Index.js

This is the entry point of the React Native application.

Import Statements:

AppRegistry from react-native: Registers the main application component.

App from ./src/App: Imports the main application component.

appName from ./app.json: Retrieves the application name.

Registering the App:

AppRegistry.registerComponent('LokalJobPortal', () => App);: This line registers the main application component (App) with the application name (LokalJobPortal).

This file is responsible for bootstrapping the React Native application.

FetchApi.js

This file handles the fetching of job data from an external API.

Import Statements:

axios: Used for making HTTP requests.

API URL:

const API_URL: Defines the base URL for fetching job data.

fetchJobs Function:

This function takes a page parameter and fetches job data from the API.

Uses axios.get to make a GET request to the API with the specified page.

Returns the results from the API response.

Catches and logs any errors that occur during the fetch process.

This file centralizes the logic for fetching job data, making it reusable across different components.

App.js

This is the main application component that sets up navigation and state management.

Import Statements:

Imports necessary libraries and components, including React, navigation libraries, Redux, AsyncStorage, and custom screens.

Navigation Setup:

Uses createBottomTabNavigator to create a bottom tab navigator with "Jobs" and "Bookmarks" tabs.

Uses createNativeStackNavigator to create a stack navigator for navigating between the main tabs and the job details screen.

MainTabs Component:

Configures the bottom tab navigator.

Sets tab bar icons and styles.

App Component:

Uses useDispatch to get the dispatch function from Redux.

useEffect hook fetches bookmarked jobs from AsyncStorage when the app loads and dispatches them to the Redux store.

Sets up the navigation container and stack navigator.

RootApp Component:

Wraps the App component with the Redux Provider to connect the app to the Redux store.

Styles:

Defines styles for the container, tab icons, and tab labels.

This file sets up the main structure of the app, including navigation and state management using Redux and AsyncStorage.

JobScreen.js

This file defines the JobScreen component, which is responsible for fetching and displaying job data in a list format. It also handles pagination and error states.

Import Statements:

React, useEffect, useState, useCallback from react: Core React and Hooks.

View, FlatList, ActivityIndicator, Alert, StyleSheet from react-native: Basic React Native components.

fetchJobs from ../api/jobsApi: Function to fetch job data from the API.

JobCard from ../components/JobCard: Custom component to display job data.

ErrorPopup from ../components/ErrorPopup: Custom component to display error messages.

AsyncStorage from @react-native-async-storage/async-storage: For storing bookmarks.

useFocusEffect from @react-navigation/native: Hook to handle focus events for navigation.

State Variables:

jobs: Stores the list of job data.

loading: Tracks if the data is currently loading.

page: Tracks the current page number for pagination.

isEndReached: Tracks if the end of the data has been reached.

error: Tracks if there is an error in fetching data.

reloadNeeded: Tracks if a reload is needed due to an error.

loadJobs Function:

Sets loading state to true.

Tries to fetch job data for the current page using fetchJobs.

Updates the job list and other state variables based on the fetched data.

Handles errors and sets the error state accordingly.

useEffect Hook:

Calls loadJobs function whenever the page state changes.

useFocusEffect Hook:

Reloads the jobs if reloadNeeded is true when the screen gains focus.

handleLoadMore Function:

Increments the page number to load more data if not currently loading and the end is not reached.

handleCloseError Function:

Closes the error popup and navigates to the Bookmarks screen.

renderJob Function:

Renders a JobCard component for each job item.

Ensures job items without a title are not rendered.

return Statement:

Renders a View containing the ErrorPopup component and a FlatList to display job data.

Shows an ActivityIndicator if data is loading for the first page.

Uses FlatList to display the job data with infinite scrolling.

Styles:

loadingContainer: Styles for the loading indicator container.

This file defines the main screen for displaying job listings, handling pagination, and displaying appropriate loading and error states.

BookmarkScreen.js

This file defines the BookmarkScreen component, which displays a list of jobs that the user has bookmarked. It retrieves the bookmarked jobs from AsyncStorage and displays them using a FlatList.

