# React Native Proposal Management App

## Project Folder Structure

```

src/

├── components/

│ ├── common/

│ │ ├── Header.tsx

│ │ ├── SearchBar.tsx

│ │ ├── FilterButton.tsx

│ │ └── LoadingSpinner.tsx

│ ├── proposal/

│ │ ├── ProposalCard.tsx

│ │ ├── ProposalList.tsx

│ │ ├── SectorFilter.tsx

│ │ └── StatusBadge.tsx

│ └── calendar/

│ ├── CalendarModal.tsx

│ └── DateRangePicker.tsx

├── screens/

│ ├── AllProposalsScreen.tsx

│ └── BulkDownloadScreen.tsx

├── services/

│ ├── api.ts

│ └── proposalService.ts

├── types/

│ └── index.ts

├── utils/

│ ├── constants.ts

│ └── helpers.ts

└── navigation/

└── AppNavigator.tsx

```

## Installation Commands

First, install the required dependencies:

```bash

# Core dependencies

npm install @react-navigation/native @react-navigation/stack

npm install react-native-screens react-native-safe-area-context

npm install react-native-gesture-handler react-native-reanimated

npm install react-native-vector-icons

npm install @react-native-community/datetimepicker

npm install react-native-calendar-picker

# For iOS

cd ios && pod install && cd ..

# For Android - add to android/app/build.gradle:

# implementation project(':react-native-vector-icons')

```

## 1. Types Definition (src/types/index.ts)

```typescript

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;

}

```

## 2. API Service (src/services/api.ts)

```typescript

const BASE\_URL = 'https://api.royalsundaram.com/v1';

class ApiService {

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

private 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();

}

async getProposals(params: FilterParams = {}) {

const queryParams = new URLSearchParams();

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

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

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

}

});

const queryString = queryParams.toString();

const endpoint = `/proposals${queryString ? `?${queryString}` : ''}`;

return this.request<ProposalsResponse>(endpoint);

}

async getSectors() {

return this.request<{success: boolean; data: {sectors: Sector[]}}>('/proposals/sectors');

}

async getStatuses() {

return this.request<{success: boolean; data: {statuses: Status[]}}>('/proposals/statuses');

}

async getDateRange() {

return this.request<{

success: boolean;

data: {

min\_date: string;

max\_date: string;

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

}

}>('/proposals/date-range');

}

async bulkDownload(proposalIds: string[], filters: any) {

return this.request('/proposals/bulk-download', {

method: 'POST',

body: JSON.stringify({

proposal\_ids: proposalIds,

filters,

download\_format: 'xlsx'

}),

});

}

}

export default new ApiService();

```

## 3. Header Component (src/components/common/Header.tsx)

```typescript

import React from 'react';

import {

View,

Text,

StyleSheet,

TouchableOpacity,

SafeAreaView,

} from 'react-native';

import Icon from 'react-native-vector-icons/MaterialIcons';

interface HeaderProps {

title: string;

onBackPress?: () => void;

showBack?: boolean;

rightComponent?: React.ReactNode;

}

const Header: React.FC<HeaderProps> = ({

title,

onBackPress,

showBack = false,

rightComponent,

}) => {

return (

<SafeAreaView style={styles.safeArea}>

<View style={styles.header}>

<View style={styles.leftSection}>

{showBack && (

<TouchableOpacity onPress={onBackPress} style={styles.backButton}>

<Icon name="arrow-back" size={24} color="#333" />

</TouchableOpacity>

)}

<Text style={styles.title}>{title}</Text>

</View>

<View style={styles.rightSection}>

<TouchableOpacity style={styles.iconButton}>

<Icon name="notifications" size={24} color="#333" />

</TouchableOpacity>

<TouchableOpacity style={styles.iconButton}>

<Icon name="search" size={24} color="#333" />

</TouchableOpacity>

<View style={styles.userInitials}>

<Text style={styles.initialsText}>AK</Text>

</View>

{rightComponent}

</View>

</View>

</SafeAreaView>

);

};

const styles = StyleSheet.create({

safeArea: {

backgroundColor: '#fff',

},

header: {

flexDirection: 'row',

justifyContent: 'space-between',

alignItems: 'center',

paddingHorizontal: 16,

paddingVertical: 12,

backgroundColor: '#fff',

borderBottomWidth: 1,

borderBottomColor: '#e0e0e0',

},

leftSection: {

flexDirection: 'row',

alignItems: 'center',

flex: 1,

},

backButton: {

marginRight: 12,

},

title: {

fontSize: 18,

fontWeight: '600',

color: '#333',

},

rightSection: {

flexDirection: 'row',

alignItems: 'center',

},

iconButton: {

padding: 8,

marginLeft: 8,

},

userInitials: {

width: 32,

height: 32,

borderRadius: 16,

backgroundColor: '#e3f2fd',

justifyContent: 'center',

alignItems: 'center',

marginLeft: 8,

},

initialsText: {

fontSize: 14,

fontWeight: '600',

color: '#1976d2',

},

});

export default Header;

