Enterprise Intranet Search Interface - Implementation Plan

Overview

Build a comprehensive Angular search interface for an employee intranet that connects to an existing Elasticsearch backend. The interface will provide search functionality, results display, filtering, sorting, and autocomplete features as specified in the PRD.

Architecture

```
graph TB
    SearchHome[Search Homepage Component] --> SearchService[Search Service]
    ResultsPage[Search Results Component] --> SearchService
    Autocomplete[Autocomplete Component] --> SearchService
    Filters[Filter Sidebar Component] --> SearchService
    SearchService --> API[Elasticsearch API]
    SearchService --> Models[Data Models/Interfaces]
    Models --> ResultItem[SearchResult Interface]
    Models --> FilterOptions[FilterOptions Interface]
    Models --> SearchQuery[SearchQuery Interface]
```

Project Structure

```
src/
├── app/
│   ├── search/
│   │   ├── search-home/
│   │   │   ├── search-home.component.ts
│   │   │   ├── search-home.component.html
│   │   │   ├── search-home.component.scss
│   │   │   └── search-home.component.spec.ts
│   │   ├── search-results/
│   │   │   ├── search-results.component.ts
│   │   │   ├── search-results.component.html
│   │   │   ├── search-results.component.scss
│   │   │   └── search-results.component.spec.ts
│   │   ├── search-bar/
│   │   │   ├── search-bar.component.ts
│   │   │   ├── search-bar.component.html
│   │   │   ├── search-bar.component.scss
│   │   │   └── search-bar.component.spec.ts
│   │   ├── autocomplete/
│   │   │   ├── autocomplete.component.ts
│   │   │   ├── autocomplete.component.html
│   │   │   └── autocomplete.component.scss
│   │   ├── filter-sidebar/
│   │   │   ├── filter-sidebar.component.ts
│   │   │   ├── filter-sidebar.component.html
│   │   │   └── filter-sidebar.component.scss
│   │   ├── result-item/
│   │   │   ├── result-item.component.ts
│   │   │   ├── result-item.component.html
│   │   │   └── result-item.component.scss
│   │   ├── search.service.ts
│   │   ├── search.models.ts
│   │   └── search-routing.module.ts
│   └── app-routing.module.ts
├── assets/
└── styles/
    └── _search-theme.scss
```

Implementation Components

1. Data Models (`search.models.ts`)

Define TypeScript interfaces for:

- `SearchResult` : title, snippet, source, author, lastModified, fileType, url, thumbnail, breadcrumb
- `SearchResponse` : results array, totalCount, searchTime, query
- `FilterOptions` : contentTypes, departments, dateRange, authors, fileFormats, sourceSystems
- `SearchQuery` : query string, filters, sort, page, pageSize
- `AutocompleteSuggestion` : text, type, count
- `SearchHistory` : query, timestamp

2. Search Service (`search.service.ts`)

Angular service handling:

- `search(query: SearchQuery): Observable<SearchResponse>` - Main search method
- `getAutocompleteSuggestions(query: string): Observable<AutocompleteSuggestion[]>` - Autocomplete
- `getTrendingSearches(): Observable<string[]>` - Trending searches
- `getSearchHistory(): Observable<SearchHistory[]>` - User search history
- `saveSearchHistory(query: string): void` - Persist search history (localStorage)
- Query parameter building for Elasticsearch API
- Error handling and retry logic

3. Search Homepage Component (`search-home.component.ts`)

Features:

- Prominent search bar (reusable component)
- Trending searches display
- Quick links to frequently accessed resources
- Search tips section
- Link to advanced search
- Responsive layout with Angular Material Grid

4. Search Bar Component (`search-bar.component.ts`)

Reusable component with:

- Material input with search icon
- Autocomplete integration (minimum 3 characters)
- Query history dropdown
- Search button
- Support for natural language queries
- Advanced search link
- Emit search events to parent

5. Autocomplete Component (`autocomplete.component.ts`)

Features:

- Dropdown suggestions overlay
- Highlight matching text
- Keyboard navigation (arrow keys, enter)
- Click to select suggestion
- Debounced API calls (200ms delay)
- Display suggestion type/category

6. Search Results Component (`search-results.component.ts`)

Main results page with:

- Persistent search bar at top
- Results count and search time display
- Result items list with pagination
- Empty state for zero results
- Loading state with skeleton loaders
- Error state handling
- Responsive layout (sidebar on desktop, collapsible on mobile)

7. Result Item Component (`result-item.component.ts`)

Individual result display:

- Title (clickable link)
- Highlighted snippet with search term highlighting
- Metadata: source system, author, last modified date, file type
- Thumbnail preview (for images/videos)
- Breadcrumb navigation
- "Was this helpful?" feedback button
- Material card layout

8. Filter Sidebar Component (`filter-sidebar.component.ts`)

Filtering interface:

- Content type filter (multi-select chips)
- Department/team filter
- Date range picker (Material datepicker)
- Author filter (autocomplete)
- File format filter
- Source system filter
- Active filters display with remove buttons

- Clear all filters button
- Collapsible on mobile (Material sidenav)

9. Sorting and Pagination

- Sort dropdown: Relevance, Date (newest/oldest), Title (A-Z), Author
- Results per page selector: 10, 25, 50
- Pagination component (Material paginator)
- URL query parameters for shareable links

10. Routing (`search-routing.module.ts`)

Routes:

- `/search` - Search homepage
- `/search/results?q=...&filters=...` - Search results page
- Query parameter handling for filters, sort, page

Key Features Implementation

Search Modifiers Support

- Boolean operators (AND, OR, NOT) parsing
- Phrase matching with quotation marks
- Field-specific search (author:, title:, date:, type:)
- Wildcard support (* and ?)

Query Understanding

- Spell check suggestions (if API provides)
- Synonym expansion (if API provides)
- Search term highlighting in results

Personalization

- User context (department, role) from service
- Recently viewed documents boost (if available)
- Personalized suggestions

Responsive Design

- Mobile-first approach
- Breakpoints: mobile (<768px), tablet (768-1024px), desktop (>1024px)
- Collapsible filter sidebar on mobile
- Touch-friendly interactions

Accessibility

- WCAG 2.1 Level AA compliance
- Keyboard navigation support
- ARIA labels and roles
- Screen reader friendly
- Focus management

Styling Approach

- Angular Material theme customization
- Custom SCSS for search-specific components

- Material components: Input, Button, Card, Chip, Select, Datepicker, Paginator, Sidenav
- Consistent spacing and typography
- Dark mode support (if theme supports)

API Integration Assumptions

Since existing API endpoints are available, the service will:

- Accept base API URL via environment configuration
- Assume RESTful endpoints:
  - `POST /api/search` - Main search endpoint
  - `GET /api/search/autocomplete?q=...` - Autocomplete
  - `GET /api/search/trending` - Trending searches
- Handle authentication tokens (if required)
- Support CORS configuration

Testing Considerations

- Unit tests for service methods
- Component tests for user interactions
- Mock API responses for testing
- Accessibility testing
- Responsive design testing

Dependencies

- @angular/core (latest)
- @angular/material (latest)
- @angular/cdk (latest)
- @angular/common/http
- rxjs (for observables)
- TypeScript 5.x

Implementation Order

1. **Setup**: Angular project structure, Material setup, routing
2. **Models & Service**: Data interfaces and API service layer
3. **Search Bar & Autocomplete**: Core search input functionality
4. **Search Homepage**: Landing page with search bar
5. **Results Display**: Basic results list and result item components
6. **Filtering**: Filter sidebar and filter logic
7. **Sorting & Pagination**: Sort options and pagination controls
8. **Polish**: Styling, accessibility, responsive design, error handling
9. **Testing**: Unit and integration tests

Future Enhancements (Out of Scope)

- AI-powered answer generation
- Voice search
- Visual search
- Browser extension
- Advanced analytics dashboard