $50/hr
Add hosting fee of $29
Due in 30 days
```

5.2 Invoice Generation (FR-200)

ID	Requirement	Priority	Status
FR-201	Generate .docx files with professional formatting	**P0**	Planned
FR-202	Generate .pdf files via built-in converter	**P0**	Planned
FR-203	Auto-calculate line item totals, subtotals, tax, grand total	**P0**	Planned
FR-204	Auto-generate invoice numbers (configurable format)	**P0**	Planned
FR-205	Apply selected template to generated invoice	**P0**	Planned
FR-206	Support multiple currency symbols and formats	**P1**	Planned
FR-207	Support configurable tax rates (single and multiple)	**P1**	Planned
FR-208	Support discounts (percentage and fixed amount)	**P1**	Planned
FR-209	Include payment terms and methods in output	**P0**	Planned
FR-210	Generate HTML preview before export	**P1**	Planned

5.3 Template Management (FR-300)

ID	Requirement	Priority	Status
FR-301	Include 5+ pre-built professional templates	**P0**	Planned
FR-302	Allow template customization (colors, fonts, layout)	**P1**	Planned
FR-303	Support custom logo upload and positioning	**P0**	Planned
FR-304	Save custom templates for reuse	**P1**	Planned
FR-305	Import/export template configurations	**P2**	Planned
FR-306	Template preview in selection UI	**P1**	Planned

5.4 Data Management (FR-400)

ID	Requirement	Priority	Status
FR-401	Store client profiles (name, address, contact, tax ID)	**P0**	Planned
FR-402	Store service/product catalog with default rates	**P1**	Planned
FR-403	Auto-complete client names from database	**P1**	Planned
FR-404	Store invoice history with full text search	**P0**	Planned
FR-405	Track invoice status (draft, sent, paid, overdue)	**P1**	Planned
FR-406	Export all data to JSON/CSV for backup	**P1**	Planned

FR-407	Import data from JSON/CSV backup	**P1**	Planned
FR-408	Store user business profile (company info, logo, payment details)	**P0**	Planned

5.5 User Interface (FR-500)

ID	Requirement	Priority	Status
FR-501	Main window with text input area and live preview	**P0**	Planned
FR-502	Side panel for template selection	**P0**	Planned
FR-503	Settings panel for business profile configuration	**P0**	Planned
FR-504	Invoice history view with filters and search	**P1**	Planned
FR-505	Client management view (CRUD operations)	**P1**	Planned
FR-506	Keyboard shortcuts for common actions	**P1**	Planned
FR-507	Dark mode support	**P2**	Planned
FR-508	Syntax highlighting for text input	**P2**	Planned
