SmartPDF Convert - Project Context for Claude

Project Overview

SmartPDF Convert is an AI-powered SaaS application that extracts tabular data from PDF documents and images, converting them into editable Excel spreadsheets. The product targets financial professionals, accountants, and businesses who need accurate extraction of invoices, bank statements, receipts, and other structured documents.

Business Model

- **Free Tier**: 3 conversions/day with generic template
- **Pro Plan**: $29/month for unlimited conversions, batch processing, and specialized templates
- **Target Market**: Accountants, bookkeepers, small businesses, financial analysts

Strategic Focus

The product differentiates through **accuracy-first positioning** in the financial document vertical, where errors have real consequences. Unlike generic PDF tools that claim "AI-powered" extraction, SmartPDF Convert provides:

- Confidence scoring to flag uncertain extractions
- Specialized templates optimized for financial documents
- Batch processing for month-end reconciliation workflows
- Split view preview to compare original PDF with extracted data

Tech Stack

Frontend

- **React 19** with TypeScript
- **Vite** for development and build
- **Tailwind CSS 4** for styling
- **shadcn/ui** components (Radix UI primitives)
- **wouter** for routing (lightweight alternative to React Router)
- **FortuneSheet** for spreadsheet editing
- **react-pdf** for PDF rendering in split view
- **react-resizable-panels** for split view layout
- **tRPC client** for type-safe API calls

Backend

- **Express 4** with TypeScript
- **tRPC 11** for type-safe API procedures
- **Drizzle ORM** for database operations
- **MySQL** database (via PlanetScale or local)

External Services

- **Supabase Auth** - Email/password + Google OAuth authentication
- **OpenRouter API** - GPT-4 Vision for AI extraction
- **Stripe** - Payment processing (live mode)
- **AWS S3** - File storage (optional)

Key Libraries

- **pdf-lib** - PDF page count detection
- **pdf2pic + GraphicsMagick** - Server-side PDF to image conversion
- **ExcelJS** - Excel file generation
- **sonner** - Toast notifications

Project Structure

```
smartpdf-convert/
├── client/                     # React frontend
│   ├── src/
│   │   ├── pages/              # Route pages (Home, Convert, Dashboard, Results, etc.)
│   │   ├── components/
│   │   │   ├── ui/             # shadcn/ui components
│   │   │   ├── upload/         # DropZone, ProcessingStatus
│   │   │   ├── editor/         # SpreadsheetEditor (FortuneSheet wrapper)
│   │   │   ├── preview/        # PDF preview, Split view components
│   │   │   ├── analysis/       # Document analysis dialog
│   │   │   ├── templates/      # Template selector
│   │   │   └── results/        # Confidence score display
│   │   ├── contexts/           # Auth context (SupabaseAuthContext)
│   │   ├── lib/                # Utilities (trpc client, cn helper)
│   │   └── App.tsx             # Main app with routing
│   └── index.html
├── server/
│   ├── _core/                  # Framework plumbing (auth, context)
│   ├── routers.ts              # All tRPC procedures
│   ├── db.ts                   # Database query helpers
│   └── *.test.ts               # Tests
├── drizzle/
│   └── schema.ts               # Database schema (conversions, users, templates, etc.)
├── shared/                     # Shared types
└── storage/                    # S3 helpers
```

Key Features Implemented

Core Extraction Flow

1. User uploads PDF/image via DropZone component
2. Server analyzes document and presents questions (AnalysisDialog)
3. User can customize extraction or use quick extract

4. GPT-4 Vision extracts tables with confidence scoring
5. Results shown in preview/edit view with export options

Batch Processing (Pro Feature)

- Upload multiple files at once
- Each file processed sequentially with progress tracking
- All tables combined into single Excel workbook with multiple sheets
- Files grouped by `batchId` in database for later retrieval

Split View Preview

- Side-by-side view of original PDF and extracted data
- Synced page navigation (for single multi-page PDFs)
- Page navigator with table count badges
- Fullscreen toggle
- Resizable panels

Dashboard

- Conversion history with batch grouping
- Click to view/re-download any conversion
- Batch conversions shown as grouped items

Database Schema (Key Tables)

```
// conversions table
{
  id: serial primary key,
  userId: varchar (nullable for anonymous),
  fileName: varchar,
  fileSize: int,
  mimeType: varchar,
  pageCount: int,
  tablesExtracted: int,
  status: enum('pending', 'processing', 'completed', 'failed'),
  extractedData: json,  // The extracted tables
  confidence: decimal,
  templateId: varchar,
  batchId: varchar,      // Groups batch uploads together
  createdAt: timestamp,
  updatedAt: timestamp
}
```

API Endpoints (tRPC Procedures)

conversion router

- `process` - Main extraction endpoint (accepts batchId for grouping)

- `analyze` - Document analysis before extraction
- `extractWithGuidance` - Extract with user-provided guidance
- `checkUsage` - Check remaining conversions for user
- `getTemplates` - Get available templates
- `history` - Get user's conversion history
- `getConversion` - Get single conversion by ID
- `getBatch` - Get all conversions in a batch by batchId

subscription router

- `upgrade` - Create Stripe checkout session
- `manage` - Create Stripe portal session

Current State & Recent Work

Recently Completed

- Split View preview with PDF rendering
- Batch upload with batchId grouping
- Dashboard batch grouping and navigation
- Results page batch view
- Fullscreen toggle for split view
- Fixed "Convert All" button positioning

Known Limitations

- Split View for batch uploads only shows the first PDF (each file in batch has separate PDF)
- Page syncing works for single multi-page PDFs, not batch uploads of separate files

Development Commands

```
pnpm dev          # Start development server (port 3000)
pnpm build        # Build for production
pnpm test         # Run tests
pnpm db:push      # Push schema changes to database
pnpm db:generate  # Generate migrations
```

Code Conventions

- 2-space indentation
- Semicolons required
- Single quotes for strings
- TypeScript strict mode
- Component files use PascalCase
- Utility files use camelCase
- CSS via Tailwind utility classes

Environment Variables Required

```
DATABASE_URL              # MySQL connection string
VITE_SUPABASE_URL         # Supabase project URL
VITE_SUPABASE_ANON_KEY    # Supabase anon key
SUPABASE_SERVICE_ROLE_KEY # Supabase service role key
OPENROUTER_API_KEY        # For GPT-4 Vision
STRIPE_SECRET_KEY         # Stripe API key
STRIPE_PRO_PRICE_ID       # Stripe price ID for Pro plan
```

Testing Notes

- Set `TESTING_MODE=true` in .env to bypass auth/payment checks during development
- Sample documents available in template selector for testing Pro features

Future Roadmap

- QuickBooks/Xero export integration
- Custom extraction templates
- API for third-party integrations
- Mobile app
- Hybrid AI (Gemini Flash for simple docs, GPT-4V for complex)