**React Native Proposals Folder Reorganization**

**New Folder Structure**

src/

├── components/

│ ├── common/

│ │ ├── Header.tsx

│ │ ├── SearchBar.tsx

│ │ ├── FilterButton.tsx

│ │ └── LoadingSpinner.tsx

│ └── proposals/ # ← New consolidated folder

│ ├── components/ # UI Components

│ │ ├── ProposalCard.tsx

│ │ ├── ProposalList.tsx

│ │ ├── SectorFilter.tsx

│ │ ├── StatusBadge.tsx

│ │ ├── CalendarModal.tsx

│ │ └── DateRangePicker.tsx

│ ├── screens/ # Screens

│ │ ├── AllProposalsScreen.tsx

│ │ ├── BulkDownloadScreen.tsx

│ │ └── ProposalDetailsScreen.tsx

│ ├── services/ # API Services

│ │ ├── proposalService.ts

│ │ └── downloadService.ts

│ ├── types/ # Type definitions

│ │ └── proposalTypes.ts

│ ├── utils/ # Utilities

│ │ ├── proposalHelpers.ts

│ │ └── proposalConstants.ts

│ └── index.ts # Barrel exports

├── services/

│ └── api.ts # Keep global API service

├── types/

│ └── index.ts # Keep global types

├── utils/

│ ├── constants.ts

│ └── helpers.ts

└── navigation/

└── AppNavigator.tsx

**File Movements Required**

**1. Move Files to New Structure**

# Create the new folder structure

mkdir -p src/components/proposals/{components,screens,services,types,utils}

# Move proposal components

mv src/components/proposal/ProposalCard.tsx src/components/proposals/components/

mv src/components/proposal/ProposalList.tsx src/components/proposals/components/

mv src/components/proposal/SectorFilter.tsx src/components/proposals/components/

mv src/components/proposal/StatusBadge.tsx src/components/proposals/components/

# Move calendar components to proposals

mv src/components/calendar/CalendarModal.tsx src/components/proposals/components/

mv src/components/calendar/DateRangePicker.tsx src/components/proposals/components/

# Move screens

mv src/screens/AllProposalsScreen.tsx src/components/proposals/screens/

mv src/screens/BulkDownloadScreen.tsx src/components/proposals/screens/

# Move proposal service

mv src/services/proposalService.ts src/components/proposals/services/

# Remove old empty directories

rmdir src/components/proposal

rmdir src/components/calendar

rmdir src/screens # if empty

**2. Create New Type File**

**src/components/proposals/types/proposalTypes.ts**

// Move all proposal-related types here from src/types/index.ts

export interface Proposal {

id: string;

proposal\_number: string;

policy\_number: string;

customer\_name: string;

sector: string;

package\_type: string;

status: 'Initiated' | 'Action Required' | 'Closed';

status\_color: 'blue' | 'red' | 'green';

inward\_code: string;

posted\_date: string;

premium: number;

currency: string;

additional\_info: string;

download\_available: boolean;

}

export interface ProposalsResponse {

success: boolean;

data: {

proposals: Proposal[];

total\_count: number;

current\_page: number;

total\_pages: number;

};

}

export interface Sector {

id: string;

name: string;

is\_default: boolean;

}

export interface Status {

id: string;

name: string;

color: string;

}

export interface FilterParams {

page?: number;

limit?: number;

sector?: string;

status?: string;

from\_date?: string;

to\_date?: string;

search?: string;

}

export interface BulkDownloadRequest {

proposal\_ids: string[];

filters: any;

download\_format: 'xlsx' | 'pdf' | 'csv';

}

**3. Create Constants File**

**src/components/proposals/utils/proposalConstants.ts**

export const PROPOSAL\_STATUSES = {

INITIATED: 'Initiated',

ACTION\_REQUIRED: 'Action Required',

CLOSED: 'Closed',

} as const;

export const STATUS\_COLORS = {

BLUE: 'blue',

RED: 'red',

GREEN: 'green',

} as const;

export const DEFAULT\_SECTORS = ['All', 'Motor', 'Health', 'Commercial'];

export const DOWNLOAD\_FORMATS = ['xlsx', 'pdf', 'csv'] as const;

export const PAGINATION\_CONFIG = {

DEFAULT\_PAGE\_SIZE: 20,

LOAD\_MORE\_THRESHOLD: 0.5,

};

export const API\_ENDPOINTS = {

PROPOSALS: '/proposals',

SECTORS: '/proposals/sectors',

STATUSES: '/proposals/statuses',

DATE\_RANGE: '/proposals/date-range',

BULK\_DOWNLOAD: '/proposals/bulk-download',

};

