⚡

#8 Create the Lesson Page with Implement Markdown Package

Introduction

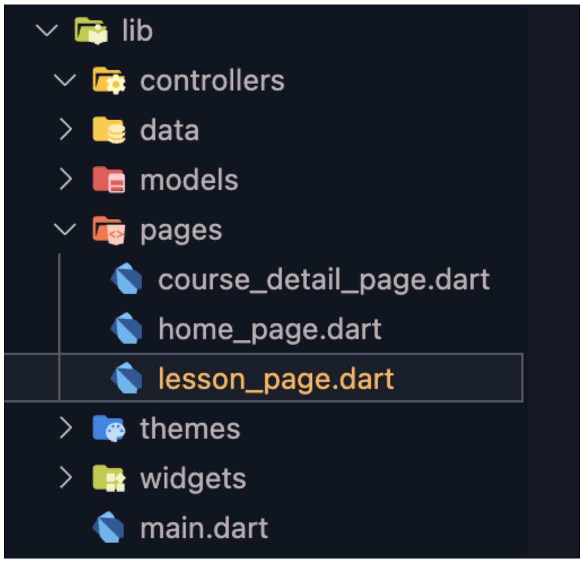
Implementing the Markdown package in a Flutter app allows for easy and efficient formatting of text. This is especially useful when creating educational content such as lesson pages. In this section, we will go through a step-by-step guide on how to create a lesson page with the Markdown package in our Flutter app, similar to the layout used on educative.io.

Step-by-step guide

To further expand on the step-by-step guide for creating a lesson page with the Markdown package in our Flutter app, let's break down each step in more detail.

Step 1 - Create a `lesson_page.dart` file

Inside `lib/pages` directory create a new file called `lesson_page.dart`.



Step 2 - Create a ConsumerStatefulWidget class

In this step, we will create a `ConsumerStatefulWidget` class for our `LessonPage`. We will also add the basic structure for the `build` method of our `LessonPageState`.

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

import '../models/course.dart';

class LessonPage extends ConsumerStatefulWidget {
  const LessonPage({super.key, required this.course});
  final Course course;

  @override
  ConsumerState<LessonPage> createState() => _LessonPageState();
}

class _LessonPageState extends ConsumerState<LessonPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: NestedScrollView(
        headerSliverBuilder: (context, _) {
          return [
            const SliverAppBar(),
          ];
        },
        body: Container(),
      ),
    );
  }
}
```

With this code, we have created a new `ConsumerStatefulWidget` class called `LessonPage`, which takes a `Course` object as a required parameter. Inside the `build` method of `LessonPageState`, we have returned a `Scaffold` widget with a `NestedScrollView`. This will allow us to create a collapsible `AppBar` section with the title of our lesson, and a scrollable `body` section for the rest of our content.

The `body` of the `NestedScrollView` is a `Container` widget that is initially empty. We will finish creating this widget section in later steps.

Step 3 - Implementation of navigation from CourseDetailPage to LessonPage

In this step, we will implement the navigation from `CourseDetailPage` to `LessonPage`. Inside `lib/pages/course_detail_page.dart`, add the following code to the `onPressed` parameter of the `ElevatedButton` widget with the text **“Start Learning”**:

```
onPressed: () {
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => LessonPage(course: widget.course),
    ),
  );
},
```

This code will navigate to the `LessonPage` when a lesson is tapped. We are passing the `Course` object as a parameter to the `LessonPage` so that we can access the title and content of the lesson. With this step, we have completed the implementation of the navigation from `CourseDetailPage` to `LessonPage`.

If successful, the result will look like the following video:

https://www.loom.com/share/3fcbf2cea1b9407d880280b15dae19d7

Step 4 - Finish creating the AppBar section.

Now we will add the title and progress bar to the `AppBar` section of our `LessonPage`. Inside the `headerSliverBuilder` of our `NestedScrollView`, add the following code:

```
 SliverAppBar(
  foregroundColor: MyColors.black,
  backgroundColor: Colors.white,
  centerTitle: false,
  pinned: true,
  title: Text(
    widget.course.title,
    style: MyTypography.titleSmall,
    overflow: TextOverflow.ellipsis,
  ),
  actions: [
    Padding(
      padding: const EdgeInsets.symmetric(horizontal: 20),
      child: Center(
        child: Text(
          '1 / 10',
          style: MyTypography.bodySmall,
        ),
      ),
    )
  ],
),
```

Here, we are using a `SliverAppBar` widget to create the collapsible `AppBar` section. We are setting the `pinned` property to `true` so that the `AppBar` will remain at the top of the screen even when the user scrolls down. We are also adding a title to the `AppBar` with the `Text` widget, and a progress bar with the `Padding` and `Center` widgets.

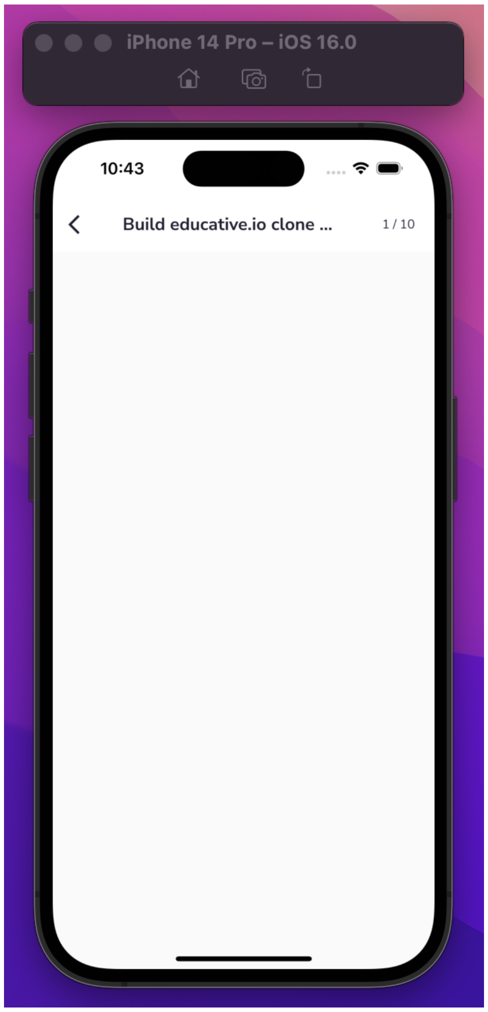
Don't forget to import the `MyColors` and `MyTypography` classes that are used in the `SliverAppBar` widget. They should be imported at the top of the file along with the other necessary imports.

```
import '../themes/colors.dart';
import '../themes/typography.dart';
```

So far, populating the `lesson_page.dart` file will look something like this:

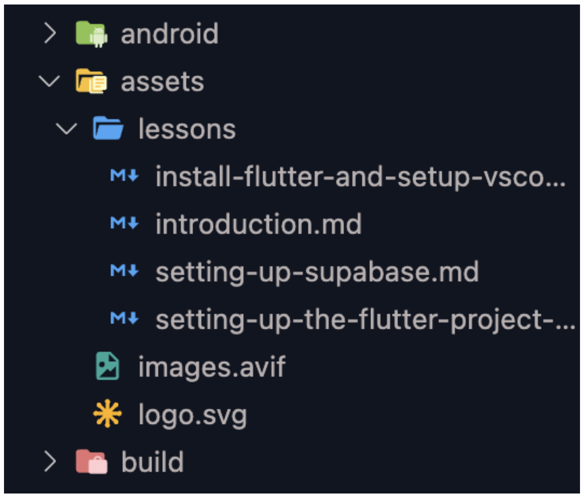
```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import '../models/course.dart';
import '../themes/colors.dart';
import '../themes/typography.dart';

class LessonPage extends ConsumerStatefulWidget {
  const LessonPage({super.key, required this.course});
  final Course course;

  @override
  ConsumerState<LessonPage> createState() => _LessonPageState();
}

class _LessonPageState extends ConsumerState<LessonPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: NestedScrollView(
        headerSliverBuilder: (context, _) {
          return [
            SliverAppBar(
              foregroundColor: MyColors.black,
              backgroundColor: Colors.white,
              centerTitle: false,
              pinned: true,
              title: Text(
                widget.course.title,
                style: MyTypography.titleSmall,
                overflow: TextOverflow.ellipsis,
              ),
              actions: [
                Padding(
                  padding: const EdgeInsets.symmetric(horizontal: 20),
                  child: Center(
                    child: Text(
                      '1 / 10',
                      style: MyTypography.bodySmall,
                    ),
                  ),
                )
              ],
            ),
          ];
        },
        body: Container(),
      ),
    );
  }
}
```

If you successfully follow step 4, the view will look like the following screenshot:



Step 5 - Complete the body section by adding lesson content using the Markdown widget.

The lesson data in Markdown format is located in the `assets` directory we added earlier, specifically in the `lessons` subdirectory. Our next step is to insert the lesson files into the widget.


