aws

Hands-On Tutorial

Build Flutter Mobile App Part One

Build Flutter Mobile App Part One: Hands-On Tutorial

Table of Contents

Build a Flutter Mobile App Using AWS Amplify - Part 1

Create a trip planner app for iOS and Android

AWS Experience	Beginner
Time to Complete	60 minutes
Cost to Complete	Free Tier eligible
Services used	AWS Amplify
Last updated	August 23, 2023

Overview

In this how-to guide, the first part of a two-part series, you will create a cross-platform Flutter mobile app using AWS Amplify. The app is a trip planner where users can create a trip and set its name, destination, and dates. Additionally, they can upload a banner image for the trip.

In the second guide of this series, you will add new features to the app, such as adding activities for the trips using a nested data model and creating the user's profile data using an Amplify function.

What you will accomplish

This how-to guide will walk you through the steps to create an app to help users plan trips. You will:

- Create a Flutter app using the terminal and structure its folders using the Feature-First approach where you will create a folder for every app’s feature.
- Use the Amplify CLI to create the Amplify backend for this app
- Add Amplify authentication capabilities to your app and use the Amplify Authenticator UI library
- Create a data model for the trips and use the GraphQL API to synchronize to the Amplify backend
- Add Amplify storage to your app to enable users to upload an image for their trips

Prerequisites

- An AWS account: If you don't already have an account, follow the Setting Up Your AWS Environment tutorial for a quick overview.
- Install and configure the Amplify CLI.
- Install and configure Flutter.
- A text editor combined with Flutter's command-line tools. For this guide we will use VSCode, but you can use your preferred IDE.

Modules

This how-to guide is divided into the following short modules. You must complete each module before moving to the next one.

1. Module 1: Create a Flutter app (15 minutes): Create a Flutter app, add its dependencies, and create its folder structure. Additionally, you will also define the app colors and the routing constants.
2. Module 2: Add Amplify authentication (10 minutes): Create the Amplify backend for the app and implement authentication using the Amplify authenticator UI library.
3. Module 3: Add API (20 minutes): Create the Amplify GraphQL API for the app and implement the CRUD operation for the trip feature.
4. Module 4: Add Amplify storage (15 minutes): Create the Amazon S3 bucket for the app and implement the ability to upload an image for the trip.

Module 1: Create a Flutter app

Overview

In this module, we will start by creating a new Flutter mobile app. Then we will add the Amplify packages and other dependencies to the app. Next, we will structure the app's folders based on the Feature-First approach. Then, we will create the constants the app will use to define its routes and colors. And finally, we will use the Amplify CLI to provision a cloud backend for the app

What you will accomplish

- Create a Flutter app
- Add the app dependencies
- Create the app folder structure
- Define the constants for the app's routes and colors
- Create an Amplify backend for the app

Implementation

Minimum time to complete	15 minutes

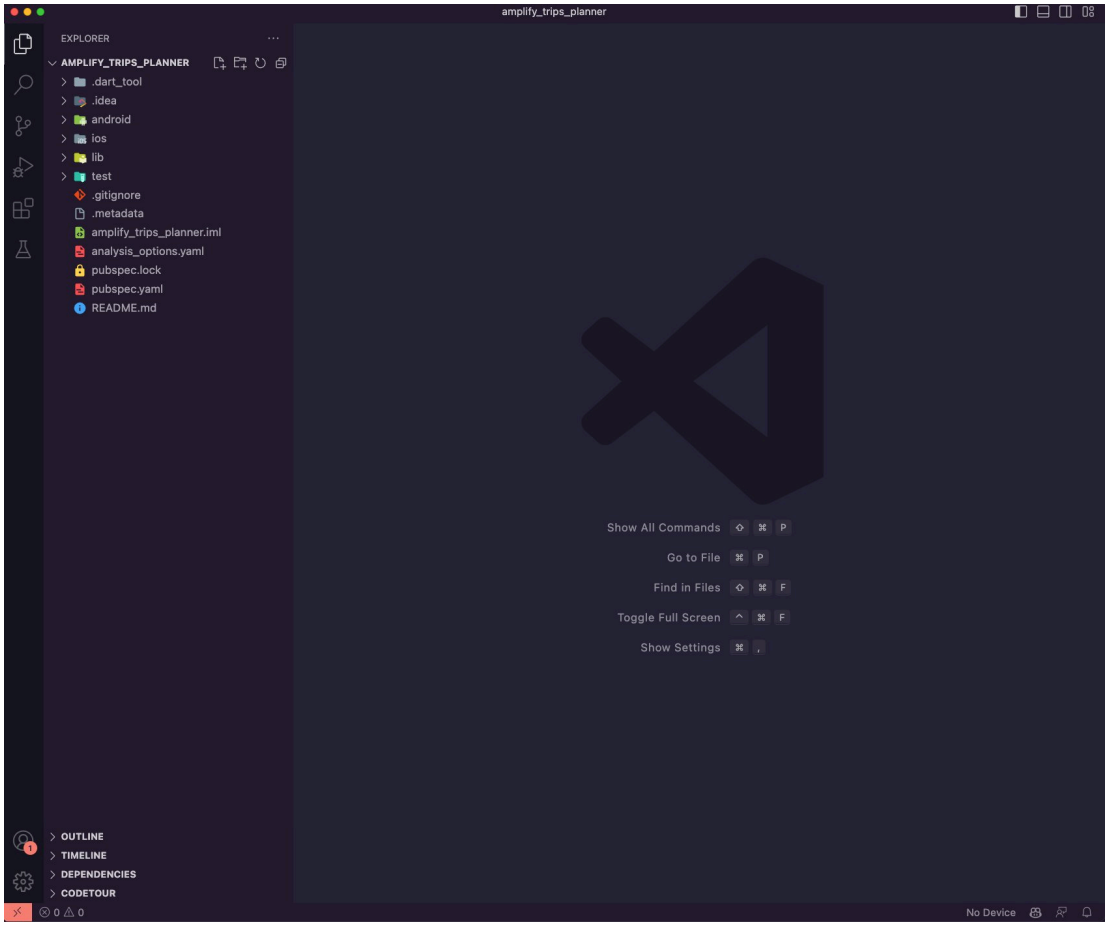
Step 1: Create a new Flutter app

1. Create a Flutter app by running the command below in your terminal.

```
flutter create amplify_trips_planner --platforms=ios,android
```

2. Open the newly created Flutter application using VSCode. You can do that by running the commands below in your terminal.

```
cd amplify_trips_planner
code . -r
```



Step 2: Add the app dependencies

1. Update the file pubspec.yaml in the application root directory to add the following Amplify packages.

```
amplify_api: ^1.0.0
amplify_auth_cognito: ^1.0.0
amplify_authenticator: ^1.0.0
amplify_flutter: ^1.0.0
amplify_storage_s3: ^1.0.0
```

2. The app will use additional dependencies for its functionality, such as State Management, Image Picker, and Navigation. Update the file pubspec.yaml in the application root directory to add the following packages as dependencies.

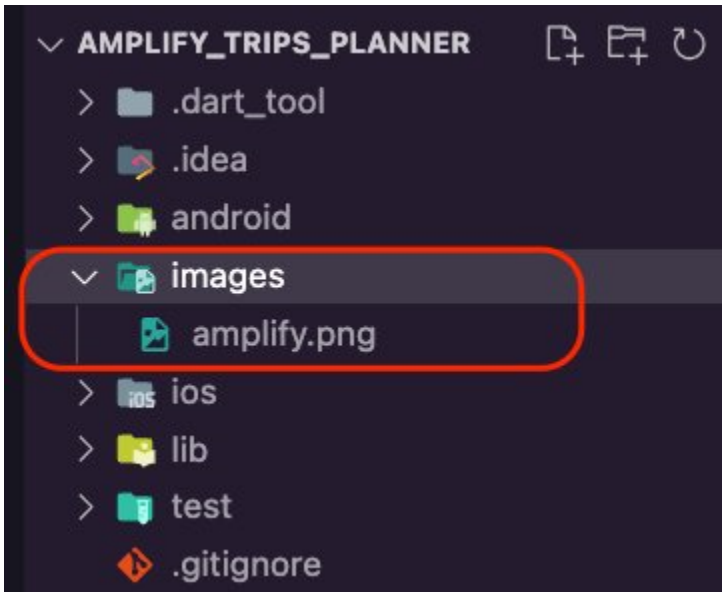
```
cached_network_image: ^3.2.3
cupertino_icons: ^1.0.5
```

```
flutter_riverpod: ^2.1.3
go_router: ^7.0.0
image_picker: ^0.8.0
intl: ^0.18.0
path: ^1.8.3
riverpod_annotation: ^2.0.1
uuid: ^3.0.7
```

3. To improve the coding experience, we will use a set of development dependencies, such as code lint packages, which encourage good coding practices, and code generation packages to simplify the State Management implementation. Update the file pubspec.yaml in the application root directory to add the following packages as dev_dependencies.

```
amplify_lints: ^3.0.0
build_runner:
custom_lint:
flutter_test:
  sdk: flutter
riverpod_generator: ^2.1.3
riverpod_lint: ^1.1.5
```

4. The app will use a placeholder image (amplify.png) when creating a new trip. After downloading the image, create a new folder in the app root directory, name it **images**, and then add the image to it.



5. Update the file pubspec.yaml to add it to the assets section.

```
flutter:
```

```
 uses-material-design: true
 assets:
   - images/amplify.png
```

The pubspec.yaml file should look like the this:

```
name: amplify_trips_planner
description: A new Flutter project.
version: 1.0.0+1

environment:
 sdk: ">=3.0.2 <4.0.0"

dependencies:
 amplify_api: ^1.0.0
 amplify_auth_cognito: ^1.0.0
 amplify_authenticator: ^1.0.0
 amplify_flutter: ^1.0.0
 amplify_storage_s3: ^1.0.0
 cached_network_image: ^3.2.3
 cupertino_icons: ^1.0.5
 flutter:
   sdk: flutter
 flutter_riverpod: ^2.1.3
 go_router: ^7.0.0
 image_picker: ^0.8.0
 intl: ^0.18.0
 path: ^1.8.3
 riverpod_annotation: ^2.0.1
 uuid: ^3.0.7

dev_dependencies:
 amplify_lints: ^3.0.0
 build_runner:
 custom_lint:
 flutter_test:
   sdk: flutter
 riverpod_generator: ^2.1.3
 riverpod_lint: ^1.1.5

flutter:
 uses-material-design: true
 assets:
```

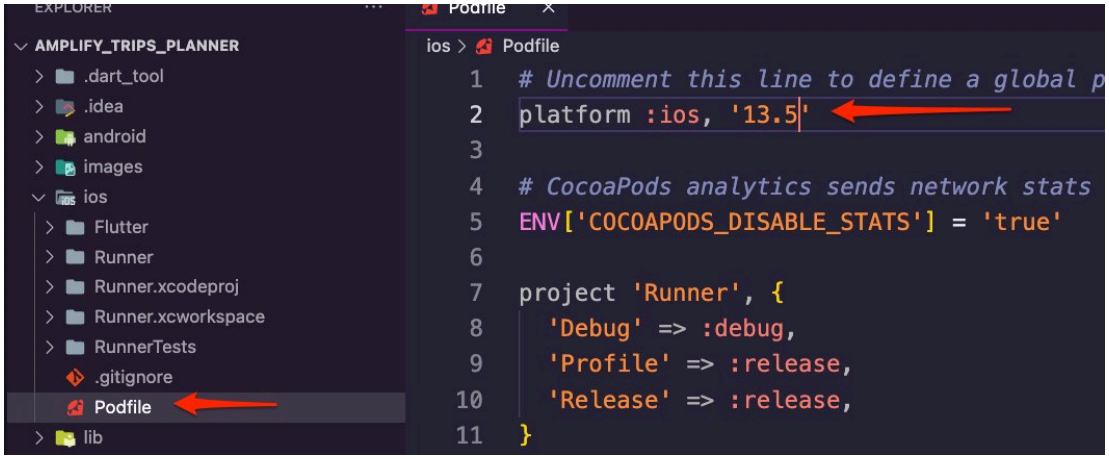
```
    - images/amplify.png
```

6. Run the following command in your terminal to install the dependencies you added to the pubspec.yaml file.

```
flutter pub get
```

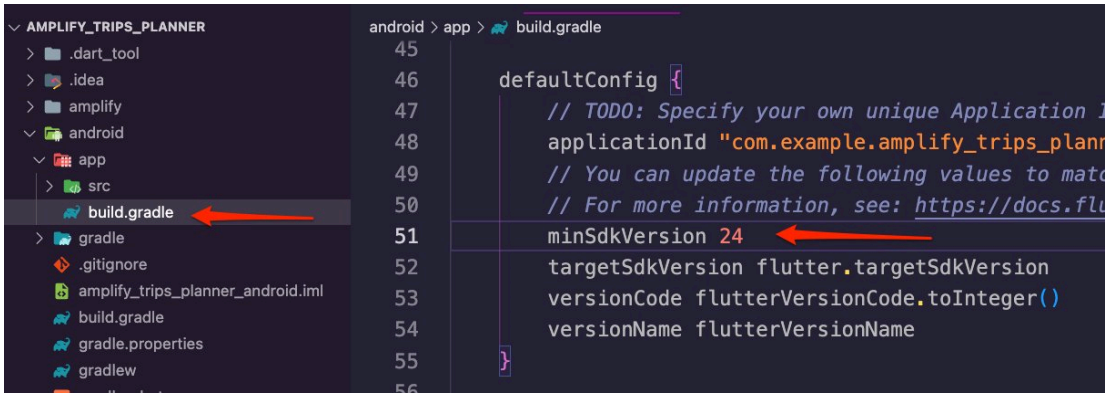
7. Update the target iOS platform by navigating to the **ios/** folder and change the Podfile to target iOS platform 13.5 or higher.

```
platform :ios, '13.5'
```



8. Update the target Android SDK by navigating to the **android/app/** folder and modifying the **build.gradle** file to update the target Android SDK version to 24 or higher.

