ชื่อ.....

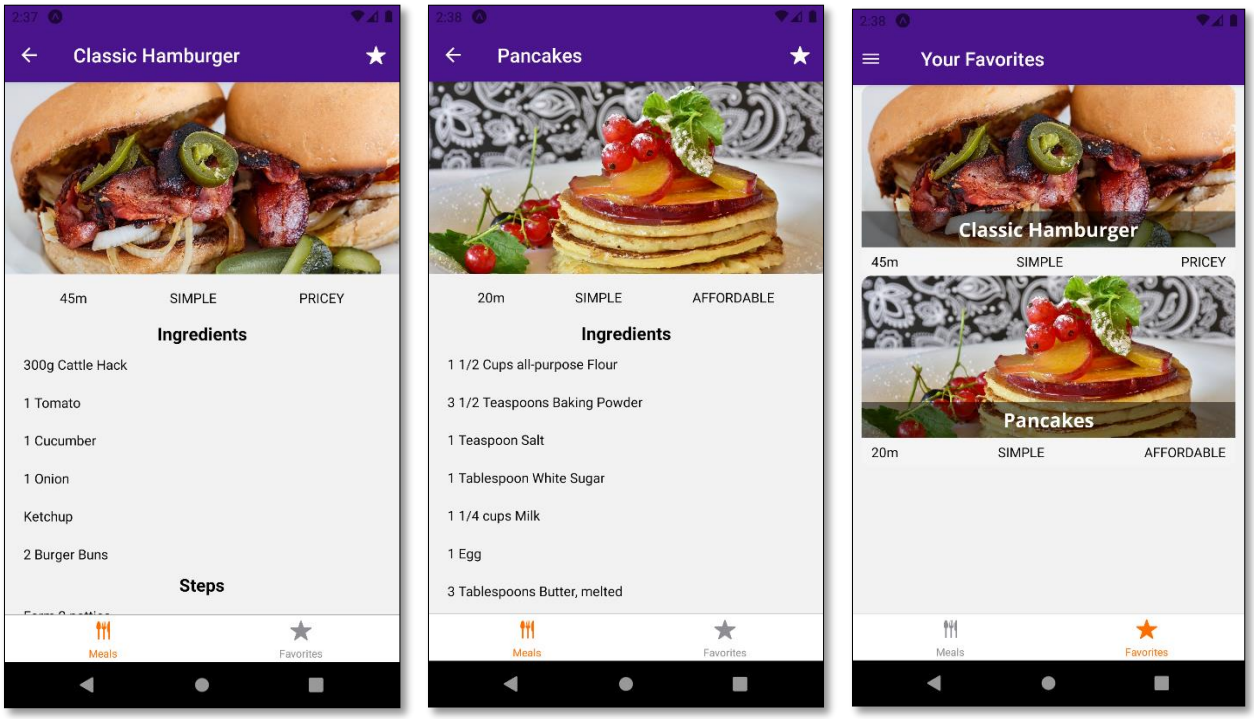
รหัสนักศึกษา.....

Lab 11 : State Management

จงปรับปรุงโปรแกรม MealApp ที่เป็นแอปพลิเคชันที่แสดงเมนูอาหารประเภทต่างๆ เพิ่มเติมจาก Lab 10 : Navigation (Part 2) ดังนี้

- ทำส่วนเลือกรายการอาหารที่ชื่นชอบให้สมบูรณ์ โดยกดปุ่มดาว ที่ตำแหน่งเฮดเดอร์ฝั่งขวาของหน้า MealDetailScreen
  - กรณีที่รายการอาหารนั้น ยังไม่ถูกเพิ่มเป็นรายการที่ชื่นชอบ จะแสดงไอคอนรูปกรอบดาว (ios-star-outline)
  - กรณีที่รายการอาหารนั้น ถูกเพิ่มเป็นรายการที่ชื่นชอบแล้ว จะแสดงไอคอนรูปดาว (ios-star)
  - เมื่อกดปุ่มดาวที่เป็นไอคอนรูป ios-star ไอคอนนั้นจะเปลี่ยนเป็นรูป ios-star-outline
  - เมื่อกดปุ่มดาวที่เป็นไอคอนรูป ios-star-outline ไอคอนนั้นจะเปลี่ยนเป็นรูป ios-star
  - รายการอาหารที่ชื่นชอบทั้งหมดจะแสดงในหน้า Favorites
- แนวคิดของการทำการทดลองครั้งนี้ มีจุดประสงค์ให้มีการใช้ Redux เพื่อจัดการ state ต่างๆ ซึ่ง state ดังกล่าว อาจมีความเกี่ยวข้องกับหลายคอมโพเนนต์ เมื่อ state มีการเปลี่ยนแปลง คอมโพเนนต์เหล่านั้น จะเห็นค่า state ล่าสุดได้ที่เพิ่งถูกเปลี่ยนแปลงได้เหมือนกัน

ตัวอย่างหน้าจอ



ชื่อ.....

