Introduction

Stacks t provides library discovery and catalog browsing across multiple library systems. Users can search for books via the Open Library API, locate nearby libraries using geolocation, and browse pre-scraped catalogs from Houston Public Library and LA County Library.

System Objectives

The Stacks application aims to:

1. Simplify Book Discovery: Search millions of books via Open Library API and locate availability in nearby libraries
2. Location-Based Library Search: Identify closest library branches with geolocation support
3. Catalog Browsing: Browse curated library catalogs by category
4. Intuitive Navigation: Category-based organization, alphabetized listings, and search bars
5. User Profiles: Save favorite items and track usage patterns
6. Accessibility & Responsiveness: Support all experience levels on desktop and mobile

Hardware, Software, and Human Interfaces

Browser Interface

- Supported: Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
- APIs: Geolocation, Fetch, Local Storage

External APIs

- Open Library API: Search books (/search.json), book details (/works/{id}.json), covers
- Nominatim API: Forward/reverse geocoding (/search, /reverse), country filtering

Frameworks & Libraries

- React 18: Component-based UI rendering, state management
- React Router DOM 6: SPA navigation
- Vite 5: Fast dev server, hot module replacement

User Interface

- NavBar: Home, Discover, Browse, Libraries, Profile
- Main content: Card-based item/library listings, responsive design
- Font: Inter (UI), Crimson Pro (headings)
- Colors: Primary Blue (#3b82f6), text dark gray (#1a1a1a), background white/light gray (#f9fafb)

Hardware Interfaces

- Input: Mouse, keyboard, touch
- Display: 320x568 minimum, 1920x1080 recommended
- Geolocation: GPS/WiFi/IP-based with permission prompt

Architectural Design

The Stacks application follows SPA architecture with a React-adapted MVC pattern:

- Model: Data adapters and service modules
- View: React UI components
- Controller: State and event handlers

Navigation is handled client-side; external APIs are queried on demand.

Major Software Components (CSCs)

CSC	Purpose	Key Modules
Routing & Navigation	Manage app routes	App.jsx, Navigation.jsx, Footer.jsx
Page Components	Top-level views	Home.jsx, Discover.jsx, Results.jsx, Library.jsx, Libraries.jsx, CategoryItems.jsx, Item.jsx, Profile.jsx
Shared UI Components	Reusable UI elements	Navigation.jsx, Footer.jsx, BookCard.jsx, LibraryCard.jsx, SearchBar.jsx
Data Adapters & Services	API integration, data transformation	openlibrary.js, finder.js, geocoding.js, bookIdentity.js
Library Registry	Static library data	california-libraries.js, socal-libraries.js

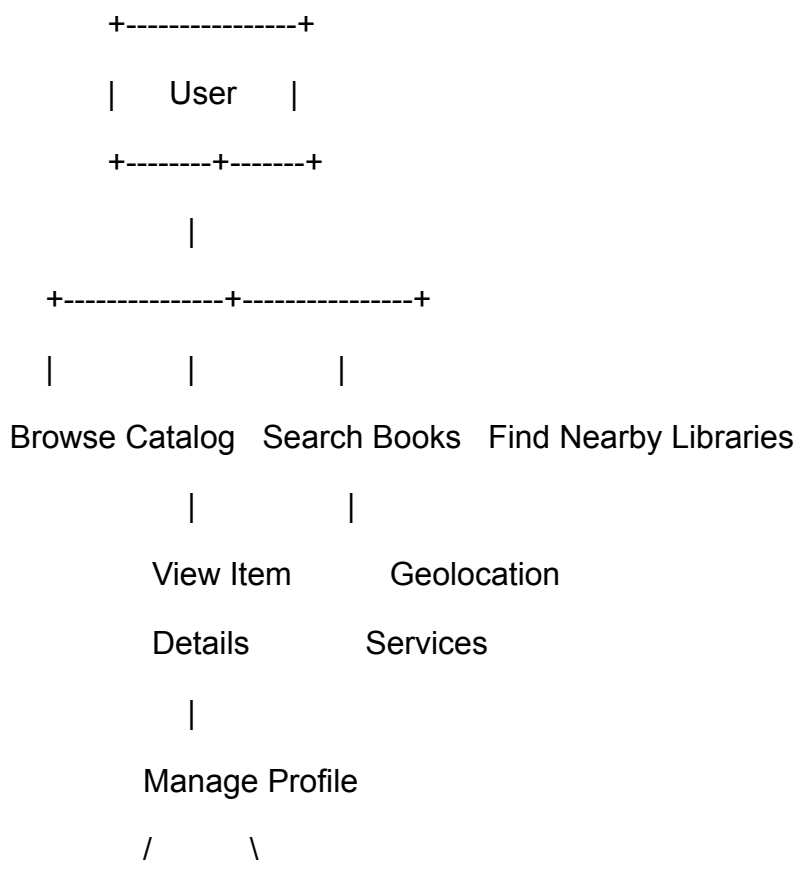
Scraper Subsystem	Catalog data extraction	scraper/ (Python runtime)

