#12 Display Lesson Content from Supabase Database

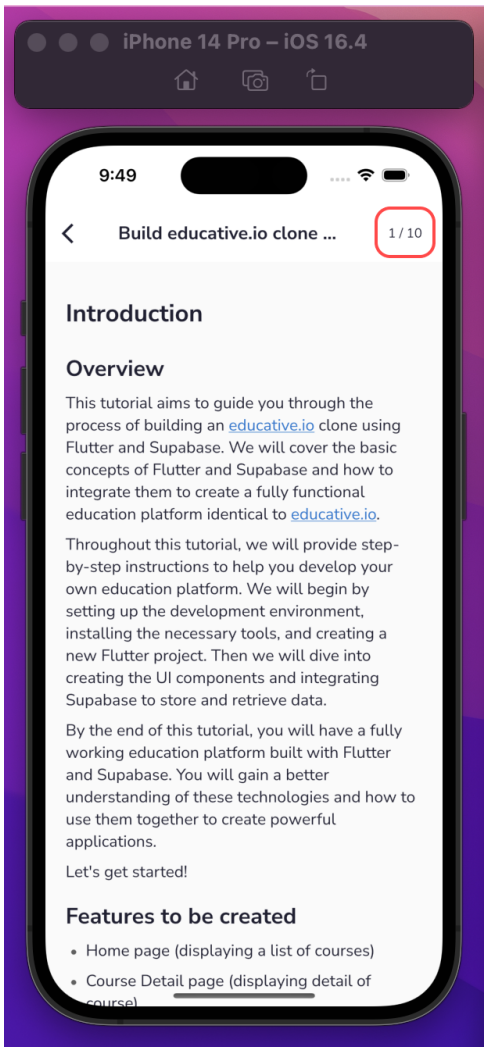
Introduction

This section will cover about integrate each widget on the `LessonPage` with data from the Supabase Database like how to display lesson content data from URL and activate the lesson progress feature on the `LessonPage` widget, etc. So, let's get started!

Implementation - Step-by-step guide

In the previous section, we have defined a class `LessonController` including the functions that we required to get data from the database and also the provider for handling the state. So, in this section, we'll focus more on the UI.

Step 1: Integrate the Progress of Lesson Widget



You can look at the image beside it, at the red circle is the progress of the lesson widget. In this step, we'll integrate it with data from Supabase Database and make it dynamic. So, let's do it!

First, define a variable that will listen for changes to `lessonChildProvider` in the `LessonPage` widget.

So, inside the `lesson_page.dart` file add the following code snippet to the `build` method of the `LessonPage` widget, specifically before returning the widget.

```
final allLessonState = ref.watch(lessonChildProvider(widget.course.id));
```

And don't forget to import `lessonChildProvider` at the top of the file along with the other necessary imports.

```
import '../controllers/lesson_controller.dart';
```

Afterward, we can use the `allLessonState` variable that we defined to build a widget.

Update the `SliverAppBar` widget in the `headerSliverBuilder` function of the `NestedScrollView` widget to look something like this:

```
            SliverAppBar(
              foregroundColor: MyColors.black,
              backgroundColor: Colors.white,
              centerTitle: false,
              pinned: true,
              title: Text(
                widget.course.title,
                style: MyTypography.titleSmall,
                overflow: TextOverflow.ellipsis,
              ),
              actions: [
                // Use allLessonState here
                allLessonState.when(
                  data: (lessons) {
                    return Padding(
                      padding: const EdgeInsets.symmetric(horizontal: 20),
                      child: Center(
                        child: Text(
                          '$currentPage / ${lessons.length}',
                          style: MyTypography.bodySmall,
                        ),
                      ),
                    );
                  },
                  loading: () {
                    return UnconstrainedBox(
                      child: Shimmer.fromColors(
                        baseColor: Colors.grey[300]!,
                        highlightColor: Colors.grey[100]!,
                        child: Container(
                          width: 50,
                          height: 20,
                          margin: const EdgeInsets.symmetric(horizontal: 16),
                          decoration: BoxDecoration(
                            color: Colors.grey,
                            borderRadius: BorderRadius.circular(10),
                          ),
                        ),
                      ),
                    );
                  },
                  error: (error, stack) {
```

```
                    return const SizedBox();
                  },
                ),
              ],
            ),
```

And then, we should import the `shimmer` package into the `lesson_page.dart` file. Because we use it in the code above.

```
import 'package:shimmer/shimmer.dart';
```

Additionally, we need to define a variable called `currentPage` . So, add the following code below into class the `_LessonPageState` :

```
int currentPage = 1;
```

