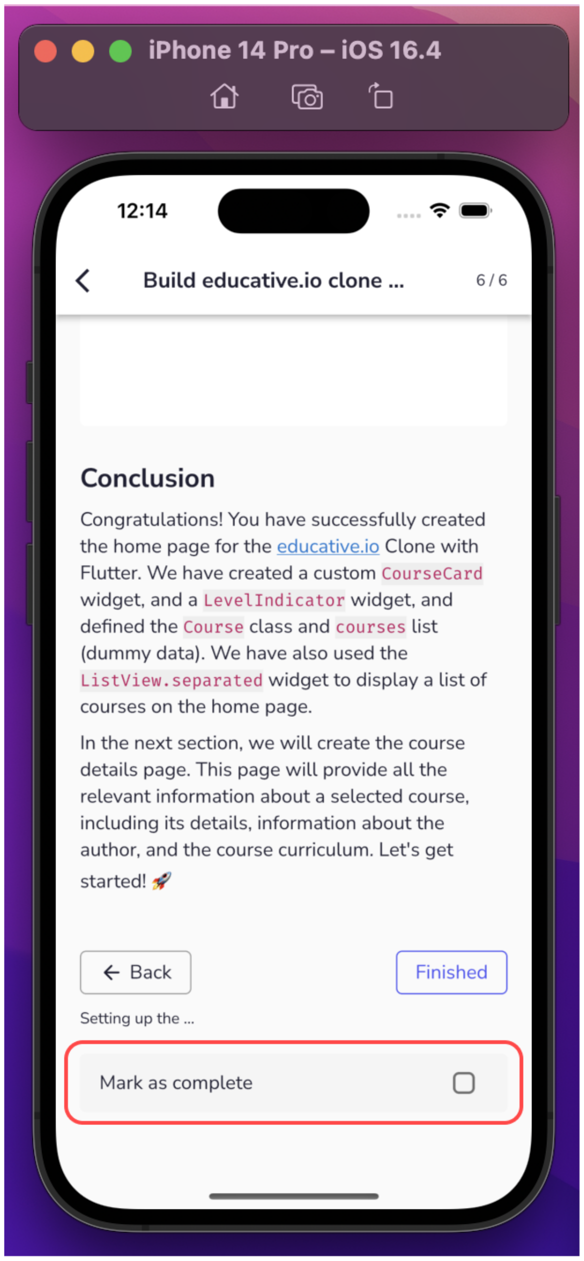
⚡

#13 Create Functionality of the Finish Course and Mark as Complete Lesson Course

Introduction

Hey there! In this section, we're going to wrap up the functionality that's required in our app, including the ability to Finish the Course and Mark as Complete each lesson course. Plus, this is the final section in our tutorial on how to **Build an educative.io Clone with Flutter**. Hope you've had fun following along!