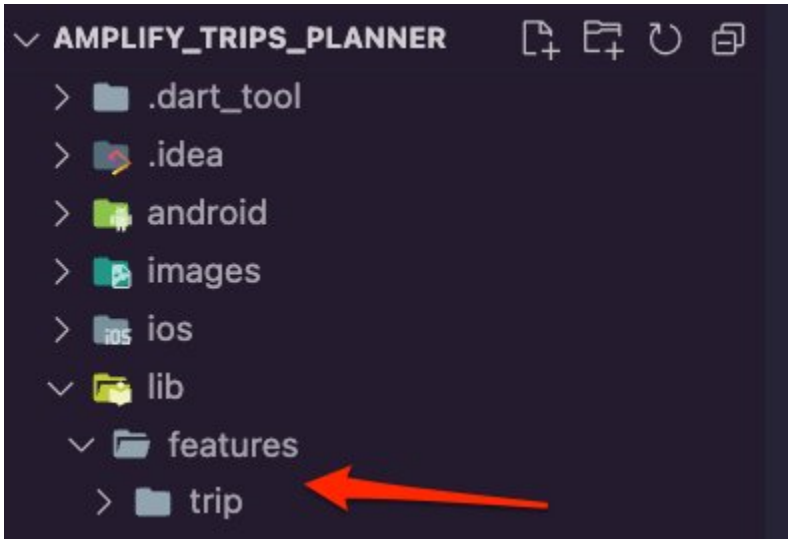
```
minSdkVersion 24
```



Step 3: Create the app folder structure

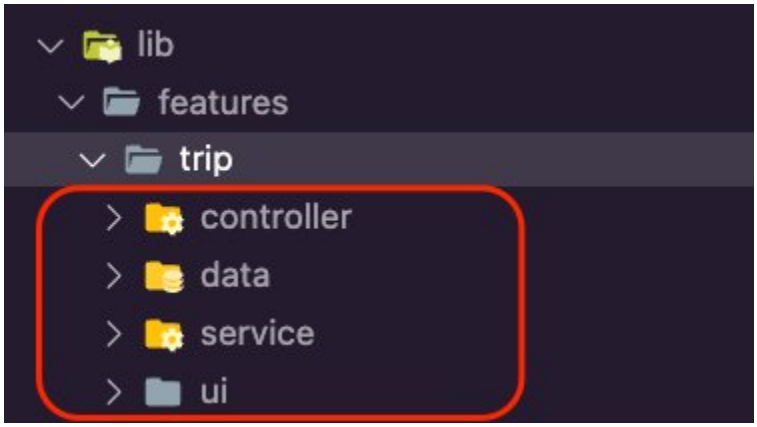
This guide will use the Feature-First approach to structure the app. We will create a folder for every app feature, and inside that folder, we can add the layers as sub-folders.

1. Create a new folder inside the **lib** folder, name it **features,** create a folder inside it, and name it **trip.**

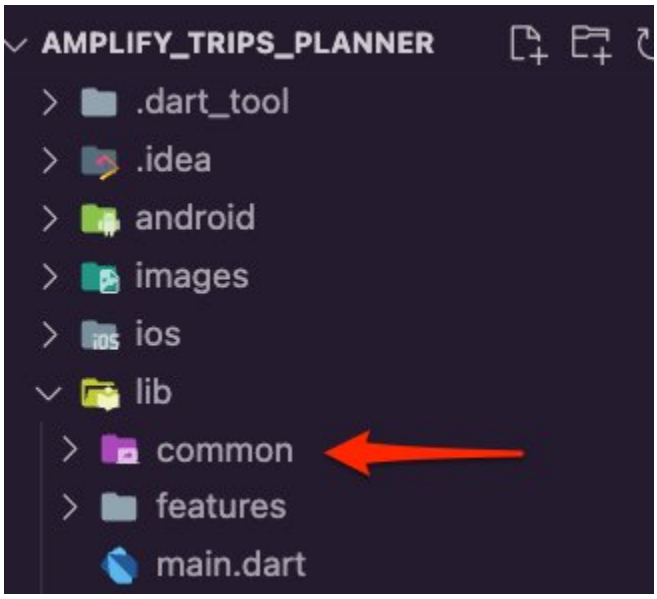


2. Create the following new folders inside the **trip** folder:

   - **service:** The layer to connect with the Amplify backend.
   - **data:** This will be the repository layer that abstracts away the networking code, specifically **service.**
   - **controller:** This is an abstract layer to connect the UI with the repository.
   - **ui:** Here, we will create the widgets and the pages that the app will present to the user.

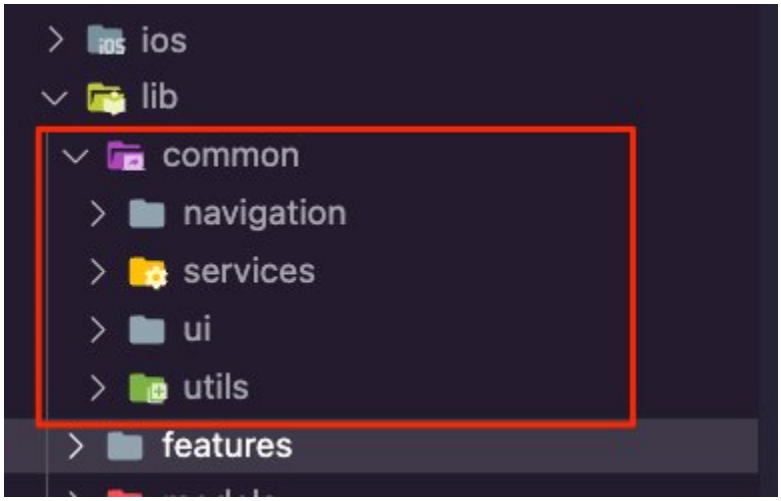


3. Create a new folder inside the **lib** folder and name it **common.** We will use this folder for the code shared across the app features.



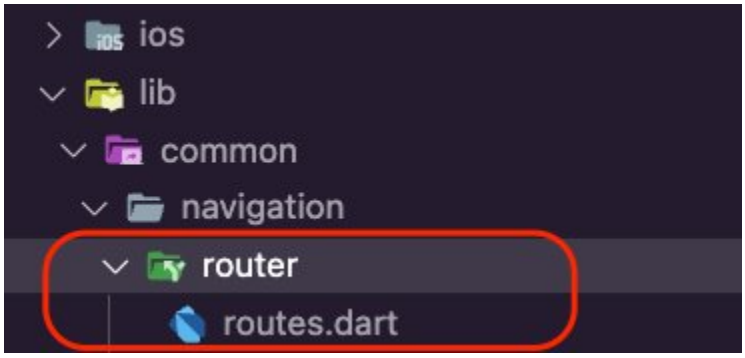
4. Create the following new folders inside the **common** folder:

   - **ui:** Here, we will create the widgets and the pages that the app will present to the user.
   - **navigation:** We will use this folder to define the app routes.
   - **services:** This is where we will create the services to connect with the Amplify backend.
   - **utils:** This folder will contain constants, such as colors, that the app will frequently use.



Step 4: Define the constants for the routes and colors

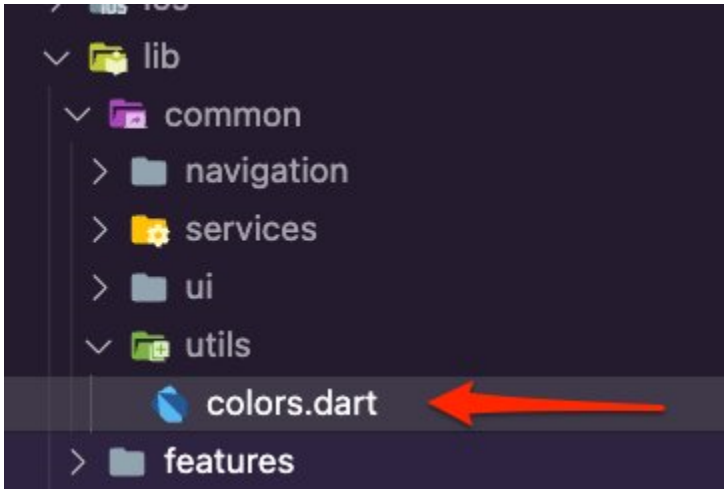
1. Create a folder inside the **lib/common/navigation** folder and call it **router,** and then create a new dart file inside it and call it **routes.dart.**



2. Open the **routes.dart** file and update it with the following code. We will use this enum to create the routing of the app.

```
enum AppRoute {
 home,
 trip,
 editTrip,
}
```

3. Create a new dart file inside the **lib/common/utils** folder and name it **colors.dart.**



4. Open the **colors.dart** file and update it with the following code to create constant variables to set the app's colors.

```
import 'package:flutter/material.dart';
const Map<int, Color> primarySwatch = {
```

```
 50: Color.fromRGBO(255, 207, 68, .1),
 100: Color.fromRGBO(255, 207, 68, .2),
 200: Color.fromRGBO(255, 207, 68, .3),
 300: Color.fromRGBO(255, 207, 68, .4),
 400: Color.fromRGBO(255, 207, 68, .5),
 500: Color.fromRGBO(255, 207, 68, .6),
 600: Color.fromRGBO(255, 207, 68, .7),
 700: Color.fromRGBO(255, 207, 68, .8),
 800: Color.fromRGBO(255, 207, 68, .9),
 900: Color.fromRGBO(255, 207, 68, 1),
};
const MaterialColor primaryColor = MaterialColor(0xFFFFCF44, primarySwatch);
const int primaryColorDark = 0xFFFD9725;
```

Step 5: Create an Amplify backend for the app

1. Navigate to the app's root folder, and provision the Amplify backend for the app by running the following command in your terminal.

```
amplify init
```

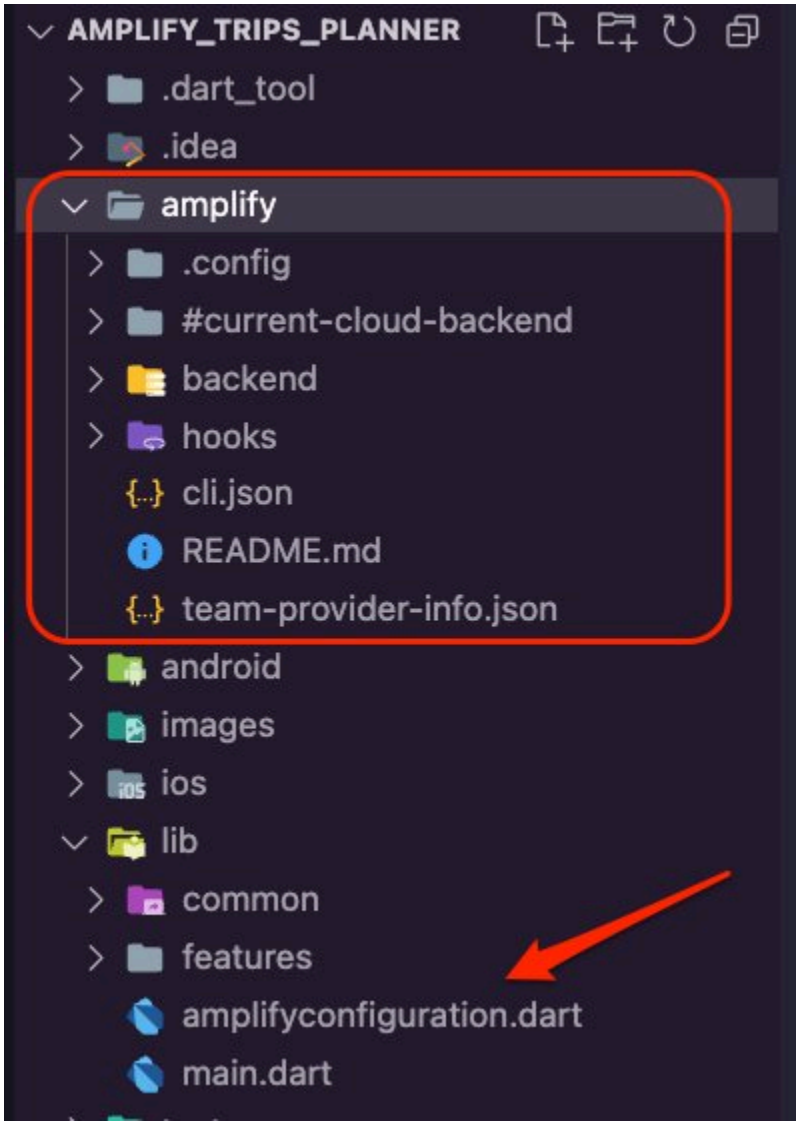
2. Enter a name for the Amplify project or accept the suggested name, then press **Enter**.

```
? Enter a name for the project amplifytripsplanner
The following configuration will be applied:

Project information
| Name: amplifytripsplanner
| Environment: dev
| Default editor: Visual Studio Code
| App type: flutter
| Configuration file location: ./lib/

? Initialize the project with the above configuration? (Y/n)
```