รหัสนักศึกษา.....

โครงสร้างของโปรแกรม

- components
  - CategoryGridTile.js
  - CustomHeaderButton.js
  - MealItem.js
  - MealList.js
- data
  - dummy-data.js
- models
  - category.js
  - meals.js
- navigation
  - MealNavigator
- screens
  - CategoriesScreen.js
  - CategoryMealsScreen.js
  - FavoritesScreen.js
  - FiltersScreen.js
  - MealDetailScreen.js
- store
  - actions
    - mealsAction
  - reducers
    - mealsReducer
- App.js

ขั้นตอนการปฏิบัติการ

1. ให้นำโปรเจคที่ทำใน Lab 10 : Navigation (Part 2) มาปรับปรุง
2. ติดตั้ง redux และ react-redux
   - expo install redux react-redux หรือ

- npm install --save redux react-redux

3. สร้างโฟลเดอร์ store เพื่อใช้จัดการการตั้งค่า Redux
    - สร้างโฟลเดอร์ actions และ reducers ภายในโฟลเดอร์ store
4. สร้างไฟล์ mealsReducer.js ภายในโฟลเดอร์ reducers เพื่อใช้จัดการ meals reducer

ปรับปรุงไฟล์ mealsReducer.js (ทำการสร้าง Reducer)

1. import {MEALS} จากไฟล์ dummy-data.js มาใช้งาน
2. กำหนด initialState เป็นอ๊อบเจ็คที่ประกอบด้วย
    - meals: รายการเมนูอาหารทั้งหมด
        - กำหนดค่าเริ่มต้น เป็น MEALS
    - filteredMeals: รายการเมนูอาหารที่ตรงตามที่กำหนดใน filter
        - กำหนดค่าเริ่มต้น เป็น MEALS
    - favoriteMeals: รายการเมนูอาหารที่ชื่นชอบ
        - กำหนดค่าเริ่มต้น เป็นอะเรย์ว่าง
3. ทำการสร้าง Reducer ด้วยการสร้างฟังก์ชัน mealReducer มีรายละเอียดดังนี้
    - ประกอบด้วยอาร์กิวเมนต์ 2 อาร์กิวเมนต์ ได้แก่ state และ action
    - กำหนดค่าเริ่มต้นของ state เป็น initialState (ใช้ในกรณีที่ Redux ทำการประมวลผล reducer เป็นครั้งแรก)
    - ให้ฟังก์ชันนี้ทำการรีเทิร์น state
4. export mealReducer

ปรับปรุงไฟล์ App.js (ทำการสร้าง Store)

1. import { createStore, combineReducers } from ‘redux’
2. Import mealsReducer
3. ในการเขียนแอปพลิเคชันที่ซับซ้อน อาจมีการกำหนด reducer หลายตัว เพื่อจัดการ state ในเรื่องที่ต่างกัน กรณีนี้ เราสามารถทำการรวม reducer ต่างๆ ด้วย combineReducers โดยมี id อ้างอิงถึง reducer แต่ละตัวได้ เช่น กำหนด meals เป็น id เพื่ออ้างอิงถึง mealsReducer (สำหรับการทดลองนี้ มีการกำหนด reducer เพียงตัวเดียว)
    - ทำการสร้าง rootReducer

- const rootReducer = combineReducers({
  meals: mealsReducer
  })

4. ทำการสร้าง store ด้วยการเรียก createStore() และกำหนดให้ rootReducer เป็น reducer ที่ใช้ติดต่อกับ store
5. import { Provider } from 'react-redux'
6. แก้ไขส่วนของ return ใน App.js
   - เดิมให้ return <MealsNavigator /> ให้เปลี่ยนโดยทำการครอบ <MealsNavigator /> ด้วย <Provider></Provider> และกำหนด property store ที่มีค่าเป็น store ที่สร้างขึ้น
   - return <Provider store={store}><MealsNavigator /></Provider>
7. ทดลองรันโปรแกรม จะพบว่าโปรแกรมยังสามารถทำงานได้ปกติ เหมือนก่อนที่จะทำการจัดการ state ด้วย Redux

