ชื่อ.....

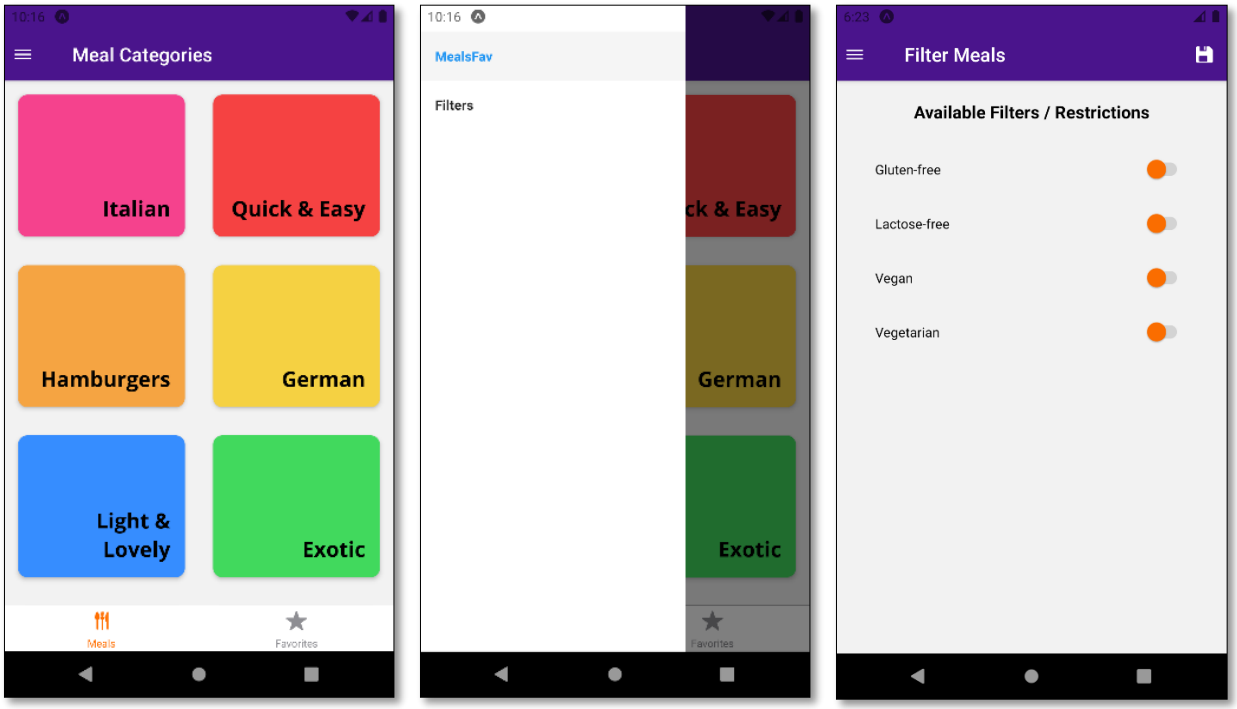
รหัสนักศึกษา.....

Lab 10 : Navigation (Part 2)

จงเขียนโปรแกรม MealApp ที่เป็นแอปพลิเคชันที่แสดงเมนูอาหารประเภทต่างๆ โดยได้มีการเพิ่มเติมจาก Lab 9 : Navigation (Part 1) ดังนี้

- สร้าง tab ด้านล่าง แสดงเมนู Meals และ Favorites
- สร้างแถบเมนูด้านข้าง แสดงเมนู Meals และ Filters
- สร้างหน้า Filters
- ปรับปรุงหน้า MealDetailScreen

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
- App.js

หมายเหตุ นักศึกษาสามารถแก้ไขโค้ดโปรแกรมที่ได้เตรียมไว้ให้ (OnLearn) ได้ตามความเหมาะสม

ขั้นตอนการปฏิบัติการ

1. ให้นำโปรเจคที่ทำใน Lab 9 : Navigation (Part 1) มาปรับปรุง
2. ติดตั้ง library และ dependencies เพื่อใช้ในการทำ Navigation ให้ครบถ้วน
   - expo install react-navigation-tabs
   - expo install react-navigation-drawer

ชื่อ.....