Major Software Interactions

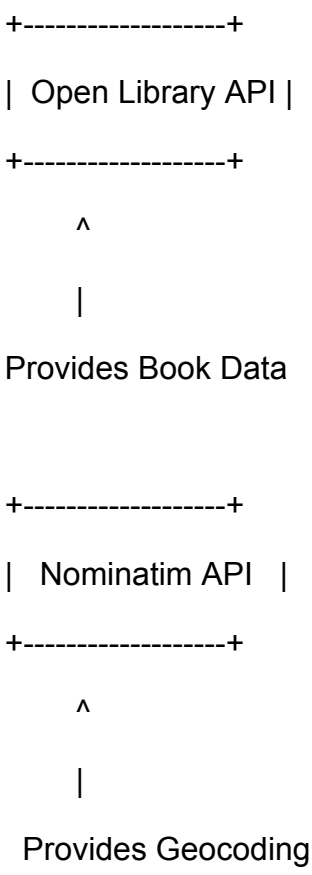
1. Page ↔ Data Adapters: Direct function calls (async), JavaScript objects, error handled via try/catch.
2. Adapters ↔ External APIs: HTTPS REST calls, JSON format, network errors handled with context.
3. Routing ↔ Page Components: React Router passing route parameters via useParams.
4. Discover Page ↔ Geolocation Services: Calls getCurrentLocation() and findNearbyLibraries(), fallback input if user denies permission.
5. Browser Storage ↔ Application State: localStorage for favorites/preferences; JSON serialization with error handling.