ปรับปรุงไฟล์ CategoryMealsScreen.js (การเรียกใช้ Store)

เนื่องจากใน Lab ที่ 9 และ 10 เราได้ทำการอ้างอิงถึงข้อมูลรายการอาหาร ด้วยการเรียกใช้ MEALS ในไฟล์ dummy-data.js แต่หลังจากนี้ เราจะทำการเรียกใช้ MEALS จาก store กลาง ที่จัดการด้วย Redux แทน

1. ให้ลบการ import MEALS จาก dummy-data.js จากไฟล์ต่างๆ เพื่อจะใช้ MEALS จาก Redux
2. import { useSelector } from 'react-redux';
   - useSelector: ทำให้เราสามารถเลือกสเตท และนำมาใช้ในคอมโพเนนต์ได้
3. ทำการปรับปรุงค่าของตัวแปร displayedMeals ดังนี้
   - สร้างตัวแปร availableMeals และกำหนดค่าเป็น filteredMeals ใน state ใน mealsReducer (ดูไฟล์ mealsReducer.js ประกอบ) โดยเรียกใช้ useSelector
     - const availableMeals = useSelector(state => state.meals.filteredMeals)
   - จาก App.js มีการเรียก combineReducers() จะพบว่า เราสามารถอ้างอิง mealsReducer ได้จาก id ชื่อ meals
   - จาก mealsReducer.js ในส่วนของการกำหนด initialState เป็นอ๊อบเจ็คซึ่งประกอบด้วย meals, filteredMeals, favoriteMeals
   - ปรับปรุงโปรแกรมในส่วนกำหนดตัวแปร displayedMeals จากเดิมที่ทำการ filter จาก MEALS ให้เปลี่ยนเป็น availableMeals แทน

ปรับปรุงไฟล์ FavoritesScreen

ให้ปรับปรุงไฟล์ FavoritesScreen ให้แสดงเมนูอาหารที่ชื่นชอบ โดยดึงข้อมูลจาก state ที่จัดการด้วย Redux (ทำคล้ายกับการแก้ไขไฟล์ CategoryMealsScreen ในหัวข้อก่อนหน้า)

1. ลบการเรียก MEALS จาก dummy-data.js
2. import { useSelector }
3. แก้ไขการกำหนด favMeals ให้อ้างอิง favoriteMeals จาก reducer แทนที่การอ้างอิงจาก MEALS ใน dummy-data.js

ปรับปรุงไฟล์ MealDetailScreen.js

1. ทำคล้ายกับไฟล์ CategoryMealsScreen.js โดยทำการปรับปรุงการกำหนดค่าของ selectedMeal ในฟังก์ชันคอมโพเนนต์ MealDetailScreen เพื่อเรียกใช้ state ใน Redux แทน MEALS
2. นอกจากการปรับปรุงโค้ดในส่วนของฟังก์ชันคอมโพเนนต์แล้ว จะพบว่าได้มีการอ้างอิงข้อมูลจาก MEALS ในส่วนของการกำหนด navigationOptions เพื่อใช้แสดงชื่อเมนูอาหารบริเวณเฮดเดอร์อีกด้วย ให้ทำการปรับปรุงโค้ดในส่วนนี้ให้เรียกใช้ state ใน Redux แต่เนื่องจาก useSelector() สามารถใช้ได้เฉพาะในฟังก์ชันคอมโพเนนต์เท่านั้น ในกรณีนี้ ให้นักศึกษาทดลองส่งค่าพารามิเตอร์จาก MealList แทน มีรายละเอียดดังนี้
    - ปรับปรุงโค้ดโปรแกรมในไฟล์ MealList.js โดยทำการปรับปรุงโค้ดที่ property onSelectMeal ของ MealItem ว่า หากมีการเลือกเมนูอาหารแล้ว นอกจากจะส่ง id ของเมนูนั้นไปยัง MealDetailScreen แล้ว ให้ส่งชื่อของเมนูอาหารนั้นไปด้วย
    - กำหนดให้พารามิเตอร์ที่ส่งเพิ่มเติมชื่อ mealTitle
    - ปรับปรุงโค้ดที่ไฟล์ MealDetailScreen.js ในส่วนของการกำหนด navigationOptions ให้รับชื่อของเมนูอาหารที่ส่งมาจากพารามิเตอร์ mealTitle แทน
3. ทดลองรันโปรแกรม จะพบว่าโปรแกรมทำงานได้ตามปกติ และจะพบว่า ณ ตอนนี้ เมื่อกดแท็บ Favorites จะพบว่า ยังไม่มีเมนูอาหารที่ชื่นชอบ (จากเดิม เรากำหนดให้เป็นเมนูอาหาร 2 รายการแรกใน MEALS)