รหัสนักศึกษา.....

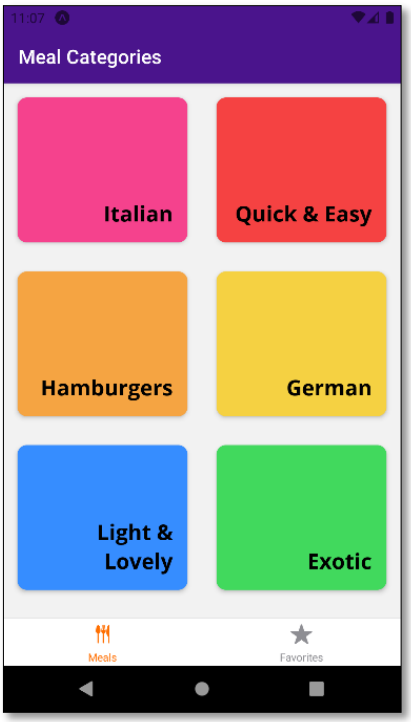
ปรับปรุงไฟล์ MealsNavigator.js เพื่อสร้าง Tab navigator ดังนี้

- Import createBottomTabNavigator เพื่อใช้ในการสร้าง tab navigator
- Import FavoritesScreen
- ให้ทำการสร้าง MealsFavTabNavigator โดยเรียก createBottomTabNavigator() โดยกำหนดอาร์กิวเมนต์แรก (RouteConfig) ดังนี้ (Slide.18)
    - Route name : Meals
        - screen เป็น MealsNavigator
        - navigationOptions ให้กำหนด property tabBarIcon ให้แสดงไอคอน ios-restaurant และกำหนดขนาดและสีตามชอบ (อาจใช้สีที่กำหนดใน Colors.js หรือใช้สีที่กำหนดใน tabBarOptions ก็ได้)
    - Route name : Favorites
        - screen เป็น FavoritesScreen
        - navigationOptions ให้กำหนด property tabBarIcon ให้แสดงไอคอน ios-star และกำหนดขนาดและสีตามชอบ (อาจใช้สีที่กำหนดใน Colors.js หรือใช้สีที่กำหนดใน tabBarOptions ก็ได้)
    - กรณีที่นศ.ต้องการเปลี่ยนข้อความที่แสดงใน tab สามารถกำหนดใน navigationOptions ผ่าน property tabBarLabel ได้

  ให้กำหนดอาร์กิวเมนต์ที่สอง (NavigationConfig) ของ createBottomTabNavigator() ดังนี้ (Slide.15)
    - tabBarOptions : { activeTintColor: สีตามชอบ,}
- ให้สร้าง app container ของ MealsFavTabNavigator แทนที่ MealsNavigator (เป็น stack navigator ที่เคยสร้างในครั้งที่แล้ว ซึ่งเป็นการเปลี่ยนหน้าเลือกเมนูอาหารต่างๆ) ตอนนี้ MealsNavigator ถือว่าซ้อนอยู่ใน Tab navigator แล้ว
- ทดลองรันและศึกษาการทำงานของโปรแกรม โดยโปรแกรมจะแสดงส่วน Tab ด้านล่าง ที่เมื่อกด Meals จะแสดงหน้าแรกของ MealsNavigator (CategoriesScreen) และเมื่อกด Favorites จะไปยังหน้า FavoritesScreen

ชื่อ.....

รหัสนักศึกษา.....



ปรับปรุงไฟล์ MealsNavigator.js

เมื่อกด tab Favorites โปรแกรมควรจะแสดงรายการเมนูอาหารที่ผู้ใช้ชื่นชอบ และสามารถกดดูรายละเอียดแต่ละเมนูอาหารได้ ซึ่งมีการเปลี่ยนหน้าในรูปแบบ Stack

1. ให้ทำการสร้าง FavNavigator โดยเรียก createStackNavigator() โดยกำหนดอาร์กิวเมนต์แรก (RouteConfig) ดังนี้ (บทที่ 9 Slide.24)
    - Route name : Favorites
        - screen เป็น FavoritesScreen
    - Route name : MealDetail
        - screen เป็น MealDetailScreen
2. ให้กำหนดอาร์กิวเมนต์ที่สอง (NavigationConfig) ของ createStackNavigator ()
    - defaultNavigationOptions : กำหนดเหมือนกับใน MealsNavigator
3. ใน MealsFavTabNavigator แก้ไขโค้ดในส่วนของ Route name : Favorites ซึ่งเดิมกำหนดให้ screen คือหน้า FavoritesScreen
    - กำหนด screen เป็น FavNavigator

ชื่อ.....