3. Press **Enter** again to accept the auto-generated options and select the AWS Authentication method. In this example, we are using an AWS profile. Amplify CLI will initialize the backend and connect the project to the cloud.

```
? Initialize the project with the above configuration? Yes
Using default provider  awscloudformation
? Select the authentication method you want to use: AWS profile

For more information on AWS Profiles, see:
https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html

? Please choose the profile you want to use AwsWest1
Adding backend environment dev to AWS Amplify app: dorapq24trw9r

Deployment completed.
Deploying root stack amplifytripsplanner [ ====================------------
        amplify-amplifytripsplanner-d… AWS::CloudFormation::Stack     CREATE_IN_PR
        UnauthRole                     AWS::IAM::Role                 CREATE_COMPL
        AuthRole                       AWS::IAM::Role                 CREATE_COMPL
        DeploymentBucket               AWS::S3::Bucket                CREATE_IN_PR

✔ Help improve Amplify CLI by sharing non sensitive configurations on failures (y/N)
· no

Deployment state saved successfully.
✔ Initialized provider successfully.
✅  Initialized your environment successfully.

Your project has been successfully initialized and connected to the cloud!

Some next steps:
"amplify status" will show you what you've added already and if it's locally configur
ed or deployed
"amplify add <category>" will allow you to add features like user login or a backend
API
"amplify push" will build all your local backend resources and provision it in the cl
oud
"amplify console" to open the Amplify Console and view your project status
"amplify publish" will build all your local backend and frontend resources (if you ha
ve hosting category added) and provision it in the cloud

Pro tip:
Try "amplify add api" to create a backend API and then "amplify push" to deploy every
thing
```

The Amplify CLI will add a new folder named **amplify** to the app's root folder, which contains the amplify project and backend details. It will also add a new dart file **(amplifyconfiguration.dart)** to the **lib/** folder. The app will use this file to know how to reach your provisioned backend resources at runtime.



Conclusion

In this module, you learned how to create a Flutter app. You also learned how to set its folder structure using the Feature-First approach. Additionally, you created constant variables for the app to use to set its routes and colors. Finally, you used the Amplify CLI to provision a cloud backend for the app.

Module 2: Add Amplify authentication

Overview

In this module, you will learn how to authenticate a user with the Amplify auth category by using Amazon Cognito, a managed user identity service.

You will also learn how to use the Amplify Authenticator library to quickly create an authentication flow for the app, allowing users to sign up, sign in, and reset their password with just a few lines of code.

What you will accomplish

- Add Amplify authentication category to the app
- Implement the authentication flow using Amplify Authenticator

Implementation

Step 1: Add Amplify authentication category to the app

1. Navigate to the app's root folder and run the following command in your terminal.

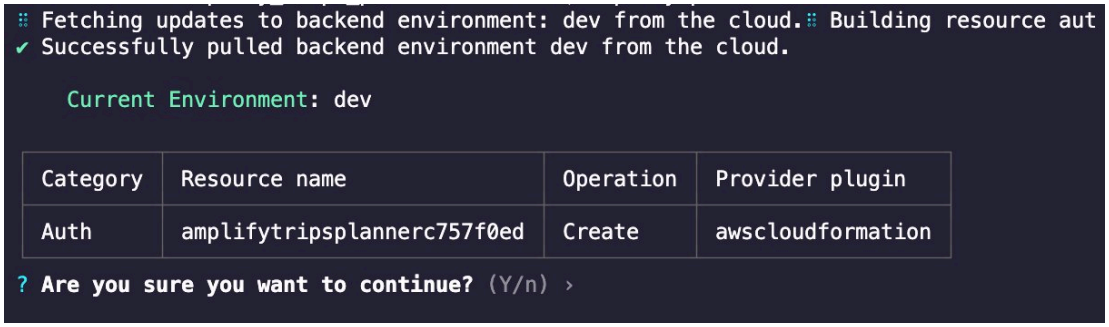
```
amplify add auth
```

2. Select the **Default configuration** option and then select **Email** as the sign-in method. Then select the **No, I am done** option.

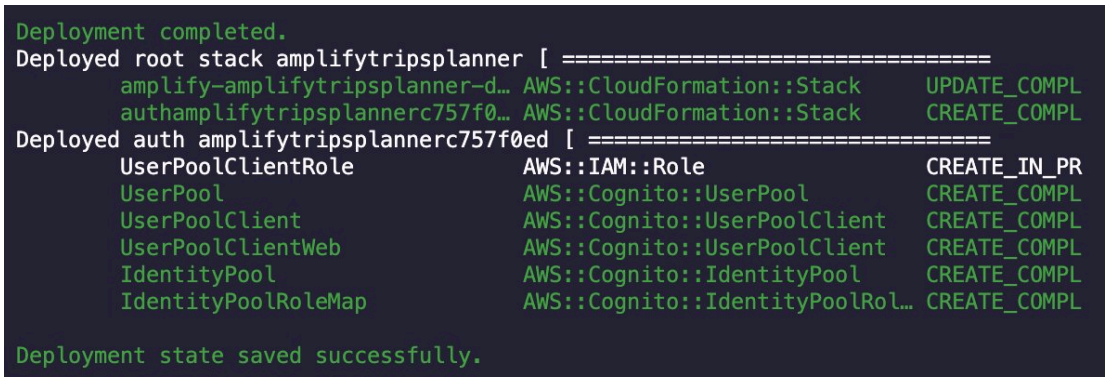
```
Using service: Cognito, provided by: awscloudformation The current configured
 provider is Amazon Cognito. Do you want to use the default authentication and
 security configuration? Default configuration Warning: you will not be able to
 edit these selections. How do you want users to be able to sign in? Email Do you
 want to configure advanced settings? No, I am done.
# Successfully added auth resource amplifytripsplanner035b18d6 locally # Some next
 steps:
"amplify push" will build all your local backend resources and provision it in the
 cloud
```

```
"amplify publish" will build all your local backend and frontend resources (if you
 have hosting category added) and provision it in the cloud
```

3. Now the Amplify Auth category is ready. Run the command **amplify push** to create the resources in the cloud.



4. Press **Enter**. The Amplify CLI will deploy the authentication resources and display a confirmation, as shown in the screenshot.

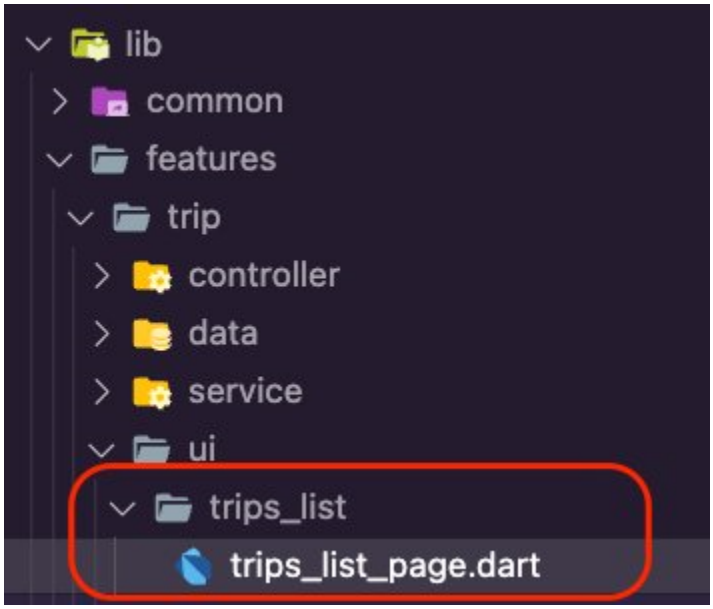


Step 2: Implement the app authentication flow using Amplify Authenticator

Minimum time to complete

10 minutes

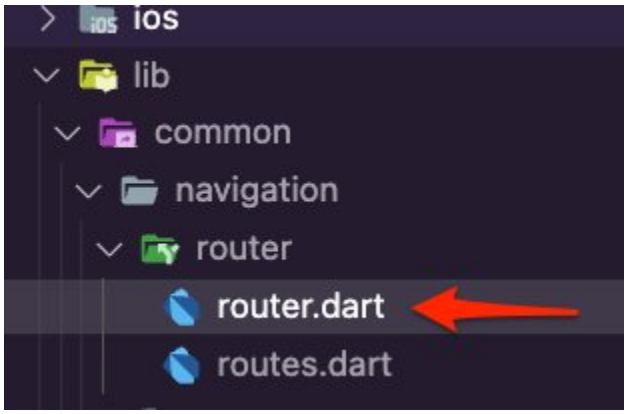
1. Create a new folder inside the **lib/features/trip/ui** folder, name it **trips_list**, and then create the file **trips_list_page.dart** inside it.



2. Open the **trips_list_page.dart** file and update it with the following code. This will be the app's homepage.

```
import 'package:flutter/material.dart';
import 'package:amplify_trips_planner/common/utils/colors.dart' as constants; class
 TripsListPage extends StatelessWidget { const TripsListPage({ super.key, });
 @override Widget build(BuildContext context) { return Scaffold( appBar:
 AppBar( centerTitle: true, title: const Text( 'Amplify Trips Planner', ),
 backgroundColor: const Color(constants.primaryColorDark), ), floatingActionButton:
 FloatingActionButton( onPressed: () {}, backgroundColor: const
 Color(constants.primaryColorDark), child: const Icon(Icons.add), ), body: const
 Center( child: Text('Trips List'), ), ); }
}
```

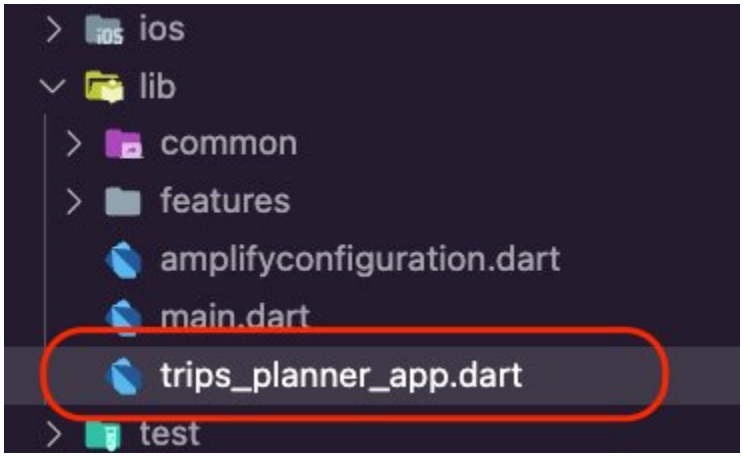
3. Create a new file inside the **lib/common/navigation/router** folder and call it **router.dart**.