iPhone 14 Pro – iOS 16.4

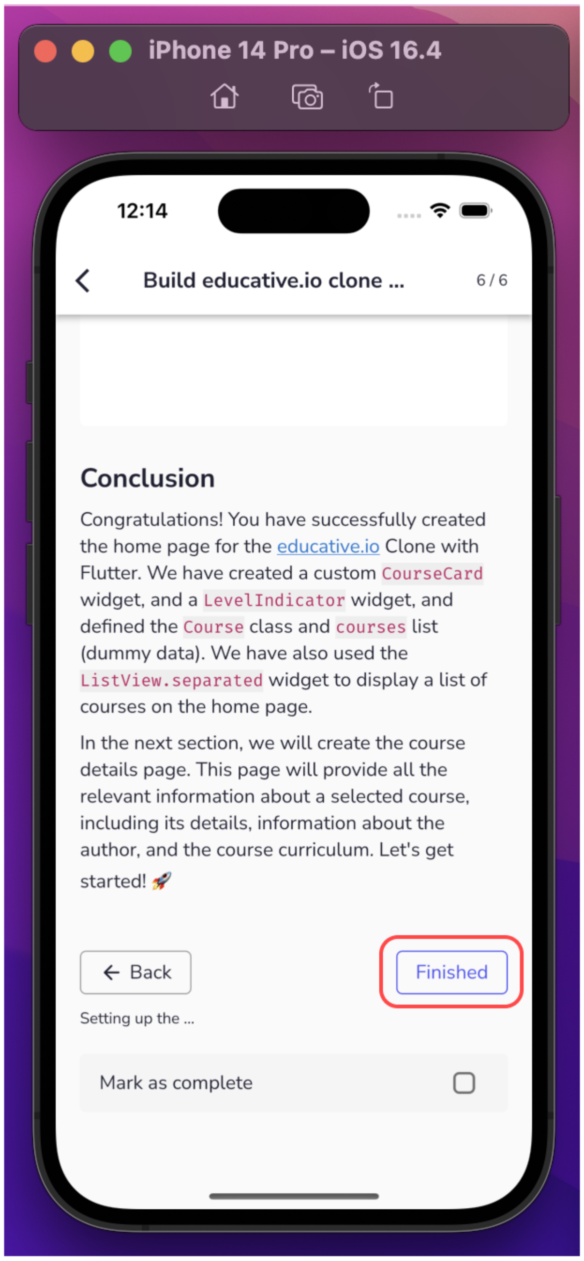
12:14

Build educative.io clone ... 6 / 6

Conclusion

Congratulations! You have successfully created the home page for the educative.io Clone with Flutter. We have created a custom `CourseCard` widget, and a `LevelIndicator` widget, and defined the `Course` class and `courses` list (dummy data). We have also used the `ListView.separated` widget to display a list of courses on the home page.

In the next section, we will create the course details page. This page will provide all the relevant information about a selected course, including its details, information about the author, and the course curriculum. Let's get started! 🚀


iPhone 14 Pro – iOS 16.4

12:14

Build educative.io clone ... 6 / 6

Conclusion

Congratulations! You have successfully created the home page for the educative.io Clone with Flutter. We have created a custom `CourseCard` widget, and a `LevelIndicator` widget, and defined the `Course` class and `courses` list (dummy data). We have also used the `ListView.separated` widget to display a list of courses on the home page.

In the next section, we will create the course details page. This page will provide all the relevant information about a selected course, including its details, information about the author, and the course curriculum. Let's get started! 🚀

Implementation - Step-by-step guide

Step 1: Create the Function of Mark as Complete the Lesson

In this step, we will create a function that allows the user to mark a lesson as complete once they have learned it. So, let's do it!

Inside the `lesson_controller.dart` file, we need to define a new method inside the `LessonController` class. The method will update the `is_completed` field of the table in the Supabase database that we created before.

Add the following code snippet into the `LessonController` class:

```
  // Complete lesson child
  Future<void> completeLessonChild(String lessonChildId,
      {required bool isComplete}) async {
    final response = await supabase
        .from('lesson_child')
        .update({'is_completed': isComplete}).eq('id', lessonChildId);

    print("response: $response");
  }
```

As usual, after we defined the function or method in a controller class we need to define a provider that will handle the state and will we use it in the UI.

So, add the following code into the `lesson_controller.dart` file:

```
final completeLessonChildProvider =
    FutureProvider.autoDispose.family<void, LessonChild>(
  (ref, lessonChild) async {
    final lessonController = LessonController();
    await lessonController.completeLessonChild(lessonChild.id,
        isComplete: lessonChild.isCompleted!);
  },
);
```

Afterward, go to the `lesson_page.dart` file. Inside that file, define the method that will handle the state when the user tapped the `Mark as complete` button on the UI.

Add the following code of method into the `lesson_page.dart` file, specifically it is in the `_LessonPageState` class:

```
 void markAsComplete(LessonChild lesson, bool? v) {
    setState(() {
      lesson.isCompleted = v;
    });

    ref
        .read(completeLessonChildProvider(lesson).future)
        .onError((error, stackTrace) {
      setState(() {
        lesson.isCompleted = false;
      });
    });
  }
```

So far, now we can call the `markAsComplete` method on the `Mark as complete` button widget. Still in the `lesson_page.dart` , update the `onChanged` and the `value` properties of the `CheckboxListTile` widget in the `buildActionButton` method.

The updated code will look something like this:

```
// 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: [
        ...
        CheckboxListTile(
          onChanged: (v) {
            // mark as complete
            markAsComplete(lessons[index], v);    // <--- Add this line
          },
          value: lessons[index].isCompleted,      // <--- Update this line
          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,
          ),
        ),
      ],
    ),
  );
}
```

Okay, if you successfully follow the steps so far, now we can `Mark as complete` a lesson. The result will look like the following video:

https://www.loom.com/share/c1a5a2f982644d00a667ebc4eba084b7?sid=f5de0d83-a96a-42fb-b03e-f6560b3399cf

---

Step 2: Define the Function of Finish the Course

In this step, we will define a function that allows the user to finish the course after marking a lesson as complete once they have learned it. So, let's do it!