รหัสนักศึกษา.....

ปรับปรุงไฟล์ MealList.js

ไฟล์ MealList.js มีจุดประสงค์ให้เป็นการแสดงรายการเมนูอาหาร ซึ่งสามารถถูกเรียกได้ทั้งจาก CategorieMealsScreen (แสดงรายการเมนูอาหารตามประเภทอาหารที่เลือก) และ FavoritesScreen (แสดงรายการเมนูอาหารที่ผู้ใช้ชื่นชอบ) จึงควรมีการแยกคอมโพเนนต์ในส่วนแสดงรายการเมนูอาหารออกมา (MealList) และรับข้อมูลรายการอาหารที่ต้องการแสดงจาก CategoryMealsScreen และ FavoritesScreen อีกที

1. ในส่วนของ return() : ให้นศ.ทำการแสดงรายการเมนูอาหาร (Hint: นศ.สามารถคัดลอกโค้ดโปรแกรมจาก CategoryMealsScreen แล้วนำมาปรับปรุง) โดยกำหนดให้ข้อมูลที่จะแสดงใน FlatList เป็นข้อมูลที่ส่งผ่าน property ชื่อ listData
2. นศ.อาจต้องเพิ่มส่วนของฟังก์ชันสำหรับ property renderItem ใน FlatList รวมถึง StyleSheet ที่จำเป็นต้องใช้

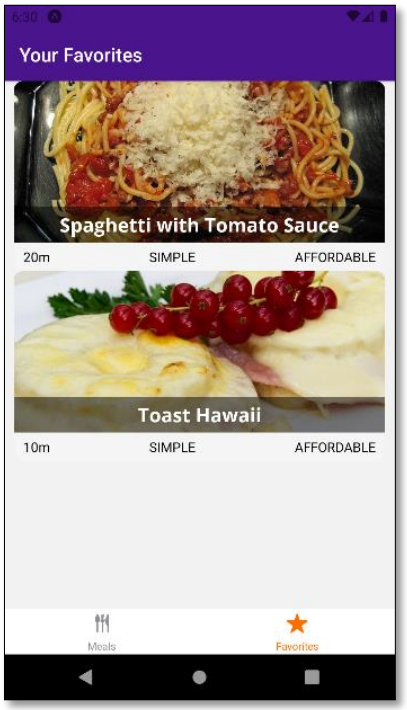
ปรับปรุงไฟล์ CategoryMealsScreen.js

1. import MealList มาใช้งาน
2. ในส่วนของ return() : ให้ทำการเรียกคอมโพเนนต์ MealList แทนที่จะสร้าง FlatList เพื่อแสดงรายการเมนูอาหารเอง โดยกำหนดให้ส่ง property ต่างๆ ดังนี้
    - listData : ลิสต์ข้อมูลเมนูอาหารที่ต้องการแสดง
    - navigation : {props.navigation} (เนื่องจาก MealList ไม่ได้ถูกกำหนดในการสร้าง Navigator จึงไม่มี property navigation ทำให้ CategoryMealsScreen ต้องส่ง property navigation ไปให้ MealList)
3. ลบหรือคอมเม้นต์โปรแกรมในส่วนที่ไม่จำเป็นต้องใช้ออก (ส่วนที่นำไปใช้ใน MealList แล้ว)
4. ทดลองรันโปรแกรม โปรแกรมจะสามารถทำงานได้ปกติ

