Tax Filing App Frontend Analysis

Executive Summary

This is a comprehensive Next.js 14 tax filing application with a sophisticated multi-step tax return process, document management with OCR capabilities, and advanced tax calculations. The frontend expects a fully functional backend with PostgreSQL database, NextAuth authentication, document processing, and tax calculation APIs.

1. Project Structure and Technology Stack

Core Technologies

- **Framework**: Next.js 14.2.28 with App Router
- **Language**: TypeScript 5.2.2
- **Database ORM**: Prisma 6.7.0 with PostgreSQL
- **Authentication**: NextAuth 4.24.11 with Prisma adapter
- **UI Framework**: React 18.2.0 with Radix UI components
- **Styling**: Tailwind CSS 3.3.3
- **State Management**: Zustand 5.0.3, Jotai 2.6.0
- **Forms**: React Hook Form 7.53.0 with Zod 3.23.8 validation
- **Charts**: Plotly.js 2.35.3, Chart.js 4.4.9, Recharts 2.15.3

Key Dependencies

- **Document Processing**: Google Cloud Document AI 9.3.0
- **File Handling**: UUID 11.1.0 for unique identifiers
- **Password Hashing**: bcryptjs 2.4.3
- **Date Handling**: date-fns 3.6.0, dayjs 1.11.13
- **HTTP Client**: Built-in fetch with SWR 2.2.4 for caching
- **Notifications**: React Hot Toast 2.4.1, Sonner 1.5.0

Project Structure

```
app/
├── api/                    # Backend API routes
│   ├── auth/              # Authentication endpoints
│   ├── tax-returns/       # Tax return CRUD operations
│   ├── documents/         # Document upload and processing
│   ├── ai/               # AI-powered tax strategies
│   └── debug/            # Debug endpoints
├── auth/                  # Authentication pages
├── dashboard/             # Main dashboard
├── tax-filing/           # Tax filing workflow
└── globals.css           # Global styles

components/
├── steps/                # Tax filing step components
├── ui/                   # Reusable UI components
├── dashboard-client.tsx  # Dashboard main component
├── tax-filing-interface.tsx # Main tax filing workflow
└── document-management.tsx # Document handling

lib/
├── db.ts                 # Database connection
├── types.ts              # TypeScript definitions
├── tax-calculations.ts   # Tax calculation logic
├── document-ai-service.ts # Document processing
└── utils.ts              # Utility functions

prisma/
└── schema.prisma         # Database schema
```

2. API Endpoints Expected by Frontend

Authentication Endpoints

POST /api/auth/signup

- **Purpose**: User registration
- **Request Body**:

```
json
  {
    "name": "string",
    "email": "string",
    "password": "string"
  }
```

- **Response**:

```
json
  {
    "message": "User created successfully",
    "user": {
      "id": "string",
      "name": "string",
      "email": "string"
    }
  }
```

GET/POST /api/auth/[...nextauth]

- **Purpose**: NextAuth authentication handler
- **Supports**: Credentials provider with email/password
- **Session Strategy**: JWT
- **Custom Pages**: `/auth/login` for sign-in

Tax Returns Endpoints

GET /api/tax-returns

- **Purpose**: Fetch all tax returns for authenticated user
- **Response**:

```
json
  [
    {
      "id": "string",
      "taxYear": "number",
      "filingStatus": "enum",
      "currentStep": "number",
      "completedSteps": "number[]",
      "isCompleted": "boolean",
      "isFiled": "boolean",
      "incomeEntries": "IncomeEntry[]",
      "deductionEntries": "DeductionEntry[]",
      "dependents": "Dependent[]",
      "createdAt": "datetime",
      "updatedAt": "datetime"
    }
  ]
```

POST /api/tax-returns

- **Purpose**: Create new tax return
- **Request Body**:

```
json
  {
    "taxYear": "number",
    "filingStatus": "SINGLE|MARRIED_FILING_JOINTLY|MARRIED_FILING_SEPARATELY|
HEAD_OF_HOUSEHOLD|QUALIFYING_SURVIVING_SPOUSE"
  }
```

GET /api/tax-returns/[id]

- **Purpose**: Fetch specific tax return with all related data
- **Response**: Complete tax return object with income, deductions, dependents

PUT /api/tax-returns/[id]

- **Purpose**: Update tax return data
- **Request Body**: Partial tax return data
- **Features**: Auto-completion of steps, last saved tracking