Still in the `lesson_page.dart`, add the following code of method into the `lesson_page.dart` file. Specifically, it is in the `_LessonPageState` class.

```
void finishLesson(List<LessonChild> lessons) {
    final isCompletedAll =
        lessons.every((lesson) => lesson.isCompleted == true);

    if (isCompletedAll) {
      // showDialog to confirm finish lesson
      showDialog(
        context: context,
        builder: (context) {
          return AlertDialog(
            title: const Text(
              'Are you sure?',
            ),
            content: const Text(
              'You have completed all the lessons. Are you sure you want to finish this course?',
            ),
            actions: [
              TextButton(
                onPressed: () {
                  Navigator.pop(context);
                },
                child: const Text(
                  'Cancel',
                ),
              ),
              TextButton(
                onPressed: () {
                  Navigator.pushReplacement(
                    context,
                    MaterialPageRoute(
                      builder: (context) => const CourseFinishPage(),
                    ),
                  );
                },
                child: const Text(
                  'Finish',
                ),
              ),
            ],
          );
        },
      );
    } else {
      showDialog(
        context: context,
        builder: (context) {
          return AlertDialog(
            title: const Text(
              'Oops!',
            ),
            content: const Text(
              'You have not completed all the lessons yet. Please complete all the lessons first.',
            ),
            actions: [
              TextButton(
                onPressed: () {
                  Navigator.pop(context);
                },
                child: const Text(
                  'OK',
                ),
              ),
            ],
          );
        },
      );
    }
  }
```

The `finishLesson` method is used to finish a course by checking if all the lessons have been completed. If all the lessons have been completed, it shows an alert dialog to confirm the user's intention to finish the course. If not, it shows an alert dialog to inform the user that they have not completed all the lessons yet and that they need to complete them first. If the user confirms the intention to finish the course, it navigates to the `CourseFinishPage`.

Because the `CourseFinishPage` widget has not been defined before, so let’s define it.

Create a new file called the `course_finish_page.dart` inside the `lib/pages` directory, and add the following code snippet into the `course_finish_page.dart` file:

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

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Spacer(flex: 2),
            const Icon(
              Icons.check_circle_rounded,
              size: 150,
              color: Colors.green,
            ),
            const SizedBox(height: 50),
            Text('Congratulations!', style: MyTypography.title),
            const SizedBox(height: 10),
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 30),
              child: Text(
                'You have completed the course successfully. Now you can continue to the next course.',
                style: MyTypography.body,
                textAlign: TextAlign.center,
              ),
            ),
            const Spacer(),
            SizedBox(
              width: MediaQuery.of(context).size.width * 0.7,
              height: 50,
              child: ElevatedButton(
                onPressed: () {
                  Navigator.pushReplacement(
                    context,
                    MaterialPageRoute(
                      fullscreenDialog: true,
                      builder: (context) => const HomePage(),
                    ),
                  );
                },
                style: ElevatedButton.styleFrom(
                  backgroundColor: MyColors.primary,
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(10),
                  ),
                ),
                child: Text(
                  'OK',
                  style: MyTypography.body.copyWith(
                    color: Colors.white,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
            ),
            const SizedBox(height: 90),
          ],
        ),
      ),
    );
  }
}
```

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

```
import 'course_finish_page.dart';
```

So far, now we can call the `finishLesson` method on the part of the `Finished` button widget. In the `lesson_page.dart` , update the `buildActionButton` method to look something like this:

```
// 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: [
            ...

            Column(
              crossAxisAlignment: CrossAxisAlignment.end,
              children: [
                OutlinedButton(
                  onPressed: () {
                    if (isLastPage) {
                      finishLesson(lessons);     // <--- Add this line
                    } 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,
                  ),
                ),
              ],
            ),
          ],
        ),

        ...
      ],
    ),
  );
}
```

Okay, if you successfully follow the steps so far, now the user can `Finished` a course. The result will look like the following video:

https://www.loom.com/share/7525fcf4e8f743ed99740e65f4e1d05b?sid=d1db0b7b-a86f-46a7-a0a3-0ad31fe57f24

Full code of the `lesson_controller.dart` file:

```
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:supabase_flutter/supabase_flutter.dart';