```

## 4. Search Bar Component (src/components/common/SearchBar.tsx)

```typescript

import React from 'react';

import {

View,

TextInput,

StyleSheet,

TouchableOpacity,

} from 'react-native';

import Icon from 'react-native-vector-icons/MaterialIcons';

interface SearchBarProps {

value: string;

onChangeText: (text: string) => void;

placeholder?: string;

onFilterPress?: () => void;

}

const SearchBar: React.FC<SearchBarProps> = ({

value,

onChangeText,

placeholder = 'Search',

onFilterPress,

}) => {

return (

<View style={styles.container}>

<View style={styles.searchContainer}>

<Icon name="search" size={20} color="#666" style={styles.searchIcon} />

<TextInput

style={styles.input}

value={value}

onChangeText={onChangeText}

placeholder={placeholder}

placeholderTextColor="#999"

/>

</View>

{onFilterPress && (

<TouchableOpacity onPress={onFilterPress} style={styles.filterButton}>

<Icon name="tune" size={20} color="#1976d2" />

</TouchableOpacity>

)}

</View>

);

};

const styles = StyleSheet.create({

container: {

flexDirection: 'row',

paddingHorizontal: 16,

paddingVertical: 12,

backgroundColor: '#fff',

},

searchContainer: {

flex: 1,

flexDirection: 'row',

alignItems: 'center',

backgroundColor: '#f5f5f5',

borderRadius: 8,

paddingHorizontal: 12,

marginRight: 12,

},

searchIcon: {

marginRight: 8,

},

input: {

flex: 1,

height: 40,

fontSize: 16,

color: '#333',

},

filterButton: {

width: 40,

height: 40,

borderRadius: 8,

backgroundColor: '#e3f2fd',

justifyContent: 'center',

alignItems: 'center',

},

});

export default SearchBar;

```

## 5. Sector Filter Component (src/components/proposal/SectorFilter.tsx)

```typescript

import React from 'react';

import {

View,

Text,

StyleSheet,

TouchableOpacity,

ScrollView,

} from 'react-native';

interface SectorFilterProps {

sectors: string[];

selectedSector: string;

onSectorSelect: (sector: string) => void;

}

const SectorFilter: React.FC<SectorFilterProps> = ({

sectors,

selectedSector,

onSectorSelect,

}) => {

return (

<ScrollView

horizontal

showsHorizontalScrollIndicator={false}

contentContainerStyle={styles.container}

>

{sectors.map((sector) => (

<TouchableOpacity

key={sector}

style={[

styles.sectorButton,

selectedSector === sector && styles.selectedSector,

]}

onPress={() => onSectorSelect(sector)}

>

<Text

style={[

styles.sectorText,

selectedSector === sector && styles.selectedSectorText,

]}

>

{sector}

</Text>

</TouchableOpacity>

))}

</ScrollView>

);

};

const styles = StyleSheet.create({

container: {

paddingHorizontal: 16,

paddingVertical: 8,

},

sectorButton: {

paddingHorizontal: 16,

paddingVertical: 8,

borderRadius: 20,

backgroundColor: '#f0f0f0',

marginRight: 8,

borderWidth: 1,

borderColor: '#e0e0e0',

},

selectedSector: {

backgroundColor: '#e8f5e8',

borderColor: '#4caf50',

},

sectorText: {

fontSize: 14,

color: '#666',

fontWeight: '500',

},

selectedSectorText: {

color: '#2e7d32',

fontWeight: '600',

},

});

export default SectorFilter;

```

## 6. Status Badge Component (src/components/proposal/StatusBadge.tsx)

```typescript

import React from 'react';

import {View, Text, StyleSheet} from 'react-native';

interface StatusBadgeProps {

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

color: 'blue' | 'red' | 'green';

}

const StatusBadge: React.FC<StatusBadgeProps> = ({status, color}) => {

const getStatusStyle = () => {

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'};

}

};

const statusStyle = getStatusStyle();

return (

<View style={[styles.badge, {backgroundColor: statusStyle.backgroundColor}]}>

<View style={[styles.dot, {backgroundColor: statusStyle.color}]} />

<Text style={[styles.text, {color: statusStyle.color}]}>{status}</Text>

</View>

);

};

const styles = StyleSheet.create({

badge: {

flexDirection: 'row',

alignItems: 'center',

paddingHorizontal: 8,

paddingVertical: 4,

borderRadius: 12,

alignSelf: 'flex-start',

},

dot: {

width: 6,

height: 6,

borderRadius: 3,

marginRight: 6,

},

text: {

fontSize: 12,

fontWeight: '500',

},

});

export default StatusBadge;

