Redux:

The Redux folder contains the code and structure for managing the application's state using Redux, a state management library for JavaScript applications. This folder includes slices for different features of the application, services for handling asynchronous data fetching, and the Redux store configuration.

Features:

newsSlice.tsx:

The newsSlice.tsx file contains the Redux slice for managing the news data in the application. It uses createSlice from Redux Toolkit to define the initial state and reducers for handling different actions.

- **Initial State**: The initial state includes a loading property set to 'idle' and a newsData property initialized as an empty array.
- **Reducers**: The file defines the following reducers:
  - newsLoading: Sets the loading state to 'pending' if it's currently 'idle'.
  - newsReceived: Updates the state with the received news data when the loading state is 'pending'.
- **Exports**: The file exports the actions newsLoading and newsReceived as well as the reducer.

portSlice.tsx:

The portSlice.tsx file contains the Redux slice for managing port-related data in the application.

- **Initial State**: The initial state includes properties for loading, portData, searchData, pathCoordinates, pathLoading, co2Data, and weatherData.
- **Reducers**: The file defines the following reducers:
  - portsDataLoading: Sets the loading state to 'pending' if it's currently 'idle'.
  - portsDataReceived: Updates the state with the received port data when the loading state is 'pending'.
  - pathCoordinatesLoading: Sets the pathLoading state to 'pending' if it's currently 'idle'.
  - portsCoordinatesReceived: Updates the state with the received path coordinates and other data when the path loading state is 'pending'.
- **Exports**: The file exports the actions portsDataLoading, portsDataReceived, pathCoordinatesLoading, and portsCoordinatesReceived as well as the reducer.

userSlice.tsx:

The userSlice.tsx file contains the Redux slice for managing user-related data in the application.

- **Initial State**: The initial state includes properties for loading, userData, error, and errorMessage.
- **Reducers**: The file defines the following reducers:
  - usersLoggingIn: Sets the loading state to 'pending' if it's currently 'idle' and resets the error properties.
  - loginUserError: Updates the state with an error message when a login error occurs.
  - signOutUser: Resets the state to the initial state when the user signs out.
  - loginUser: Updates the state with the logged-in user data.
  - saveUserRoute: Updates the user data with a saved route.
- **Exports**: The file exports the actions usersLoggingIn, loginUser, loginUserError, saveUserRoute, and signOutUser as well as the reducer.

Services:

newsServices.tsx:

The newsServices.tsx file contains asynchronous action creators for fetching news data.

- **loadNews**: Fetches news data from the API and dispatches newsLoading and newsReceived actions based on the API response.

portServices.tsx:
The portServices.tsx file contains asynchronous action creators for fetching port-related data and path coordinates.

- **loadPortData**: Fetches port data from the API and dispatches portsDataLoading and portsDataReceived actions based on the API response.
- **loadPathCoordinates**: Fetches path coordinates from the API and dispatches pathCoordinatesLoading and portsCoordinatesReceived actions based on the API response.

userServices.tsx:
The userServices.tsx file contains asynchronous action creators for managing user authentication and routes.

- **userSignupFunction**: Handles user signup by dispatching the appropriate actions based on the API response.
- **userLoginFunction**: Handles user login by dispatching the appropriate actions based on the API response.
- **googleSignIn**: Handles Google sign-in by dispatching the appropriate actions based on the API response.
- **userSaveRoutes**: Saves user routes by dispatching the appropriate actions based on the API response.

Store:

store.tsx:
The store.tsx file contains the configuration for the Redux store.

- **configureStore**: Configures the Redux store with the reducers from the slices (user, port, and news) and optional middleware.
- **TypedUseSelectorHook**: Defines a typed version of the useSelector hook for use in the application.

i18n.tsx Documentation:

The i18n.tsx file is responsible for setting up internationalization (i18n) in the application using the i18next library and its React integration library, initReactI18next. It configures the available languages, default language, and provides a fallback language to ensure seamless translations in the application.

Key Components:

- i18next: A powerful internationalization library that helps in translating and managing languages in your application.
- initReactI18next: A React integration library for i18next that allows you to use i18n functions and hooks within React components.
- enTranslation, knTranslation, frTranslation: JSON files containing translations for English, Kannada, and French languages, respectively.
- LanguageDetector: A language detection plugin for i18next that detects the user's preferred language and sets it accordingly.
- useSelector: A hook from Redux that is used to select and access values from the Redux store.
- LOOKUP: An object from the static lookup module that provides constants such as language codes.

Code Breakdown:

Resources:

The resources object contains language resources for the application. Each language code (en, fr, kn) is associated with its respective translations.

- en: English translations from enTranslation.
- fr: French translations from frTranslation.
- kn: Kannada translations from knTranslation.

I
18n Initialization:
The i18n object is configured using the .use() method from i18next to include the LanguageDetector and initReactI18next plugins. The init() method sets up the internationalization with the following options:

- resources: Specifies the language resources for the application.
- lng: Specifies the default language, which is set to English (LOOKUP.LANGUAGES.EN).
- fallbackLng: Specifies the fallback language in case the user's preferred language is not supported, which is also set to English (LOOKUP.LANGUAGES.EN).
- interpolation: Disables escaping values to prevent potential issues with HTML output in translations.

Export:
The i18n object is exported as the default export from the file, allowing other parts of the application to access the configured internationalization functionality.

Usage:

To use the configured internationalization in your application:

- Import i18n from the file.
- Utilize the i18next functions and hooks provided by initReactI18next to manage translations and language detection.

The i18n.tsx file provides a well-configured setup for internationalization in the application. It includes language resources, sets default and fallback languages, and initializes the i18n object for seamless translation and language detection capabilities throughout the application.