FR-509	Real-time validation feedback	**P0**	Planned

6. Non-Functional Requirements

6.1 Performance (NFR-100)

ID	Requirement	Target	Priority
NFR-101	Application cold start time	< 3 seconds	**P0**
NFR-102	Invoice generation time (text to document)	< 5 seconds	**P0**
NFR-103	PDF conversion time	< 3 seconds	**P0**
NFR-104	Memory usage (idle)	< 200 MB	**P1**
NFR-105	Memory usage (active)	< 500 MB	**P1**
NFR-106	Database query response time	< 100 ms	**P1**
NFR-107	UI responsiveness (input lag)	< 50 ms	**P0**

6.2 Reliability (NFR-200)

ID	Requirement	Target
NFR-201	Crash-free sessions	> 99.5%
NFR-202	Data integrity (no corruption on unexpected shutdown)	100%
NFR-203	Auto-save draft invoices	Every 30 seconds
NFR-204	Graceful error handling with user-friendly messages	All errors

6.3 Compatibility (NFR-300)

ID	Requirement
NFR-301	Windows 10 (version 1903+) 64-bit support
NFR-302	Windows 11 (all versions) 64-bit support
NFR-303	Minimum screen resolution: 1280x720
NFR-304	High DPI display support (125%, 150%, 200% scaling)
NFR-305	Generated DOCX compatible with Word 2016+, LibreOffice 6+, Google Docs
NFR-306	Generated PDF compatible with all major PDF readers

6.4 Security (NFR-400)

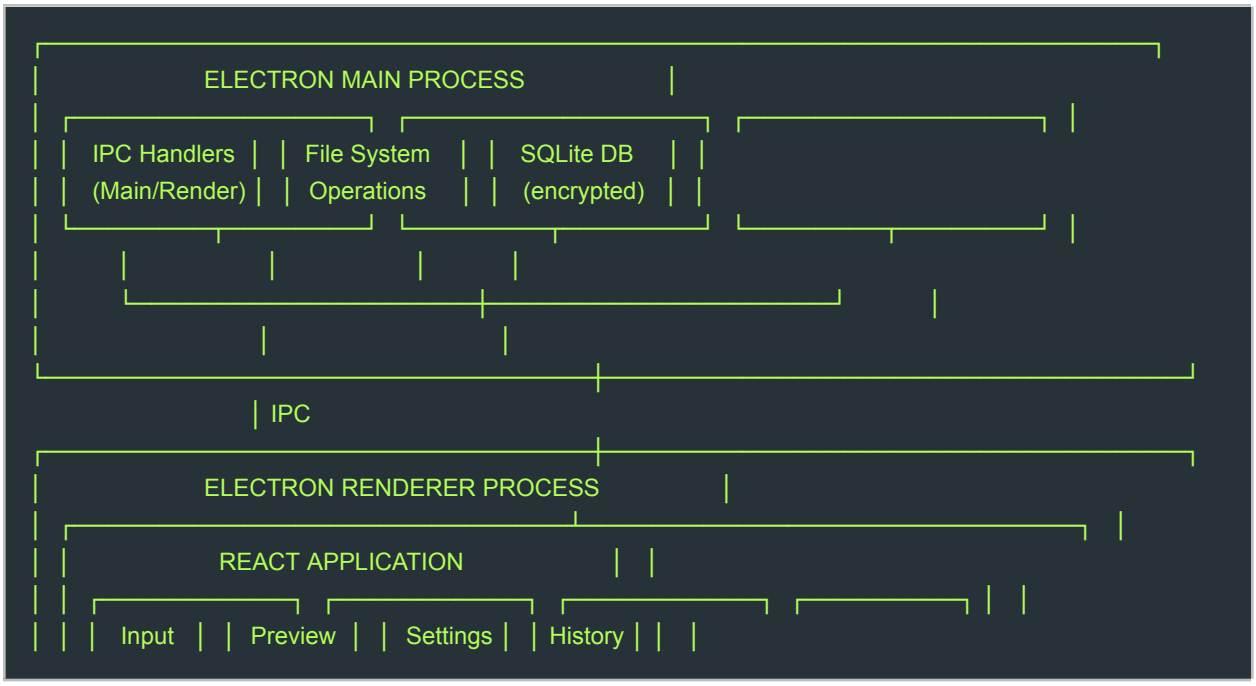
ID	Requirement
NFR-401	Zero network connections - complete offline operation
NFR-402	No telemetry or analytics data collection
NFR-403	Local database encryption option (AES-256)
NFR-404	Code-signed installer and executable
NFR-405	No admin privileges required for installation or operation

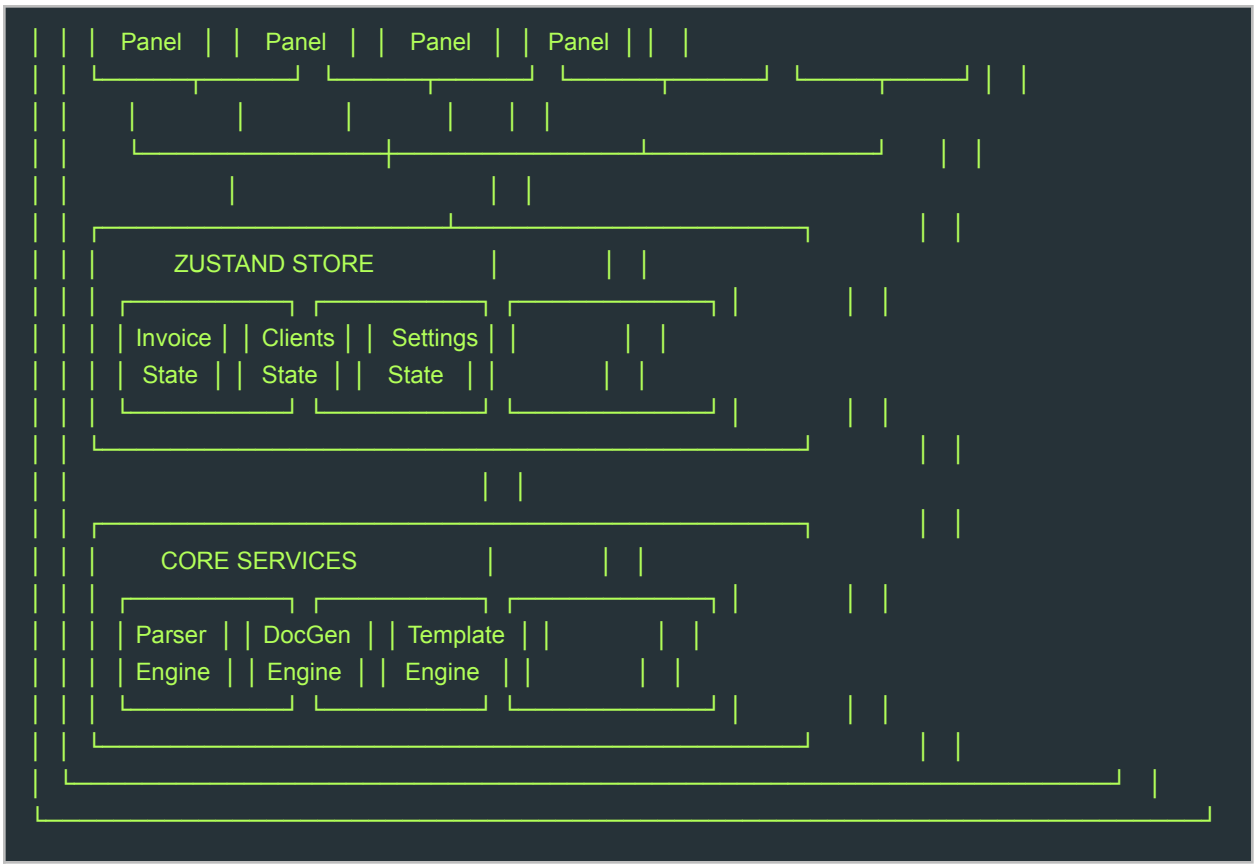
7. Technical Architecture

7.1 Technology Stack

Layer	Technology	Justification
Runtime	Electron 28.x	Cross-platform desktop framework, mature ecosystem
Frontend Framework	React 18.x	Component-based UI, excellent developer tooling
State Management	Zustand	Lightweight, simple API, good TypeScript support
Styling	Tailwind CSS + shadcn/ui	Rapid development, consistent design system
Language	TypeScript 5.x	Type safety, better maintainability
Database	SQLite + better-sqlite3	Embedded, zero-config, fast, reliable
Document Generation	docx (npm package)	Pure JS, no external dependencies
PDF Generation	Puppeteer (bundled)	High-fidelity HTML-to-PDF conversion
Build Tool	Vite + electron-builder	Fast builds, excellent Electron support
Testing	Vitest + Playwright	Fast unit tests, reliable E2E testing

7.2 Architecture Diagram





7.3 Directory Structure