```

## 7. Proposal Card Component (src/components/proposal/ProposalCard.tsx)

```typescript

import React from 'react';

import {

View,

Text,

StyleSheet,

TouchableOpacity,

} from 'react-native';

import Icon from 'react-native-vector-icons/MaterialIcons';

import StatusBadge from './StatusBadge';

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

interface ProposalCardProps {

proposal: Proposal;

onPress?: () => void;

onDownloadPress?: () => void;

showCheckbox?: boolean;

isSelected?: boolean;

onSelectionChange?: (selected: boolean) => void;

}

const ProposalCard: React.FC<ProposalCardProps> = ({

proposal,

onPress,

onDownloadPress,

showCheckbox = false,

isSelected = false,

onSelectionChange,

}) => {

return (

<TouchableOpacity style={styles.card} onPress={onPress}>

<View style={styles.header}>

{showCheckbox && (

<TouchableOpacity

style={styles.checkbox}

onPress={() => onSelectionChange?.(!isSelected)}

>

<Icon

name={isSelected ? 'check-box' : 'check-box-outline-blank'}

size={20}

color={isSelected ? '#4caf50' : '#ccc'}

/>

</TouchableOpacity>

)}

<View style={styles.statusContainer}>

<StatusBadge status={proposal.status} color={proposal.status\_color} />

<Text style={styles.additionalInfo}>{proposal.additional\_info}</Text>

</View>

</View>

<View style={styles.content}>

<View style={styles.row}>

<Text style={styles.proposalNumber}>{proposal.proposal\_number}</Text>

<Text style={styles.policyNumber}>{proposal.policy\_number}</Text>

</View>

<View style={styles.row}>

<Text style={styles.customerName}>{proposal.customer\_name}</Text>

<Text style={styles.packageType}>{proposal.package\_type}</Text>

</View>

<View style={styles.bottomRow}>

<View style={styles.detailsContainer}>

<View style={styles.detailItem}>

<Text style={styles.detailLabel}>Inward Code</Text>

<Text style={styles.detailValue}>{proposal.inward\_code}</Text>

</View>

<View style={styles.detailItem}>

<Text style={styles.detailLabel}>Posted Date</Text>

<Text style={styles.detailValue}>{proposal.posted\_date}</Text>

</View>

<View style={styles.detailItem}>

<Text style={styles.detailLabel}>Premium</Text>

<Text style={styles.detailValue}>

{proposal.currency}{proposal.premium}

</Text>

</View>

</View>

{proposal.download\_available && onDownloadPress && (

<TouchableOpacity

style={styles.downloadButton}

onPress={onDownloadPress}

>

<Icon name="file-download" size={20} color="#1976d2" />

</TouchableOpacity>

)}

</View>

</View>

</TouchableOpacity>

);

};

const styles = StyleSheet.create({

card: {

backgroundColor: '#fff',

marginHorizontal: 16,

marginVertical: 4,

borderRadius: 8,

padding: 16,

elevation: 2,

shadowColor: '#000',

shadowOffset: {width: 0, height: 1},

shadowOpacity: 0.2,

shadowRadius: 1.41,

},

header: {

flexDirection: 'row',

justifyContent: 'space-between',

alignItems: 'flex-start',

marginBottom: 12,

},

checkbox: {

marginRight: 12,

},

statusContainer: {

flex: 1,

alignItems: 'flex-end',

},

additionalInfo: {

fontSize: 12,

color: '#666',

marginTop: 4,

},

content: {

flex: 1,

},

row: {

flexDirection: 'row',

justifyContent: 'space-between',

marginBottom: 8,

},

proposalNumber: {

fontSize: 16,

fontWeight: '600',

color: '#333',

},

policyNumber: {

fontSize: 14,

color: '#666',

},

customerName: {

fontSize: 14,

color: '#333',

},

packageType: {

fontSize: 14,

color: '#666',

},

bottomRow: {

flexDirection: 'row',

justifyContent: 'space-between',

alignItems: 'flex-end',

marginTop: 8,

},

detailsContainer: {

flexDirection: 'row',

flex: 1,

},

detailItem: {

marginRight: 16,

},

detailLabel: {

fontSize: 10,

color: '#999',

marginBottom: 2,

},

detailValue: {

fontSize: 12,

color: '#333',

fontWeight: '500',

},

downloadButton: {

padding: 4,

},

});

export default ProposalCard;