4. Open the **router.dart** file and update it with the following code to create the routes for the app.

```
import 'package:amplify_trips_planner/common/navigation/router/routes.dart';
import 'package:amplify_trips_planner/features/trip/ui/trips_list/
trips_list_page.dart';
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart'; final router = GoRouter( routes:
 [ GoRoute( path: '/', name: AppRoute.home.name, builder: (context, state) =>
 const TripsListPage(), ), ], errorBuilder: (context, state) => Scaffold( body:
 Center( child: Text(state.error.toString()), ), ),
);
```

5. Create a new dart file inside the **lib** folder and call it **trips_planner_app.dart**.



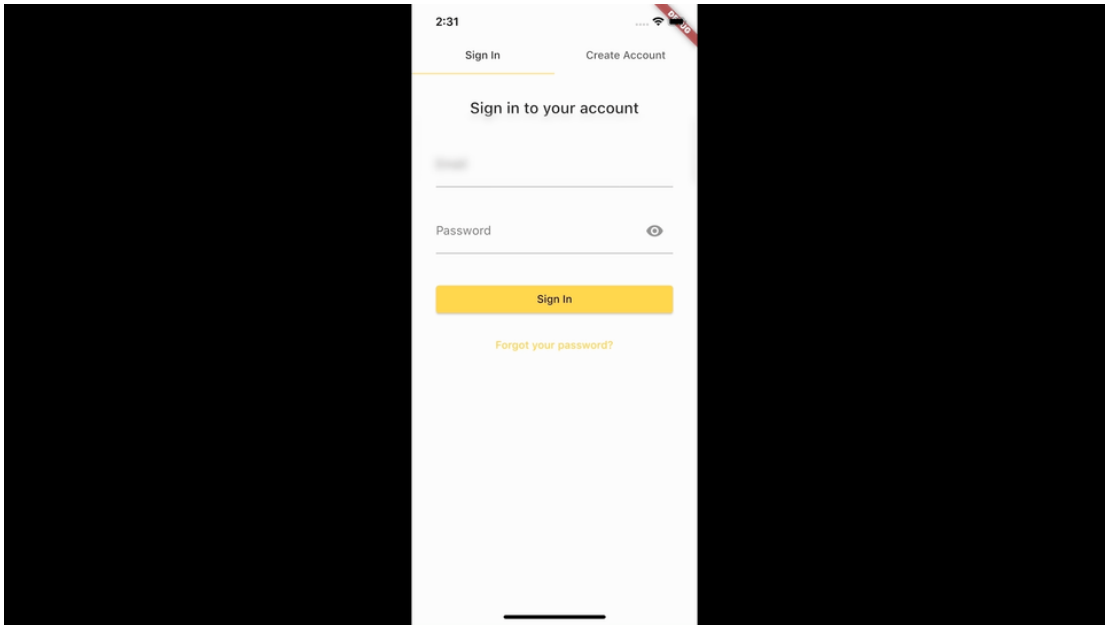
6. Open the **trips_planner_app.dart** file and update it with the following code to wrap the **MaterialApp** in an Amplify Authenticator widget.

```
import 'package:amplify_authenticator/amplify_authenticator.dart';
import 'package:amplify_trips_planner/common/navigation/router/router.dart';
import 'package:amplify_trips_planner/common/utils/colors.dart' as constants;
import 'package:flutter/material.dart'; class TripsPlannerApp extends
 StatelessWidget { const TripsPlannerApp({ super.key, }); @override
 Widget build(BuildContext context) { return Authenticator( child:
 MaterialApp.router( routerConfig: router, builder: Authenticator.builder(),
 theme: ThemeData( colorScheme: ColorScheme.fromSwatch(primarySwatch:
 constants.primaryColor) .copyWith( background: const
 Color(0xff82CFEA), ), ), ), ); }
}
```

7. Open the **main.dart** file and update it with the following code. This contains the logic of configuring Amplify for the Auth category.

```
import 'package:amplify_auth_cognito/amplify_auth_cognito.dart';
import 'package:amplify_flutter/amplify_flutter.dart';
import 'package:amplify_trips_planner/trips_planner_app.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
 import 'amplifyconfiguration.dart'; Future<void> main() async
 { WidgetsFlutterBinding.ensureInitialized(); try { await _configureAmplify(); } on
 AmplifyAlreadyConfiguredException { debugPrint('Amplify configuration failed.'); }
 runApp( const ProviderScope( child: TripsPlannerApp(), ), );
} Future<void> _configureAmplify() async { await
 Amplify.addPlugins([ AmplifyAuthCognito(), ]); await
 Amplify.configure(amplifyconfig);
}
```

8. Run the app in the simulator and use the authentication flow to create a user. The following is an example using an iPhone simulator.



Conclusion

In this module, you implemented a user authentication flow in your app with just a few lines of code. In the next module, we will add an API to your app.

Module 3: Add API

Overview

In this module, you will add an API to your app using the Amplify CLI to retrieve and persist your trip data. The API you will create is a GraphQL API that uses AWS AppSync (a managed GraphQL service) backed by DynamoDB (a NoSQL database).

What you will accomplish

- Add Amplify API to the app
- Add the trip data model to the app
- Implement the CRUD operations and flow for the trip feature
- Implement the trips listing UI

Implementation

Minimum time to complete	20 minutes

Step 1: Add Amplify API to the app

1. Navigate to the root folder of the app and provision an Amplify API resource by running the following command in your terminal.

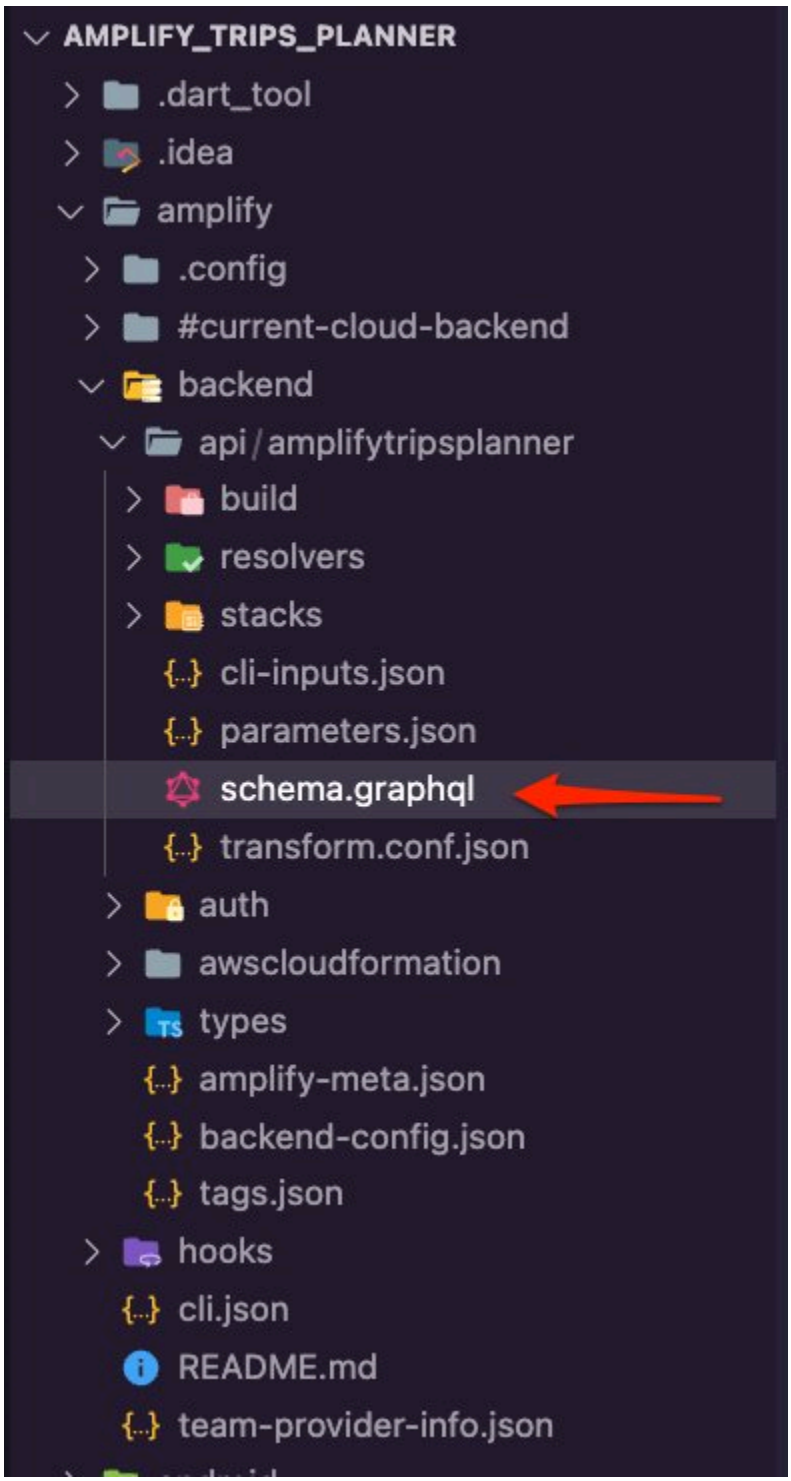
```
amplify add api
```

2. To create the GraphQL API, enter the following when prompted:

```
? Select from one of the below mentioned services: GraphQL
? Here is the GraphQL API that we will create. Select a setting to edit or continue
 A
uthorization modes: API key (default, expiration time: 7 days from now)
? Choose the default authorization type for the API Amazon Cognito User Pool
Use a Cognito user pool configured as a part of this project.
? Configure additional auth types? No
```

```
? Here is the GraphQL API that we will create. Select a setting to edit or continue
 C
ontinue
? Choose a schema template: Blank Schema
# GraphQL schema compiled successfully.
```

The Amplify CLI will add a new folder for the API, including the **schema.graphql** file where you will define the models for the app.



3. Open the **schema.graphql** file and update it with the following to define the trip model.

```
type Trip @model @auth(rules: [{ allow: owner }]) {
 id: ID!
 tripName: String!
 destination: String!
```

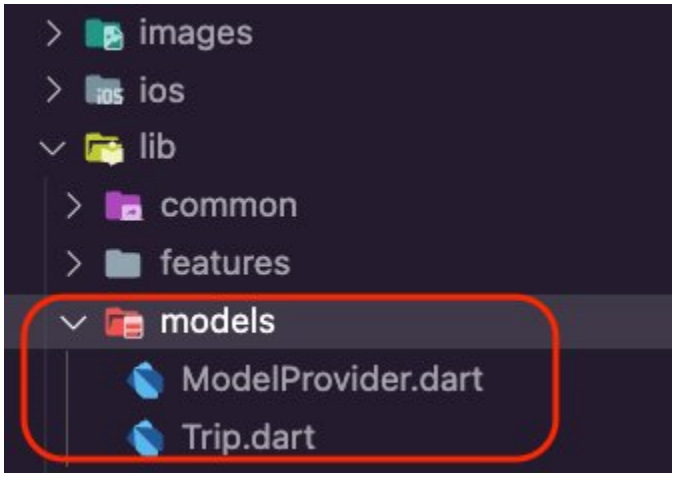
```
 startDate: AWSDate!
 endDate: AWSDate!
 tripImageUrl: String
 tripImageKey: String
}
```

Step 2: Add trip data model to the app

1. Run the following command in the root folder of the app to generate the trip model file.

   ```
   amplify codegen models
   ```

   The Amplify CLI will generate the dart files in the **lib/models** folder.

2. Run the command **amplify push** to create the resources in the cloud.

   ```
   Current Environment: dev

| Category | Resource name | Operation | Provider plugin |
| --- | --- | --- | --- |
| Api | amplifytripsplanner | Create | awscloudformation |
| Auth | amplifytripsplannerc757f0ed | No Change | awscloudformation |

? Are you sure you want to continue? (Y/n) ›
   ```

3. Press **Enter**. The Amplify CLI will deploy the resources and display a confirmation, as shown in the screenshot.

```
        authamplifytripsplannerc757f0… AWS::CloudFormation::Stack     UPDATE_COMPL
        apiamplifytripsplanner         AWS::CloudFormation::Stack     CREATE_COMPL
Deployed api amplifytripsplanner [ ========================================
        GraphQLAPI                     AWS::AppSync::GraphQLApi       CREATE_COMPL
        GraphQLAPINONEDS95A13CF0       AWS::AppSync::DataSource       CREATE_COMPL
        GraphQLAPITransformerSchema3C… AWS::AppSync::GraphQLSchema    CREATE_COMPL
        Trip                           AWS::CloudFormation::Stack     CREATE_IN_PR

Deployment state saved successfully.

GraphQL endpoint: https://idixlcamijfkvfzpt4d3fycnxa.appsync-api.us-west-1.amazonaws.
com/graphql

GraphQL transformer version: 2
```

4. Open **main.dart** file and update the **_configureAmplify()** function as shown in the code to add the Amplify API plugin.

```
Future<void> _configureAmplify() async {
 await Amplify.addPlugins([
   AmplifyAuthCognito(),
   AmplifyAPI(modelProvider: ModelProvider.instance),
 ]);
 await Amplify.configure(amplifyconfig);
}
```

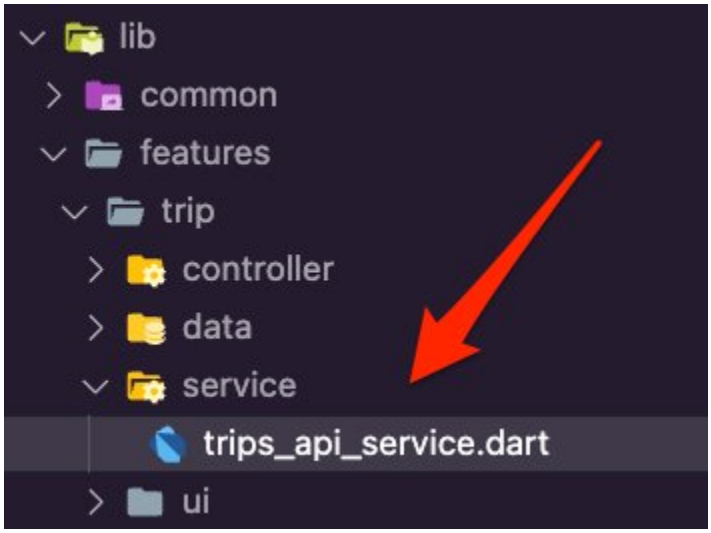
The **main.dart** file should now look like the following code snippet.

```
import 'package:amplify_api/amplify_api.dart';
import 'package:amplify_auth_cognito/amplify_auth_cognito.dart';
import 'package:amplify_flutter/amplify_flutter.dart';
import 'package:amplify_trips_planner/models/ModelProvider.dart';
import 'package:amplify_trips_planner/trips_planner_app.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

import 'amplifyconfiguration.dart';

Future<void> main() async {
 WidgetsFlutterBinding.ensureInitialized();
 try {
   await _configureAmplify();
 } on AmplifyAlreadyConfiguredException {
   debugPrint('Amplify configuration failed.');
 }

 runApp(
```

```
    const ProviderScope(
      child: TripsPlannerApp(),
    ),
  );
}

Future<void> _configureAmplify() async {
  await Amplify.addPlugins([
    AmplifyAuthCognito(),
    AmplifyAPI(modelProvider: ModelProvider.instance),
  ]);
  await Amplify.configure(amplifyconfig);
}
```

Step 3: Implement the CRUD operations and flow for trip feature

1. Create a new dart file inside the **lib/features/trip/service** folder and call it **trips_api_service.dart.**



2. Open the **trips_api_service.dart** file and update it with the following code snippet to create the **TripsAPIService,** which contains the following functions:

   - **getTrips** - This function will query the Amplify API for the **active and upcoming** trips and return a list of them.
   - **getPastTrips** - This function will query the Amplify API for **past** trips and return a list of them.
   - **getTrip** - This function will query the Amplify API for a specific trip.

