CET215: Mobile Application Development

Lecture 6: Navigations & **User Inputs with Forms**

© Spring 2025– **Dr. Ahmed Elrefaiy**
Ahmed.elrefai@sut.edu.eg

Navigation in Flutter

- Navigation in Flutter allows users to move between different screens (pages) in an app.
- It helps manage the user flow within the application.
- How to make many pages:
    - Make sure that the main function return MaterialApp
    - Create class for each new page, and return Scaffold from each page
    - Navigator occurs between Scaffold classes, where there are one MaterialApp Widget in the main

Types of Navigation in Flutter

- Stack-Based Navigation (Imperative)
  - Uses Navigator.push() and Navigator.pop().
  - Each new page is pushed onto a stack, and the back button pops it off.
- Named Routes (Declarative Navigation)
  - Defines routes in MaterialApp.
  - Uses Navigator.pushNamed() and Navigator.pop()
- Bottom Navigation Bar
  - using BottomNavigationBar
- Drawer Navigation
  - Using side menu (drawer)

Stack-Based Navigation

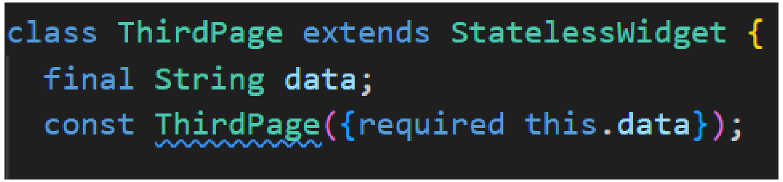
```
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => SecondPage())
);

Navigator.pop(context); // Returns to the previous screen
```

- Navigator → is a built-in Flutter class that manages the navigation stack.
  - Navigator in widget, and to work flutter search for it inside the provided context
- When a new screen is pushed, it appears on top of the previous screen.
- Pop method removes the top page and goes back.
- The **context** represents the current location in the widget tree.
- **MaterialPageRoute** → is a widget that helps transition between screens.
- **builder** → is a function that tells Flutter which page to display when navigating.

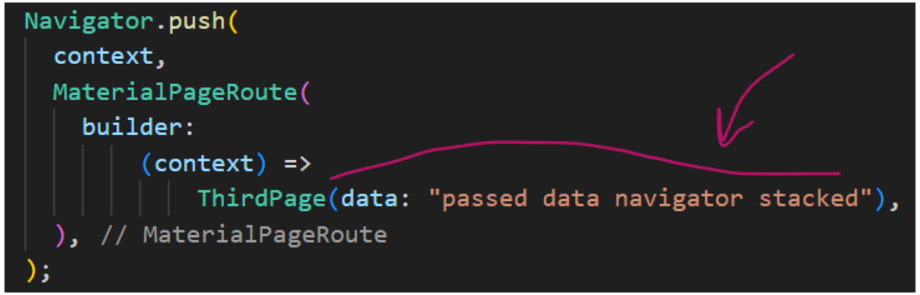
Stack-Based Navigation: Coding Example

Stack-Based Navigation with Parameters



```
class ThirdPage extends StatelessWidget {
  final String data;
  const ThirdPage({required this.data});
```

- Create you class with overloaded constructor → Pass data to constructor while navigation



```
Navigator.push(
  context,
  MaterialPageRoute(
    builder:
        (context) =>
            ThirdPage(data: "passed data navigator stacked"),
  ), // MaterialPageRoute
);
```

Stack-Based Navigation with Parameters : Coding Example

Named Routes Navigation

- Define all routes from the MaterialApp
  - Give each class widget name
  - / used for the initial route
- Push to the stack using route name.
- **<u>Can be used with navbar pages from the home</u>**
- What if need to pass dynamic data to SecondPage from home page?