กำหนด Dispatch Actions ใน Reducer

1. สร้างไฟล์ mealsAction.js ในโฟลเดอร์ store/actions/
2. สร้างตัวแปร TOGGLE_FAVORITE ดังนี้
    - export const TOGGLE_FAVORITE = "TOGGLE_FAVORITE";

3. สร้างฟังก์ชัน toggleFavorite เพื่อกำหนด action ที่มี type เป็น TOGGLE_FAVORITE และ การส่งข้อมูล mealId ดังนี้
    - export const toggleFavorite = (id) => {
      return { type: TOGGLE_FAVORITE, mealId: id };
      };
    - การกำหนด action นี้ เพื่อที่จะใช้แจ้งให้ Reducer รู้ว่า มีการทำ action กดปุ่มดาว (เพิ่ม/ลบเมนูให้อยู่ในรายการที่ชื่นชอบ) เพื่อให้ Reducer จัดการกับ state ที่เกี่ยวกับรายการเมนูที่เป็นที่ชื่นชอบอย่างไร

ปรับปรุงไฟล์ mealsReducer.js ในโฟลเดอร์ store/reducers/

1. กำหนดการจัดการข้อมูล state ตามประเภทของ action ใน mealsReducer โดยใช้ switch-case statement ให้เลือกทำงานตาม action.type
2. กรณีที่ action.type เป็น TOGGLE_FAVORITE ให้เขียนโปรแกรม ดังนี้
    - ทำการค้นหาตำแหน่ง mealId (ส่งมากับ action) ใน state.favoriteMeals ผ่านคำสั่ง findIndex() ซึ่งถ้าค้นหาพบ mealId นั้น จะคืนค่าเป็นตำแหน่งของเมนูอาหารที่มี mealId นั้น แต่ถ้าหาไม่พบจะคืนค่า -1
    - ตรวจสอบผลลัพธ์การ findIndex() หากมีค่ามากกว่าหรือเท่ากับ 0 (พบข้อมูลนั้น) ให้ทำการลบข้อมูลนั้นออกจาก state.favoriteMeals (สามารถใช้ฟังก์ชัน splice())
        - แสดงว่ารายการที่เรากดปุ่มดาว เป็นเมนูที่ชื่นชอบอยู่แล้ว ดังนั้น เมื่อกดปุ่มดาวอีกครั้ง จึงเป็นการเอาเมนูนั้นออกจากเมนูที่ชื่นชอบ
    - กรณีที่ค้นหา mealId นั้นไม่พบ ให้ทำการเพิ่มรายเมนูอาหารนั้นเข้าไปใน state.favoriteMeals
        - แสดงว่ารายการที่เรากดปุ่มดาว เป็นเมนูที่ยังไม่อยู่ในรายการที่ชื่นชอบ ดังนั้น เมื่อกดปุ่มดาว จึงเป็นการเพิ่มเมนูนั้นเข้าในรายการเมนูที่ชื่นชอบ

ปรับปรุงไฟล์ MealDetailScreen.js (ทำการ dispatch action)

1. import { useDispatch } from ‘react-redux’
2. import {toggleFavorite} จาก mealsAction
3. สร้างส่วน dispatch ในฟังก์ชันคอมโพเนนต์ MealDetailScreen
    - const dispatch = useDispatch()