1. Import Statements:

- React , useEffect , useState , useCallback from react : Core React and Hooks.
- View , FlatList , Text , ActivityIndicator , StyleSheet , Image from react-native : Basic React Native components.
- AsyncStorage from @react-native-async-storage/async-storage : For storing and retrieving bookmarked jobs.
- useFocusEffect from @react-navigation/native : Hook to handle focus events for navigation.
- JobCard from ../components/JobCard : Custom component to display job data.

2. State Variables:

- bookmarkedJobs : Stores the list of bookmarked job data.
- loading : Tracks if the data is currently loading.

3. loadBookmarkedJobs Function:

- Tries to retrieve bookmarked jobs from AsyncStorage.
- If bookmarked jobs are found, parses and sets them in the bookmarkedJobs state.
- Sets the loading state to false once the process is complete.

4. useEffect Hook:

- Calls loadBookmarkedJobs function once when the component mounts.

5. useFocusEffect Hook:

- Reloads the bookmarked jobs whenever the screen gains focus, ensuring the list is updated if bookmarks have changed.

6. renderJob Function:

- Renders a JobCard component for each job item.
- Passes the job data to the JobCard and navigates to the JobDetails screen when a job is pressed.

7. Conditional Rendering:

- If the data is still loading, displays an ActivityIndicator in a centered view.
- If there are no bookmarked jobs, displays a message and an image indicating no bookmarks are available.
- If there are bookmarked jobs, displays them in a FlatList .

8. Styles:

- container : Styles for the main container, with a dark background.
- loadingContainer : Styles for the loading indicator container.
- emptyContainer : Styles for the empty state container.
- image : Styles for the empty state image.

Summary:

- The BookmarkScreen component ensures that bookmarked jobs are loaded from AsyncStorage when the component mounts or gains focus.
- It handles loading and empty states gracefully, providing feedback to the user.
- It uses a FlatList to display the bookmarked jobs, leveraging the JobCard component for each job item.

This component is essential for providing a smooth user experience, allowing users to view and manage their bookmarked jobs efficiently.

JobCard.js

The JobCard component is a reusable card component designed to display job information. It also includes functionality to bookmark jobs.

1. Import Statements:

- React , useEffect from react : Core React and Hooks.
- View , Text , StyleSheet , TouchableOpacity , Image from react-native : Basic React Native components.
- FontAwesome6 from react-native-vector-icons/FontAwesome6 : For using font icons.
- useDispatch , useSelector from react-redux : For interacting with the Redux store.
- addBookmark , removeBookmark from ../features/bookmarks/bookmarksSlice : Redux actions to add or remove bookmarks.

2. Props and Destructuring:

- The component receives a job object and an onPress function as props.
- The job object is destructured to extract specific properties such as id , title , company_name , primary_details , job_tags , and creatives .

3. Abbreviating Title:

- The title is abbreviated if its length exceeds 30 characters to ensure it fits within the card's layout.

4. Redux Dispatch and Selector:

- dispatch is used to send actions to the Redux store.
- bookmarks is obtained from the Redux store to check if the job is already bookmarked.

5. Bookmark Handling:

- isBookmarked is a boolean indicating whether the current job is bookmarked.
- toggleBookmark function adds or removes the job from bookmarks based on its current state.

6. Component Layout:

- TouchableOpacity wraps the entire card to make it clickable.
- The card layout includes:
  - head : Contains the job title and the bookmark icon.
  - companyContainer : Displays the company name.
  - tags : Displays the location, job type, and qualification details.

7. Styles:

- card : Styles for the card container, including border, background color, and padding.
- head : Styles for the header section, with a row layout and space between elements.
- content : Styles for the main content section, with padding.
- title : Styles for the job title text.
- companyContainer : Styles for the company name container.
- company : Styles for the company name text.
- details : Styles for the job details tags.
- tags : Styles for the tags container, with a row layout and wrapping.
- icon : Styles for the icon color.
- image : Styles for the bookmark icon size.
- location : Styles for the location icon size.
- loc : Styles for the location and job type container.

Summary:

- The JobCard component displays a job's information in a card format.
- It handles abbreviating long titles, displays the company name, location, job type, and qualification.
- Users can bookmark or remove bookmarks for a job directly from the card.
- The card is styled to fit the app's theme and provides a clean, user-friendly interface for job listings.

This component is essential for displaying individual job details in a structured format, enhancing the user experience by allowing bookmarking functionality directly within the job card.