```

## 8. All Proposals Screen (src/screens/AllProposalsScreen.tsx)

```typescript

import React, {useState, useEffect, useCallback} from 'react';

import {

View,

StyleSheet,

FlatList,

Text,

RefreshControl,

TouchableOpacity,

Alert,

} from 'react-native';

import Header from '../components/common/Header';

import SearchBar from '../components/common/SearchBar';

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

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

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

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

const AllProposalsScreen: React.FC = ({navigation}: any) => {

const [proposals, setProposals] = useState<Proposal[]>([]);

const [loading, setLoading] = useState(false);

const [refreshing, setRefreshing] = useState(false);

const [searchQuery, setSearchQuery] = useState('');

const [selectedSector, setSelectedSector] = useState('All');

const [totalCount, setTotalCount] = useState(0);

const [currentPage, setCurrentPage] = useState(1);

const [hasMore, setHasMore] = useState(true);

const sectors = ['All', 'Motor', 'Health', 'Commercial'];

const fetchProposals = useCallback(async (

page: number = 1,

resetData: boolean = true

) => {

try {

setLoading(true);

const params: FilterParams = {

page,

limit: 20,

search: searchQuery || undefined,

sector: selectedSector === 'All' ? undefined : selectedSector.toLowerCase(),

};

const response = await ApiService.getProposals(params);

if (response.success) {

const newProposals = response.data.proposals;

if (resetData) {

setProposals(newProposals);

} else {

setProposals(prev => [...prev, ...newProposals]);

}

setTotalCount(response.data.total\_count);

setCurrentPage(response.data.current\_page);

setHasMore(response.data.current\_page < response.data.total\_pages);

}

} catch (error) {

console.error('Error fetching proposals:', error);

Alert.alert('Error', 'Failed to fetch proposals. Please try again.');

} finally {

setLoading(false);

setRefreshing(false);

}

}, [searchQuery, selectedSector]);

useEffect(() => {

fetchProposals(1, true);

}, [fetchProposals]);

const handleRefresh = () => {

setRefreshing(true);

fetchProposals(1, true);

};

const handleLoadMore = () => {

if (!loading && hasMore) {

fetchProposals(currentPage + 1, false);

}

};

const handleSearch = (text: string) => {

setSearchQuery(text);

};

const handleSectorChange = (sector: string) => {

setSelectedSector(sector);

};

const handleFilterPress = () => {

// Navigate to filter screen or show filter modal

console.log('Filter pressed');

};

const handleProposalPress = (proposal: Proposal) => {

// Navigate to proposal details

console.log('Proposal pressed:', proposal.id);

};

const handleDownloadPress = (proposal: Proposal) => {

// Handle individual download

Alert.alert('Download', `Download proposal ${proposal.proposal\_number}?`);

};

const handleBulkDownloadPress = () => {

navigation.navigate('BulkDownload');

};

const renderProposal = ({item}: {item: Proposal}) => (

<ProposalCard

proposal={item}

onPress={() => handleProposalPress(item)}

onDownloadPress={() => handleDownloadPress(item)}

/>

);

const renderListHeader = () => (

<View>

<SearchBar

value={searchQuery}

onChangeText={handleSearch}

placeholder="Search"

onFilterPress={handleFilterPress}

/>

<SectorFilter

sectors={sectors}

selectedSector={selectedSector}

onSectorSelect={handleSectorChange}

/>

<View style={styles.listHeader}>

<Text style={styles.resultCount}>

Showing {totalCount} proposals

</Text>

<TouchableOpacity

style={styles.bulkDownloadButton}

onPress={handleBulkDownloadPress}

>

<Text style={styles.bulkDownloadText}>Bulk download</Text>

</TouchableOpacity>

</View>

</View>

);

const renderEmptyState = () => (

<View style={styles.emptyState}>

<Text style={styles.emptyTitle}>No proposals this time</Text>

<Text style={styles.emptySubtitle}>

There are no proposals to show at this time for you.

</Text>

</View>

);

return (

<View style={styles.container}>

<Header title="All Proposals" />

<FlatList

data={proposals}

keyExtractor={(item) => item.id}

renderItem={renderProposal}

ListHeaderComponent={renderListHeader}

ListEmptyComponent={!loading ? renderEmptyState : null}

refreshControl={

<RefreshControl refreshing={refreshing} onRefresh={handleRefresh} />

}

onEndReached={handleLoadMore}

onEndReachedThreshold={0.5}

showsVerticalScrollIndicator={false}

/>

</View>

);

};

const styles = StyleSheet.create({

container: {

flex: 1,

backgroundColor: '#f5f5f5',

},

listHeader: {

flexDirection: 'row',

justifyContent: 'space-between',

alignItems: 'center',

paddingHorizontal: 16,

paddingVertical: 12,

backgroundColor: '#fff',

},

resultCount: {

fontSize: 14,

color: '#666',

},

bulkDownloadButton: {

paddingVertical: 4,

},

bulkDownloadText: {

fontSize: 14,

color: '#1976d2',

fontWeight: '500',

textDecorationLine: 'underline',

},

emptyState: {

alignItems: 'center',

justifyContent: 'center',

paddingTop: 100,

paddingHorizontal: 32,

},

emptyTitle: {

fontSize: 18,

fontWeight: '600',

color: '#333',

marginBottom: 8,

},

emptySubtitle: {

fontSize: 14,

color: '#666',

textAlign: 'center',

lineHeight: 20,

},

});

export default AllProposalsScreen;