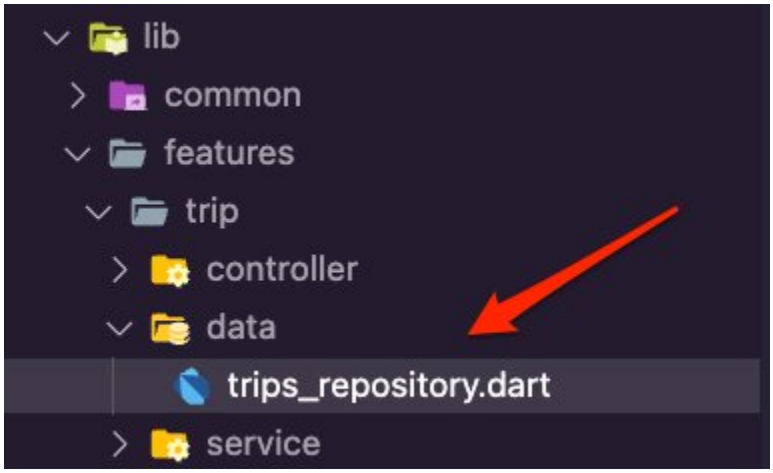
- **addTrip** , **deleteTrip** , and **updateTrip** is for adding, deleting, or updating the trips in the Amplify API.

```
import 'dart:async';

import 'package:amplify_api/amplify_api.dart';
import 'package:amplify_flutter/amplify_flutter.dart';
import 'package:amplify_trips_planner/models/ModelProvider.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

final tripsAPIServiceProvider = Provider<TripsAPIService>((ref) {
  final service = TripsAPIService();
  return service;
});

class TripsAPIService {
  TripsAPIService();

  Future<List<Trip>> getTrips() async {
    try {
      final request = ModelQueries.list(Trip.classType);
      final response = await Amplify.API.query(request: request).response;

      final trips = response.data?.items;
      if (trips == null) {
        safePrint('getTrips errors: ${response.errors}');
        return const [];
      }
      trips.sort(
        (a, b) =>
            a!.startDate.getDateTime().compareTo(b!.startDate.getDateTime()),
      );
      return trips
          .map((e) => e as Trip)
          .where(
            (element) => element.endDate.getDateTime().isAfter(DateTime.now()),
          )
          .toList();
    } on Exception catch (error) {
      safePrint('getTrips failed: $error');

      return const [];
```

```
    }
  }

  Future<List<Trip>> getPastTrips() async {
    try {
      final request = ModelQueries.list(Trip.classType);
      final response = await Amplify.API.query(request: request).response;

      final trips = response.data?.items;
      if (trips == null) {
        safePrint('getPastTrips errors: ${response.errors}');
        return const [];
      }
      trips.sort(
        (a, b) =>
            a!.startDate.getDateTime().compareTo(b!.startDate.getDateTime()),
      );
      return trips
          .map((e) => e as Trip)
          .where(
            (element) => element.endDate.getDateTime().isBefore(DateTime.now()),
          )
          .toList();
    } on Exception catch (error) {
      safePrint('getPastTrips failed: $error');

      return const [];
    }
  }

  Future<void> addTrip(Trip trip) async {
    try {
      final request = ModelMutations.create(trip);
      final response = await Amplify.API.mutate(request: request).response;

      final createdTrip = response.data;
      if (createdTrip == null) {
        safePrint('addTrip errors: ${response.errors}');
        return;
      }
    } on Exception catch (error) {
      safePrint('addTrip failed: $error');
    }
  }
```

```
  Future<void> deleteTrip(Trip trip) async {
    try {
      await Amplify.API
          .mutate(
            request: ModelMutations.delete(trip),
          )
          .response;
    } on Exception catch (error) {
      safePrint('deleteTrip failed: $error');
    }
  }

  Future<void> updateTrip(Trip updatedTrip) async {
    try {
      await Amplify.API
          .mutate(
            request: ModelMutations.update(updatedTrip),
          )
          .response;
    } on Exception catch (error) {
      safePrint('updateTrip failed: $error');
    }
  }

  Future<Trip> getTrip(String tripId) async {
    try {
      final request = ModelQueries.get(
        Trip.classType,
        TripModelIdentifier(id: tripId),
      );
      final response = await Amplify.API.query(request: request).response;

      final trip = response.data!;
      return trip;
    } on Exception catch (error) {
      safePrint('getTrip failed: $error');
      rethrow;
    }
  }
}
```

3. Create a new dart file inside the **lib/features/trip/data** folder and call it **trips_repository.dart.**

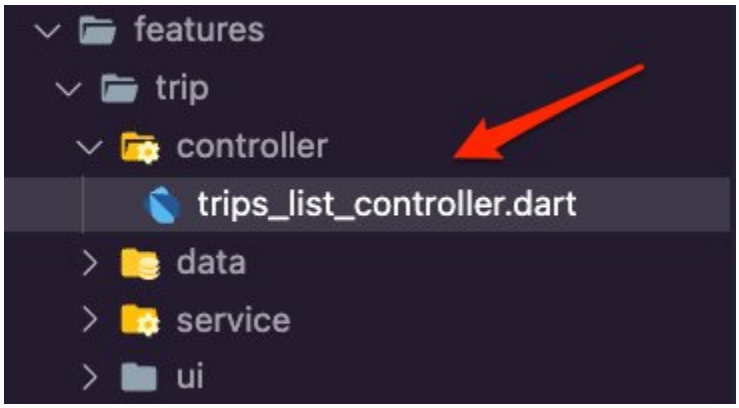


4. Open the **trips_repository.dart** file and update it with the following code:

```
import 'package:amplify_trips_planner/features/trip/service/
trips_api_service.dart';
import 'package:amplify_trips_planner/models/Trip.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

final tripsRepositoryProvider = Provider<TripsRepository>((ref) {
 final tripsAPIService = ref.read(tripsAPIServiceProvider);
 return TripsRepository(tripsAPIService);
});

class TripsRepository {
 TripsRepository(this.tripsAPIService);

 final TripsAPIService tripsAPIService;

 Future<List<Trip>> getTrips() {
   return tripsAPIService.getTrips();
 }

 Future<List<Trip>> getPastTrips() {
   return tripsAPIService.getPastTrips();
 }

 Future<void> add(Trip trip) async {
   return tripsAPIService.addTrip(trip);
 }

 Future<void> update(Trip updatedTrip) async {
   return tripsAPIService.updateTrip(updatedTrip);
```

```
 }

 Future<void> delete(Trip deletedTrip) async {
   return tripsAPIService.deleteTrip(deletedTrip);
 }

 Future<Trip> getTrip(String tripId) async {
   return tripsAPIService.getTrip(tripId);
 }
}
```

5. Create a new dart file inside the **lib/feature/trip/controller** folder and call it **trips_list_controller.dart**.



6. Open the **trips_list_controller.dart** file and update it with the following code. The UI will use the controller to add a new trip by creating the trip item and passing it as a parameter to the **tripsRepository.add(trip)** function.

> **Note**
>
> VSCode will show errors due to the missing the **trips_list_controller.g.dart** file. You will fix that in the next step.

```
import 'dart:async';

import 'package:amplify_flutter/amplify_flutter.dart';
import 'package:amplify_trips_planner/features/trip/data/trips_repository.dart';
import 'package:amplify_trips_planner/models/ModelProvider.dart';
import 'package:riverpod_annotation/riverpod_annotation.dart';

part 'trips_list_controller.g.dart';
```

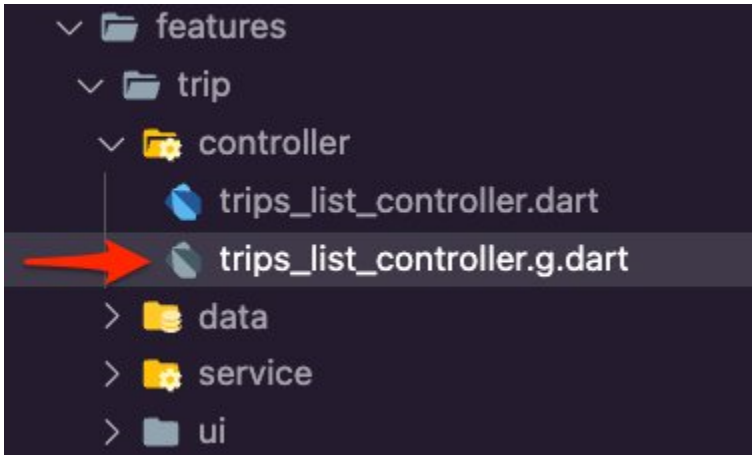
```
@riverpod
class TripsListController extends _$TripsListController {
 Future<List<Trip>> _fetchTrips() async {
   final tripsRepository = ref.read(tripsRepositoryProvider);
   final trips = await tripsRepository.getTrips();
   return trips;
 }

 @override
 FutureOr<List<Trip>> build() async {
   return _fetchTrips();
 }

 Future<void> addTrip({
   required String name,
   required String destination,
   required String startDate,
   required String endDate,
 }) async {
   final trip = Trip(
     tripName: name,
     destination: destination,
     startDate: TemporalDate(DateTime.parse(startDate)),
     endDate: TemporalDate(DateTime.parse(endDate)),
   );

   state = const AsyncValue.loading();

   state = await AsyncValue.guard(() async {
     final tripsRepository = ref.read(tripsRepositoryProvider);
     await tripsRepository.add(trip);
     return _fetchTrips();
   });
 }

 Future<void> removeTrip(Trip trip) async {
   state = const AsyncValue.loading();
   state = await AsyncValue.guard(() async {
     final tripsRepository = ref.read(tripsRepositoryProvider);
     await tripsRepository.delete(trip);

     return _fetchTrips();
   });
```

```
  }
}
```

7. Navigate to the app's root folder and run the command below in your terminal.

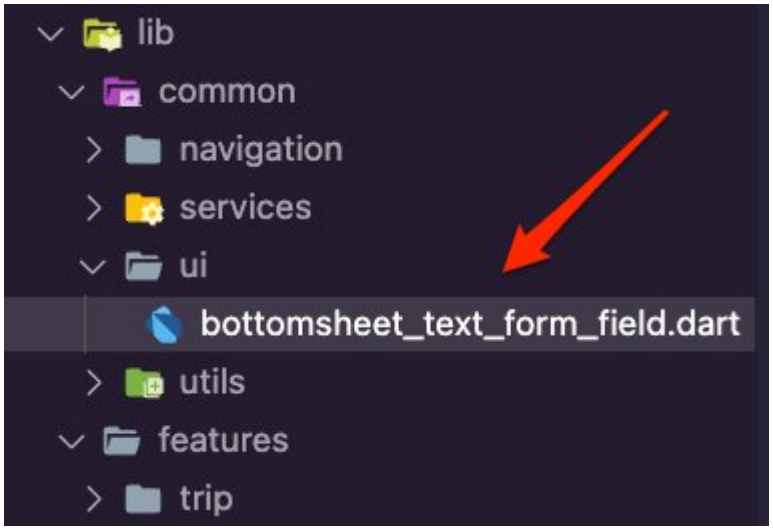
```
dart run build_runner build -d
```

This will generate the **trips_list_controller.g.dart** file inside the **lib/feature/trip/controller** folder.