Finally, we also need to update the code in the `body` property of the `NestedScrollView` widget. Specifically, in the `onPageChanged` property of the `PageView.builder` . Update the code to look something like this:

```
        // Body
        body: PageView.builder(
          controller: _pageController,
          itemCount: lessonsContents.length,
          onPageChanged: (value) {
            // Update the currentPage value
            setState(() {
              currentPage = value + 1;
            });
          },
          physics: const NeverScrollableScrollPhysics(),
          itemBuilder: (context, index) {
            bool isLastPage = index == lessonsContents.length - 1;

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

The `onPageChanged` property is a callback that is called when the current page of the `PageView` changes. It receives a `value` parameter that represents the index of the current page. And we update the `currentPage` variable by setting it to the `value` plus one since the `value` starts from zero but the `currentPage` starts from one.

So far, here is the populate of the `lesson_page.dart` file:

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

import '../controllers/lesson_controller.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() {
    ...
  }

  @override
  void dispose() {
    ...
  }

  void nextPage() {
    ...
  }

  void previousPage() {
    ...
  }

  int currentPage = 1;    // <-- Add this line
```

```
  @override
  Widget build(BuildContext context) {
    // Define allLessonState here
    final allLessonState = ref.watch(lessonChildProvider(widget.course.id));

    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: [
                // Use allLessonState here
                allLessonState.when(
                  data: (lessons) {
                    return Padding(
                      padding: const EdgeInsets.symmetric(horizontal: 20),
                      child: Center(
                        child: Text(
                          '$currentPage / ${lessons.length}',
                          style: MyTypography.bodySmall,
                        ),
                      ),
                    );
                  },
                  loading: () {
                    return UnconstrainedBox(
                      child: Shimmer.fromColors(
                        baseColor: Colors.grey[300]!,
                        highlightColor: Colors.grey[100]!,
                        child: Container(
                          width: 50,
                          height: 20,
                          margin: const EdgeInsets.symmetric(horizontal: 16),
                          decoration: BoxDecoration(
                            color: Colors.grey,
                            borderRadius: BorderRadius.circular(10),
                          ),
                        ),
                      ),
                    );
                  },
                  error: (error, stack) {
                    return const SizedBox();
                  },
```

```
                ),
              ],
            ),
          ];
        },
        // Body
        body: PageView.builder(
          controller: _pageController,
          itemCount: lessonsContents.length,
          onPageChanged: (value) {
            setState(() {
              currentPage = value + 1;
            });
          },
          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) {
    ...
  }
}
```

If you have successfully followed the steps so far, the result will look like the video shown in the progress of lesson widget.

https://www.loom.com/share/924a15b0fc2c41c68b79b00897433219?sid=c1b4946e-c234-46b8-a574-caad5e1df6ae

---

Step 2: Integrate the Lesson Content

In the previous section, we used the dummy data to display lesson content on the UI. So in this section, we'll integrate or use the lesson content data from the database to

display it on the UI. So, let's do it!

Inside the `lesson_page.dart` file or the `LessonPage` widget, update the code in the `body` property of the `NestedScrollView` widget to resemble the following:

```
        // Body
        body: allLessonState.when(
          data: (lessons) {
            return PageView.builder(
              controller: _pageController,
              itemCount: lessons.length,
              onPageChanged: (value) {
                setState(() {
                  currentPage = value + 1;
                });
              },
              physics: const NeverScrollableScrollPhysics(),
              itemBuilder: (context, index) {
                bool isLastPage = index == lessons.length - 1;
                return LessonContent(
                  lesson: lessons[index],
                  child: buildActionButton(
                    index,
                    isLastPage,
                    lessons,
                  ),
                );
              },
            );
          },
          loading: () => const LessonLoading(),
          error: (error, stack) => Center(
            child: Text(error.toString()),
          ),
        ),
```

After then, we need to define the `LessonLoading` widget as a custom widget when the state is loading using the `shimmer` package.

Create a new file called `lesson_loading.dart` inside the `lib/widgets` directory. And, add the following code snippet into the `lesson_loading.dart` file.

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

class LessonLoading extends StatelessWidget {
  const LessonLoading({super.key});
```

```
  @override
  Widget build(BuildContext context) {
    return Shimmer.fromColors(
      baseColor: Colors.grey[300]!,
      highlightColor: Colors.grey[100]!,
      period: const Duration(milliseconds: 1000),
      direction: ShimmerDirection.ltr,
      child: SingleChildScrollView(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Container(
              width: 100.0 * 2.0,
              height: 16.0 * 2,
              margin: const EdgeInsets.symmetric(vertical: 8.0),
              decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: BorderRadius.circular(10),
              ),
            ),
            for (var i = 0; i < 3; i++)
              Container(
                width: double.infinity,
                height: 16.0,
                margin: const EdgeInsets.symmetric(vertical: 8.0),
                decoration: BoxDecoration(
                  color: Colors.white,
                  borderRadius: BorderRadius.circular(10),
                ),
              ),
            Container(
              width: 100.0,
              height: 16.0,
              margin: const EdgeInsets.symmetric(vertical: 8.0),
              decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: BorderRadius.circular(10),
              ),
            ),