```
invoiceforge-pro/
├── electron/
│   ├── main.ts              # Electron main process entry
│   ├── preload.ts           # Context bridge for IPC
│   ├── ipc/
│   │   ├── file-handlers.ts    # File system operations
│   │   ├── db-handlers.ts      # Database operations
│   │   └── pdf-handlers.ts     # PDF generation
│   └── utils/
│       └── paths.ts         # App paths management
├── src/
│   ├── main.tsx             # React entry point
│   ├── App.tsx              # Root component
│   ├── components/
│   │   ├── ui/              # shadcn/ui components
│   │   ├── input/
│   │   │   ├── TextInputPanel.tsx
│   │   │   └── InputToolbar.tsx
│   │   ├── preview/
```

```
│   │   │   ├── InvoicePreview.tsx
│   │   │   └── PreviewControls.tsx
│   │   ├── templates/
│   │   │   ├── TemplateSelector.tsx
│   │   │   └── TemplateEditor.tsx
│   │   ├── settings/
│   │   │   ├── BusinessProfile.tsx
│   │   │   └── Preferences.tsx
│   │   └── history/
│   │       ├── InvoiceList.tsx
│   │       └── InvoiceSearch.tsx
│   ├── core/
│   │   ├── parser/
│   │   │   ├── index.ts
│   │   │   ├── tokenizer.ts
│   │   │   ├── key-value-parser.ts
│   │   │   ├── line-item-parser.ts
│   │   │   └── natural-language-parser.ts
│   │   ├── generator/
│   │   │   ├── index.ts
│   │   │   ├── docx-generator.ts
│   │   │   └── html-generator.ts
│   │   └── templates/
│   │       ├── index.ts
│   │       ├── template-engine.ts
│   │       └── built-in/
│   │           ├── professional.ts
│   │           ├── minimal.ts
│   │           ├── modern.ts
│   │           ├── classic.ts
│   │           └── creative.ts
│   ├── stores/
│   │   ├── invoice-store.ts
│   │   ├── client-store.ts
│   │   ├── template-store.ts
│   │   └── settings-store.ts
│   ├── hooks/
│   │   ├── useInvoice.ts
│   │   ├── useParser.ts
│   │   └── useDatabase.ts
│   ├── types/
│   │   ├── invoice.ts
│   │   ├── client.ts
│   │   ├── template.ts
│   │   └── settings.ts
│   └── utils/
```