Step 4: Implement the trips listing UI

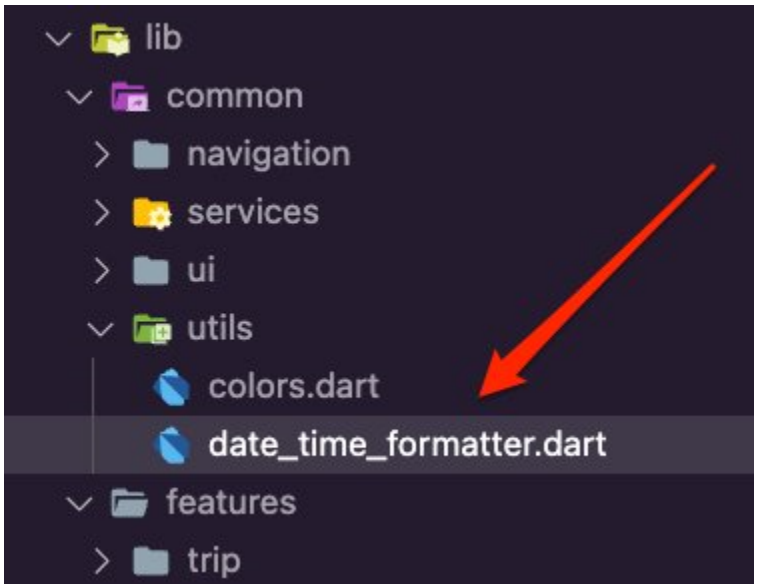
1. Create a new dart file in the **lib/common/ui** folder and call it **bottomsheet_text_form_field.dart**.



2. Open the **bottomsheet_text_form_field.dart** file and update it with the following code to create the **BottomSheetTextFormField** widget to build a **TextFormField** that the App will use in a form to create a new trip.

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

class BottomSheetTextFormField extends StatelessWidget {
 const BottomSheetTextFormField({
   required this.labelText,
   required this.controller,
   required this.keyboardType,
   this.onTap,
   super.key,
 });

 final String labelText;
 final TextEditingController controller;
 final TextInputType keyboardType;
 final void Function()? onTap;

 @override
 Widget build(BuildContext context) {
   return TextFormField(
     controller: controller,
     keyboardType: keyboardType,
     autofocus: true,
     autocorrect: false,
     textInputAction: TextInputAction.next,
     validator: (value) {
       if (value == null || value.isEmpty) {
         return 'Please enter a value';
       }

       return null;
     },
     decoration: InputDecoration(
       labelText: labelText,
     ),
     onTap: onTap,
   );
 }
}
```

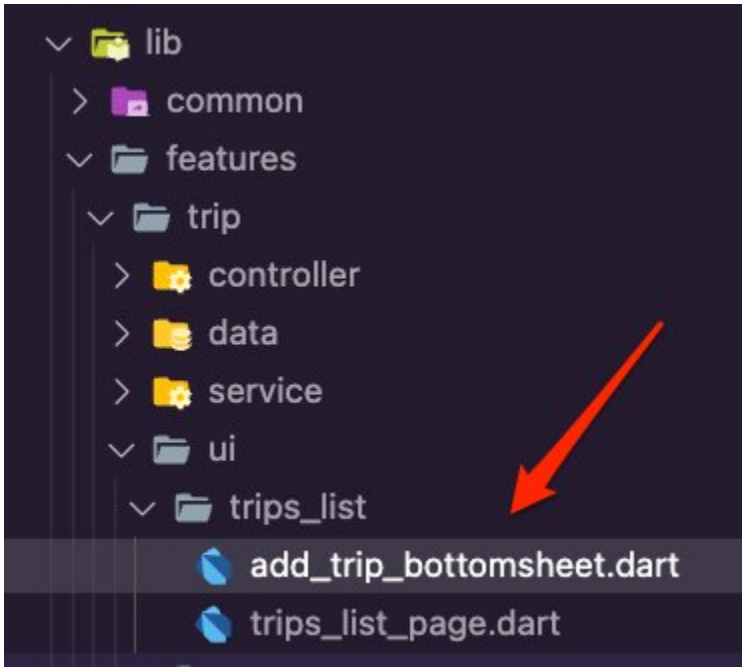
3. Create a new dart file in the **lib/common/utils** folder and call it **date_time_formatter.dart**.



4. Open the **date_time_formatter.dart** file and update it with the following code to create the **DateTimeFormatter** extension to format the DateTime value.

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

extension DateTimeFormatter on DateTime {
 String format(String format) {
   return DateFormat(format).format(this);
 }
}
```

5. Create a new dart file inside the **lib/features/trip/ui/trips_list** folder and call it **add_trip_bottomsheet.dart**.



6. Open the **add_trip_bottomsheet.dart** file and update it with the following code. This will allow us to present a form to the user to submit the required details to create a new trip.

   Note how we are using the **BottomSheetTextFormField** widget and the **DateTimeFormatter** extension in the form.

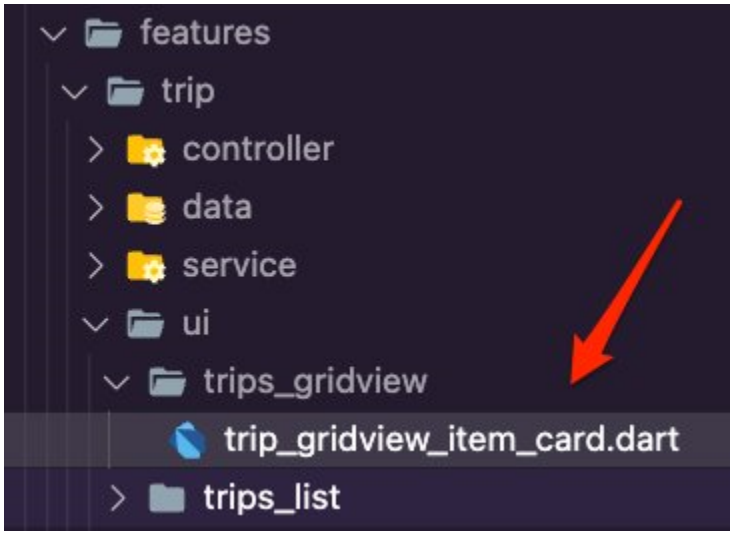
```
import 'package:amplify_trips_planner/common/ui/bottomsheet_text_form_field.dart';
import 'package:amplify_trips_planner/common/utils/date_time_formatter.dart';
import 'package:amplify_trips_planner/features/trip/controller/
trips_list_controller.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:go_router/go_router.dart';

class AddTripBottomSheet extends ConsumerStatefulWidget {
 const AddTripBottomSheet({
   super.key,
 });

 @override
 AddTripBottomSheetState createState() => AddTripBottomSheetState();
}

class AddTripBottomSheetState extends ConsumerState<AddTripBottomSheet> {
 final formGlobalKey = GlobalKey<FormState>();
```

```
final tripNameController = TextEditingController();
final destinationController = TextEditingController();
final startDateController = TextEditingController();
final endDateController = TextEditingController();

@override
Widget build(BuildContext context) {
  return Form(
    key: formGlobalKey,
    child: Container(
      padding: EdgeInsets.only(
        top: 15,
        left: 15,
        right: 15,
        bottom: MediaQuery.of(context).viewInsets.bottom + 15,
      ),
      width: double.infinity,
      child: Column(
        mainAxisSize: MainAxisSize.min,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          BottomSheetTextFormField(
            labelText: 'Trip Name',
            controller: tripNameController,
            keyboardType: TextInputType.name,
          ),
          const SizedBox(
            height: 20,
          ),
          BottomSheetTextFormField(
            labelText: 'Trip Destination',
            controller: destinationController,
            keyboardType: TextInputType.name,
          ),
          const SizedBox(
            height: 20,
          ),
          BottomSheetTextFormField(
            labelText: 'Start Date',
            controller: startDateController,
            keyboardType: TextInputType.datetime,
            onTap: () async {
              final pickedDate = await showDatePicker(
                context: context,
```

```
                initialDate: DateTime.now(),
                firstDate: DateTime(2000),
                lastDate: DateTime(2101),
              );

              if (pickedDate != null) {
                print(pickedDate.format('yyyy-MM-dd'));
                startDateController.text = pickedDate.format('yyyy-MM-dd');
                print(startDateController.text);
              }
            },
          ),
          const SizedBox(
            height: 20,
          ),
          BottomSheetTextFormField(
            labelText: 'End Date',
            controller: endDateController,
            keyboardType: TextInputType.datetime,
            onTap: () async {
              if (startDateController.text.isNotEmpty) {
                final pickedDate = await showDatePicker(
                  context: context,
                  initialDate: DateTime.parse(startDateController.text),
                  firstDate: DateTime.parse(startDateController.text),
                  lastDate: DateTime(2101),
                );

                if (pickedDate != null) {
                  print(pickedDate.format('yyyy-MM-dd'));
                  endDateController.text = pickedDate.format('yyyy-MM-dd');
                }
              }
            },
          ),
          const SizedBox(
            height: 20,
          ),
          TextButton(
            child: const Text('OK'),
            onPressed: () async {
              final currentState = formGlobalKey.currentState;
              if (currentState == null) {
                return;
```

```
                }
                if (currentState.validate()) {
                  await ref.watch(tripsListControllerProvider.notifier).addTrip(
                        name: tripNameController.text,
                        destination: destinationController.text,
                        startDate: startDateController.text,
                        endDate: endDateController.text,
                      );

                  if (context.mounted) {
                    context.pop();
                  }
                }
              }, //,
            ),
          ],
        ),
      ),
    );
  }
}
```

7. Create a new folder inside the **lib/features/trip/ui** folder, name it **trips_gridview**, and then create the file **trip_gridview_item_card.dart** inside it.



8. Open the **trip_gridview_item_card.dart** file and update it with the following code. This will create a Card widget to display the trip details in the trips list page.

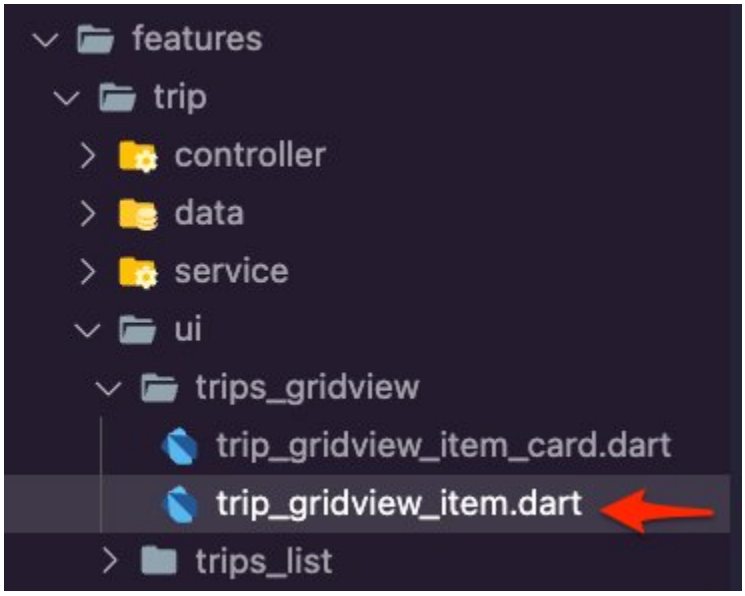
```
import 'package:amplify_trips_planner/common/utils/colors.dart' as constants;
import 'package:amplify_trips_planner/common/utils/date_time_formatter.dart';
```

```
import 'package:amplify_trips_planner/models/ModelProvider.dart';
import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/material.dart';

class TripGridViewItemCard extends StatelessWidget {
 const TripGridViewItemCard({
   required this.trip,
   super.key,
 });

 final Trip trip;

 @override
 Widget build(BuildContext context) {
   return Card(
     clipBehavior: Clip.antiAlias,
     shape: RoundedRectangleBorder(
       borderRadius: BorderRadius.circular(15),
     ),
     elevation: 5,
     child: Column(
       children: [
         Expanded(
           child: Container(
             height: 500,
             alignment: Alignment.center,
             color: const Color(constants.primaryColorDark),
             child: Stack(
               children: [
                 Positioned.fill(
                   child: trip.tripImageUrl != null
                       ? Stack(
                           children: [
                             const Center(child: CircularProgressIndicator()),
                             CachedNetworkImage(
                               errorWidget: (context, url, dynamic error) =>
                                   const Icon(Icons.error_outline_outlined),
                               imageUrl: trip.tripImageUrl!,
                               cacheKey: trip.tripImageKey,
                               width: double.maxFinite,
                               height: 500,
                               alignment: Alignment.topCenter,
                               fit: BoxFit.fill,
                             ),
```

```
                        ],
                      )
                    : Image.asset(
                        'images/amplify.png',
                        fit: BoxFit.contain,
                      ),
              ),
              Positioned(
                bottom: 16,
                left: 16,
                right: 16,
                child: FittedBox(
                  fit: BoxFit.scaleDown,
                  alignment: Alignment.centerLeft,
                  child: Text(
                    trip.destination,
                    style: Theme.of(context)
                        .textTheme
                        .headlineSmall!
                        .copyWith(color: Colors.white),
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
      Padding(
        padding: const EdgeInsets.fromLTRB(2, 8, 8, 4),
        child: DefaultTextStyle(
          softWrap: false,
          overflow: TextOverflow.ellipsis,
          style: Theme.of(context).textTheme.titleMedium!,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Padding(
                padding: const EdgeInsets.only(bottom: 8),
                child: Text(
                  trip.tripName,
                  style: Theme.of(context)
                      .textTheme
                      .titleMedium!
                      .copyWith(color: Colors.black54),
```

```
                  ),
                ),
                Text(
                  trip.startDate.getDateTime().format('MMMM dd, yyyy'),
                  style: const TextStyle(fontSize: 12),
                ),
                Text(
                  trip.endDate.getDateTime().format('MMMM dd, yyyy'),
                  style: const TextStyle(fontSize: 12),
                ),
              ],
            ),
          ),
        ),
      ],
    ),
  );
 }
}
```