JobDetailScreen.js

1. Import Statements:

- React , useEffect , useState from react : Core React and Hooks.
- View , Text , Button , StyleSheet , Linking , TouchableOpacity , ScrollView from react-native : Basic React Native components.
- useDispatch , useSelector from react-redux : For interacting with the Redux store.
- addBookmark , removeBookmark from ../features/bookmarks/bookmarksSlice : Redux actions to add or remove bookmarks.

2. Props and Destructuring:

- The component receives a job object from route.params and a navigation object from props.
- The job object contains detailed information about the job.

3. Redux Dispatch and Selector:

- dispatch is used to send actions to the Redux store.
- bookmarks is obtained from the Redux store to check if the job is already bookmarked.
- isBookmarked is a boolean indicating whether the current job is bookmarked.

4. Bookmark Handling:

- toggleBookmark function adds or removes the job from bookmarks based on its current state.

5. OpenURLButton Component:

- This component handles opening URLs in the device's default web browser.
- It takes a url and children as props and uses Linking.canOpenURL and Linking.openURL to open the URL.

6. Layout and Content:

- The main content is wrapped in a ScrollView to allow for scrolling.
- The job details are displayed using various Text and View components.
- The job details include:

- company_name
- title
- job_location_slug
- Job highlights such as job_category , job_hours , job_role , salary_min , salary_max , openings_count , and num_applications .
- other_details
- Contact button that uses the OpenURLButton component.
- created_on and expire_on dates
- fees_charged
- whatsapp_no

- A Button at the bottom allows users to bookmark or remove the bookmark for the job.

7. Styles:

- container : Styles for the main container, with padding and background color.
- companyName , highlights , description : Styles for the text elements with font size, weight, margin, and color.
- detailContainer : Styles for containers holding details, with a row layout.
- detailLabel , detailText : Styles for detail labels and text.
- details : Styles for job details tags.
- tags : Styles for the tags container, with row layout and wrapping.
- border : Styles for a border separating sections.
- callbutton : Styles for the contact button.
- link : Styles for the text within the contact button.
- contactButtons : Styles for the contact buttons container, with row layout.

Summary:

- The JobDetailScreen component displays detailed job information in a user-friendly format.
- It provides functionalities to bookmark the job, open related links, and view various job details.
- The component is styled to match the app's theme and ensures that all information is easily accessible to the user.

This component is essential for providing detailed job information and enhancing user interaction with bookmarking and external link functionalities.

store.js

The store.js file is responsible for setting up the Redux store for the application. It uses the @reduxjs/toolkit library to configure the store, which simplifies the process of creating and managing the store.

1. Import Statements:

- configureStore from @reduxjs/toolkit : This function helps in setting up the Redux store with good default configurations.

- jobsReducer from ../features/jobs/jobsSlice : The reducer function to handle the state and actions related to jobs.

- bookmarkReducer from ../features/bookmarks/bookmarksSlice : The reducer function to handle the state and actions related to bookmarks.

2. Configuring the Store:

- configureStore is called with an object that contains the reducer property. This property is an object where each key is a slice of the state, and the value is the corresponding reducer function.

- The jobs key is assigned the jobsReducer function, which manages the state related to job listings.

- The bookmarks key is assigned the bookmarkReducer function, which manages the state related to bookmarked jobs.

3. Exporting the Store:

- The configured store is exported as the default export from this file, making it available for use throughout the application.

Summary:

- This store.js file sets up the Redux store by combining the jobs and bookmarks slices of the state.

- The jobsReducer manages the state related to job listings, while the bookmarkReducer manages the state related to bookmarked jobs.

- The configureStore function from @reduxjs/toolkit is used to create the store with the necessary configurations.

- The configured store is exported for use in the application, allowing components to access and update the state as needed.

This setup ensures that the application's state is managed efficiently and is organized into slices, making it easier to maintain and extend.

jobsSlice.js

The `jobsSlice.js` file utilizes the `@reduxjs/toolkit` package to define a Redux slice for managing job-related state, including asynchronous data fetching.

1. Imports:

- createSlice and createAsyncThunk from @reduxjs/toolkit : These functions are used to create a Redux slice with asynchronous action creators.

- fetchJobs from ../../api/jobsApi : The function responsible for fetching job data from an API.