```
│       ├── currency.ts
│       ├── date.ts
│       └── validation.ts
├── database/
│   ├── schema.sql
│   └── migrations/
├── resources/
│   ├── icons/
│   └── templates/
├── tests/
│   ├── unit/
│   ├── integration/
│   └── e2e/
├── package.json
├── electron-builder.yml
├── tsconfig.json
├── vite.config.ts
└── tailwind.config.js
```

8. Data Models & Schema

8.1 Core Entities

Invoice

```
interface Invoice {
  id: string;                  // UUID
  invoiceNumber: string;       // User-facing number (e.g., "INV-2025-0042")
  status: 'draft' | 'sent' | 'paid' | 'overdue' | 'cancelled';

  // Parties
  clientId: string;            // FK to Client
  businessProfileId: string;   // FK to BusinessProfile

  // Dates
  issueDate: Date;
  dueDate: Date;
  paidDate?: Date;

  // Line Items
  lineItems: LineItem[];

  // Financials
  subtotal: number;
  discounts: Discount[];
  taxes: Tax[];
  total: number;
  currency: string;            // ISO 4217 (e.g., "USD")

  // Content
  notes?: string;
  terms?: string;
  paymentInstructions?: string;

  // Metadata
  templateId: string;          // FK to Template
  originalText: string;        // Raw input text
  createdAt: Date;
  updatedAt: Date;
}
```

LineItem

```
interface LineItem {
  id: string;
  invoiceId: string;

  description: string;
  details?: string;              // Extended description

  quantity: number;
  unit?: string;                 // "hours", "units", "months", etc.
  unitPrice: number;

  discount?: number;             // Line-level discount
  discountType?: 'percentage' | 'fixed';

  taxable: boolean;
  taxRate?: number;

  amount: number;                // Calculated: qty * price - discount

  sortOrder: number;
}
```

Client

```
interface Client {
  id: string;

  // Identity
  name: string;
  displayName?: string;          // Short name for UI
  type: 'individual' | 'company';

  // Contact
  email?: string;
  phone?: string;
  website?: string;

  // Address
  addressLine1?: string;
  addressLine2?: string;
  city?: string;
  state?: string;
  postalCode?: string;
  country?: string;
```

```
  // Financial
  taxId?: string;
  defaultCurrency?: string;
  defaultPaymentTerms?: number;  // Days

  // Metadata
  notes?: string;
  tags?: string[];
  createdAt: Date;
  updatedAt: Date;
}
```

BusinessProfile

```
interface BusinessProfile {
  id: string;

  // Identity
  companyName: string;
  legalName?: string;
  tagline?: string;

  // Contact
  email: string;
  phone?: string;
  website?: string;

  // Address
  addressLine1: string;
  addressLine2?: string;
  city: string;
  state: string;
  postalCode: string;
  country: string;

  // Financial
  taxId?: string;
  defaultCurrency: string;
  defaultPaymentTerms: number;
  defaultTaxRate?: number;

  // Branding
  logoPath?: string;
```

```
  primaryColor?: string;
  accentColor?: string;

  // Payment Methods
  paymentMethods: PaymentMethod[];

  // Invoice Settings
  invoicePrefix: string;        // e.g., "INV"
  invoiceNumberFormat: string;  // e.g., "{PREFIX}-{YEAR}-{SEQ:4}"
  nextInvoiceSequence: number;

  createdAt: Date;
  updatedAt: Date;
}
```

8.2 Database Schema (SQLite)

```
-- Clients table
CREATE TABLE clients (
    id TEXT PRIMARY KEY,
    name TEXT NOT NULL,
    display_name TEXT,
    type TEXT CHECK(type IN ('individual', 'company')) DEFAULT 'company',
    email TEXT,
    phone TEXT,
    website TEXT,
    address_line1 TEXT,
    address_line2 TEXT,
    city TEXT,
    state TEXT,
    postal_code TEXT,
    country TEXT,
    tax_id TEXT,
    default_currency TEXT DEFAULT 'USD',
    default_payment_terms INTEGER DEFAULT 30,
    notes TEXT,
    tags TEXT,  -- JSON array
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Business profiles table
CREATE TABLE business_profiles (
```