Architectural Diagrams

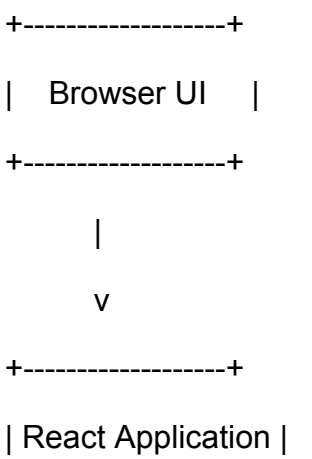
1. Use Case Diagram

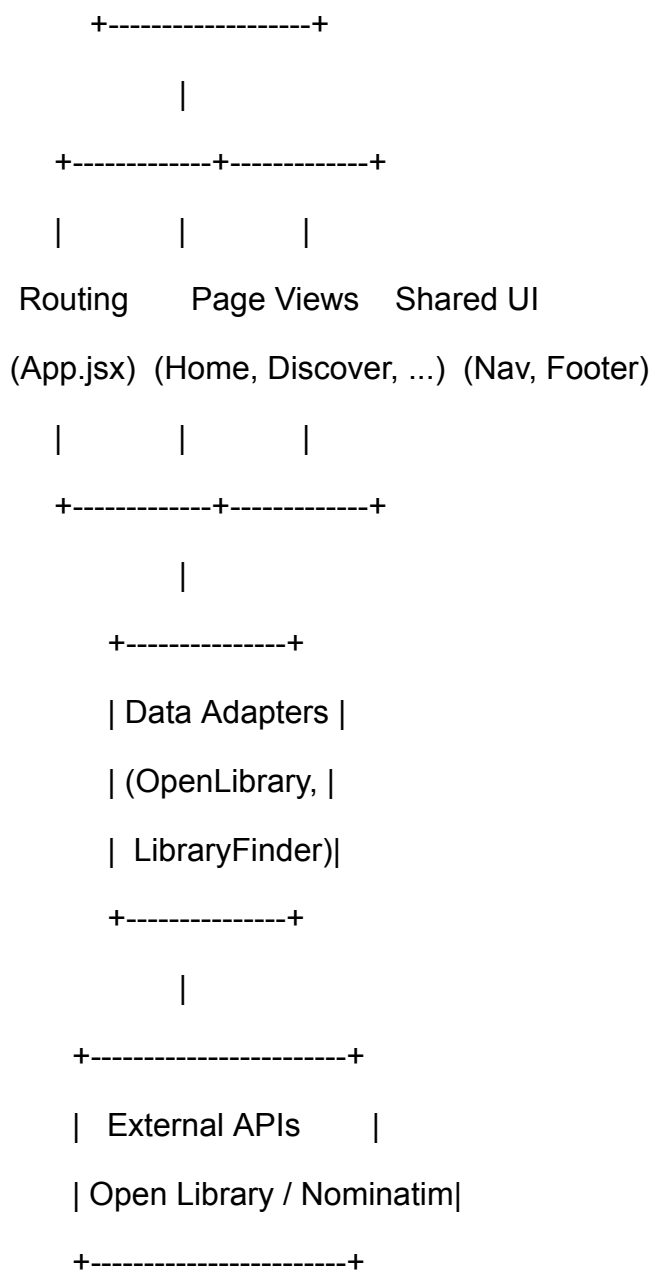


Save Favorites View History

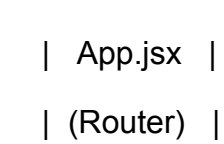


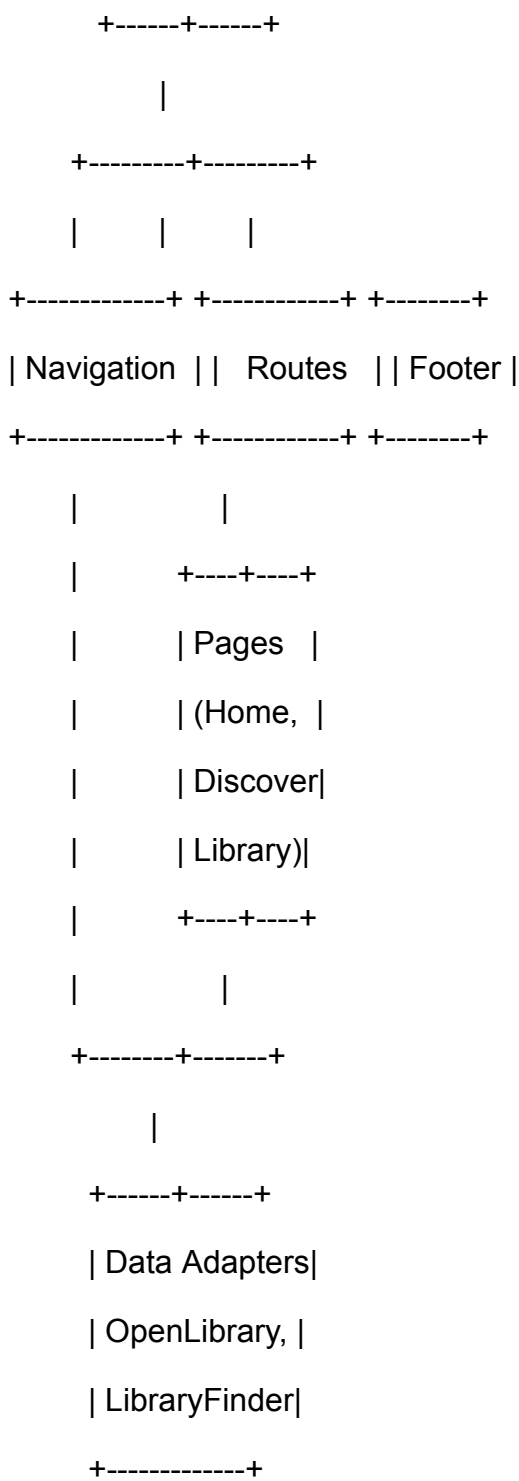
2. Component Diagram



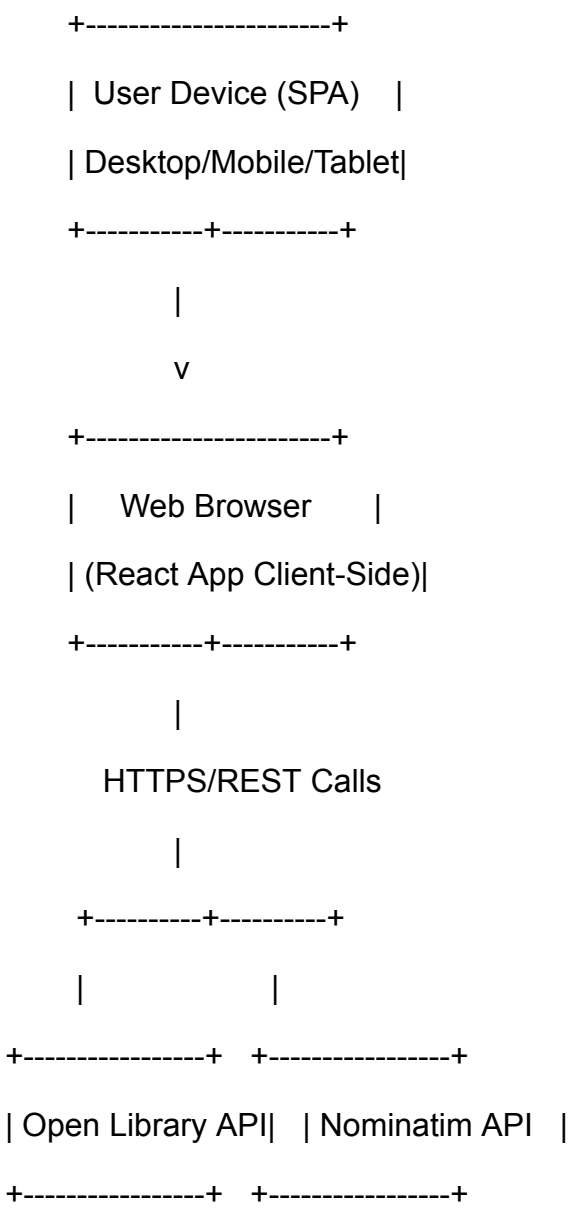


3. Class Diagram





4. Deployment Diagram: User Device (browser) ↔ HTTPS ↔ Web Server ↔ External APIs (Open Library, Nominatim).



CSC and CSU Descriptions

Routing and Navigation

- CSU: App.jsx: Manages routes, renders page components
- CSU: Navigation.jsx: Renders navigation bar
- CSU: Footer.jsx: Site footer

Page Components

- CSUs: Home.jsx, Discover.jsx, Results.jsx, Library.jsx, Libraries.jsx, CategoryItems.jsx, Item.jsx, Profile.jsx
- Responsibilities: Layout, data fetching, state management, user interactions

Data Adapters & Services

- CSU: OpenLibraryAdapter: Search books, get details, fetch covers
- CSU: LibraryFinder: Find nearby libraries, calculate distance, geocode location

Library Registry

- Stores static library metadata and coordinates for distance calculation

Scraper Subsystem

- Extracts, cleans, and exports library catalog data (Python runtime, separate from SPA)

Database Design and Description

The Stacks application does not use a backend database; all data is fetched dynamically from APIs or pre-scraped JSON catalogs. No database ER diagrams or access/security policies are required.