```

## 9. Bulk Download Screen (src/screens/BulkDownloadScreen.tsx)

```typescript

import React, {useState, useEffect} from 'react';

import {

View,

Text,

StyleSheet,

ScrollView,

TouchableOpacity,

Alert,

FlatList,

} from 'react-native';

import Icon from 'react-native-vector-icons/MaterialIcons';

import Header from '../components/common/Header';

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

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

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

const BulkDownloadScreen: React.FC = ({navigation, route}: any) => {

const [selectedProposals, setSelectedProposals] = useState<string[]>([]);

const [proposals, setProposals] = useState<Proposal[]>([]);

const [selectAll, setSelectAll] = useState(false);

const [loading, setLoading] = useState(false);

const [dateRange, setDateRange] = useState('23 JAN - 27 MAR 24');

const [selectedSector, setSelectedSector] = useState('All sectors');

const [downloadFormat, setDownloadFormat] = useState('xlsx');

// Get filter params from navigation or use defaults

const filterParams = route?.params?.filters || {

sector: 'all',

from\_date: '2024-01-23',

to\_date: '2024-03-27',

};

useEffect(() => {

fetchFilteredProposals();

}, []);

const fetchFilteredProposals = async () => {

try {

setLoading(true);

const response = await ApiService.getProposals(filterParams);

if (response.success) {

setProposals(response.data.proposals);

}

} catch (error) {

console.error('Error fetching proposals:', error);

Alert.alert('Error', 'Failed to fetch proposals');

} finally {

setLoading(false);

}

};

const handleSelectAll = () => {

if (selectAll) {

setSelectedProposals([]);

} else {

setSelectedProposals(proposals.map(p => p.id));

}

setSelectAll(!selectAll);

};

const handleProposalSelection = (proposalId: string, selected: boolean) => {

if (selected) {

setSelectedProposals(prev => [...prev, proposalId]);

} else {

setSelectedProposals(prev => prev.filter(id => id !== proposalId));

}

// Update select all state

const newSelectedCount = selected

? selectedProposals.length + 1

: selectedProposals.length - 1;

setSelectAll(newSelectedCount === proposals.length);

};

const handleDownload = async () => {

if (selectedProposals.length === 0) {

Alert.alert('No Selection', 'Please select at least one proposal to download.');

return;

}

try {

setLoading(true);

const response = await ApiService.bulkDownload(selectedProposals, {

...filterParams,

download\_format: downloadFormat,

});

if (

# Complete React Native Proposal Management App

## 9. Complete Bulk Download Screen (src/screens/BulkDownloadScreen.tsx)

```typescript

import React, {useState, useEffect} from 'react';

import {

View,

Text,

StyleSheet,

ScrollView,

TouchableOpacity,

Alert,

FlatList,

} from 'react-native';

import Icon from 'react-native-vector-icons/MaterialIcons';

import Header from '../components/common/Header';

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

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

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

const BulkDownloadScreen: React.FC = ({navigation, route}: any) => {

const [selectedProposals, setSelectedProposals] = useState<string[]>([]);

const [proposals, setProposals] = useState<Proposal[]>([]);

const [selectAll, setSelectAll] = useState(false);

const [loading, setLoading] = useState(false);

const [dateRange, setDateRange] = useState('23 JAN - 27 MAR 24');

const [selectedSector, setSelectedSector] = useState('All sectors');

const [downloadFormat, setDownloadFormat] = useState('xlsx');

// Get filter params from navigation or use defaults

const filterParams = route?.params?.filters || {

sector: 'all',

from\_date: '2024-01-23',

to\_date: '2024-03-27',

};

useEffect(() => {

fetchFilteredProposals();

}, []);

const fetchFilteredProposals = async () => {

try {

setLoading(true);

const response = await ApiService.getProposals(filterParams);

if (response.success) {

setProposals(response.data.proposals);

}

} catch (error) {

console.error('Error fetching proposals:', error);

Alert.alert('Error', 'Failed to fetch proposals');

} finally {

setLoading(false);

}

};

const handleSelectAll = () => {

if (selectAll) {

setSelectedProposals([]);

} else {

setSelectedProposals(proposals.map(p => p.id));

}

setSelectAll(!selectAll);

};

const handleProposalSelection = (proposalId: string, selected: boolean) => {

if (selected) {

setSelectedProposals(prev => [...prev, proposalId]);

} else {

setSelectedProposals(prev => prev.filter(id => id !== proposalId));

}

// Update select all state

const newSelectedCount = selected

? selectedProposals.length + 1

: selectedProposals.length - 1;

setSelectAll(newSelectedCount === proposals.length);

};

const handleDownload = async () => {

if (selectedProposals.length === 0) {

Alert.alert('No Selection', 'Please select at least one proposal to download.');

return;

}

try {

setLoading(true);

const response = await ApiService.bulkDownload(selectedProposals, {

...filterParams,

download\_format: downloadFormat,

});

if (response.success) {

Alert.alert(

'Download Started',

`Your download has been initiated. You will receive ${selectedProposals.length} files.`

);

navigation.goBack();

}

} catch (error) {

console.error('Error downloading:', error);

Alert.alert('Download Failed', 'Failed to start download. Please try again.');

} finally {

setLoading(false);

}

};

const handleEditFilters = () => {

// Navigate to filter screen or show filter modal

navigation.navigate('FilterModal', {

currentFilters: filterParams,

onApplyFilters: (newFilters: any) => {

// Handle filter updates

console.log('New filters:', newFilters);

}

});