```
    id TEXT PRIMARY KEY,
    company_name TEXT NOT NULL,
    legal_name TEXT,
    tagline TEXT,
    email TEXT NOT NULL,
    phone TEXT,
    website TEXT,
    address_line1 TEXT NOT NULL,
    address_line2 TEXT,
    city TEXT NOT NULL,
    state TEXT NOT NULL,
    postal_code TEXT NOT NULL,
    country TEXT NOT NULL,
    tax_id TEXT,
    default_currency TEXT DEFAULT 'USD',
    default_payment_terms INTEGER DEFAULT 30,
    default_tax_rate REAL DEFAULT 0,
    logo_path TEXT,
    primary_color TEXT DEFAULT '#1565C0',
    accent_color TEXT DEFAULT '#2E7D32',
    invoice_prefix TEXT DEFAULT 'INV',
    invoice_number_format TEXT DEFAULT '{PREFIX}-{YEAR}-{SEQ:4}',
    next_invoice_sequence INTEGER DEFAULT 1,
    payment_methods TEXT,  -- JSON array
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Invoices table
CREATE TABLE invoices (
    id TEXT PRIMARY KEY,
    invoice_number TEXT UNIQUE NOT NULL,
    status TEXT CHECK(status IN ('draft', 'sent', 'paid', 'overdue', 'cancelled')) DEFAULT 'draft',
    client_id TEXT NOT NULL REFERENCES clients(id),
    business_profile_id TEXT NOT NULL REFERENCES business_profiles(id),
    issue_date DATE NOT NULL,
    due_date DATE NOT NULL,
    paid_date DATE,
    subtotal REAL NOT NULL,
    discount_total REAL DEFAULT 0,
    tax_total REAL DEFAULT 0,
    total REAL NOT NULL,
    currency TEXT DEFAULT 'USD',
    notes TEXT,
    terms TEXT,
```

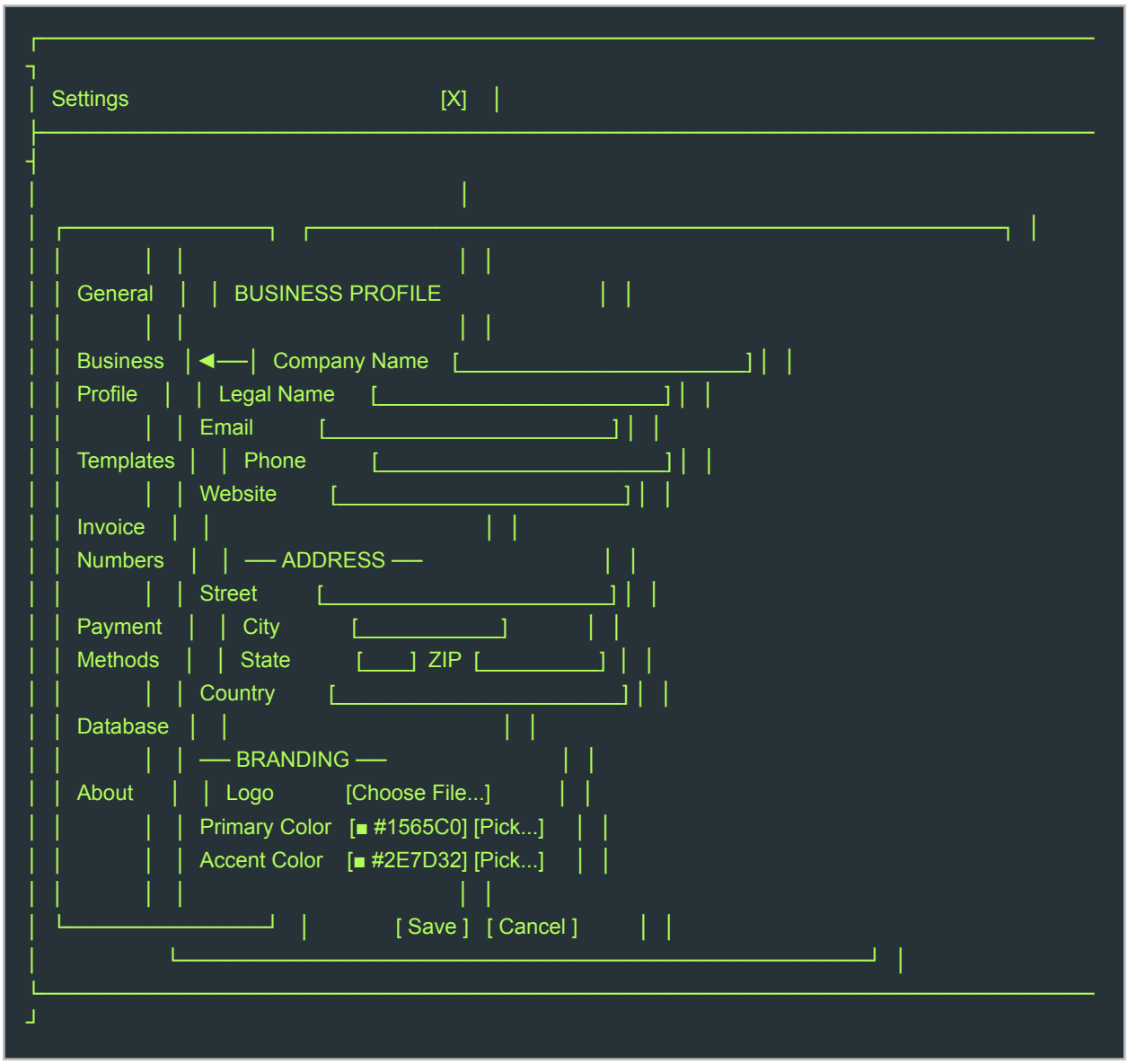
```
    payment_instructions TEXT,
    template_id TEXT,
    original_text TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Line items table
CREATE TABLE line_items (
    id TEXT PRIMARY KEY,
    invoice_id TEXT NOT NULL REFERENCES invoices(id) ON DELETE CASCADE,
    description TEXT NOT NULL,
    details TEXT,
    quantity REAL NOT NULL,
    unit TEXT,
    unit_price REAL NOT NULL,
    discount REAL DEFAULT 0,
    discount_type TEXT CHECK(discount_type IN ('percentage', 'fixed')),
    taxable INTEGER DEFAULT 1,
    tax_rate REAL DEFAULT 0,
    amount REAL NOT NULL,
    sort_order INTEGER DEFAULT 0
);

