Better Auth Implementation Documentation

Overview

This document explains how Better Auth is implemented in the DocuSeal application, including all files involved and their connections.

Files Structure

1. Core Better Auth Configuration Files

`/src/lib/auth.ts` - Server-side Auth Configuration

Purpose: Main Better Auth server configuration
What it does:

- Configures Better Auth with Prisma database adapter
- Sets up email/password authentication
- Configures Google OAuth provider
- Defines session settings (expiration, update frequency)
- Sets security secret and base URL

Key Features:

```
export const auth = betterAuth({
  database: prismaAdapter(db, { provider: "postgresql" }),
  emailAndPassword: { enabled: true, requireEmailVerification: false },
  socialProviders: {
    google: {
      clientId: process.env.GOOGLE_CLIENT_ID!,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET!,
      redirectURI: `${process.env.BETTER_AUTH_URL}/api/auth/callback/google`,
    },
  },
  session: { expiresIn: 60 * 60 * 24 * 7, updateAge: 60 * 60 * 24 },
  secret: process.env.BETTER_AUTH_SECRET!,
  baseURL: process.env.BETTER_AUTH_URL || "http://localhost:3000",
})
```

`/src/lib/auth-client.ts` - Client-side Auth Interface

Purpose: Client-side Better Auth interface for React components
What it does:

- Creates auth client for frontend usage
- Exports authentication functions (signIn, signUp, signOut, useSession)
- Handles client-side authentication state

Key Exports:

```
export const { signIn, signUp, signOut, useSession } = authClient
```

2. API Routes

`/src/app/api/auth/[...all]/route.ts` - Auth API Handler

Purpose: Handles all Better Auth API requests
What it does:

- Processes GET and POST requests to `/api/auth/*`
- Routes authentication requests to Better Auth handler
- Handles sign-in, sign-up, OAuth callbacks, session management

Implementation:

```
export async function GET(request: Request) {
  return auth.handler(request)
}
export async function POST(request: Request) {
  return auth.handler(request)
}
```

3. Database Schema

`/prisma/schema.prisma` - Database Models

Purpose: Defines database structure for Better Auth
What it includes:

- `User` model: Stores user information
- `Account` model: Stores authentication provider data
- `Session` model: Manages user sessions
- `Verification` model: Handles email verification tokens

Key Models:

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

model Account {
  id                    String    @id @default(cuid())
  accountId             String    @default("")
  providerId            String    @default("credentials")
  userId                String
  accessToken           String?
  refreshToken          String?
  // ... other OAuth fields
  user                  User      @relation(fields: [userId], references: [id], onDelete:
  @@unique([providerId, accountId])
}

model Session {
  id        String   @id @default(cuid())
  expiresAt DateTime
  token     String   @unique
  userId    String
  user      User     @relation(fields: [userId], references: [id], onDelete: Cascade)
}
```

4. Frontend Components

`/src/components/navbar.tsx` - Navigation Component

Purpose: Main navigation with user authentication state

Better Auth Usage:

- Uses `useSession()` hook to get current user
- Uses `signOut()` function for logout
- Displays user avatar and profile information
- Shows sign-in button for unauthenticated users

Key Implementation:

```
const { data: session } = useSession();
// Conditional rendering based on session state
{session ? (
  // User dropdown with profile and logout
) : (
  // Sign in button
)}
```

`/src/app/page.tsx` - Homepage Component

Purpose: Main dashboard/homepage
Better Auth Usage:

- Uses `useSession()` to check authentication status
- Shows loading skeleton while session is being fetched
- Redirects unauthenticated users to welcome screen
- Displays personalized content for authenticated users

Key Implementation:

```
const { data: session, isPending } = useSession();
if (isPending) return <DashboardSkeleton />;
if (!session) return <WelcomeScreen />;
// Authenticated user content
```

`/src/app/auth/signin/page.tsx` - Sign In Page

Purpose: User authentication interface
Better Auth Usage:

- Uses `signIn.email()` for email/password authentication

- Uses `signIn.social()` for Google OAuth
- Handles authentication errors and success states

Key Implementation:

```
const handleSignIn = async () => {
  try {
    await signIn.email({ email, password });
    window.location.href = "/";
  } catch (error) {
    setError(error.message);
  }
};

const handleGoogleSignIn = async () => {
  try {
    await signIn.social({ provider: "google" });
  } catch (error) {
    setError(error.message);
  }
};
```

`/src/app/auth/signup/page.tsx` - Sign Up Page

Purpose: User registration interface
Better Auth Usage:

- Uses `signUp.email()` for user registration
- Automatically signs in user after successful registration
- Uses `signIn.social()` for Google OAuth registration

Key Implementation:

```
const handleSignUp = async () => {
  try {
    await signUp.email({
      email,
      password,
      name: email.split("@")[0],
    });
    await signIn.email({ email, password });
    router.push("/");
  } catch (err) {
    setError(err.message);
  }
};
```

`/src/components/client-providers.tsx` - App Providers

Purpose: Wraps app with necessary providers
Better Auth Usage:

- Removed NextAuth SessionProvider
- Only includes ThemeProvider now
- Better Auth handles session state internally

5. Environment Configuration

`.env` - Environment Variables

Required Variables:

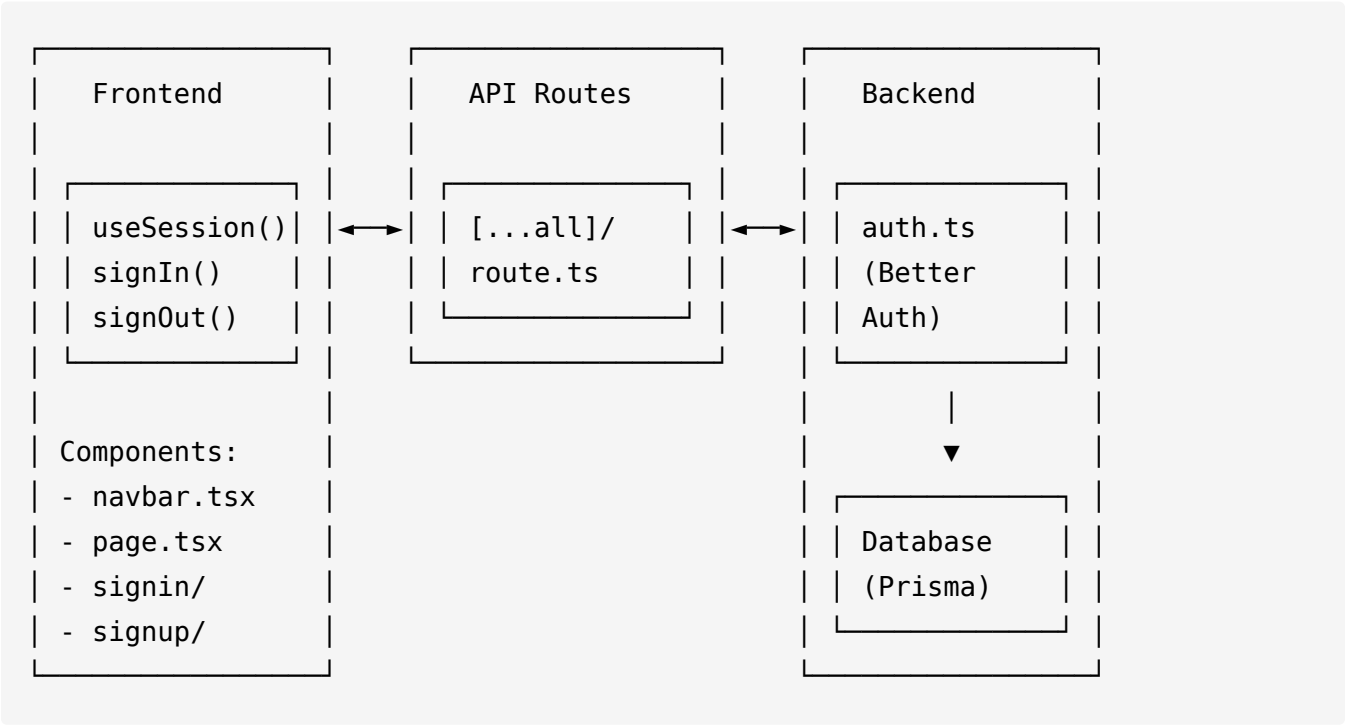
```
# Database
DATABASE_URL="your_database_url"
DIRECT_DATABASE_URL="your_direct_database_url"