ปรับปรุงไฟล์ FavoritesScreen.js

1. กำหนด headerTitle ของ FavoritesScreen เป็น “Your Favorites” (กำหนดผ่าน navigationOptions – Slide.33)
2. Import MealList และ {MEALS} มาใช้งาน
3. ทดลองสร้างตัวแปร favMeals กำหนดข้อมูลเมนูอาหารที่ชื่นชอบไว้ก่อน เพื่อใช้ทดสอบการทำงานของโปรแกรม
    - const favMeals = MEALS.filter((meal) => meal.id === "m1" || meal.id === "m2");
    - เป็นการกำหนดให้เมนูอาหารที่มี id เป็น ‘m1’ หรือ ‘m2’ เป็นรายการที่ชื่นชอบ

4. ในส่วนของ return() : ให้ทำการเรียกคอมโพเนนต์ MealList เพื่อแสดงรายการเมนูอาหารเอง โดยให้แสดงเมนูตามที่กำหนดใน favMeals (คล้ายกับหัวข้อปรับปรุงไฟล์ CategoryMealsScreen.js)
5. ทดลองรันโปรแกรม จะพบว่าเมื่อกด tab Favorites จะแสดงเมนูอาหารตามที่กำหนดใน favMeals ซึ่งเมื่อเข้าไปในเมนูหนึ่งๆ ก็แสดงหน้ารายละเอียดของเมนูนั้นต่อ



Draw Navigation

ปรับปรุงไฟล์ MealsNavigator.js

1. Import createDrawerNavigator เพื่อใช้ในการสร้าง drawer navigator
2. ให้ทำการสร้าง FiltersNavigator โดยเรียก createStackNavigator() โดยกำหนดอาร์กิวเมนต์แรก (RouteConfig) ดังนี้
    - Route name : Filters
        - screen เป็น FiltersScreen
3. ให้ทำการสร้าง MainNavigator โดยเรียก createDrawerNavigator() โดยกำหนดอาร์กิวเมนต์แรก (RouteConfig) ดังนี้
    - Route name : MealsFav
        - screen เป็น MealsFavTabNavigator
    - Route name : Filters
        - screen เป็น FiltersNavigator

4. ให้สร้าง app container ของ MainNavigator แทนที่ MealsFavTabNavigator (MealsFavTabNavigator ถือว่าอยู่ใน drawer navigator แล้ว)
5. เมื่อทดลองรัน นศ.จะพบว่าโปรแกรมจะเปิด drawer จากข้างจอ ต่อไปจะทำการสร้างปุ่มเปิด drawer ที่บริเวณเฮดเดอร์

การสร้างปุ่มเมนู (CategoriesScreen.js)

ส่วนนี้จะทำการสร้างปุ่มเมนูที่หน้า CategoriesScreen โดยได้เตรียมไฟล์ CustomHeaderButton.js (ในโฟลเดอร์ components) เพื่อใช้ในการกำหนดคุณลักษณะของปุ่มบนเฮดเดอร์ (Slide.33)

1. ติดตั้ง react-navigation-header-buttons และ @expo/vector-icons
2. ศึกษาโปรแกรมใน CustomHeaderButton.js
3. ปรับปรุงไฟล์ CategoriesScreen.js โดยให้ import คอมโพเนนต์ที่จำเป็น ได้แก่ {HeaderButton, Item} และ CustomHeaderButton เป็นต้น
4. ปรับปรุงส่วน CategoriesScreen.navigationOptions
    - ปรับให้เขียนในรูปแบบฟังก์ชัน มีโครงสร้างดังนี้