-- Templates table
CREATE TABLE templates (
    id TEXT PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT,
    type TEXT CHECK(type IN ('built-in', 'custom')) DEFAULT 'custom',
    config TEXT NOT NULL,  -- JSON configuration
    preview_image TEXT,
    is_default INTEGER DEFAULT 0,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Services catalog table
CREATE TABLE services (
    id TEXT PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT,
    default_rate REAL,
    default_unit TEXT,
    category TEXT,
```

```
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Create indexes
CREATE INDEX idx_invoices_client ON invoices(client_id);
CREATE INDEX idx_invoices_status ON invoices(status);
CREATE INDEX idx_invoices_date ON invoices(issue_date);
CREATE INDEX idx_line_items_invoice ON line_items(invoice_id);
CREATE INDEX idx_clients_name ON clients(name);

-- Full-text search
CREATE VIRTUAL TABLE invoices_fts USING fts5(
    invoice_number,
    notes,
    original_text,
    content='invoices',
    content_rowid='rowid'
);
```

9. User Interface Specifications

9.1 Main Window Layout

9.2 Settings Panel



9.3 Keyboard Shortcuts

Shortcut	Action
Ctrl + Enter	Generate invoice from current text
Ctrl + Shift + D	Export as DOCX
Ctrl + Shift + P	Export as PDF
Ctrl + N	New invoice
Ctrl + O	Open text file
Ctrl + S	Save draft

Ctrl + ,	Open settings
Ctrl + H	Open invoice history
Ctrl + T	Cycle templates
F5	Refresh preview
Esc	Close modal / Cancel

10. Input/Output Specifications

10.1 Input Text Format

The parser supports multiple input formats. Users can mix formats within a single input.

Standard Key-Value Format

```
# Invoice Metadata
Invoice: INV-2025-0042
Client: Delta Service Company
Date: January 15, 2025
Due: February 14, 2025
Currency: USD

# Line Items (pipe-separated)
Description | Quantity | Rate | Unit
Development Work | 40 | 50 | hours
Design Services | 20 | 75 | hours
Hosting (Monthly) | 1 | 29 | month

# Optional Sections
Discount: 10%
Tax: 8.25%
Notes: Thank you for your business!
Terms: Payment due within 30 days.
```

Compact Format

```
Client: Acme Corp
Date: 2025-01-15

Web Development, 40h @ $50
API Integration, 20h @ $60
Server Setup, 8h @ $75
Hosting, $29/mo

Discount: $50
Due: Net 30
```

Natural Language Format (P2)

```
Bill Acme Corporation for January 2025 work:
```

```
- 40 hours of web development at $50 per hour
- 20 hours of API integration at $60/hr
- Monthly hosting fee of $29

Apply a 10% loyalty discount.
Payment is due in 30 days.
```

10.2 Parser Output Structure