# Better Auth
BETTER_AUTH_SECRET="your_random_secret_key"
BETTER_AUTH_URL="http://localhost:3000"
NEXT_PUBLIC_BETTER_AUTH_URL="http://localhost:3000"

# Google OAuth
GOOGLE_CLIENT_ID="your_google_client_id"
GOOGLE_CLIENT_SECRET="your_google_client_secret"
```

How Components Are Connected

Authentication Flow

1. **Initial Load**:
   ```
   App Start → client-providers.tsx → Components use useSession()
   ```
2. **Sign In Process**:
   ```
   signin/page.tsx → signIn.email() → /api/auth/[...all] → auth.ts → Database
   ```
3. **Session Management**:
   ```
   useSession() → auth-client.ts → /api/auth/get-session → auth.ts → Database
   ```
4. **Google OAuth**:
   ```
   signIn.social() → Google → /api/auth/callback/google → auth.ts → Database
   ```

Data Flow Diagram



Component Dependencies

```
auth.ts (server config)
    ↓
auth-client.ts (client interface)
    ↓
Components (navbar, pages, etc.)
    ↓
API Routes ([...all]/route.ts)
    ↓
Database (Prisma schema)
```

Key Features Implemented

✅ Authentication Methods

- **Email/Password**: Traditional credential-based auth
- **Google OAuth**: Social authentication

- **Session Management**: Automatic session handling
- **Security**: Encrypted sessions with secret key

✅ User Interface

- **Protected Routes**: Automatic redirection for unauthenticated users
- **Loading States**: Skeleton screens during auth checks
- **Error Handling**: User-friendly error messages
- **Responsive Design**: Works on all devices

✅ Database Integration

- **Prisma ORM**: Type-safe database operations
- **PostgreSQL**: Robust relational database
- **Migration Support**: Schema versioning and updates
- **Relationships**: Proper foreign key constraints

Migration from NextAuth

What Was Replaced

- `next-auth/react` → `@/lib/auth-client`
- `NextAuthOptions` → `betterAuth()` configuration
- `[...nextauth].ts` → `[...all]/route.ts`
- `SessionProvider` → Built-in session management
- `getServerSession` → `auth.$Infer.Session`

Benefits of Better Auth

- **Smaller bundle size**: More lightweight than NextAuth
- **Better TypeScript support**: Full type safety
- **Simpler configuration**: Less boilerplate code
- **Modern architecture**: Built for React Server Components
- **Better performance**: Optimized for Next.js 15

Troubleshooting

Common Issues

1. **Database connection errors**: Check DATABASE_URL and DIRECT_DATABASE_URL
2. **Google OAuth failures**: Verify GOOGLE_CLIENT_ID and redirect URIs
3. **Session not persisting**: Check BETTER_AUTH_SECRET and baseURL
4. **CORS issues**: Ensure BETTER_AUTH_URL matches your domain

Debug Steps

1. Check environment variables are loaded
2. Verify database schema is up to date
3. Test API routes directly
4. Check browser network tab for auth requests
5. Verify Google Cloud Console configuration

Security Considerations

Best Practices Implemented

- **Environment variables**: Sensitive data not in code
- **Secure sessions**: Encrypted with random secret
- **HTTPS in production**: Secure data transmission
- **Input validation**: Sanitized user inputs
- **Database constraints**: Proper foreign keys and unique constraints

This documentation provides a complete overview of the Better Auth implementation in your DocuSeal application.