      CategoriesScreen.navigationOptions = (navigationData) => {

      return {...}

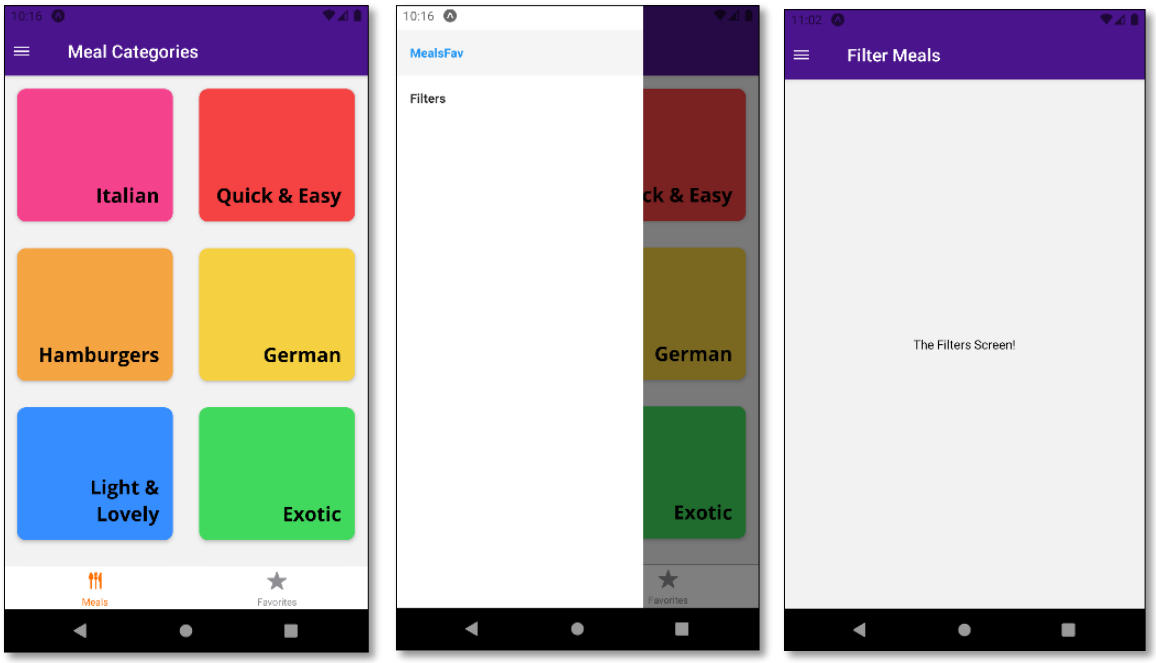
      }
    - ในส่วน return {...} กำหนดให้
        - headerTitle เป็น Meal Categories (เหมือนที่เคยกำหนดมาแล้ว)
        - headerLeft เป็นการอ้างถึง คอมโพเนนต์ HeaderButtons เพื่อใช้แสดงปุ่มเมนู โดยกำหนดให้ Item มี title เป็น ‘Menu’ และแสดงไอคอน ‘ios-menu’ และเมื่อกดปุ่มเมนูให้ทำการเรียก navigationData.navigation.toggleDrawer()

ปรับปรุงไฟล์ FiltersScreen.js

1. กำหนด navigationOptions ให้มี headerTitle เป็น “Filter Meals” และมีปุ่มเมนูเหมือนกับไฟล์ CategoriesScreen.js
2. ทดลองรันโปรแกรม จะพบว่า ปุ่มเมนูจะปรากฏที่เฮดเดอร์ด้านซ้าย และเมื่อกดที่ปุ่ม จะขึ้นแถบเมนูที่หน้าจอฝั่งซ้าย และสามารถกดปุ่มเลือกเมนูได้ทั้งที่หน้า CategoriesScreen และ FiltersScreen

ชื่อ.....

รหัสนักศึกษา.....



ปรับปรุงไฟล์ MealsNavigator.js

1. ในส่วนของ MainNavigator ให้ทำการกำหนด drawerLabel ของ MealsFav เป็น “Meals” (ทำโดยกำหนดผ่าน navigationOptions) (Slide.21)
2. นศ.สามารถตั้งค่าต่างๆ ของ drawer navigator ได้ โดยการกำหนดอาร์กิวเมนต์ที่สอง (NavigationConfig) ของ createDrawerNavigator() ให้กำหนดค่าดังนี้

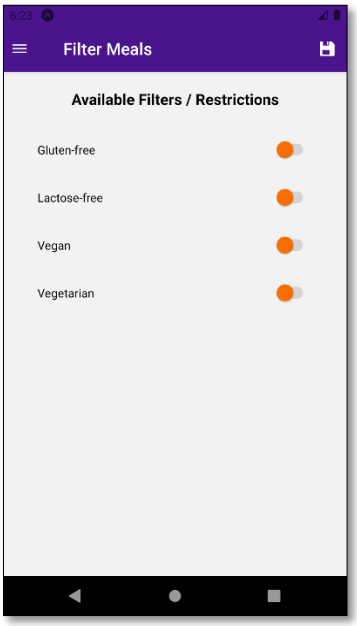
   { contentOptions: { activeTintColor: สีตามชอบ } }
3. ทดลองรันโปรแกรม จะพบว่าส่วนของแถบเมนู จะแสดงหัวข้อ Meals และ Filters ซึ่งหัวข้อที่ทำงานอยู่จะมีสีตามที่กำหนด

ชื่อ.....

