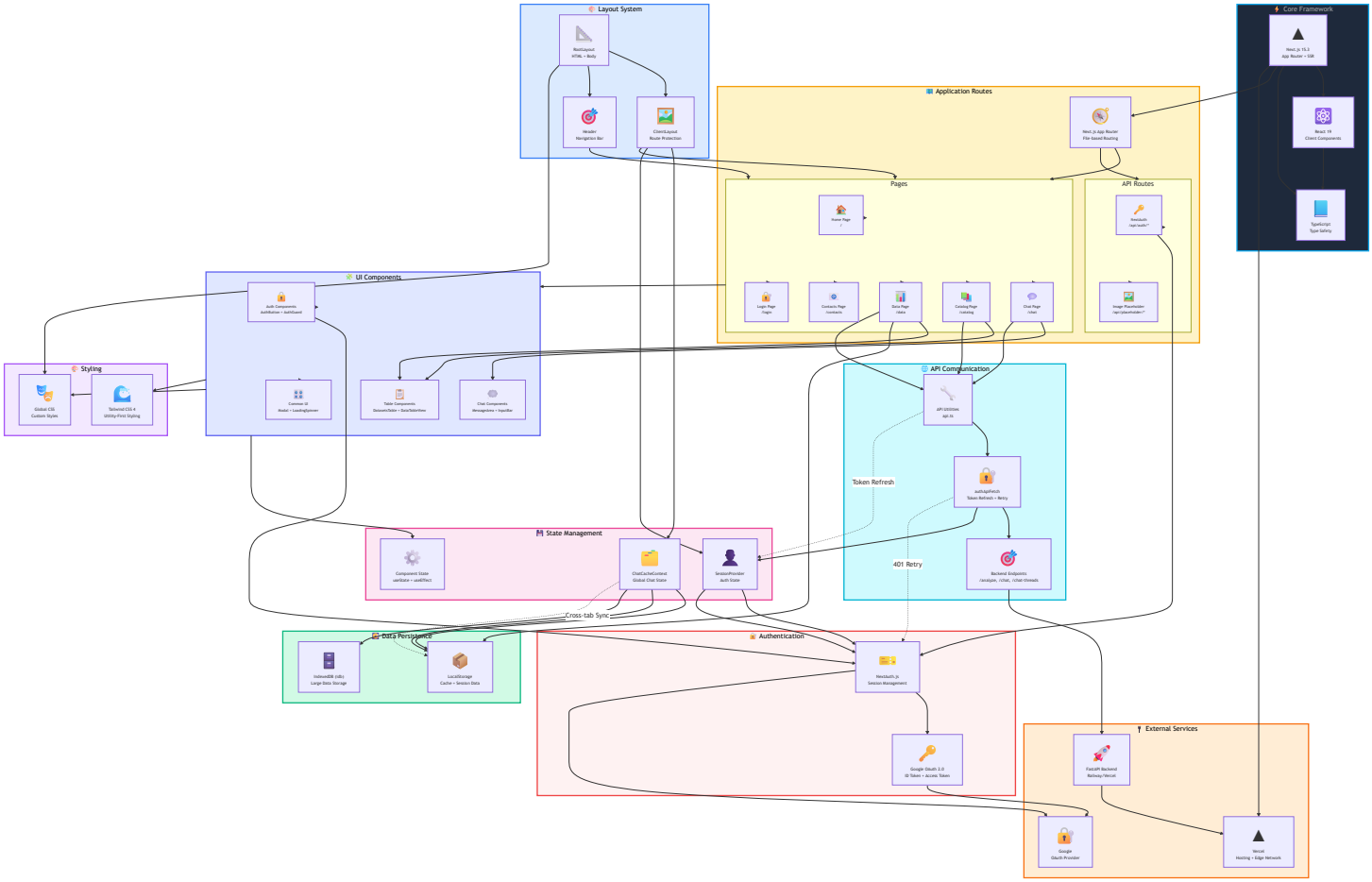
Frontend Architecture Diagram



Architecture Overview

🏗️ Key Architectural Patterns

1. **Component-Based Architecture**: React components organized by feature and reusability
2. **Layered Architecture**: Clear separation between routing, layout, UI, state, and services
3. **Context Pattern**: Global state management using React Context API
4. **Repository Pattern**: Centralized API communication via `api.ts`
5. **Protected Route Pattern**: Authentication guards on private routes
6. **Token Refresh Pattern**: Automatic token refresh on 401 responses