POST /api/tax-returns/[id]/auto-save

- **Purpose**: Auto-save functionality for form data
- **Request Body**: Form data to save

- **Response**:
```
json
  {
    "taxReturn": "TaxReturn",
    "savedAt": "datetime"
  }
```

POST /api/tax-returns/[id]/complete-step

- **Purpose**: Mark step as completed and advance workflow
- **Request Body**:
```
json
  {
    "stepNumber": "number",
    "data": "object"
  }
```

Income Management Endpoints

POST /api/tax-returns/[id]/income

- **Purpose**: Add income entry
- **Request Body**:
```
json
  {
    "incomeType": "W2_WAGES|INTEREST|DIVIDENDS|BUSINESS_INCOME|CAPITAL_GAINS|OTHER_INCOME|
UNEMPLOYMENT|RETIREMENT_DISTRIBUTIONS|SOCIAL_SECURITY",
    "amount": "decimal",
    "description": "string?",
    "employerName": "string?",
    "employerEIN": "string?",
    "payerName": "string?",
    "payerTIN": "string?"
  }
```

PUT /api/tax-returns/[id]/income/[entryId]

- **Purpose**: Update income entry

DELETE /api/tax-returns/[id]/income/[entryId]

- **Purpose**: Delete income entry

Deduction Management Endpoints

POST /api/tax-returns/[id]/deductions

- **Purpose**: Add deduction entry
- **Request Body**:
```
json
  {
    "deductionType": "MORTGAGE_INTEREST|STATE_LOCAL_TAXES|CHARITABLE_CONTRIBUTIONS|MEDIC-
AL_EXPENSES|BUSINESS_EXPENSES|STUDENT_LOAN_INTEREST|IRA_CONTRIBUTIONS|OTHER_DEDUCTIONS",
    "amount": "decimal",
    "description": "string?"
  }
```

PUT /api/tax-returns/[id]/deductions/[entryId]

- **Purpose**: Update deduction entry

DELETE /api/tax-returns/[id]/deductions/[entryId]

- **Purpose**: Delete deduction entry

Document Management Endpoints

POST /api/documents/upload

- **Purpose**: Upload tax documents
- **Request**: FormData with file and taxReturnId
- **File Types**: PDF, PNG, JPEG, TIFF (max 10MB)
- **Response**:

```
json
  {
    "id": "string",
    "fileName": "string",
    "fileType": "string",
    "fileSize": "number",
    "documentType": "W2|FORM_1099_INT|FORM_1099_DIV|etc",
    "processingStatus": "PENDING"
  }
```

POST /api/documents/[id]/process

- **Purpose**: Process document with OCR/AI extraction
- **Response**: Streaming response with extracted data
- **Features**:
- Google Document AI integration (primary)
- LLM fallback processing
- Structured data extraction for tax forms

GET /api/tax-returns/[id]/documents

- **Purpose**: Fetch all documents for tax return

DELETE /api/documents/[id]

- **Purpose**: Delete document

AI Integration Endpoints

POST /api/ai/tax-strategies

- **Purpose**: AI-powered tax optimization suggestions
- **Request Body**: Tax return data
- **Response**: Optimization recommendations

GET /api/debug/tax-data

- **Purpose**: Debug endpoint for tax calculations

3. Data Models and Database Schema

Core Models

User

```
model User {
  id            String    @id @default(cuid())
  name          String?
  email         String    @unique
  emailVerified DateTime?
  image         String?
  password      String?
  accounts      Account[]
  sessions      Session[]
  taxReturns    TaxReturn[]
  createdAt     DateTime  @default(now())
  updatedAt     DateTime  @updatedAt
}
```