```
interface ParsedInvoice {
  // Metadata
  invoiceNumber?: string;
  client: {
    name: string;
    address?: ParsedAddress;
    email?: string;
  };
  dates: {
    issue: Date;
    due: Date;
  };
  currency: string;

  // Line Items
  lineItems: {
    description: string;
    quantity: number;
    unit?: string;
    rate: number;
    amount: number;
  }[];

  // Financials
  subtotal: number;
  discounts: {
    description?: string;
    type: 'percentage' | 'fixed';
    value: number;
    amount: number;
  }[];
  taxes: {
    description?: string;
    rate: number;
```

```
    amount: number;
  }[];
  total: number;

  // Content
  notes?: string;
  terms?: string;

  // Validation
  warnings: ParseWarning[];
  errors: ParseError[];
}
```

10.3 Export Formats

Format	Library	Notes
DOCX	docx (npm)	Native generation, no external deps
PDF	Puppeteer	HTML → PDF conversion, high fidelity
HTML	React rendering	For preview and email embedding
JSON	Native	For data backup and interoperability

10.4 File Naming Convention

```
Default pattern: {InvoiceNumber}_{ClientName}_{Date}.{ext}

Examples:
- INV-2025-0042_DeltaServiceCompany_2025-01-15.docx
- INV-2025-0042_DeltaServiceCompany_2025-01-15.pdf

Configurable tokens:
- {InvoiceNumber}  - Full invoice number
- {ClientName}     - Sanitized client name (no spaces/special chars)
- {Date}           - Issue date (YYYY-MM-DD)
- {Year}           - 4-digit year
- {Month}          - 2-digit month
- {Seq}            - Sequence number
```

11. Build & Deployment

11.1 Build Configuration

electron-builder.yml

```yaml
appId: com.invoiceforge.pro
productName: InvoiceForge Pro
copyright: Copyright © 2025 InvoiceForge

directories:
  output: dist
  buildResources: resources

files:
  - "**/*"
  - "!**/*.ts"
  - "!**/*.map"
  - "!**/node_modules/*/{CHANGELOG.md,README.md}"
  - "!**/node_modules/.cache"

win:
  target:
    - target: nsis
      arch:
        - x64
  icon: resources/icons/icon.ico
  artifactName: ${productName}-Setup-${version}.exe

nsis:
  oneClick: false
  perMachine: false
  allowToChangeInstallationDirectory: true
  installerIcon: resources/icons/icon.ico
  uninstallerIcon: resources/icons/icon.ico
  installerHeaderIcon: resources/icons/icon.ico
  createDesktopShortcut: true
  createStartMenuShortcut: true
  shortcutName: InvoiceForge Pro
  license: LICENSE.txt

extraResources:
  - from: resources/templates
    to: templates
  - from: database/schema.sql
```

```
    to: database/schema.sql

publish: null  # Disable auto-update for offline-first
```

package.json (scripts)