};

const renderProposal = ({item}: {item: Proposal}) => (

<ProposalCard

proposal={item}

showCheckbox={true}

isSelected={selectedProposals.includes(item.id)}

onSelectionChange={(selected) => handleProposalSelection(item.id, selected)}

/>

);

const renderHeader = () => (

<View style={styles.headerContent}>

<View style={styles.filterInfo}>

<Text style={styles.filterText}>

{selectedSector} | {dateRange}

</Text>

<TouchableOpacity onPress={handleEditFilters}>

<Icon name="edit" size={16} color="#1976d2" />

</TouchableOpacity>

</View>

<TouchableOpacity style={styles.selectAllButton} onPress={handleSelectAll}>

<Icon

name={selectAll ? 'check-box' : 'check-box-outline-blank'}

size={20}

color={selectAll ? '#4caf50' : '#ccc'}

/>

<Text style={styles.selectAllText}>Select All</Text>

</TouchableOpacity>

</View>

);

const renderFooter = () => (

<View style={styles.footer}>

<View style={styles.formatSelector}>

<Text style={styles.formatLabel}>Save as</Text>

<TouchableOpacity style={styles.formatButton}>

<Icon name="description" size={16} color="#4caf50" />

<Text style={styles.formatText}>.xlsx File</Text>

<Icon name="keyboard-arrow-down" size={16} color="#666" />

</TouchableOpacity>

</View>

<TouchableOpacity

style={[

styles.downloadButton,

selectedProposals.length === 0 && styles.downloadButtonDisabled

]}

onPress={handleDownload}

disabled={selectedProposals.length === 0 || loading}

>

<Text style={[

styles.downloadButtonText,

selectedProposals.length === 0 && styles.downloadButtonTextDisabled

]}>

{loading ? 'Downloading...' : 'Download'}

</Text>

</TouchableOpacity>

</View>

);

return (

<View style={styles.container}>

<Header

title="Bulk download"

showBack={true}

onBackPress={() => navigation.goBack()}

rightComponent={

<TouchableOpacity onPress={() => navigation.goBack()}>

<Text style={styles.clearAllText}>Clear all</Text>

</TouchableOpacity>

}

/>

<FlatList

data={proposals}

keyExtractor={(item) => item.id}

renderItem={renderProposal}

ListHeaderComponent={renderHeader}

ListFooterComponent={renderFooter}

showsVerticalScrollIndicator={false}

style={styles.list}

/>

</View>

);

};

const styles = StyleSheet.create({

container: {

flex: 1,

backgroundColor: '#f5f5f5',

},

list: {

flex: 1,

},

headerContent: {

backgroundColor: '#fff',

padding: 16,

borderBottomWidth: 1,

borderBottomColor: '#e0e0e0',

},

filterInfo: {

flexDirection: 'row',

justifyContent: 'space-between',

alignItems: 'center',

marginBottom: 16,

},

filterText: {

fontSize: 14,

color: '#666',

},

selectAllButton: {

flexDirection: 'row',

alignItems: 'center',

},

selectAllText: {

fontSize: 16,

color: '#333',

marginLeft: 8,

},

footer: {

backgroundColor: '#fff',

padding: 16,

borderTopWidth: 1,

borderTopColor: '#e0e0e0',

flexDirection: 'row',

justifyContent: 'space-between',

alignItems: 'center',

},

formatSelector: {

flexDirection: 'row',

alignItems: 'center',

},

formatLabel: {

fontSize: 14,

color: '#666',

marginRight: 12,

},

formatButton: {

flexDirection: 'row',

alignItems: 'center',

paddingHorizontal: 12,

paddingVertical: 8,

borderRadius: 8,

backgroundColor: '#f5f5f5',

},

formatText: {

fontSize: 14,

color: '#333',

marginLeft: 4,

marginRight: 4,

},

downloadButton: {

backgroundColor: '#1976d2',

paddingHorizontal: 24,

paddingVertical: 12,

borderRadius: 8,

minWidth: 120,

alignItems: 'center',

},

downloadButtonDisabled: {

backgroundColor: '#ccc',

},

downloadButtonText: {

color: '#fff',

fontSize: 16,

fontWeight: '600',

},

downloadButtonTextDisabled: {

color: '#999',

},

clearAllText: {

fontSize: 14,

color: '#1976d2',

textDecorationLine: 'underline',

},

});

export default BulkDownloadScreen;

```

## 10. Filter Modal Component (src/components/common/FilterModal.tsx)

```typescript

import React, {useState, useEffect} from 'react';

import {

View,

Text,

StyleSheet,

TouchableOpacity,

Modal,

SafeAreaView,

} from 'react-native';

import Icon from 'react-native-vector-icons/MaterialIcons';

import CalendarPicker from 'react-native-calendar-picker';

import {Picker} from '@react-native-picker/picker';

interface FilterModalProps {

visible: boolean;

onClose: () => void;

onApply: (filters: any) => void;

initialFilters?: any;

}

const FilterModal: React.FC<FilterModalProps> = ({

visible,

onClose,

onApply,

initialFilters = {},

}) => {

const [fromDate, setFromDate] = useState(new Date());

const [toDate, setToDate] = useState(new Date());

const [selectedStatus, setSelectedStatus] = useState('');

const [selectedSector, setSelectedSector] = useState('');

const statuses = [

{ label: 'All Status', value: '' },

{ label: 'Initiated', value: 'initiated' },

{ label: 'Action Required', value: 'action\_required' },

{ label: 'Closed', value: 'closed' },

];

const sectors = [

{ label: 'All Sectors', value: '' },

{ label: 'Motor', value: 'motor' },

{ label: 'Health', value: 'health' },

{ label: 'Commercial', value: 'commercial' },

];

useEffect(() => {

if (initialFilters.from\_date) {

setFromDate(new Date(initialFilters.from\_date));