TaxReturn (Primary Entity)

```
model TaxReturn {
  id                 String         @id @default(cuid())
  userId             String
  taxYear            Int
  filingStatus       FilingStatus

  // Personal Information
  firstName          String?
  lastName           String?
  ssn                String?
  spouseFirstName    String?
  spouseLastName     String?
  spouseSsn          String?
  address            String?
  city               String?
  state              String?
  zipCode            String?

  // Tax Calculations (Decimal precision for currency)
  totalIncome        Decimal        @default(0) @db.Decimal(12, 2)
  adjustedGrossIncome Decimal       @default(0) @db.Decimal(12, 2)
  standardDeduction Decimal         @default(0) @db.Decimal(12, 2)
  itemizedDeduction Decimal         @default(0) @db.Decimal(12, 2)
  taxableIncome      Decimal        @default(0) @db.Decimal(12, 2)
  taxLiability       Decimal        @default(0) @db.Decimal(12, 2)
  totalCredits       Decimal        @default(0) @db.Decimal(12, 2)
  refundAmount       Decimal        @default(0) @db.Decimal(12, 2)
  amountOwed         Decimal        @default(0) @db.Decimal(12, 2)

  // Workflow Management
  currentStep        Int            @default(1)
  completedSteps     Int[]          @default([])
  lastSavedAt        DateTime?
  isCompleted        Boolean        @default(false)
  isFiled            Boolean        @default(false)

  // Relationships
  user               User           @relation(fields: [userId], references: [id], onDel
ete: Cascade)
  incomeEntries      IncomeEntry[]
  deductionEntries   DeductionEntry[]
  dependents         Dependent[]
  documents          Document[]

  @@unique([userId, taxYear])
}
```

IncomeEntry

```
model IncomeEntry {
  id             String        @id @default(cuid())
  taxReturnId    String
  incomeType     IncomeType
  description    String?
  amount         Decimal       @db.Decimal(12, 2)

  // W-2 specific fields
  employerName   String?
  employerEIN    String?

  // 1099 specific fields
  payerName      String?
  payerTIN       String?

  taxReturn      TaxReturn     @relation(fields: [taxReturnId], references: [id], onDel
ete: Cascade)
  extractedEntries DocumentExtractedEntry[]
}
```

DeductionEntry

```
model DeductionEntry {
  id             String        @id @default(cuid())
  taxReturnId    String
  deductionType DeductionType
  description    String?
  amount         Decimal       @db.Decimal(12, 2)

  taxReturn      TaxReturn     @relation(fields: [taxReturnId], references: [id], onDel
ete: Cascade)
  extractedEntries DocumentExtractedEntry[]
}
```

Document (OCR/AI Processing)

```
model Document {
  id               String        @id @default(cuid())
  taxReturnId      String
  fileName         String
  fileType         String
  fileSize         Int
  filePath         String
  documentType     DocumentType
  processingStatus ProcessingStatus @default(PENDING)

  // OCR and extraction results
  ocrText          String?       @db.Text
  extractedData    Json?

  // Verification workflow
  isVerified       Boolean       @default(false)
  verifiedBy       String?
  verificationNotes String?      @db.Text

  taxReturn        TaxReturn     @relation(fields: [taxReturnId], references: [id], onD
elete: Cascade)
  extractedEntries DocumentExtractedEntry[]
}
```

Enums

FilingStatus

- SINGLE
- MARRIED_FILING_JOINTLY
- MARRIED_FILING_SEPARATELY
- HEAD_OF_HOUSEHOLD
- QUALIFYING_SURVIVING_SPOUSE

IncomeType

- W2_WAGES
- INTEREST
- DIVIDENDS
- BUSINESS_INCOME
- CAPITAL_GAINS
- OTHER_INCOME
- UNEMPLOYMENT
- RETIREMENT_DISTRIBUTIONS
- SOCIAL_SECURITY

DocumentType

- W2, W2_CORRECTED, W3
- FORM_1099_INT, FORM_1099_DIV, FORM_1099_MISC, FORM_1099_NEC, FORM_1099_R, FORM_1099_G, FORM_1099_K
- FORM_1098, FORM_1098_E, FORM_1098_T
- FORM_5498, SCHEDULE_K1
- OTHER_TAX_DOCUMENT, RECEIPT, STATEMENT, UNKNOWN

ProcessingStatus

- PENDING
- PROCESSING
- COMPLETED
- FAILED
- MANUAL_REVIEW_REQUIRED

4. Authentication and User Management

NextAuth Configuration

- **Provider**: Credentials (email/password)
- **Adapter**: Prisma adapter for database sessions
- **Session Strategy**: JWT
- **Password Hashing**: bcryptjs with salt rounds 12
- **Custom Pages**: `/auth/login` for sign-in

Session Management

- Server-side session validation using `getServerSession()`
- Client-side session provider wrapping entire app
- Automatic redirect to login for protected routes
- User context available throughout application

Security Features

- Password hashing with bcrypt
- CSRF protection via NextAuth
- Secure session tokens
- Environment-based secrets