9. Create the file **trip_gridview_item.dart** in the **lib/features/trip/ui/trips_gridview** folder.

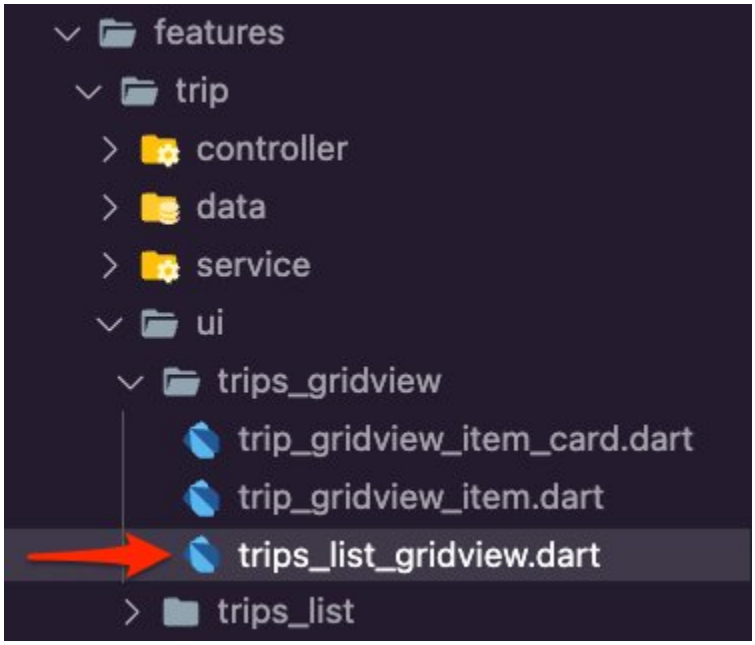


10. Open the **trip_gridview_item.dart** file and update it with the following code. The App will use this for the trips list grid.

```
import 'package:amplify_trips_planner/common/navigation/router/routes.dart';
import 'package:amplify_trips_planner/features/trip/ui/trips_gridview/
trip_gridview_item_card.dart';
import 'package:amplify_trips_planner/models/ModelProvider.dart';
```

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

class TripGridViewItem extends StatelessWidget {
 const TripGridViewItem({
   required this.trip,
   super.key,
 });

 final Trip trip;

 @override
 Widget build(BuildContext context) {
   return InkWell(
     splashColor: Theme.of(context).primaryColor,
     borderRadius: BorderRadius.circular(15),
     onTap: () {
       context.goNamed(
         AppRoute.trip.name,
         pathParameters: {'id': trip.id},
         extra: trip,
       );
     },
     child: TripGridViewItemCard(
       trip: trip,
     ),
   );
 }
}
```

11. Create the file **trips_list_gridview.dart** inside the **lib/features/trip/ui/trips_gridview** folder.



12. Open the **trips_list_gridview.dart** file and update it with the following code. This will cause the gridview to display the trips list.

```
import 'package:amplify_trips_planner/features/trip/ui/trips_gridview/
trip_gridview_item.dart';
import 'package:amplify_trips_planner/models/ModelProvider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

class TripsListGridView extends StatelessWidget {
 const TripsListGridView({
   required this.tripsList,
   super.key,
 });

 final AsyncValue<List<Trip>> tripsList;

 @override
 Widget build(BuildContext context) {
   switch (tripsList) {
     case AsyncData(:final value):
       return value.isEmpty
           ? const Center(
               child: Text('No Trips'),
             )
           : OrientationBuilder(
```

```
                builder: (context, orientation) {
                  return GridView.count(
                    crossAxisCount:
                        (orientation == Orientation.portrait) ? 2 : 3,
                    mainAxisSpacing: 4,
                    crossAxisSpacing: 4,
                    padding: const EdgeInsets.all(4),
                    childAspectRatio:
                        (orientation == Orientation.portrait) ? 0.9 : 1.4,
                    children: value.map((tripData) {
                      return TripGridViewItem(
                        trip: tripData,
                      );
                    }).toList(growable: false),
                  );
                },
              );

      case AsyncError():
        return const Center(
          child: Text('Error'),
        );
      case AsyncLoading():
        return const Center(
          child: CircularProgressIndicator(),
        );

      case _:
        return const Center(
          child: Text('Error'),
        );
    }
  }
}
```

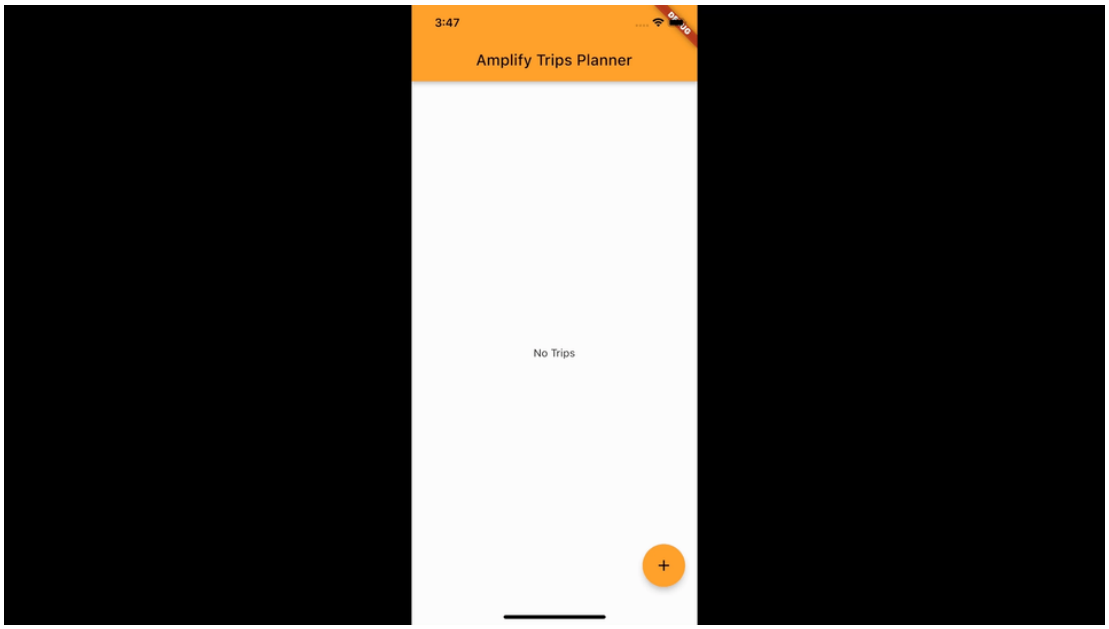
13. Open the **trips_list_page.dart** file and update it with the following code to use the **TripsListGridView** widget you created above for displaying the trips.

```
import 'package:amplify_trips_planner/common/utils/colors.dart' as constants;
import 'package:amplify_trips_planner/features/trip/controller/
trips_list_controller.dart';
import 'package:amplify_trips_planner/features/trip/ui/trips_gridview/
trips_list_gridview.dart';
```

```
import 'package:amplify_trips_planner/features/trip/ui/trips_list/
add_trip_bottomsheet.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

class TripsListPage extends ConsumerWidget {
 const TripsListPage({
   super.key,
 });

 Future<void> showAddTripDialog(BuildContext context) =>
     showModalBottomSheet<void>(
       isScrollControlled: true,
       elevation: 5,
       context: context,
       builder: (sheetContext) {
         return const AddTripBottomSheet();
       },
     );

 @override
 Widget build(BuildContext context, WidgetRef ref) {
   final tripsListValue = ref.watch(tripsListControllerProvider);
   return Scaffold(
     appBar: AppBar(
       centerTitle: true,
       title: const Text(
         'Amplify Trips Planner',
       ),
       backgroundColor: const Color(constants.primaryColorDark),
     ),
     floatingActionButton: FloatingActionButton(
       onPressed: () {
         showAddTripDialog(context);
       },
       backgroundColor: const Color(constants.primaryColorDark),
       child: const Icon(Icons.add),
     ),
     body: TripsListGridView(
       tripsList: tripsListValue,
     ),
   );
 }
```

```
}
```

14. Run the app in the simulator and create a trip. The following is an example using an iPhone simulator.



Conclusion

In this module, you added a GraphQL API for trips using Amplify, and configured create, read, update, and delete trips functionality in your app.

Module 4: Add Amplify storage

Overview

In this module, you will add the ability to upload an image for each trip. You will add Amplify storage to enable image uploading and rendering.

The Amplify Storage category comes with default built-in support for Amazon Simple Storage Service (Amazon S3). The Amplify CLI helps you create and configure your app's storage buckets.

What you will accomplish

- Add Amplify storage to the app
- Add the Trip Details page to the app
- Implement the upload image feature

Implementation

Minimum time to complete	15 minutes

Step 1: Add Amplify storage to the app

1. Navigate to the root folder of the app and set up a storage resource by running the following command in your terminal.

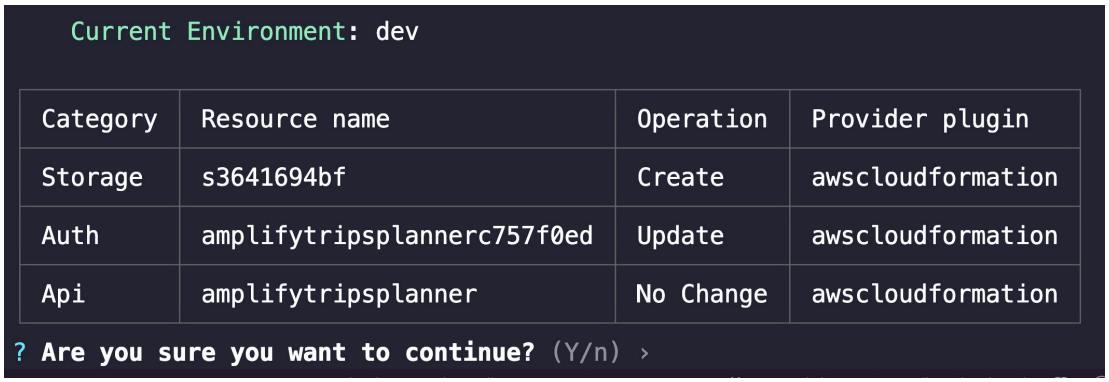
```
amplify add storage
```

2. To create the Storage category, enter the following when prompted:

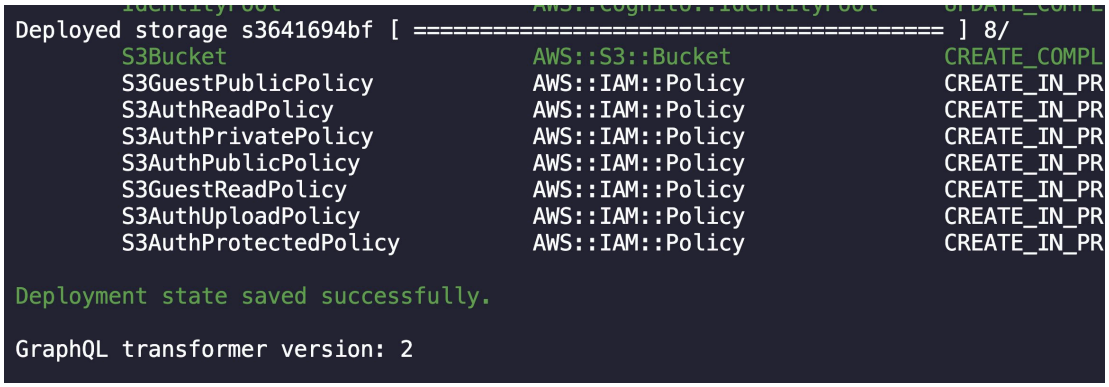
```
? Select from one of the below mentioned services: Content (Images, audio, video,
 etc.)
# Provide a friendly name for your resource that will be used to label this
 category in the project: · s3cf3f0a40
# Provide bucket name: · amplifytripsplannerstorage
# Who should have access: · Auth and guest users
```

