**Redux design**

**store:**

categories[ members]

four categories: family[], friend[]s, vacation[]s, work[]s

store = { members: [] }

// member object

member: {

id: uuid, v4

categoryID: [0-4] // get it from state in NewMember.js

name: string

zip: string

}

**Actions**

**/\*\*\* actions/types.js \*\*\*/**

export const ADD\_MEMBER = “ADD\_MEMBER”

export const GET\_MEMBER = “GET\_MEMBER”

export const UPDATE\_MEMBER = “UPDATE\_MEMBER”

export const DETETE\_MEMBER = “DELETE\_MEMBER”

-----------

**/\*\*\* actions/creators.js \*\*\*/**

export const addMember = name => {

return { type: ADD\_MEMBER, data: createMember(categoryID, name, zip) }

}

export const getMember = name => {

return { type: GET\_MEMBER, data: { name}

}

export const updateMember = name => {

return { type: UPDATE\_MEMBER, data: name }

}

export const deleteMember = name => {

return { type: DELETE\_MEMBER, data: name }

}

**Reducers**

**/\*\*\* reducers/members.js \*\*\*/**

// member should be created by \_createMember(…)

function withNewMember(oldMembers, member) {

return oldMembers.map( m => {

if (m.categoryID === member.categoryID) {

oldMembers.addMember(member) // addMember is actions/creators.js

return deck

} else {

return category

}

}

}

const reducer = (state =[], action) => {

switch(action.type) {

case ADD\_MEMMBER:

return withNewMemberr(state, action.data)

}

return state; // if not the type return unmodiied state

}

export default reducer

**/\*\*\* reducers/index.js \*\*\*/**

import CategoriesReducer from ./Categories.js

const initialState = () => {

return { categories: MockCategories }

}

export const reducer = (state = initialState(), action) => {

let categories = CategoriesReducer(state.decks, action)

return { categories: categories }

}

**Create store**

**// add to HomeWeather.js**

HomeWeather.txt

import React, { Component} from 'react'

import Drawer from './Drawer'

import { reducer } from “../reducers/index”

let store = createStore(reducer)

class HomeWeather extends Component {

render() {

return(

<Provider store={store} >

<Drawer />

</Provider>

)

}

}

export default HomeWeather

**Connect to store**

need mapStateToProps() and mapDispatchToProps for <Category /> and <Member />

**<Category />**

// I should create <Category />

components/CategoryScreen/index.js, Category.js

// mapStateToProps and mapDispatchToProps should be **in index.js**

1. **mapStateToProps** describes how the Redux **store’s state will be provided to this component as props**.

Our state includes an array of categories. We’ll want to calculate the counts here, too.

const **mapStateToProps** = state => {

return {

categories: state.categories,

//k reduce(reducer function, currentValue)

counts: state.categories.reduce(

// category is elem in categories, sum is return value

// this is reduer function

(sum, category) => {

sum[category.id] = category.members.length

return sum

},

{} // curentValue is empty {}

) => counts = { 4, 5, 6, 5} length of each category

}

}

// Array.reduce

const array1 = [1,2,3,4]

const reducer = (accumulator, currentValue) => accumulator + currentValue

alert(array1.reduce(reducer)) //=> 10

alert(array1.reduce(reducer,5)) //=> 15

1. **mapDispatchToProps** defines the **props that a component will receive**, which can be used to dispatch actions.

import { addCategory } from ./../../actions/creators

// this provide createCategory props for create Category

const mapDispatchToProps = dispatch => {

return {

**createCategory:** categoryAction => { displatch(categoryAction) }

}

}

export default connect(mapStateToProps, mapDispatchToProps)(Categories)

Together, we can use these new props (**createCategory, categories, and counts**) in our component.

Now, <Categories /> will render based on props received from Redux, and it will also dispatch Redux actions instead of modifying state directly

----------------------------

**<NewMember />**

// I should create <NewMember />

components/NewMember/ index.js

const mapStateProps = state => {

return { categories: state.categories }

}

const mapDispatchToProps = dispatch => {

return {

createMember: (name, zip, categoryID) => {

dispatch(addMember(name, zip, categoryID))

}

}

}

export default connect(mapStateToProps, mapDispatchToProps)(NewMember)