5. Configuration and Environment Variables

Required Environment Variables

```
# Database
DATABASE_URL="postgresql://user:password@host:port/database"

# NextAuth
NEXTAUTH_URL="http://localhost:3000"
NEXTAUTH_SECRET="your-secret-key"

# AbacusAI (for LLM fallback)
ABACUSAI_API_KEY="your-api-key"

# Google Document AI (Optional)
GOOGLE_CLOUD_PROJECT_ID="your-project-id"
GOOGLE_CLOUD_LOCATION="us"
GOOGLE_CLOUD_W2_PROCESSOR_ID="processor-id"
GOOGLE_CLOUD_1099_PROCESSOR_ID="processor-id"
GOOGLE_APPLICATION_CREDENTIALS="/path/to/service-account.json"
```

Configuration Files

- **next.config.js**: Next.js configuration with image optimization disabled
- **tailwind.config.ts**: Tailwind CSS with custom theme
- **tsconfig.json**: TypeScript configuration
- **prisma/schema.prisma**: Database schema definition

6. Frontend Routing and Page Structure

App Router Structure

```
app/
├── page.tsx                    # Landing page
├── layout.tsx                  # Root layout with providers
├── globals.css                 # Global styles
├── auth/
│   ├── login/page.tsx         # Login page
│   └── signup/page.tsx        # Registration page
├── dashboard/
│   └── page.tsx               # Main dashboard
└── tax-filing/
    └── [id]/page.tsx          # Tax filing workflow
```

Protected Routes

- `/dashboard` - Requires authentication
- `/tax-filing/[id]` - Requires authentication and tax return ownership

Navigation Flow

1. **Landing Page** → Authentication
2. **Dashboard** → Tax return management
3. **Tax Filing Workflow** → 7-step process:
   - Step 1: Getting Started
   - Step 2: Personal Information
   - Step 3: Income
   - Step 4: Deductions
   - Step 5: Tax Calculation
   - Step 6: Review
   - Step 7: Filing

7. Key Features and Functionality

Multi-Step Tax Filing Workflow

- **7-step guided process** with progress tracking
- **Auto-save functionality** with 2-second debouncing
- **Step completion tracking** with validation
- **Navigation controls** with prev/next buttons
- **Real-time tax calculations** as data is entered

Document Management System

- **File upload** with drag-and-drop support

- **OCR processing** using Google Document AI
- **LLM fallback** for document extraction
- **Document verification** workflow
- **Supported formats**: PDF, PNG, JPEG, TIFF (max 10MB)

Advanced Tax Calculations

- **Real-time tax liability** calculations
- **Standard vs itemized deduction** comparison
- **Tax optimization suggestions** using AI
- **Multiple income types** support
- **Dependent management** with credit calculations

Interactive Features

- **What-if scenarios** for tax planning
- **Data visualization** with charts and graphs
- **Progress tracking** with completion percentages
- **Auto-save notifications** with timestamps
- **Responsive design** for mobile/desktop

AI Integration

- **Document extraction** with structured data parsing
- **Tax strategy recommendations**
- **Optimization suggestions** based on user data
- **Fallback processing** when Document AI unavailable

8. Backend Implementation Requirements

Database Setup

- PostgreSQL database with Prisma ORM
- Run `prisma migrate dev` to create tables
- Run `prisma db seed` to populate initial data
- Ensure proper indexing for performance

File Storage

- Local file system storage in `uploads/documents/`
- UUID-based file naming for security
- File type validation and size limits
- Proper cleanup for deleted documents

Document Processing Pipeline

1. File upload and validation
2. Document type detection
3. OCR processing (Google Document AI or LLM)
4. Data extraction and structuring
5. Verification workflow
6. Integration with tax calculations

Tax Calculation Engine

- Implement tax brackets and rates
- Standard deduction calculations
- Credit calculations (Child Tax Credit, EITC)
- State tax considerations
- Real-time recalculation triggers

API Security

- Authentication middleware for all protected routes
- User ownership validation for tax returns
- Input validation and sanitization
- Rate limiting for document processing
- Error handling and logging

Performance Considerations

- Database query optimization
- File processing queues for large documents
- Caching for tax calculation results
- Streaming responses for document processing
- Auto-save debouncing to prevent excessive requests

This comprehensive analysis provides all the information needed to implement a compatible backend for the tax filing application. The frontend is well-structured and expects a robust backend with proper authentication, database management, document processing, and tax calculation capabilities.