```
> android
v assets
  v lessons
      install-flutter-and-setup-vsco...
      introduction.md
      setting-up-supabase.md
      setting-up-the-flutter-project-...
    images.avif
    logo.svg
> build
```

To complete the body section, we need to add the `PageView` widget, which will contain the content for each lesson. Additionally, we will implement navigation between lessons and the ability to mark lessons as completed.

Inside the `body` parameter of the `NestedScrollView` widget add the following `PageView` widget:

```
PageView.builder(
  controller: _pageController,
  itemCount: lessonContents.length,
  onPageChanged: (value) {},
  physics: const NeverScrollableScrollPhysics(),
  itemBuilder: (context, index) {
    bool isLastPage = index == lessons.length - 1;

    return LessonContent(
      lesson: lessonsContents[index],
      child: buildActionButton(index, isLastPage, lessonsContents),
    );
  },
),
```

Then, we need to create a variable `_pageController` with type `PageController` and initialize the variable inside `initState` function, and also don't forget to dispose of the `_pageController` inside the `dispose` function into class the `_LessonPageState` like this:

```
late PageController _pageController;

@override
void initState() {
    super.initState();
    _pageController = PageController();
  }

  @override
  void dispose() {
    _pageController.dispose();
    super.dispose();
  }
```

Next, we must define dummy data inside the `lib/data/dummy_data.dart` file. We will store the list of lesson content in a variable called `lessonsContents`.

Here is variable the list of lesson content:

```
// List of dummy lesson childs
final List<LessonChild> lessonsContents = [
  LessonChild(
    id: '',
    title: 'Introduction',
    content: 'assets/lessons/introduction.md',
    isCompleted: false,
    lessonId: '',
    createdAt: DateTime.now(),
  ),
  LessonChild(
    id: '',
    title: 'Installing Flutter',
    content: 'assets/lessons/install-flutter-and-setup-vscode.md',
    isCompleted: false,
    lessonId: '',
    createdAt: DateTime.now(),
  ),
  LessonChild(
    id: '',
    title: 'Installing Supabase',
    content: 'assets/lessons/setting-up-supabase.md',
    isCompleted: false,
    lessonId: '',
    createdAt: DateTime.now(),
  ),
  LessonChild(
    id: '',
    title: 'Setting Up the Database',
    content: 'assets/lessons/setting-up-the-flutter-project-and-connect-to-the-supabase-ap
i.md',
    isCompleted: false,
    lessonId: '',
    createdAt: DateTime.now(),
  ),
];
```

For now, populate the `lib/data/dummy_data.dart` file will look something like this:

```
import '../models/course.dart';
import '../models/lesson.dart';

final List<Course> courses = [
  ...
];

// List of dummy lessons
final List<Lesson> lessons = [
  ...
];

// List of dummy lesson children
final List<LessonChild> lessonsContents = [
  LessonChild(
    id: '',
    title: 'Introduction',
    content: 'assets/lessons/introduction.md',
    isCompleted: false,
    lessonId: '',
    createdAt: DateTime.now(),
  ),
  LessonChild(
    id: '',
    title: 'Installing Flutter',
    content: 'assets/lessons/install-flutter-and-setup-vscode.md',
    isCompleted: false,
    lessonId: '',
    createdAt: DateTime.now(),
  ),
  LessonChild(
    id: '',
    title: 'Installing Supabase',
    content: 'assets/lessons/setting-up-supabase.md',
    isCompleted: false,
    lessonId: '',
    createdAt: DateTime.now(),
  ),
  LessonChild(
    id: '',
    title: 'Setting Up the Database',
    content: 'assets/lessons/setting-up-the-flutter-project-and-connect-to-the-supabase-ap
i.md',
    isCompleted: false,
    lessonId: '',
    createdAt: DateTime.now(),
  ),
];
```

Once the `lessonsContents` list has been defined, it can be used in the `lesson_page.dart` file. Specifically, the length of the list can be passed to the `itemCount` parameter of the `PageView` widget, allowing for the display of a list of lesson content.

To import variable `lessonsContents` list into `lesson_page.dart`, use the following syntax:

```
import '../data/dummy_data.dart';
```

So far, populating the `lesson_page.dart` file will look something like this:

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

import '../data/dummy_data.dart';
import '../models/course.dart';
import '../themes/colors.dart';
import '../themes/typography.dart';

class LessonPage extends ConsumerStatefulWidget {
  const LessonPage({super.key, required this.course});
  final Course course;

  @override
  ConsumerState<LessonPage> createState() => _LessonPageState();
}

class _LessonPageState extends ConsumerState<LessonPage> {
  late PageController _pageController;

  @override
  void initState() {
    super.initState();
    _pageController = PageController();
  }

  @override
  void dispose() {
    _pageController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: NestedScrollView(
        // AppBar
        headerSliverBuilder: (context, innerBoxIsScrolled) {
          return [
            SliverAppBar(
              foregroundColor: MyColors.black,
              backgroundColor: Colors.white,
              centerTitle: false,
              pinned: true,
              title: Text(
                widget.course.title,
                style: MyTypography.titleSmall,
                overflow: TextOverflow.ellipsis,
              ),
              actions: [
                Padding(
                  padding: const EdgeInsets.symmetric(horizontal: 20),
                  child: Center(
                    child: Text(
                      '1 / 10',
                      style: MyTypography.bodySmall,
                    ),
                  ),
                )
              ],
            ),
          ];
        },
        // Body
        body: PageView.builder(
          controller: _pageController,
          itemCount: lessonsContents.length,
          onPageChanged: (value) {},
          physics: const NeverScrollableScrollPhysics(),
          itemBuilder: (context, index) {
            bool isLastPage = index == lessons.length - 1;

            return LessonContent(
              lesson: lessonsContents[index],
              child: buildActionButton(index, isLastPage, lessonsContents),
            );
          },
        ),
      ),
    );
  }
}
```

The `itemBuilder` property of `PageView` is used to define how each page in the `PageView` should be built. In this case, we are using the `lessonsContents` list to populate each page with a `LessonContent` widget.

The `buildActionButton` method is also called, which returns a widget containing the appropriate action buttons (back, next, or completed) depending on the current page.

The `LessonContent` widget and the `buildActionButton` method are defined later in the next step.

Step 6 - Define the `LessonContent` widget.

To define the `LessonContent` widget, create a new file called `lesson_content.dart` inside the `lib/widgets` directory. Inside the file, define a new `StatefulWidget` class called `LessonContent` which accepts a `LessonChild` object as a required parameter.

The `LessonContent` widget will be responsible for displaying the content of each lesson. We will use the `MarkdownBody` widget from the `flutter_markdown` package to render the lesson content in Markdown format. The `LessonContent` widget will also contain the action buttons for navigating between lessons and marking lessons as completed.

Add the following code to `lesson_content.dart`:

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_markdown/flutter_markdown.dart';
import 'package:url_launcher/url_launcher.dart';

import '../models/lesson.dart';

class LessonContent extends StatelessWidget {
  const LessonContent({super.key, required this.lesson, required this.child});
  final LessonChild lesson;
  final Widget child;

  @override
  Widget build(BuildContext context) {
    return FutureBuilder(
      future: rootBundle.loadString(lesson.content),
      builder: (context, snapshot) {
        if (snapshot.hasData) {
          return SingleChildScrollView(
            controller: ScrollController(),
            padding: const EdgeInsets.only(
              bottom: 40,
              left: 20,
              right: 20,
              top: 30,
            ),
            child: Column(
              children: [
                // Lesson Body with Markdown
                MarkdownBody(
                  softLineBreak: true,
                  fitContent: true,
                  shrinkWrap: true,
                  selectable: true,
                  data: snapshot.data.toString(),
                  styleSheet: markdownStyleSheet(context),
                  builders: markdownBuilders(context),
                  inlineSyntaxes: markdownInlineSyntaxes,
                  imageBuilder: (uri, title, alt) {
                    return Padding(
                      padding: const EdgeInsets.only(bottom: 10, top: 5),
                      child: GestureDetector(
                        onTap: () {
                          debugPrint('Link tapped: $uri, $title, $alt');
                          launchUrl(Uri.parse(alt!));
                        },
                        child: ClipRRect(
                          borderRadius: BorderRadius.circular(5),
                          child: Image.network(
                            uri.toString(),
                            fit: BoxFit.cover,
                          ),
                        ),
                      ),
                    );
                  },
                  onTapLink: (text, href, title) {
                    debugPrint('Link tapped: $text, $href, $title');
                    launchUrl(Uri.parse(href!));
                  },
                ),
                // Actions Button
                child,
              ],
            ),
          );
        }