รหัสนักศึกษา.....

ปรับปรุงไฟล์ FiltersScreen.js

หน้าจอนี้จะแสดงส่วนการตั้งค่าเพื่อใช้กรองเมนูอาหารที่ต้องการ โดยผู้ใช้สามารถกรองเมนูอาหารจากการตั้งค่า Gluten-free, Lactose-free, Vegan และ Vegetarian ตามลำดับ ซึ่งแสดงดังรูป



4. ให้กำหนดสเตท (state) ที่เกี่ยวข้องกับการตั้งค่าทั้งหมดในไฟล์ FiltersScreen.js
    - ชื่อสเตท : isGlutenFree, isLactoseFree, isVegan, isVegetarian

- เมธอดที่ใช้กำหนดค่าสเตท : setIsGlutenFree, setIsLactoseFree, setIsVegan, setIsVegetarian
- ค่าเริ่มต้นของสเตทเป็น false ทั้งหมด

5. ทำการปรับปรุงหน้าจอ FiltersScreen ดังนี้
   - แสดงข้อความ “Available Filters / Restrictions” ในหน้าจอ
   - แสดงส่วนของการตั้งค่า ซึ่งประกอบด้วย
     - ข้อความแสดงหัวข้อที่ต้องการตั้งค่า เช่น Gluten-free และ Lactose-free เป็นต้น
     - คอมโพเนนต์ Switch เป็น toggle ที่ผู้ใช้สามารถเปิดปิด เพื่อบอกว่าจะกรองข้อมูลในหัวข้อนั้นๆ หรือไม่
6. ให้กำหนดคอมโพเนนต์ Switch ด้วย property ดังนี้
   - trackColor : เป็นสีของแถบของ toggle
     - trackColor = {{ true: สีตามชอบ, false: สีตามชอบ}}
     - กรณีที่ Switch ถูกเปิด/ปิด จะแสดงแถบสีตามที่กำหนด
   - thumbColor : เป็นสีหมุดของ toggle
   - value : เป็นค่าสเตทที่สอดคล้องกับหัวข้อของ Switch นั้นๆ
   - onValueChange : กรณีที่ค่า Switch ถูกเปลี่ยนแปลง จะทำการอัพเดทค่าให้กับสเตทที่เกี่ยวข้อง
7. ให้สร้างปุ่ม Save บริเวณเฮดเดอร์ด้านขวา
   - title : Save
   - iconName : ios-save
8. ทดลองรันโปรแกรม จะได้หน้าจอตามรูปข้างต้น และสามารถกด Switch เลือกกรองข้อมูลต่างๆ ได้

ปรับปรุงไฟล์ MealDetailScreen.js

ให้ปรับปรุงหน้าแสดงรายละเอียดของเมนูอาหาร ดังนี้

- ให้สามารถเลื่อนหน้าจอขึ้นลงได้
  - สามารถใช้ ScrollView ช่วยในการเลื่อนหน้าจอขึ้นลง
- สามารถแสดงรูปภาพและมีรายละเอียดใต้ภาพ
  - สามารถศึกษารูปแบบการแสดงรูปภาพและรายละเอียดใต้ภาพจาก MealItem.js ได้
- มีรายละเอียดของวัตถุดิบและวิธีทำอาหารของเมนูนั้นๆ
  - สามารถใช้ map() เพื่อช่วยในการแสดงรายการวัตถุดิบและวิธีทำอาหารได้
- ให้สร้างปุ่มดาว ที่บริเวณเฮดเดอร์ด้านขวา เพื่อใช้เพิ่มรายการอาหารนั้นเป็นเมนูที่ชื่นชอบ

ชื่อ.....

รหัสนักศึกษา.....