**4. Create Helper Functions**

**src/components/proposals/utils/proposalHelpers.ts**

import { Proposal, FilterParams } from '../types/proposalTypes';

export const formatCurrency = (amount: number, currency: string): string => {

return `${currency}${amount.toLocaleString()}`;

};

export const formatDate = (dateString: string): string => {

const date = new Date(dateString);

return date.toLocaleDateString('en-US', {

day: '2-digit',

month: 'short',

year: '2-digit',

});

};

export const getStatusStyle = (color: 'blue' | 'red' | 'green') => {

switch (color) {

case 'blue':

return { backgroundColor: '#e3f2fd', color: '#1976d2' };

case 'red':

return { backgroundColor: '#ffebee', color: '#d32f2f' };

case 'green':

return { backgroundColor: '#e8f5e8', color: '#2e7d32' };

default:

return { backgroundColor: '#f5f5f5', color: '#666' };

}

};

export const buildQueryParams = (params: FilterParams): string => {

const queryParams = new URLSearchParams();

Object.entries(params).forEach(([key, value]) => {

if (value !== undefined && value !== null && value !== '') {

queryParams.append(key, value.toString());

}

});

return queryParams.toString();

};

export const filterProposalsByStatus = (

proposals: Proposal[],

status?: string

): Proposal[] => {

if (!status || status === 'All') return proposals;

return proposals.filter(proposal => proposal.status === status);

};

export const filterProposalsBySector = (

proposals: Proposal[],

sector?: string

): Proposal[] => {

if (!sector || sector === 'All') return proposals;

return proposals.filter(proposal =>

proposal.sector.toLowerCase() === sector.toLowerCase()

);

};

**5. Create Barrel Export File**

**src/components/proposals/index.ts**

// Components

export { default as ProposalCard } from './components/ProposalCard';

export { default as ProposalList } from './components/ProposalList';

export { default as SectorFilter } from './components/SectorFilter';

export { default as StatusBadge } from './components/StatusBadge';

export { default as CalendarModal } from './components/CalendarModal';

export { default as DateRangePicker } from './components/DateRangePicker';

// Screens

export { default as AllProposalsScreen } from './screens/AllProposalsScreen';

export { default as BulkDownloadScreen } from './screens/BulkDownloadScreen';

// Services

export { default as ProposalService } from './services/proposalService';

// Types

export \* from './types/proposalTypes';

// Utils

export \* from './utils/proposalConstants';

export \* from './utils/proposalHelpers';

**6. Create Dedicated Proposal Service**

**src/components/proposals/services/proposalService.ts**

import ApiService from '../../../services/api';

import { ProposalsResponse, FilterParams, BulkDownloadRequest } from '../types/proposalTypes';

import { buildQueryParams, API\_ENDPOINTS } from '../utils/proposalHelpers';

class ProposalService {

async getProposals(params: FilterParams = {}): Promise<ProposalsResponse> {

const queryString = buildQueryParams(params);

const endpoint = `${API\_ENDPOINTS.PROPOSALS}${queryString ? `?${queryString}` : ''}`;

return ApiService.request<ProposalsResponse>(endpoint);

}

async getSectors() {

return ApiService.request<{success: boolean; data: {sectors: Sector[]}}>(

API\_ENDPOINTS.SECTORS

);

}

async getStatuses() {

return ApiService.request<{success: boolean; data: {statuses: Status[]}}>(

API\_ENDPOINTS.STATUSES

);

}

async getDateRange() {

return ApiService.request<{

success: boolean;

data: {

min\_date: string;

max\_date: string;

default\_range: {from: string; to: string};

}

}>(API\_ENDPOINTS.DATE\_RANGE);

