

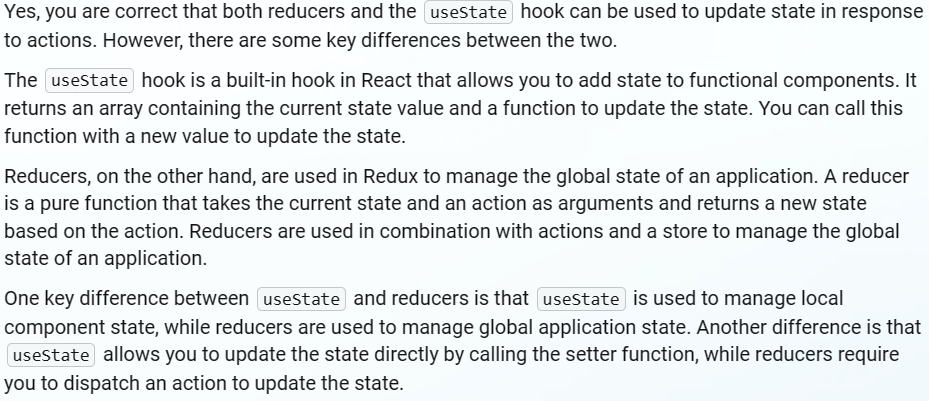
* **createAsyncThunk i**s a tool that helps you manage data that comes from outside your app, like

from an API. It does this by creating a special kind of function called a "thunk" that can handle the

process of getting the data and updating your app's state with the new data. You give it a name and tell.

it what to do to get the data, and it takes care of the rest.

* A **Redux Slice** is a way to organize the code for a single feature in a Redux app. It contains the reducer logic and actions for that feature. Think of it as a way to split up the overall Redux state into smaller parts, or “slices”, for easier management.
* **Reducers** are functions that determine changes to an application’s state. They take the current state and an action as arguments and return a new state.
* Sure! Imagine you have a big toy box **(the Redux store)** that holds all your **toys (the state).** You want to keep your toys organized, so you divide the toy box into sections **(slices)** using cardboard dividers. Each section holds a different type of toy, like cars, dolls, or blocks.
* Now, let’s say you want to play with your toys and change how they are arranged in the toy box. You can’t do it by yourself because you’re too small, so you ask your mom or dad **(the reducer)** to help you. You tell them what you want to do (the action), like “put the red car in the car section” or “take out all the dolls.” Your mom or dad listens to what you say and makes the changes for you.
* That’s how the Redux store, reducers, and slices work! The store is like a big toy box that holds all the data for your app. The reducers are like your mom or dad who listen to what you want to do and make changes to the data. And the slices are like the sections in the toy box that keep everything organized.
* A **bearer** token is a type of authentication token that is used to authenticate HTTP requests. When an HTTP request includes a bearer token in its 'Authorization' header, the server can use the token to verify the identity of the user making the request.
* An **async thunk** is a special kind of Redux action creator that can return a function instead of an action object. This function can contain asynchronous logic, such as making an API call, and can dispatch additional actions based on the result of the asynchronous operation.
* A **reducer** is a function that takes the current state and an action as arguments and returns a new state based on the action. Reducers are used to update the state in response to actions.

* **Extra reducers** are additional reducer functions that can be added to a createSlice call in Redux Toolkit. These reducers can handle actions that are not generated by the slice’s action creators. Extra reducers are useful for handling actions from other slices or external sources.
* 
* **bcryptjs**: It is a library that helps you hash passwords and compare them to previously hashed passwords. Hashing is a one-way process that takes an input (in this case, a password) and generates a fixed-length string of characters that represents that input. The idea behind hashing passwords is that if someone gains access to your database, they won’t be able to see the actual passwords because they are stored as hashes.
* **jwt**: JSON Web Tokens (JWTs) are a way of securely transmitting information between parties as a JSON object. They are often used for authentication purposes in web applications. A JWT consists of three parts: a header, a payload, and a signature. The header contains information about how the JWT should be signed and encrypted, the payload contains the actual data being transmitted (such as user information), and the signature is used to verify that the JWT has not been tampered with.

`````````````````````````````````````````````````````````````````````````````

* **config**: This folder contains all the configuration files for your backend application such as db.js which contains the database connection details.
* **controllers**: This folder contains all the functions for your APIs such as goalController.js and userController.js. These functions are responsible for handling incoming requests and returning responses.
* **middleware**: This folder contains all the middleware functions such as authMiddleware.js and errorMiddleware.js. Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application’s request-response cycle.
* **models**: This folder contains all your schema files such as goalModel.js and userModel.js. Schemas define the structure of documents within a collection.
* **routes**: This folder contains all the routes that you have created using Express Router such as goalRoutes.js and userRoutes.js. Routes define how your application responds to client requests to a particular endpoint.
* **server.js**: This file typically handles the app startup, routing, and other functions of the application.