4. สร้างฟังก์ชัน toggleFavoriteHandler ซึ่งทำการส่ง action toggleFavorite พร้อมกับค่า mealId ไปยัง Reducer
    - const toggleFavoriteHandler = () => {
        dispatch(toggleFavorite(mealId));
      };
5. ฟังก์ชัน toggleFavoriteHandler ควรถูกเรียกใช้ทำงานเมื่อมีการกดปุ่มดาว (จะทำการส่ง action ไปใน reducer อัพเดทค่าสเตทตามที่กำหนด) ดังนั้น ต้องทำการส่งฟังก์ชัน toggleFavoriteHandler ไปยัง navigationOptions เพื่อกำหนดว่า เมื่อมีการกด Item ใน headerRight จะทำการเรียก toggleFavoriteHandler นั่นเอง
    - ในฟังก์ชันคอมโพเนนต์ MealDetailScreen ให้ทำการส่งพารามิเตอร์ผ่าน navigation property ด้วยคำสั่ง props.navigation.setParams({ toggleFav: toggleFavoriteHandler }) โดยให้กำหนดคำสั่งนี้ภายใต้ useEffect() และจะมีการเรียกคำสั่งนี้เมื่อ toggleFavoriteHandler มีการเปลี่ยนแปลง
        - useEffect(() => {
            ...
          }, [...]);
6. เพื่อหลีกเลี่ยงการเกิด infinite loop ให้เรียกใช้ useCallback()
    - import { useCallback } from 'react'
    - ปรับฟังก์ชัน toggleFavoriteHandler โดยการครอบด้วย useCallback() และกำหนด dependencies เป็น dispatch และ mealId
        - const toggleFavoriteHandler = useCallback(() => {
            ...
          }, [...]);
7. ในส่วนของ navigationOptions ให้สร้างตัวแปร toggleFavorite ซึ่งทำการดึงค่าจากพารามิเตอร์ toggleFav ข้างต้น (จากข้อ 5) โดยใช้ getParam() และให้ทำการเรียก toggleFavorite เมื่อมีการกดปุ่มดาวที่อยู่บริเวณเฮดเดอร์ฝั่งขวา
8. ทดลองรันโปรแกรม จะพบว่าเราสามารถเพิ่มเมนูอาหารที่ชื่นชอบ ผ่านการกดปุ่มดาวบริเวณเฮดเดอร์ฝั่งขวา และหากกดปุ่มดาวที่เมนูอาหารที่ชื่นชอบอีกครั้ง เมนูนั้นจะถูกนำออกจากรายการที่ชื่นชอบ (พิจารณาจากแท็บ Favorites) แต่ไอคอนรูปดาวจะยังเหมือนกัน ไม่ว่าเมนูอาหารนั้นจะเป็นที่ชื่นชอบหรือไม่

ชื่อ.....

รหัสนักศึกษา.....

ปรับปรุงไฟล์ MealDetailScreen.js เพิ่มเติม

หากพิจารณาการกำหนด navigationOptions จะพบว่า เราจะต้องมีการแก้ไขโค้ดของ headerRight (การแสดงไอคอนปุ่มดาว) โดยกำหนดว่า

- หากเมนูอาหารนั้นเป็นเมนูที่ชื่นชอบ จะแสดงไอคอนเป็นรูปดาวที่มีสีพื้น (ios-star)
- หากเมนูอาหารนั้นไม่ใช่เมนูชื่นชอบ จะแสดงไอคอนเป็นรูปกรอบดาว (ios-star-outline)
- ดังนั้น จำเป็นต้องมีการสร้างตัวแปรเพื่อเก็บค่าว่าเมนูนั้นเป็นที่ชื่นชอบหรือไม่ (isFav) ซึ่งสามารถทำได้ในฟังก์ชันคอมโพเนนต์ MealDetailScreen แล้วจึงส่งพารามิเตอร์ผ่าน navigation property

1. สร้างตัวแปร currentMealIsFav โดยตรวจสอบว่าเมนูอาหารนั้นอยู่ในรายการอาหารที่ชื่นชอบ (state.meals.favoriteMeals) หรือไม่ โดยเรียกใช้ some()
   - const currentMealIsFav = useSelector((state) =>
     state.meals.favoriteMeals.some((meal) => meal.id === mealId)
     );
   - ให้ส่งค่า currentMealIsFav นี้ให้กับ navigationOptions ผ่าน setParam() และใช้ useEffect() โดยกำหนดให้ส่งพารามิเตอร์ชื่อ isFav และกำหนด dependencies เป็น currentMealIsFav
2. เพิ่มเติมโปรแกรมในส่วนของ navigationOptions
   - กำหนดตัวแปร isFavorite ซึ่งทำการดึงค่าพารามิเตอร์ isFav จาก navigation property (กำหนดจากข้อ 1)
   - ปรับปรุงการกำหนด iconName โดยกำหนดว่า
     - ถ้า isFavorite เป็น true ให้ iconName เป็น ios-star
     - ถ้า isFavorite เป็น false ให้ iconName เป็น ios-star-outline
3. ทดลองรันโปรแกรม จะพบว่าปุ่มดาวที่เฮดเดอร์ฝั่งขวาของเมนูอาหารแต่ละรายการจะแสดงไอคอนที่ต่างกันขึ้นกับว่าเมนูอาหารนั้นอยู่ในรายการที่ชื่นชอบหรือไม่