}

async bulkDownload(request: BulkDownloadRequest) {

return ApiService.request(API\_ENDPOINTS.BULK\_DOWNLOAD, {

method: 'POST',

body: JSON.stringify(request),

});

}

async downloadSingleProposal(proposalId: string, format: string = 'pdf') {

return ApiService.request(`${API\_ENDPOINTS.PROPOSALS}/${proposalId}/download`, {

method: 'POST',

body: JSON.stringify({ format }),

});

}

}

export default new ProposalService();

**Import Statement Updates Required**

**1. Update AllProposalsScreen.tsx**

// OLD imports

import StatusBadge from '../components/proposal/StatusBadge';

import ProposalCard from '../components/proposal/ProposalCard';

import SectorFilter from '../components/proposal/SectorFilter';

import ApiService from '../services/api';

import {Proposal, FilterParams} from '../types';

// NEW imports

import { StatusBadge, ProposalCard, SectorFilter } from '../components/proposals';

import ProposalService from '../components/proposals/services/proposalService';

import { Proposal, FilterParams } from '../components/proposals/types/proposalTypes';

import { DEFAULT\_SECTORS, PAGINATION\_CONFIG } from '../components/proposals/utils/proposalConstants';

import { formatCurrency, formatDate } from '../components/proposals/utils/proposalHelpers';

**2. Update BulkDownloadScreen.tsx**

// OLD imports

import ProposalCard from '../components/proposal/ProposalCard';

import ApiService from '../services/api';

import {Proposal} from '../types';

// NEW imports

import { ProposalCard } from '../components/proposals';

import ProposalService from '../components/proposals/services/proposalService';

import { Proposal, BulkDownloadRequest } from '../components/proposals/types/proposalTypes';

import { DOWNLOAD\_FORMATS } from '../components/proposals/utils/proposalConstants';

**3. Update Navigation File**

**src/navigation/AppNavigator.tsx**

// OLD imports

import AllProposalsScreen from '../screens/AllProposalsScreen';

import BulkDownloadScreen from '../screens/BulkDownloadScreen';

// NEW imports

import { AllProposalsScreen, BulkDownloadScreen } from '../components/proposals';

**4. Update Component Files**

In each component file, update the imports:

**ProposalCard.tsx**

// OLD

import StatusBadge from './StatusBadge';

import {Proposal} from '../../types';

// NEW

import StatusBadge from './StatusBadge';

import { Proposal } from '../types/proposalTypes';

import { formatCurrency, formatDate } from '../utils/proposalHelpers';

**StatusBadge.tsx**

// OLD - inline styles

// NEW

import { getStatusStyle } from '../utils/proposalHelpers';

import { STATUS\_COLORS } from '../utils/proposalConstants';

**5. Update Global API Service**

**src/services/api.ts** - Make request method public:

class ApiService {

private token: string = 'your\_bearer\_token\_here';

// Make this method public so ProposalService can use it

public async request<T>(endpoint: string, options: RequestInit = {}): Promise<T> {

const url = `${BASE\_URL}${endpoint}`;

const config: RequestInit = {

headers: {

'Content-Type': 'application/json',

'Authorization': `Bearer ${this.token}`,

...options.headers,

},

...options,

};

const response = await fetch(url, config);

if (!response.ok) {

throw new Error(`HTTP error! status: ${response.status}`);

}

return response.json();

}

// Remove proposal-specific methods as they're now in ProposalService

// Keep only generic/global methods here

}

**Benefits of This Reorganization**

1. **Better Organization**: All proposal-related code is in one place
2. **Easier Maintenance**: Changes to proposal features are contained
3. **Scalability**: Easy to add new proposal features
4. **Reusability**: Clear separation of concerns with barrel exports
5. **Type Safety**: Dedicated type definitions for proposals
6. **Testing**: Easier to test proposal functionality in isolation

**Migration Checklist**

* [ ] Create new folder structure
* [ ] Move all files to new locations
* [ ] Update import statements in all files
* [ ] Create new type definitions file
* [ ] Create constants and helpers files
* [ ] Create barrel export file
* [ ] Update API service to make request method public
* [ ] Test all proposal screens to ensure imports work
* [ ] Update any other files that import proposal components
* [ ] Remove old empty directories