2. Async Action Creator (loadJobs):

- createAsyncThunk is used to define an asynchronous action creator named loadJobs . It takes a string 'jobs/loadJobs' as its first argument (the action type prefix) and an async function as its second argument.

- Inside loadJobs , fetchJobs(page) is awaited to fetch job data from the API. Upon success, it logs the response data and returns response.jobs .

3. Initial State and Reducer (jobsSlice):

- createSlice is used to define the Redux slice named 'jobs' .

- initialState defines the initial state of the jobs slice, including an empty array jobs , loading set to false , and error initialized to null .

- The reducers field is an empty object because this slice only uses the extraReducers field to handle actions dispatched by createAsyncThunk .

4. extraReducers :

- extraReducers is a builder callback that listens to actions dispatched by loadJobs .

- .addCase(loadJobs.pending, (state) => { ... }) sets state.loading to true when the loadJobs action is pending.

- .addCase(loadJobs.fulfilled, (state, action) => { ... }) updates state.jobs by appending the fetched action.payload (job data) when the loadJobs action is fulfilled. It also sets state.loading to false .

- .addCase(loadJobs.rejected, (state, action) => { ... }) updates state.error with the action.error when the loadJobs action is rejected, and sets state.loading to false .

5. Export:

- export default jobsSlice.reducer; exports the reducer function generated by createSlice , which manages the state updates for the jobs slice.

Summary:

- The jobsSlice.js file defines a Redux slice named 'jobs' that manages job-related state including jobs , loading , and error .

- It uses createAsyncThunk to create an asynchronous action creator loadJobs that fetches job data from an API and updates the state based on the API response.

- extraReducers handles the lifecycle of loadJobs actions, updating loading , jobs , and error state based on whether the action is pending, fulfilled, or rejected.

- This setup ensures that job data fetching and state management are handled efficiently and in a structured manner using Redux and Redux Toolkit.

This file structure ensures that job-related data fetching and state management in your Redux store are handled cleanly and efficiently.

bookmarkSlice.js

The bookmarkSlice.js file utilizes @reduxjs/toolkit to define a Redux slice responsible for managing bookmarked jobs, including adding, removing, and loading bookmarks from AsyncStorage .

1. Imports:

- createSlice from @reduxjs/toolkit : Used to define a Redux slice.

- AsyncStorage from @react-native-async-storage/async-storage : Provides asynchronous storage API for React Native applications.

2. Redux Slice (bookmarkSlice):

- createSlice is used to define the Redux slice named 'bookmarks' .
- initialState defines the initial state of the bookmarks slice, which includes an empty array bookmarks and error set to null .
- reducers field contains three reducer functions:
  - setBookmarks : Updates state.bookmarks with action.payload .
  - addBookmark : Adds action.payload (a job object) to state.bookmarks and updates AsyncStorage with the updated list of bookmarks.
  - removeBookmark : Filters out the bookmark with action.payload.id from state.bookmarks and updates AsyncStorage with the updated list.

3. Action Creators (setBookmarks , addBookmark , removeBookmark):

- These are exported from the slice and can be dispatched to update the Redux state.

4. loadBookmarks Thunk Action Creator:

- loadBookmarks is an async thunk action creator that loads bookmarked jobs from AsyncStorage .
- It retrieves bookmarkedJobs from AsyncStorage , parses it, and dispatches setBookmarks to update the state with the loaded bookmarks.

5. Export:

- export default bookmarkSlice.reducer; exports the reducer function generated by createSlice , which manages the state updates for the bookmarks slice.

Summary:

- The bookmarkSlice.js file defines a Redux slice named 'bookmarks' that manages bookmarked job data.
- It uses createSlice to define the initial state (bookmarks array and error).
- Reducer functions (setBookmarks , addBookmark , removeBookmark) handle updating the bookmarks state array and persisting changes to AsyncStorage .
- loadBookmarks thunk action creator asynchronously loads bookmarked jobs from AsyncStorage and updates the state using setBookmarks .

- This setup ensures bookmarked job data is stored persistently and managed efficiently using Redux and AsyncStorage in a React Native application.

This file structure ensures that bookmarked job data can be added, removed, loaded, and stored persistently across sessions in your React Native application using Redux and AsyncStorage .