        return const Center(
          child: CircularProgressIndicator(),
        );
      },
    );
  }
}
```

Here, the `LessonContent` widget is defined as a stateless widget that takes in a `LessonChild` object and a child widget. The widget uses the `FutureBuilder` widget to asynchronously load the Markdown content of the lesson file. Once the content is loaded, it is displayed using the `MarkdownBody` widget from the `flutter_markdown` package. The `MarkdownBody` widget also includes builders for images and links and provides a callback function for when links are tapped. Finally, the `child` widget is displayed below the Markdown content, which contains the action buttons for navigating between lessons and marking lessons as completed.

In the `LessonContent` widget, we define a `markdownStyleSheet` to customize the styling of the Markdown content, a `markdownBuilder` to add custom widgets to the Markdown content, and a `markdownInlineSyntaxes` method to define custom inline syntaxes for the Markdown content. These will be defined and explained in the next step.

For now, just comment on them and the `MarkdownBody` widget will look something like this:

```
// Lesson Body with Markdown
MarkdownBody(
  softLineBreak: true,
  fitContent: true,
  shrinkWrap: true,
  selectable: true,
  data: snapshot.data.toString(),
  // styleSheet: markdownStyleSheet(context),
  // builders: markdownBuilders(context),
  // inlineSyntaxes: markdownInlineSyntaxes,
  imageBuilder: (uri, title, alt) {
    return Padding(
      padding: const EdgeInsets.only(bottom: 10, top: 5),
      child: GestureDetector(
        onTap: () {
          debugPrint('Link tapped: $uri, $title, $alt');
          launchUrl(Uri.parse(alt!));
        },
        child: ClipRRect(
          borderRadius: BorderRadius.circular(5),
          child: Image.network(
            uri.toString(),
            fit: BoxFit.cover,
          ),
        ),
      ),
    );
  },
  onTapLink: (text, href, title) {
    debugPrint('Link tapped: $text, $href, $title');
    launchUrl(Uri.parse(href!));
  },
),
```

Now that the `LessonContent` widget has been defined, we can import it into the `lesson_page.dart` file and can use it in the `itemBuilder` property of the `PageView`.

```
import '../widgets/lesson_content.dart';
```

Step 7 - Define the `buildActionButton` method

In this step, we will define the `buildActionButton` method inside the `_LessonPageState` class. This method will return a widget containing the appropriate action buttons (back, next, or completed) depending on the current page.

Here is the implementation of the `buildActionButton` method:

```
  // Back Button, Next Button, and Completed Button
  Widget buildActionButton(int index, bool isLastPage, List<LessonChild> lessons,) {
    return Padding(
      padding: const EdgeInsets.symmetric(vertical: 40),
      child: Column(
        children: [
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              if (index == 0) const Spacer(),
              if (index != 0)
                Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    OutlinedButton(
                      onPressed: () {
                        previousPage();
                      },
                      style: OutlinedButton.styleFrom(
                        foregroundColor: MyColors.primary,
                        side: const BorderSide(
                          color: Colors.grey,
                          width: 1,
                        ),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(5),
                        ),
                      ),
                      child: Row(
                        mainAxisSize: MainAxisSize.min,
                        children: [
                          Icon(
                            Icons.arrow_back_rounded,
                            size: 20,
                            color: MyColors.black,
                          ),
                          const SizedBox(width: 5),
                          Text(
                            'Back',
                            style: MyTypography.body,
                          ),
                        ],
                      ),
                    ),
                    const SizedBox(height: 5),
                    SizedBox(
                      width: MediaQuery.of(context).size.width * 0.3,
                      child: Text(
                        lessons[index - 1].title,
                        style: MyTypography.bodySmall,
                        overflow: TextOverflow.ellipsis,
                      ),
                    ),
                  ],
                ),
              Column(
                crossAxisAlignment: CrossAxisAlignment.end,
                children: [
                  OutlinedButton(
                    onPressed: () {
                      if (isLastPage) {
                        // TODO: finish lesson
                      } else {
                        nextPage();
                      }
                    },
                    style: OutlinedButton.styleFrom(
                      foregroundColor: MyColors.primary,
                      side: BorderSide(
                        color: MyColors.primary,
                        width: 1,
                      ),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(5),
                      ),
                    ),
                    child: Row(
                      mainAxisSize: MainAxisSize.min,
                      children: [
                        Text(
                          isLastPage ? 'Finished' : 'Next',
                          style: MyTypography.body.copyWith(
                            color: MyColors.primary,
                          ),
                        ),
                        if (!isLastPage) const SizedBox(width: 5),
                        if (!isLastPage)
                          Icon(
                            Icons.arrow_forward_rounded,
                            size: 20,
                            color: MyColors.primary,
                          ),
                      ],
                    ),
                  ),
                  const SizedBox(height: 5),
                  SizedBox(
                    width: MediaQuery.of(context).size.width * 0.3,
                    child: Text(
                      isLastPage ? '' : lessons[index + 1].title,
                      style: MyTypography.bodySmall,
                      overflow: TextOverflow.ellipsis,
                    ),
                  ),
                ],
              ),
            ],
          ),
          const SizedBox(height: 20),
          CheckboxListTile(
            onChanged: (v) {
              // TODO: mark as complete
            },
            value: false,
            tileColor: Colors.grey[100],
            activeColor: MyColors.primary.withOpacity(0.8),
            dense: true,
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(5),
            ),
            checkboxShape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(5),
            ),
            title: Text(
              'Mark as complete',
              style: MyTypography.body,
            ),
          ),
        ],
      ),
    );
  }
```

This method includes the `previousPage()` and `nextPage()` functions for navigating to the previous or next page, respectively. Additionally, it includes a `CheckboxListTile` widget that can be used to mark a page as complete.

`previousPage()` and `nextPage()` are methods that are not defined in the code snippet provided above. However, they are likely functions that will be defined in the `_LessonPageState` class. `previousPage()` will be called when the user clicks the back button, and it will navigate the user to the previous page in the `PageView`. `nextPage()` will be called when the user clicks the next button, and it will navigate the user to the next page in `PageView`.

To implement the `previousPage()` and `nextPage()` functions, we need to define them inside the `_LessonPageState` class.

```
  void nextPage() {
    _pageController.nextPage(
      duration: const Duration(milliseconds: 300),
      curve: Curves.easeIn,
    );
  }