}

if (initialFilters.to\_date) {

setToDate(new Date(initialFilters.to\_date));

}

setSelectedStatus(initialFilters.status || '');

setSelectedSector(initialFilters.sector || '');

}, [initialFilters]);

const handleDateChange = (date: Date, type: string) => {

if (type === 'END\_DATE') {

setToDate(date);

} else {

setFromDate(date);

}

};

const handleApply = () => {

const filters = {

from\_date: fromDate.toISOString().split('T')[0],

to\_date: toDate.toISOString().split('T')[0],

status: selectedStatus,

sector: selectedSector,

};

onApply(filters);

onClose();

};

const handleClearAll = () => {

setFromDate(new Date());

setToDate(new Date());

setSelectedStatus('');

setSelectedSector('');

};

return (

<Modal visible={visible} animationType="slide" presentationStyle="pageSheet">

<SafeAreaView style={styles.container}>

<View style={styles.header}>

<View style={styles.headerLeft}>

<TouchableOpacity onPress={onClose}>

<Icon name="close" size={24} color="#333" />

</TouchableOpacity>

<Text style={styles.title}>Filter</Text>

</View>

<TouchableOpacity onPress={handleClearAll}>

<Text style={styles.clearAllText}>Clear all</Text>

</TouchableOpacity>

</View>

<View style={styles.content}>

<View style={styles.dateSection}>

<View style={styles.dateInputs}>

<View style={styles.dateInput}>

<Text style={styles.dateLabel}>From</Text>

<TouchableOpacity style={styles.dateButton}>

<Icon name="event" size={20} color="#666" />

<Text style={styles.dateText}>

{fromDate.toLocaleDateString('en-US', {

month: 'short',

year: 'numeric'

})}

</Text>

</TouchableOpacity>

</View>

<View style={styles.dateInput}>

<Text style={styles.dateLabel}>To</Text>

<TouchableOpacity style={styles.dateButton}>

<Icon name="event" size={20} color="#666" />

<Text style={styles.dateText}>

{toDate.toLocaleDateString('en-US', {

month: 'short',

year: 'numeric'

})}

</Text>

</TouchableOpacity>

</View>

</View>

<CalendarPicker

startFromMonday={true}

allowRangeSelection={true}

selectedStartDate={fromDate}

selectedEndDate={toDate}

onDateChange={handleDateChange}

todayBackgroundColor="#e3f2fd"

selectedDayColor="#4caf50"

selectedDayTextColor="#FFFFFF"

scaleFactor={375}

textStyle={styles.calendarText}

/>

</View>

<View style={styles.dropdownSection}>

<Text style={styles.sectionTitle}>Status</Text>

<View style={styles.pickerContainer}>

<Picker

selectedValue={selectedStatus}

onValueChange={(itemValue) => setSelectedStatus(itemValue)}

style={styles.picker}

>

{statuses.map((status) => (

<Picker.Item

key={status.value}

label={status.label}

value={status.value}

/>

))}

</Picker>

</View>

</View>

<View style={styles.dropdownSection}>

<Text style={styles.sectionTitle}>Select sector</Text>

<View style={styles.pickerContainer}>

<Picker

selectedValue={selectedSector}

onValueChange={(itemValue) => setSelectedSector(itemValue)}

style={styles.picker}

>

{sectors.map((sector) => (

<Picker.Item

key={sector.value}

label={sector.label}

value={sector.value}

/>

))}

</Picker>

</View>

</View>

</View>

<View style={styles.footer}>

<TouchableOpacity style={styles.applyButton} onPress={handleApply}>

<Text style={styles.applyButtonText}>Apply</Text>

</TouchableOpacity>

</View>

</SafeAreaView>

</Modal>

);

};