```
{
  "name": "invoiceforge-pro",
  "version": "1.0.0",
  "main": "electron/main.js",
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
    "preview": "vite preview",
    "electron:dev": "concurrently \"vite\" \"electron .\"",
    "electron:build": "npm run build && electron-builder",
    "electron:build:win": "npm run build && electron-builder --win",
    "test": "vitest",
    "test:e2e": "playwright test",
    "lint": "eslint src --ext .ts,.tsx",
    "typecheck": "tsc --noEmit"
  },
  "dependencies": {
    "docx": "^8.5.0",
    "better-sqlite3": "^9.4.0",
    "uuid": "^9.0.0",
    "date-fns": "^3.3.0",
    "zustand": "^4.5.0"
  },
  "devDependencies": {
    "electron": "^28.2.0",
    "electron-builder": "^24.9.0",
    "vite": "^5.0.0",
    "typescript": "^5.3.0",
    "@types/react": "^18.2.0",
    "vitest": "^1.2.0",
    "playwright": "^1.41.0"
  }
}
```

11.2 Build Process

7. Install dependencies: npm install

8. Run TypeScript compilation: npm run typecheck
9. Run tests: npm test
10. Build renderer (Vite): npm run build
11. Package with electron-builder: npm run electron:build:win
12. Output: dist/InvoiceForge Pro-Setup-1.0.0.exe

11.3 Installer Specifications

Property	Value
Installer Type	NSIS (Nullsoft Scriptable Install System)
Target Architecture	x64 only (64-bit Windows)
Installation Mode	Per-user (no admin required)
Default Install Path	%LOCALAPPDATA%\Programs\InvoiceForge Pro
User Data Path	%APPDATA%\InvoiceForge Pro
Estimated Size	~120 MB installed
Installer Size	~80 MB compressed
Code Signing	EV Code Signing Certificate (required)
Uninstaller	Included, removes all files except user data

11.4 System Requirements

Component	Minimum	Recommended
OS	Windows 10 v1903 (64-bit)	Windows 11 (64-bit)
Processor	Intel Core i3 / AMD Ryzen 3	Intel Core i5 / AMD Ryzen 5
RAM	4 GB	8 GB
Storage	500 MB available	1 GB available
Display	1280 x 720	1920 x 1080
Graphics	DirectX 11 compatible	Hardware acceleration

12. Testing Requirements

12.1 Test Strategy

Test Type	Tool	Coverage Target	Run Frequency
Unit Tests	Vitest	> 80%	Every commit
Integration	Vitest	> 70%	Every PR
E2E Tests	Playwright	Critical paths	Pre-release
Visual Regression	Playwright	All templates	Weekly
Performance	Custom benchmarks	All NFRs	Pre-release

12.2 Critical Test Cases

Parser Tests

- TC-P001: Parse standard key-value format correctly
- TC-P002: Parse pipe-separated line items
- TC-P003: Parse CSV-style line items
- TC-P004: Handle missing required fields with appropriate errors
- TC-P005: Parse multiple date formats (ISO, US, natural)
- TC-P006: Parse currency amounts with various formats ($1,234.56)
- TC-P007: Calculate line item totals correctly
- TC-P008: Apply percentage discounts correctly
- TC-P009: Apply fixed amount discounts correctly
- TC-P010: Calculate tax on subtotal after discounts

Document Generation Tests

- TC-D001: Generate valid DOCX file that opens in Word
- TC-D002: Generate valid PDF file that opens in readers
- TC-D003: All template fields populated correctly
- TC-D004: Logo renders at correct size and position
- TC-D005: Currency formatting matches locale settings
- TC-D006: Line items table renders with correct columns
- TC-D007: Totals section calculates correctly
- TC-D008: Multi-page invoices paginate correctly

E2E Tests

- TC-E001: Complete flow - text input to DOCX export
- TC-E002: Complete flow - text input to PDF export
- TC-E003: Create new client and use in invoice
- TC-E004: Search and load historical invoice
- TC-E005: Change template and regenerate invoice
- TC-E006: Modify business profile and verify in output
- TC-E007: Import/export data backup
- TC-E008: Keyboard shortcuts work correctly

13. Security Considerations

13.1 Security Architecture

- Complete offline operation - zero network connections
- No telemetry, analytics, or crash reporting
- Local SQLite database with optional AES-256 encryption
- Code-signed installer and executable (EV certificate)
- No elevated privileges required
- Sandboxed renderer process (Electron security best practices)

13.2 Data Protection

Data Type	Protection Method
Database file	Optional encryption (AES-256-CBC with user password)
Exported documents	Standard file system permissions
Configuration files	JSON in user app data directory
Logo images	Copied to app data, original path not stored
Backup exports	Optional password-protected ZIP

13.3 Electron Security Checklist

- nodeIntegration: false (always)
- contextIsolation: true (always)
- sandbox: true for renderer
- webSecurity: true
- No remote module usage
- CSP headers configured
- IPC channels validated and typed
- No shell.openExternal with user input

14. Future Roadmap

v1.1 (Q2 2025)

- macOS support (DMG installer)
- Linux support (AppImage, .deb)
- Natural language parsing improvements
- Additional templates (10 total)

- Recurring invoice scheduling

v1.2 (Q3 2025)

- Email integration (optional, user-configured SMTP)
- QuickBooks export compatibility
- Multi-language support (i18n)
- Custom template designer

v2.0 (Q4 2025)

- Optional cloud sync (encrypted, user-controlled)
- Mobile companion app (read-only)
- Team/multi-user support
- Payment gateway integration

15. Appendices

Appendix A: Glossary

Term	Definition
DXA	Document eXtensible Architecture units (1/20 of a point, 1440 = 1 inch)
Electron	Framework for building cross-platform desktop apps with web technologies
IPC	Inter-Process Communication between Electron main and renderer
NSIS	Nullsoft Scriptable Install System for Windows installers
P0/P1/P2	Priority levels: P0=Critical, P1=High, P2=Medium

Appendix B: References

- Electron Documentation: https://electronjs.org/docs
- docx npm package: https://docx.js.org
- electron-builder: https://electron.build
- SQLite: https://sqlite.org/docs.html
- Puppeteer: https://pptr.dev

Appendix C: Document History

Version	Date	Author	Changes
1.0	2025-01-08	Product Team	Initial release

Document Approval

This PRD has been reviewed and approved for development.