Academy Frontend Project Documentation

1. Project Overview

Name: Academy Booking Frontend **Purpose**: A web application for managing academy bookings, hall reservations, and event scheduling. **Tech Stack**:

- **Framework**: Next.js 14+ (App Router)
- **Language**: TypeScript
- **Styling**: Tailwind CSS
- **Icons**: Lucide React
- **Animation**: Framer Motion
- **State Management**: React Context ([AuthContext](file:///d:/AcademyFrontend/academy-frontend/context/auth-context.tsx#17-25))

2. System Design & Architecture

Architecture: Client-Side Rendering with Mock API

The application is designed as a **Single Page Application (SPA)** feel using Next.js Client Components for interactivity.

- **Frontend Layer**: React components handling UI and user interaction.
- **Data Layer ([/lib/api.ts](file:///d:/AcademyFrontend/academy-frontend/lib/api.ts))**: A mock service layer that simulates backend calls. It returns promises with delays to mimic network requests.
- **Auth Layer ([/context/auth-context.tsx](file:///d:/AcademyFrontend/academy-frontend/context/auth-context.tsx))**: Manages user sessions, login state (`isAuthenticated`), and user profile data. It persists state in memory (for this demo version).

Key Design Patterns

- **Layout Pattern**: Usage of [layout.tsx](file:///d:/AcademyFrontend/academy-frontend/app/layout.tsx) for persistent UI (Header/Footer) and global auth checks.
- **Context Pattern**: Global accessible state for Authentication.
- **Container/Presenter**: Separation of logic (hooks, API calls) from pure UI components where applicable.
- **Guard Pattern**: Protected routes redirect unauthenticated users to Login via [layout.tsx](file:///d:/AcademyFrontend/academy-frontend/app/layout.tsx) and specific page logic.

3. File Structure & Organization

```
academy-frontend/
├── app/                        # Next.js App Router (Pages & Layouts)
│   ├── (public)/               # Publicly accessible pages without user l
│   │   └── page.tsx            # Landing Page (Home)
│   ├── (auth)/                 # Authentication pages
│   │   └── login/page.tsx      # Login Page
│   ├── (user)/                 # User-facing pages (Wrapped in UserLayout
│   │   ├── layout.tsx          # Main Layout (Header, Footer, Auth Check)
│   │   ├── book/               # Booking Page
│   │   ├── calendar/           # Calendar View
│   │   ├── my-bookings/        # User's Booking Management
│   │   └── about/              # About Us Page
│   └── layout.tsx              # Root Layout (Fonts, Metadata)
├── components/                 # Reusable UI Components
│   ├── academy/                # Academy & Hall Cards/Lists
│   ├── booking/                # Booking Forms & Stats
│   ├── calendar/               # Calendar Views
│   ├── home/                   # Landing Page Sections (Locations, Hero)
│   ├── layout/                 # Header & Footer
│   └── ui/                     # Shadcn/Radix UI Primitives (Button, Inpu
├── context/
│   └── auth-context.tsx        # Authentication Provider
├── lib/
│   └── api.ts                  # Mock API Service
└── types/                      # TypeScript Definitions
```

4. User Flows

4.1. Public Navigation Flow

1. **Landing Page**: Users arrive at `/`.
    - **Action**: Click **"Book Your Place"** -> Redirects to `/book`.
    - **Action**: Click an **Academy Location Card** -> Redirects to `/book?academyId=...` (Pre-filters the list).
2. **Public Access**: Users can freely visit `/book`, `/calendar`, and `/about` without logging in.
3. **Navbar**: "Book" and "Calendar" tabs are visible to all users.

4.2. Booking Process (The Core Flow)

1. **Discovery**: User views the **Hall List** on `/book`.
    - **Filter**: User can filter halls by "Academy" using the dropdown.
2. **Selection**: User clicks **"Book Now"** on a Hall Card.
3. **Auth Check**:
    - **If Guest**: Redirected to `/login`. After login, redirected back to booking.

    - **If User**: Redirected to `/book?action=book&hallId=...`.
4. **Booking Form**:
    - The **Booking Form** component renders (only for authenticated users).
    - User fills details (Date, Time, Event Name).
    - **Submit**: Mock submission to API.
5. **Confirmation**: User is redirected to `/my-bookings` to see their new request.

4.3. Authentication Flow

1. **Login**: accessing `/login`.
2. **Credential Check**: Mock verification (accepts any email/password for demo, except specific test cases if defined).
3. **Session Start**: [AuthContext](file:///d:/AcademyFrontend/academy-frontend/context/auth-context.tsx#17-25) updates `isAuthenticated = true`.
4. **Redirect**: User sent to Dashboard or previous attempted page.
5. **Logout**: Clears session, redirects to `/login`.

4.4. Management Flow

1. **My Bookings**: internal route `/my-bookings`.
2. **Access**: PROTECTED. Guests are redirected to login.
3. **View**: User sees list of their past/upcoming bookings with status (Approved/Pending).

5. Key Components Details

- **[Header](file:///d:/AcademyFrontend/academy-frontend/components/layout/header.tsx#13-181) ([components/layout/header.tsx](file:///d:/AcademyFrontend/academy-frontend/components/layout/header.tsx))**:

    - Responsive navigation bar.
    - Handles showing/hiding links based on `isAuthenticated`.
    - Shows User Profile if logged in, "Login" button if guest.

- **[HallCard](file:///d:/AcademyFrontend/academy-frontend/components/academy/hall-card.tsx#14-82) ([components/academy/hall-card.tsx](file:///d:/AcademyFrontend/academy-frontend/components/academy/hall-card.tsx))**:

    - Displays Hall image, name, and parent Academy.
    - Contains the "Book Now" logic (Router push + Auth check).

- **[BookPage](file:///d:/AcademyFrontend/academy-frontend/app/%28user%29/book/page.tsx#93-100) ([app/(user)/book/page.tsx](file:///d:/AcademyFrontend/academy-frontend/app/%28user%29/book/page.tsx))**:

    - **State**: Manages `selectedAcademy` filter.
    - **Render**: Switches between [HallList](file:///d:/AcademyFrontend/academy-frontend/components/academy/hall-list.tsx#12-71) (View Mode) and [BookingForm](file:///d:/AcademyFrontend/academy-frontend/components/booking/booking-form.tsx#39-619) (Book Mode).
    - **Effect**: Syncs URL parameters (`?academyId=1`) with local state.

- **[AuthContext](file:///d:/AcademyFrontend/academy-frontend/context/auth-context.tsx#17-25) ([context/auth-context.tsx](file:///d:/AcademyFrontend/academy-frontend/context/auth-context.tsx))**:

    - Exposes: `user`, [login(email, pass)](file:///d:/AcademyFrontend/academy-frontend/context/auth-context.tsx#48-85), [logout()](file:///d:/AcademyFrontend/academy-frontend/context/auth-context.tsx#86-92), `isAuthenticated`, `isLoading`.
    - Wraps the entire application to provide session state everywhere.

6. How to Use this Document

To save this as a PDF:

1. **VS Code**: Open this markdown file -> Open Command Palette (`Ctrl+Shift+P`) -> Type "Markdown PDF: Export (pdf)". (Requires extension).
2. **Browser**: Copy this content to a markdown viewer (like StackEdit.io) or print the raw view to PDF.