            // image placeholder
            Container(
              width: double.infinity,
              height: 200.0,
              margin: const EdgeInsets.symmetric(vertical: 8.0),
              decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: BorderRadius.circular(10),
              ),
            ),
            for (var i = 0; i < 3; i++)
              Container(
                width: double.infinity,
```

```
                height: 16.0,
                margin: const EdgeInsets.symmetric(vertical: 8.0),
                decoration: BoxDecoration(
                  color: Colors.white,
                  borderRadius: BorderRadius.circular(10),
                ),
              ),
            Container(
              width: 100.0,
              height: 16.0,
              margin: const EdgeInsets.symmetric(vertical: 8.0),
              decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: BorderRadius.circular(10),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

And after we defined the `LessonLoading` widget, let's import it to the `lesson_page.dart` file.

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

Finally, we should update something code inside the `lib/widgets/lesson_content.dart` file. Specifically, in the `build` method of the `LessonContent` widget, change the value of the `future` property of the `FutureBuilder` widget to resemble the following:

```
future: NetworkAssetBundle(Uri.parse(lesson.content))
          .loadString(lesson.content),
```

And don't forget to import the `NetworkAssetBundle` widget. Add the following package to the `lesson_content.dart` file:

```
import 'package:flutter/services.dart';
```

`NetworkAssetBundle` is a subclass of `AssetBundle` that retrieves resources from the network. It is useful for loading assets that are not packaged with the app but are

instead located on a server. The `NetworkAssetBundle` constructor takes a `Uri` object representing the URL of the asset, and it returns an `AssetBundle` object that can be used to load the asset.

The `NetworkAssetBundle` is often used in conjunction with `FutureBuilder` to asynchronously load and display remote assets in the app.

Here's the full code of the `lesson_content.dart` file:

```
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';
import 'markdown_style.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: NetworkAssetBundle(Uri.parse(lesson.content))
          .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(),
        );
      },
    );
  }
}
```

If you have successfully followed the steps so far, the result will look like the video:

https://www.loom.com/share/b4135469188949ca832f14ce0a39133b?sid=9df10781-b286-47ca-a6a5-c0e4945dac18

Full Code of `lesson_page.dart` File

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

import '../controllers/lesson_controller.dart';
import '../models/course.dart';
import '../models/lesson.dart';
import '../themes/colors.dart';
import '../themes/typography.dart';
import '../widgets/lesson_content.dart';
import '../widgets/lesson_loading.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,
    );
  }

  int currentPage = 1;

  @override
```

```
  Widget build(BuildContext context) {
    final allLessonState = ref.watch(lessonChildProvider(widget.course.id));

    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: [
                // Use allLessonState here
                allLessonState.when(
                  data: (lessons) {
                    return Padding(
                      padding: const EdgeInsets.symmetric(horizontal: 20),
                      child: Center(
                        child: Text(
                          '$currentPage / ${lessons.length}',
                          style: MyTypography.bodySmall,
                        ),
                      ),
                    );
                  },
                  loading: () {
                    return UnconstrainedBox(
                      child: Shimmer.fromColors(
                        baseColor: Colors.grey[300]!,
                        highlightColor: Colors.grey[100]!,
                        child: Container(
                          width: 50,
                          height: 20,
                          margin: const EdgeInsets.symmetric(horizontal: 16),
                          decoration: BoxDecoration(
                            color: Colors.grey,
                            borderRadius: BorderRadius.circular(10),
                          ),
                        ),
                      ),
                    );
                  },
                  error: (error, stack) {
                    return const SizedBox();
                  },
                ),
              ],
```

```
            ),
          ];
        },
        // Body
        body: allLessonState.when(
          data: (lessons) {
            return PageView.builder(
              controller: _pageController,
              itemCount: lessons.length,
              onPageChanged: (value) {
                setState(() {
                  currentPage = value + 1;
                });
              },
              physics: const NeverScrollableScrollPhysics(),
              itemBuilder: (context, index) {
                bool isLastPage = index == lessons.length - 1;
                return LessonContent(
                  lesson: lessons[index],
                  child: buildActionButton(
                    index,
                    isLastPage,
                    lessons,
                  ),
                );
              },
            );
          },
          loading: () => const LessonLoading(),
          error: (error, stack) => Center(
            child: Text(error.toString()),
          ),
        ),
      ),
    );
  }

  // 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,
            ),
          ),
        ],
      ),
    );
  }
}
```

Conclusion

Fantastic! We have integrated widgets in the `LessonPage` like the progress of the lesson widget and also the lesson content widget with data from the database. Now, we are nearing the end of the tutorial on creating an educative.io clone app. So, in the next section, we will finish the functionality that is required in our app like the function of Finish Lesson Course and Mark as Complete each lesson course. Happy coding! 🚀