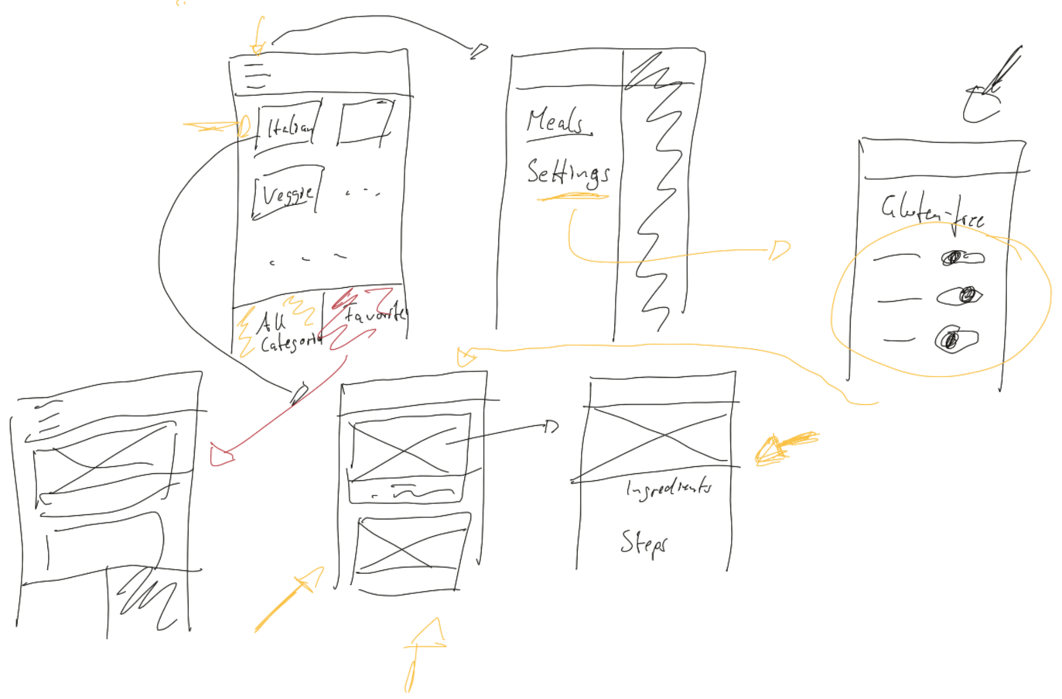
Week 8

Navigation and Multiple Pages

- What happens if we have more than one screen/page?
    - Navigation & Screen
    - Pushing and Popping Pages
    - Tabs & Drawers
    - Passing data between screens
- We will try to build a MEALS APP. The app will have categories and meals. User can add to favorites of any meal. Also, user can filter the data.



- Technically a Scaffold is a Page (or Screen).
- Create a separate file for Categories screen and create a stateless widget that returns a GridView (items are listed in multiple columns and rows):

```
class CategoryScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return GridView(
      children: [],
      gridDelegate: const SliverGridDelegateWithMaxCrossAxisExtent(
        maxCrossAxisExtent: 200, //if you have a device with 300px width only one category is shown in a row if you
have 500px then two categories are lied side by side in a row
        childAspectRatio: 3/2, // for 200px width, I need 300px height (for extra spacing)
        crossAxisSpacing: 20,
        mainAxisSpacing: 20,
      ),
    );
  }
}
```

- Add a new data model class to your app:

```
class Category{
 final String id;
 final String title;
 final Color color;
 Category({
  required this.id,
  required this.title,
  this.color=Colors.orange});
}
```

- Add some dummy data:

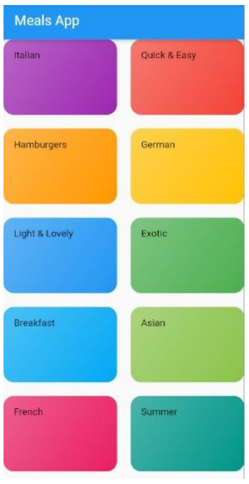
```
const DUMMY_CATEGORIES = const [
 Category(
  id: 'c1',
  title: 'Italian',
  color: Colors.purple,
 ),
 Category(
  id: 'c2',
  title: 'Quick & Easy',
  color: Colors.red,
 ),
 Category(
  id: 'c3',
  title: 'Hamburgers',
  color: Colors.orange,
 ),
 Category(
  id: 'c4',
  title: 'German',
  color: Colors.amber,
 ),
 Category(
  id: 'c5',
  title: 'Light & Lovely',
  color: Colors.blue,
 ),
 Category(
  id: 'c6',
  title: 'Exotic',
  color: Colors.green,
 ),
 Category(
  id: 'c7',
  title: 'Breakfast',
  color: Colors.lightBlue,
 ),
 Category(
  id: 'c8',
  title: 'Asian',
  color: Colors.lightGreen,
 ),
 Category(
  id: 'c9',
  title: 'French',
  color: Colors.pink,
 ),
 Category(
  id: 'c10',
  title: 'Summer',
  color: Colors.teal,
 ),
];
```

- Add a new class named CategoryItem:

```
class CategoryItem extends StatelessWidget{
  final String id;
  final String title;
  final Color color;
  CategoryItem(this.id, this.title, this.color);

  @override
  Widget build(BuildContext context){
    return Container(
      padding: const EdgeInsets.all(15),
      child: Text(title),
      decoration: BoxDecoration(
        gradient: LinearGradient(
          colors: [
            color.withOpacity(0.7),
            color,
          ],
          begin: Alignment.topLeft,
          end: Alignment.bottomRight,
        ),
        borderRadius: BorderRadius.circular(15),
      ),
    );
  }
}
```

- Now change the **children:** of **CategoryScreen**:
- *DUMMY_CATEGORIES.map((catData)=>CategoryItem(catData.id, catData.title, catData.color)).toList(),*
- Wrap the **GridView** with a **Scaffold** to make it a screen, then call this from a MaterialApp's **home:** parameter:

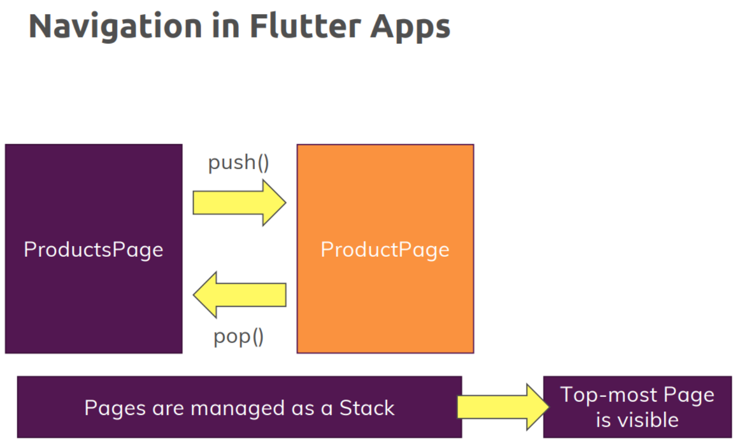


```
void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Meals App',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: CategoryScreen(),
    );
  }
}
class CategoryScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Meals App'),),
      body: GridView(
        children: DUMMY_CATEGORIES.map((catData)=>CategoryItem(catData.id, catData.title,catData.color)).toList(),
        gridDelegate: const SliverGridDelegateWithMaxCrossAxisExtent(
          maxCrossAxisExtent: 200,
          childAspectRatio: 3/2,
          crossAxisSpacing: 20,
          mainAxisSpacing: 20,
        ),
      ),
    );
  }
}
```

- You can add extra styling and theming for the GridView.
- Create a new page/screen for the meals in a specific category:

```
class CategoryMealsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Meals'),
      ),
      body: Center(
        child: Text('Recipes for this category:'),
      ),
    );
  }
}
```

- We will make the category items in the GridView clickable and when a user hits a category item, we will route to CategoryMealsScreen.