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

final lessonProvider = FutureProvider.autoDispose.family<List<Lesson>, String>(
  (ref, courseId) async {
    final lessonController = LessonController();
    final lessons = await lessonController.getLessonsByCourseId(courseId);

    return lessons;
  },
);

final lessonChildProvider =
    FutureProvider.autoDispose.family<List<LessonChild>, String>(
  (ref, lessonId) async {
    final lessonController = LessonController();
    final lessonChild =
        await lessonController.getLessonChildByLessonId(lessonId);
    return lessonChild;
  },
);

final completeLessonChildProvider =
    FutureProvider.autoDispose.family<void, LessonChild>(
  (ref, lessonChild) async {
    final lessonController = LessonController();
    await lessonController.completeLessonChild(lessonChild.id,
        isComplete: lessonChild.isCompleted!);
  },
);

class LessonController {
  final supabase = Supabase.instance.client;

  Future<List<Lesson>> getLessonsByCourseId(String courseId) async {
    final response = await supabase
        .from('lesson')
        .select('*, lesson_child!inner(*)')
        .eq('course_id', courseId);

    print("response: $response");

    final lessons = response
        .map((lesson) => Lesson.fromJson(lesson))
        .toList()
        .cast<Lesson>();

    print("lessons: ${lessons.length}");

    return lessons;
  }

  // Get all lesson child by course id
  Future<List<LessonChild>> getLessonChildByLessonId(String courseId) async {
    final response = await supabase
        .from('lesson')
        .select('*, lesson_child!inner(*)')
        .eq('course_id', courseId)
        .order('created_at', ascending: true);

    // debugPrint("response: $response");

    final lessons = response
        .map((lesson) => Lesson.fromJson(lesson))
        .toList()
        .cast<Lesson>();

    final lessonChilds = <LessonChild>[];

    for (var lesson in lessons) {
      lessonChilds.addAll(lesson.lessons);
    }

    print("lessonChild: ${lessonChilds.length}");

    return lessonChilds;
  }
  // Complete lesson child
  Future<void> completeLessonChild(String lessonChildId,
      {required bool isComplete}) async {
    final response = await supabase
        .from('lesson_child')
        .update({'is_completed': isComplete}).eq('id', lessonChildId);

    print("response: $response");
  }
}
```

Full code 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 '../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';
import 'course_finish_page.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) {
                        finishLesson(lessons);
                      } 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) {
              // mark as complete
              markAsComplete(lessons[index], v);
            },
            value: lessons[index].isCompleted,
            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,
            ),
          ),
        ],
      ),
    );
```

```
  }

  void markAsComplete(LessonChild lesson, bool? v) {
    setState(() {
      lesson.isCompleted = v;
    });

    ref
        .read(completeLessonChildProvider(lesson).future)
        .onError((error, stackTrace) {
      setState(() {
        lesson.isCompleted = false;
      });
    });
  }

  void finishLesson(List<LessonChild> lessons) {
    final isCompletedAll =
        lessons.every((lesson) => lesson.isCompleted == true);

    if (isCompletedAll) {
      // showDialog to confirm finish lesson
      showDialog(
        context: context,
        builder: (context) {
          return AlertDialog(
            title: const Text(
              'Are you sure?',
            ),
            content: const Text(
              'You have completed all the lessons. Are you sure you want to finish this course?',
            ),
            actions: [
              TextButton(
                onPressed: () {
                  Navigator.pop(context);
                },
                child: const Text(
                  'Cancel',
                ),
              ),
              TextButton(
                onPressed: () {
                  Navigator.pushReplacement(
                    context,
                    MaterialPageRoute(
                      builder: (context) => const CourseFinishPage(),
                    ),
                  );
                },
                child: const Text(
                  'Finish',
                ),
              ),
            ],
          );
        },
      );
    } else {
      showDialog(
        context: context,
        builder: (context) {
          return AlertDialog(
            title: const Text(
              'Oops!',
            ),
            content: const Text(
              'You have not completed all the lessons yet. Please complete all the lessons first.',
            ),
            actions: [
              TextButton(
                onPressed: () {
                  Navigator.pop(context);
                },
                child: const Text(
                  'OK',
                ),
              ),
            ],
          );
        },
      );
    }
  }
}
```

Conclusion

Overall, creating functionality for finishing a course and marking lessons as complete is an essential aspect of any online learning platform. With the ability to easily track progress and mark lessons as complete, learners can more effectively manage their time and stay motivated to complete their courses. By implementing the features discussed in this section, we can create a more user-friendly and engaging online learning experience.

And also we have covered the final steps required to finish the online learning app, including the ability to mark lessons as complete and finish courses. Congrats! 🔥