  void previousPage() {
    _pageController.previousPage(
      duration: const Duration(milliseconds: 300),
      curve: Curves.easeIn,
    );
  }
```

The `previousPage` function calls the `previousPage` method of the `_pageController` to navigate to the previous page in the `PageView`. Similarly, the `nextPage` function calls the `nextPage` method of the `_pageController` to navigate to the next page in the `PageView`.

So far, populating the `lesson_page.dart` file will look something like this:

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

import '../data/dummy_data.dart';
import '../models/course.dart';
import '../models/lesson.dart';
import '../themes/colors.dart';
import '../themes/typography.dart';
import '../widgets/lesson_content.dart';

class LessonPage extends ConsumerStatefulWidget {
  const LessonPage({super.key, required this.course});
  final Course course;

  @override
  ConsumerState<LessonPage> createState() => _LessonPageState();
}

class _LessonPageState extends ConsumerState<LessonPage> {
  late PageController _pageController;

  @override
  void initState() {
    super.initState();
    _pageController = PageController();
  }

  @override
  void dispose() {
    _pageController.dispose();
    super.dispose();
  }

  void nextPage() {
    _pageController.nextPage(
      duration: const Duration(milliseconds: 300),
      curve: Curves.easeIn,
    );
  }

  void previousPage() {
    _pageController.previousPage(
      duration: const Duration(milliseconds: 300),
      curve: Curves.easeIn,
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: NestedScrollView(
        // AppBar
        headerSliverBuilder: (context, innerBoxIsScrolled) {
          return [
            SliverAppBar(
              foregroundColor: MyColors.black,
              backgroundColor: Colors.white,
              centerTitle: false,
              pinned: true,
              title: Text(
                widget.course.title,
                style: MyTypography.titleSmall,
                overflow: TextOverflow.ellipsis,
              ),
              actions: [
                Padding(
                  padding: const EdgeInsets.symmetric(horizontal: 20),
                  child: Center(
                    child: Text(
                      '1 / 10',
                      style: MyTypography.bodySmall,
                    ),
                  ),
                )
              ],
            ),
          ];
        },
        // Body
        body: PageView.builder(
          controller: _pageController,
          itemCount: lessonsContents.length,
          onPageChanged: (value) {},
          physics: const NeverScrollableScrollPhysics(),
          itemBuilder: (context, index) {
            bool isLastPage = index == lessonsContents.length - 1;

            return LessonContent(
              lesson: lessonsContents[index],
              child: buildActionButton(index, isLastPage, lessonsContents),
            );
          },
        ),
      ),
    );
  }

  // Back Button, Next Button, and Completed Button
  Widget buildActionButton(int index, bool isLastPage, List<LessonChild> lessons,) {
    return Padding(
      padding: const EdgeInsets.symmetric(vertical: 40),
      child: Column(
        children: [
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              if (index == 0) const Spacer(),
              if (index != 0)
                Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    OutlinedButton(
                      onPressed: () {
                        previousPage();
                      },
                      style: OutlinedButton.styleFrom(
                        foregroundColor: MyColors.primary,
                        side: const BorderSide(
                          color: Colors.grey,
                          width: 1,
                        ),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(5),
                        ),
                      ),
                      child: Row(
                        mainAxisSize: MainAxisSize.min,
                        children: [
                          Icon(
                            Icons.arrow_back_rounded,
                            size: 20,
                            color: MyColors.black,
                          ),
                          const SizedBox(width: 5),
                          Text(
                            'Back',
                            style: MyTypography.body,
                          ),
                        ],
                      ),
                    ),
                    const SizedBox(height: 5),
                    SizedBox(
                      width: MediaQuery.of(context).size.width * 0.3,
                      child: Text(
                        lessons[index - 1].title,
                        style: MyTypography.bodySmall,
                        overflow: TextOverflow.ellipsis,
                      ),
                    ),
                  ],
                ),
              Column(
                crossAxisAlignment: CrossAxisAlignment.end,
                children: [
                  OutlinedButton(
                    onPressed: () {
                      if (isLastPage) {
                        // TODO: finish lesson
                      } else {
                        nextPage();
                      }
                    },
                    style: OutlinedButton.styleFrom(
                      foregroundColor: MyColors.primary,
                      side: BorderSide(
                        color: MyColors.primary,
                        width: 1,
                      ),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(5),
                      ),
                    ),
                    child: Row(
                      mainAxisSize: MainAxisSize.min,
                      children: [
                        Text(
                          isLastPage ? 'Finished' : 'Next',
                          style: MyTypography.body.copyWith(
                            color: MyColors.primary,
                          ),
                        ),
                        if (!isLastPage) const SizedBox(width: 5),
                        if (!isLastPage)
                          Icon(
                            Icons.arrow_forward_rounded,
                            size: 20,
                            color: MyColors.primary,
                          ),
                      ],
                    ),
                  ),
                  const SizedBox(height: 5),
                  SizedBox(
                    width: MediaQuery.of(context).size.width * 0.3,
                    child: Text(
                      isLastPage ? '' : lessons[index + 1].title,
                      style: MyTypography.bodySmall,
                      overflow: TextOverflow.ellipsis,
                    ),
                  ),
                ],
              ),
            ],
          ),
          const SizedBox(height: 20),
          CheckboxListTile(
            onChanged: (v) {
              // TODO: mark as complete
            },
            value: false,
            tileColor: Colors.grey[100],
            activeColor: MyColors.primary.withOpacity(0.8),
            dense: true,
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(5),
            ),
            checkboxShape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(5),
            ),
            title: Text(
              'Mark as complete',
              style: MyTypography.body,
            ),
          ),
        ],
      ),
    );
  }
}
```

Testing the App

If you successfully follow step-by-step this guide, now let's test our app. Run the application on an emulator or device using the terminal command. The command `flutter run` will build the app and install it on your device. The result will resemble the video below:

https://www.loom.com/share/c7b93cf714c7420c94fb1b4302d9940c

In the video, you can see that the `Back` button is disabled on the first lesson page, and the `Next` button is replaced with a `Finished` button on the last lesson page. Regarding the lesson progress, the function of the `Mark as Complete` button, and the function of the `Finished` button, we will handle this in a later section.

Conclusion

In this section, we created a lesson page with a `PageView` and implemented `previousPage()` and `nextPage()` functions to navigate between pages. We also added `Back` and `Next` buttons, which are disabled on the first and last pages, respectively. Finally, we added a `Mark as complete` button, which we will handle in a later section. Congratulations, you have successfully created the lesson page for an educative.io clone with Flutter!