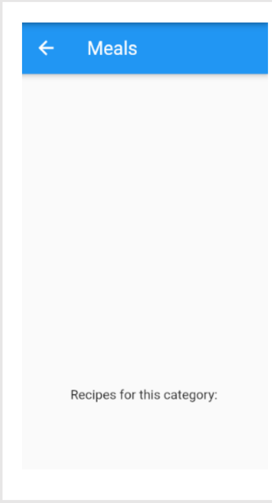


- Wrap the Container in the CategoryItem class with an InkWell widget:

```
class CategoryItem extends StatelessWidget {
  final String id;
  final String title;
  final Color color;
  CategoryItem(this.id, this.title, this.color);

  void selectCategory(BuildContext ctx){
    Navigator.of(ctx).push(
      MaterialPageRoute(
        builder: (_){
          return CategoryMealsScreen();
        },
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    return InkWell(
      onTap: () => selectCategory(context),
      splashColor: Colors.orange,
      borderRadius: BorderRadius.circular(15),
      child: Container(
```



- How can we pass data from one screen to another?
- Add two variables and a constructor to the **CategoryMealsScreen** class:

```
  final String categoryId;
  final String categoryTitle;
  CategoryMealsScreen(this.categoryId,this.categoryTitle);
```

- Now change the **return CategoryMealsScreen();** line of **selectCategory** of **CategoryItem** to: return CategoryMealsScreen(id,title);
- Another way of passing data from one screen to another> using Named Routes
- First add comments to the variables and the constructor of **CategoryMealsScreen**:

```
//final String categoryId;
//final String categoryTitle;
//CategoryMealsScreen(this.categoryId,this.categoryTitle);
```

- Add **routes:** parameter after the **home:** of **MyApp**:

```
  routes: {
    '\category-meals': (ctx) => CategoryMealsScreen(),
  }
```

- Change the **selectCategory** of **CategoryItem** class:

```
void selectCategory(BuildContext ctx){
  Navigator.of(ctx).pushNamed(
    '/category-meals',
    arguments: {
      'id': id,
      'title' : title,
    },
  );
}
```

- Add this inside of the **build** method of **CategoryMealsScreen** to extract the named route's arguments:

```
  final routeArgs = ModalRoute.of(context)!.settings.arguments as Map<String,String>;
  final String categoryTitle = routeArgs['title']!;
```

- The named routes are easy to manage multiple screen projects.
- You can add a route to your main screen and disable the **home:** parameter and add an **initialRoute:** parameter instead.
- You can add a **static const routeName ='…';** to your every screen and use it in whenever you want just by typing for ex. **CategoryMealsScreen.routeName**
- Define a new model class named Meal and enums:

```
class Meal {
  final String id;
  final List<String> categories;
  final String title;
  final String imageUrl;
  final List<String> ingredients;
  final List<String> steps;
  final int duration;
  final Complexity complexity;
  final Affordability affordability;
  final bool isGlutenFree;
  final bool isLactoseFree;
  final bool isVegan;
  final bool isVegetarian;
  const Meal({
    required this.id,
    required this.categories,
    required this.title,
    required this.imageUrl,
    required this.ingredients,
    required this.steps,
    required this.duration,
    required this.complexity,
    required this.affordability,
    required this.isGlutenFree,
    required this.isLactoseFree,
    required this.isVegan,
    required this.isVegetarian,
  });
}

enum Complexity {
  Simple,
  Challenging,
  Hard,
}

enum Affordability {
  Affordable,
  Pricey,
  Luxurious,
}
```

- Download the Week8_dummy_data.dart from Canvas and copy+paste the content into your Dartpad or Android Studio file. Please remember if you have already the DUMMY_CATEGORIES, copy only the DUMMY_MEALS.
- Extract the variable **id** from route arguments in **CategoryMealsScreen**'s **build** method:

```
    final String categoryId = routeArgs['id']!;
```

- Use where method of the **DUMMY_MEALS** to find the exact meals of the selected category:

```
    final categoryMeals = DUMMY_MEALS.where((meal){
      return meal.categories.contains(categoryId);
    }).toList();
```

- Change the body of CategoryMealsScreen:

```
      body: Center(
        child: ListView.builder(
          itemBuilder: (ctx,index){
            return Text(categoryMeals[index].title);
          },
          itemCount: categoryMeals.length,
        ),
      ),
```

References

- https://www.udemy.com/course/learn-flutter-dart-to-build-ios-android-apps/