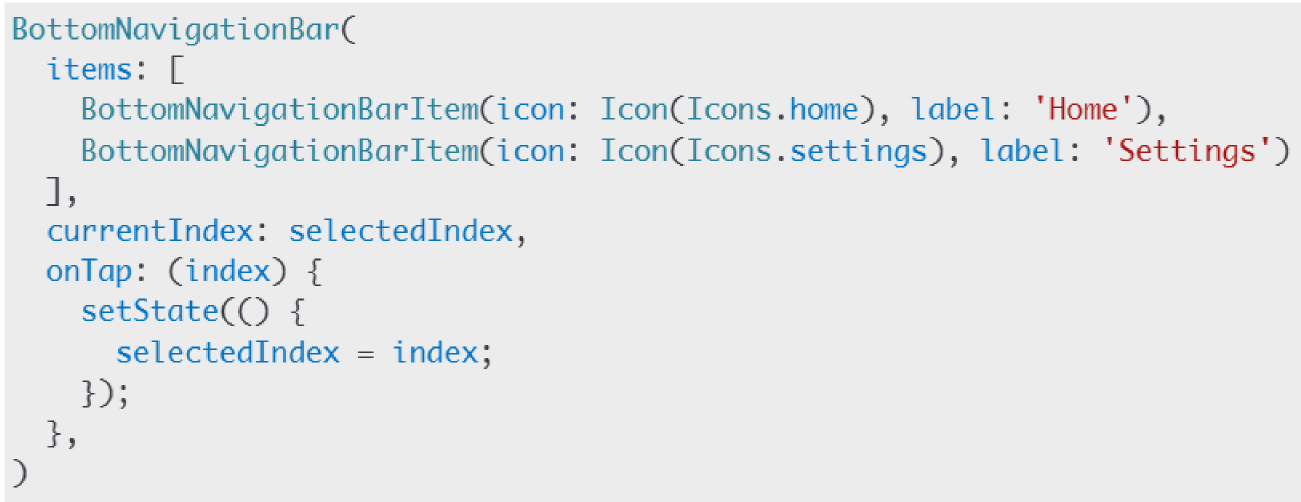
```
void main() {
  runApp(MaterialApp(
    initialRoute: '/',
    routes: {
      '/': (context) => HomePage(),
      '/second': (context) => SecondPage(),
    },
  ));
}

Navigator.pushNamed(context, '/second');
```

Named Routes Navigation: Coding Example

Bottom Navigation Bar

- BottomNavigationBar → used to state navigation buttons.
- Must be used with Stateful Widget, as it required manage states.
- On tap → flutter return selected index, so we need to refresh the current Index with clicked one



```
BottomNavigationBar(
  items: [
    BottomNavigationBarItem(icon: Icon(Icons.home), label: 'Home'),
    BottomNavigationBarItem(icon: Icon(Icons.settings), label: 'Settings')
  ],
  currentIndex: selectedIndex,
  onTap: (index) {
    setState(() {
      selectedIndex = index;
    });
  },
)
```

Bottom Navigation Bar: Coding Example

Drawer Navigation

- Drawer → is the sidebar with items you need to navigate.

```
Drawer(
  child: ListView(
    children: [
      ListTile(
        title: Text("Home"),
        onTap: () {
          Navigator.pushNamed(context, '/');
        },
      ),
      ListTile(
        title: Text("Settings"),
        onTap: () {
          Navigator.pushNamed(context, '/settings');
        },
      )
    ],
  ),
)
```

Drawer Navigation : Coding Example

Gesture Detector

- A widget that detects gestures like taps, swipes, double taps, and long presses.
- Does not have a visible UI but wraps around other widgets to add gesture functionality.
- Where Can We Use It?
    - Tap gestures: Navigate to a new page when tapping a widget.
    - Swipe gestures: Navigate between pages like a photo gallery.
    - Long press: Show extra options or navigate.
    - Double tap: Zoom in an image or trigger a special action.

```
GestureDetector(
  onLongPress: () {
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => SecondPage()
    );
  },
  child: ElevatedButton(
    onPressed: () {},
    child: Text("Long Press to Navigate"),
  ),
)
```

Gesture Detector : Coding Example

User Inputs and Forms

- Forms allow users to enter and submit data
- Used for login, registration, and other input fields.
- Ensures correct data entry with validation.
- Steps to make form with user input:
  - Create form using **Form** widget with unique form key
  - Place **TextFormField** Widget inside it
  - Write Validator for each **TextFormField** that validate its value

```
final _formKey = GlobalKey<FormState>();
```

```
TextFormField(
  decoration: InputDecoration(labelText: "Email"),
  validator: (value) => value!.isEmpty ? "Enter email" : null,
)
```

```
ElevatedButton(
  onPressed: () {
    if (_formKey.currentState!.validate()) {
      print("Login Successful");
    }
  },
  child: Text("Login"),
)
```

User Inputs and Forms

- **TextEditingController** → Widget class used to control and retrieve text from **TextField** or **TextFormField**

```
final TextEditingController emailController = TextEditingController();
```

```
TextFormField(
  controller: emailController, // Connects controller to the text field
  decoration: InputDecoration(labelText: "Email"),
)
```

```
String email = emailController.text; // Retrieves the entered email
```

User Inputs and Forms: Coding Example