📋 Core Features

1. Routing & Navigation

- File-based routing using Next.js App Router
- Server and client components separation
- Dynamic route parameters for flexible navigation
- API routes for server-side endpoints

2. State Management

- **ChatCacheContext**: Manages chat threads, messages, pagination
  - LocalStorage persistence for offline support
  - Cross-tab synchronization for consistent state
  - Optimistic updates for better UX
- **SessionProvider**: Manages authentication state
  - Token refresh mechanism
  - Session persistence across page reloads

3. Authentication Flow

- Google OAuth 2.0 integration via NextAuth
- Protected routes with AuthGuard component
- Automatic token refresh on expiry
- Session state synchronization

4. Data Flow

```
User Action → Component → Context/State → API Utils → Backend
     ↓                                                    ↓
  UI Update ← Component ← Context Update ← Response ← Backend
```

5. Caching Strategy

- **Hot Data**: ChatCacheContext (in-memory)
- **Warm Data**: LocalStorage (48-hour cache)
- **Cold Data**: IndexedDB (large datasets)
- **Pagination**: Incremental loading (10 threads per page)

6. **Performance Optimizations**

- Lazy loading of components with Suspense
- Pagination for large data sets (threads, messages)
- Debounced search inputs
- Memoized expensive computations
- Automatic code splitting via Next.js

🔄 Data Flow Patterns

Chat Message Flow:

1. User enters message in `InputBar`
2. Message sent via `authApiFetch` to backend `/analyze`
3. Optimistic update in `ChatCacheContext`
4. Backend processes with LangGraph agent
5. Response received and cached
6. UI updates via `MessageArea`

Authentication Flow:

1. User clicks login in `AuthButton`
2. Redirected to Google OAuth
3. NextAuth handles callback
4. Session created and stored
5. Protected routes become accessible
6. Token auto-refreshes on expiry

Data Persistence Flow:

1. State changes in `ChatCacheContext`
2. Automatic save to LocalStorage
3. Cross-tab synchronization via storage events
4. Page refresh loads from LocalStorage
5. Stale data triggers API refresh

🎯 Component Hierarchy

```
RootLayout (Server)
└── SessionProviderWrapper (Client)
    └── ClientLayout (Client)
        ├── Header
        │   ├── Navigation Links
        │   └── AuthButton
        └── Pages
            ├── ChatPage
            │   ├── Sidebar (Thread List)
            │   ├── MessageArea
            │   ├── InputBar
            │   └── FollowupPrompts
            ├── CatalogPage
            │   └── DatasetsTable
            ├── DataPage
            │   └── DataTableView
            └── Other Pages
```

📦 Technology Stack Summary

Layer	Technologies
Framework	Next.js 15.3, React 19, TypeScript
Styling	Tailwind CSS 4, Custom CSS
Authentication	NextAuth, Google OAuth 2.0
State Management	React Context API, useState/useEffect
Data Persistence	LocalStorage, IndexedDB (idb)
API Communication	Fetch API, Custom authApiFetch wrapper
Utilities	uuid, markdown-to-jsx
Deployment	Vercel Edge Network

🔐 Security Measures

1. **Token Management**: Automatic refresh on 401 responses

2. **Route Protection**: AuthGuard wraps protected pages
3. **CSRF Protection**: NextAuth built-in protection
4. **Secure Storage**: Sensitive data never in LocalStorage
5. **HTTPS Only**: All communication over secure channels

🚀 Deployment Architecture

```
User Browser
    ↓
Vercel Edge Network (CDN)
    ↓
Next.js Frontend (Vercel Serverless)
    ↓ (via /api proxy)
FastAPI Backend (Railway/Vercel)
```

Note: This diagram represents the current frontend architecture as of the implementation. The architecture follows modern React and Next.js best practices with emphasis on performance, security, and user experience.