const styles = StyleSheet.create({

container: {

flex: 1,

backgroundColor: '#fff',

},

header: {

flexDirection: 'row',

justifyContent: 'space-between',

alignItems: 'center',

padding: 16,

borderBottomWidth: 1,

borderBottomColor: '#e0e0e0',

},

headerLeft: {

flexDirection: 'row',

alignItems: 'center',

},

title: {

fontSize: 18,

fontWeight: '600',

color: '#333',

marginLeft: 16,

},

clearAllText: {

fontSize: 14,

color: '#1976d2',

textDecorationLine: 'underline',

},

content: {

flex: 1,

padding: 16,

},

dateSection: {

marginBottom: 24,

},

dateInputs: {

flexDirection: 'row',

justifyContent: 'space-between',

marginBottom: 16,

},

dateInput: {

flex: 1,

marginHorizontal: 4,

},

dateLabel: {

fontSize: 14,

color: '#666',

marginBottom: 8,

},

dateButton: {

flexDirection: 'row',

alignItems: 'center',

padding: 12,

borderWidth: 1,

borderColor: '#e0e0e0',

borderRadius: 8,

backgroundColor: '#f9f9f9',

},

dateText: {

fontSize: 14,

color: '#333',

marginLeft: 8,

},

calendarText: {

color: '#333',

},

dropdownSection: {

marginBottom: 24,

},

sectionTitle: {

fontSize: 16,

fontWeight: '600',

color: '#333',

marginBottom: 12,

},

pickerContainer: {

borderWidth: 1,

borderColor: '#e0e0e0',

borderRadius: 8,

backgroundColor: '#f9f9f9',

},

picker: {

height: 50,

},

footer: {

padding: 16,

borderTopWidth: 1,

borderTopColor: '#e0e0e0',

},

applyButton: {

backgroundColor: '#1976d2',

paddingVertical: 16,

borderRadius: 8,

alignItems: 'center',

},

applyButtonText: {

color: '#fff',

fontSize: 16,

fontWeight: '600',

},

});

export default FilterModal;

```

## 11. Loading Spinner Component (src/components/common/LoadingSpinner.tsx)

```typescript

import React from 'react';

import {View, ActivityIndicator, StyleSheet} from 'react-native';

interface LoadingSpinnerProps {

size?: 'small' | 'large';

color?: string;

}

const LoadingSpinner: React.FC<LoadingSpinnerProps> = ({

size = 'large',

color = '#1976d2',

}) => {

return (

<View style={styles.container}>

<ActivityIndicator size={size} color={color} />

</View>

);

};

const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: 'center',

alignItems: 'center',

},

});

export default LoadingSpinner;

```

## 12. App Navigator (src/navigation/AppNavigator.tsx)

```typescript

import React from 'react';

import {NavigationContainer} from '@react-navigation/native';

import {createStackNavigator} from '@react-navigation/stack';

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

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

const Stack = createStackNavigator();

const AppNavigator: React.FC = () => {

return (

<NavigationContainer>

<Stack.Navigator

screenOptions={{

headerShown: false,

}}

>

<Stack.Screen name="AllProposals" component={AllProposalsScreen} />

<Stack.Screen name="BulkDownload" component={BulkDownloadScreen} />

</Stack.Navigator>

</NavigationContainer>

);

};

export default AppNavigator;

```

## 13. Main App Component (App.tsx)

```typescript

import React from 'react';

import {StatusBar} from 'react-native';

import AppNavigator from './src/navigation/AppNavigator';

const App: React.FC = () => {

return (

<>

<StatusBar barStyle="dark-content" backgroundColor="#fff" />

<AppNavigator />

</>

);

};

export default App;

```

## 14. Constants (src/utils/constants.ts)

```typescript

export const API\_ENDPOINTS = {

PROPOSALS: '/proposals',

SECTORS: '/proposals/sectors',

STATUSES: '/proposals/statuses',

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

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

};

export const COLORS = {

PRIMARY: '#1976d2',

SUCCESS: '#4caf50',

ERROR: '#d32f2f',

WARNING: '#ff9800',

BACKGROUND: '#f5f5f5',

WHITE: '#ffffff',

TEXT\_PRIMARY: '#333333',

TEXT\_SECONDARY: '#666666',

BORDER: '#e0e0e0',

};

export const PROPOSAL\_STATUS = {

INITIATED: 'Initiated',

ACTION\_REQUIRED: 'Action Required',

CLOSED: 'Closed',

};

export const STATUS\_COLORS = {

BLUE: 'blue',

RED: 'red',

GREEN: 'green',

};

```

## 15. Helper Functions (src/utils/helpers.ts)

```typescript

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

const date = new Date(dateString);

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

day: '2-digit',

month: '2-digit',

year: 'numeric',

});

};

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

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

};

export const getStatusColor = (status: string): 'blue' | 'red' | 'green' => {

switch (status.toLowerCase()) {

case 'initiated':

return 'blue';

case 'action required':

return 'red';

case 'closed':

return 'green';

default:

return 'blue';

}

};

export const debounce = (func: Function, wait: number) => {

let timeout: NodeJS.Timeout;

return function executedFunction(...args: any[]) {

const later = () => {

clearTimeout(timeout);

func(...args);

};

clearTimeout(timeout);

timeout = setTimeout(later, wait);

};

};

```

## 16. Additional Dependencies to Install

```bash

# Additional dependencies for complete functionality

npm install @react-native-picker/picker

npm install react-native-modal

npm install @react-navigation/bottom-tabs

npm install react-native-date-picker

# For iOS

cd ios && pod install && cd ..

```

## 17. Android Configuration

Add to `android/app/build.gradle`:

```gradle

dependencies {

implementation project(':react-native-vector-icons')

implementation project(':react-native-calendar-picker')

implementation project(':@react-native-picker\_picker')

}

```

Add to `MainApplication.java`:

```java

import com.oblador.vectoricons.VectorIconsPackage;

import com.reactnativecommunity.picker.RNCPickerPackage;

@Override

protected List<ReactPackage> getPackages() {

return Arrays.<ReactPackage>asList(

new MainReactPackage(),

new VectorIconsPackage(),

new RNCPickerPackage()

);

}

```

This completes the React Native Proposal Management App with all the necessary components, screens, navigation, and functionality as shown in your design mockups.