```
# What kind of access do you want for Authenticated users? · create/update, read,
 delete
# What kind of access do you want for Guest users? · read
# Do you want to add a Lambda Trigger for your S3 Bucket? (y/N) · no
```

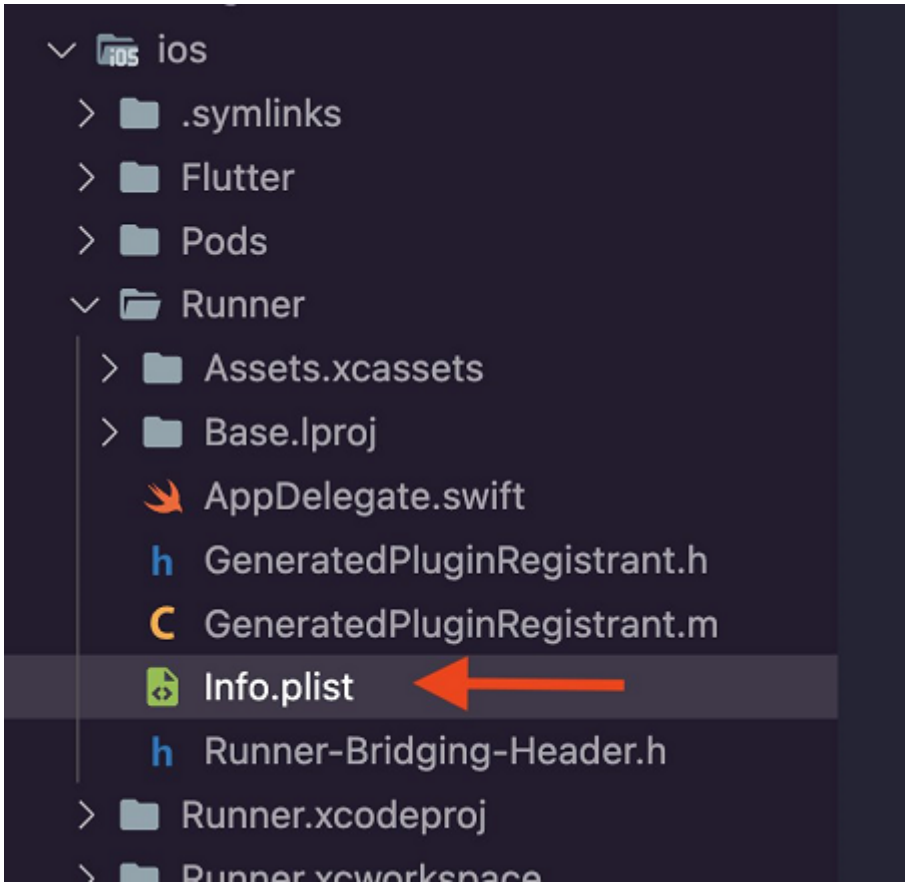
3. Run the command **amplify push** to create the resources in the cloud.



4. Press **Enter.** The Amplify CLI will deploy the resources and display a confirmation.



5. For iOS, open the file **ios/Runner/info.plist**. There are no configurations required for Android to access the phone camera and photo library.



Add the following keys and values.

```
<key>NSCameraUsageDescription</key>
<string>Some Description</string>
<key>NSMicrophoneUsageDescription</key>
<string>Some Description</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>Some Description</string>
```

The **ios/Runner/info.plist** file will look like the following.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
   <key>CFBundleDevelopmentRegion</key>
   <string>$(DEVELOPMENT_LANGUAGE)</string>
   <key>CFBundleDisplayName</key>
```

```
    <string>Amplify Trips Planner</string>
    <key>CFBundleExecutable</key>
    <string>$(EXECUTABLE_NAME)</string>
    <key>CFBundleIdentifier</key>
    <string>$(PRODUCT_BUNDLE_IDENTIFIER)</string>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleName</key>
    <string>amplify_trips_planner</string>
    <key>CFBundlePackageType</key>
    <string>APPL</string>
    <key>CFBundleShortVersionString</key>
    <string>$(FLUTTER_BUILD_NAME)</string>
    <key>CFBundleSignature</key>
    <string>????</string>
    <key>CFBundleVersion</key>
    <string>$(FLUTTER_BUILD_NUMBER)</string>
    <key>LSRequiresIPhoneOS</key>
    <true/>
    <key>UILaunchStoryboardName</key>
    <string>LaunchScreen</string>
    <key>UIMainStoryboardFile</key>
    <string>Main</string>
    <key>UISupportedInterfaceOrientations</key>
    <array>
        <string>UIInterfaceOrientationPortrait</string>
        <string>UIInterfaceOrientationLandscapeLeft</string>
        <string>UIInterfaceOrientationLandscapeRight</string>
    </array>
    <key>UISupportedInterfaceOrientations~ipad</key>
    <array>
        <string>UIInterfaceOrientationPortrait</string>
        <string>UIInterfaceOrientationPortraitUpsideDown</string>
        <string>UIInterfaceOrientationLandscapeLeft</string>
        <string>UIInterfaceOrientationLandscapeRight</string>
    </array>
    <key>UIViewControllerBasedStatusBar</key> Appearance
    <false/>
    <key>CADisableMinimumFrameDurationOnPhone</key>
    <true/>
    <key>UIApplicationSupportsIndirectInputEvents</key>
    <true/>
    <key>NSCameraUsageDescription</key>
    <string>Some Description</string>
```

```
   <key>NSMicrophoneUsageDescription</key>
   <string>Some Description</string>
   <key>NSPhotoLibraryUsageDescription</key>
   <string>Some Description</string>
</dict>
</plist>
```

6. Open the **main.dart** file and update the **_configureAmplify()** function as shown in the following code to add the Amplify storage plugin.

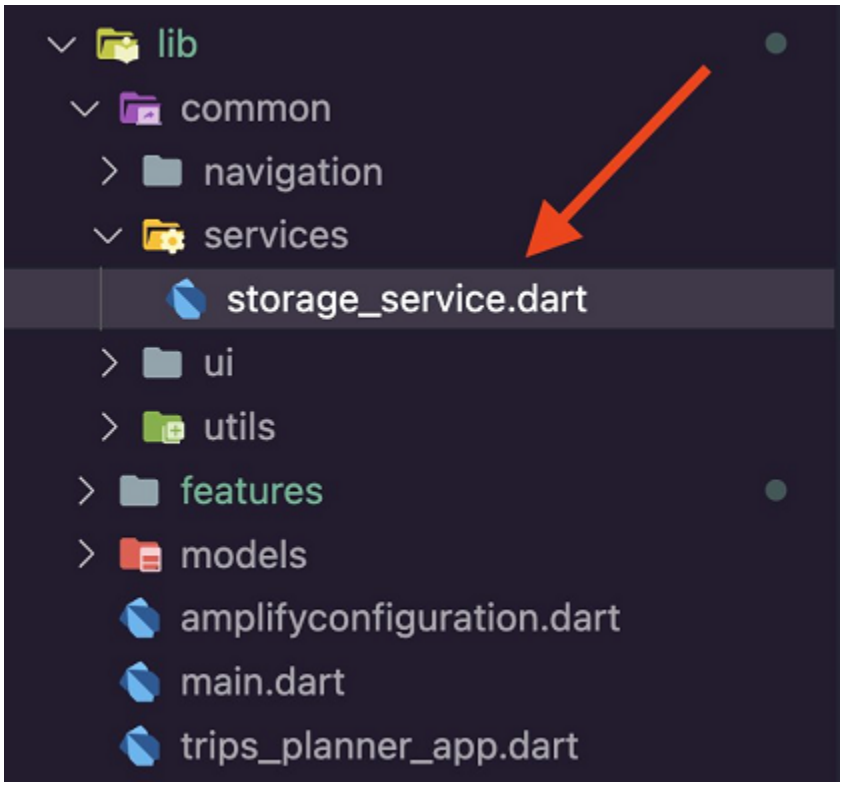
```
Future<void> _configureAmplify() async {
 await Amplify.addPlugins([
   AmplifyAuthCognito(),
   AmplifyAPI(modelProvider: ModelProvider.instance),
    AmplifyStorageS3()
 ]);
 await Amplify.configure(amplifyconfig);
}
```

The **main.dart** file should now look like the following.

```
import 'package:amplify_api/amplify_api.dart';
import 'package:amplify_auth_cognito/amplify_auth_cognito.dart';
import 'package:amplify_flutter/amplify_flutter.dart';
import 'package:amplify_trips_planner/models/ModelProvider.dart';
import 'package:amplify_trips_planner/trips_planner_app.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:amplify_storage_s3/amplify_storage_s3.dart';

import 'amplifyconfiguration.dart';

Future<void> main() async {
 WidgetsFlutterBinding.ensureInitialized();
 try {
   await _configureAmplify();
 } on AmplifyAlreadyConfiguredException {
   debugPrint('Amplify configuration failed.');
 }

 runApp(
   const ProviderScope(
     child: TripsPlannerApp(),
```

```
    ),
  );
}

Future<void> _configureAmplify() async {
 await Amplify.addPlugins([
   AmplifyAuthCognito(),
   AmplifyAPI(modelProvider: ModelProvider.instance),
    AmplifyStorageS3()
 ]);
 await Amplify.configure(amplifyconfig);
}
```

7. Create a new dart file inside the folder **lib/common/services** and name it **storage_service.dart.**



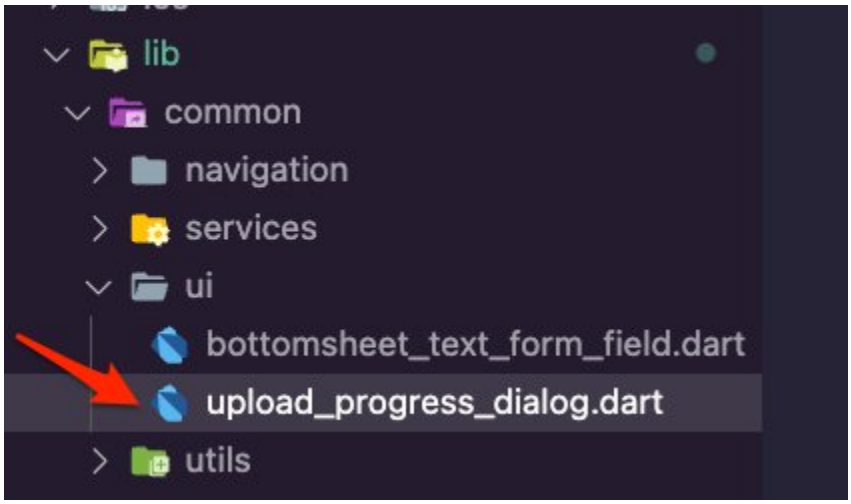
8. Open **storage_service.dart** file and update it with the following code to create the **StorageService**. In this service, you will find the **uploadFile** function, which uses the Amplify storage library to upload an image into an Amazon S3 bucket. Additionally, the service provides a **ValueNotifier** object to track the progress of the image upload.

```
import 'dart:io';
```

```
import 'package:amplify_flutter/amplify_flutter.dart';
import 'package:amplify_storage_s3/amplify_storage_s3.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:path/path.dart' as p;
import 'package:uuid/uuid.dart';

final storageServiceProvider = Provider<StorageService>((ref) {
 return StorageService(ref: ref);
});

class StorageService {
 StorageService({
   required Ref ref,
 });

 ValueNotifier<double> uploadProgress = ValueNotifier<double>(0);
 Future<String> getImageUrl(String key) async {
   final result = await Amplify.Storage.getUrl(
     key: key,
     options: const StorageGetUrlOptions(
       pluginOptions: S3GetUrlPluginOptions(
         validateObjectExistence: true,
         expiresIn: Duration(days: 1),
       ),
     ),
   ).result;
   return result.url.toString();
 }

 ValueNotifier<double> getUploadProgress() {
   return uploadProgress;
 }

 Future<String?> uploadFile(File file) async {
   try {
     final extension = p.extension(file.path);
     final key = const Uuid().v1() + extension;
     final awsFile = AWSFile.fromPath(file.path);

     await Amplify.Storage.uploadFile(
       localFile: awsFile,
       key: key,
       onProgress: (progress) {
```

```
          uploadProgress.value = progress.fractionCompleted;
        },
      ).result;

      return key;
    } on Exception catch (e) {
      debugPrint(e.toString());
      return null;
    }
  }

  void resetUploadProgress() {
    uploadProgress.value = 0;
  }
}
```

9. Create a new dart file inside the folder **lib/common/ui** and name it **upload_progress_dialog.dart.**



10. Open the **upload_progress_dialog.dart** file and update it with the following code to create a dialog that uses a progress indicator for the image upload.

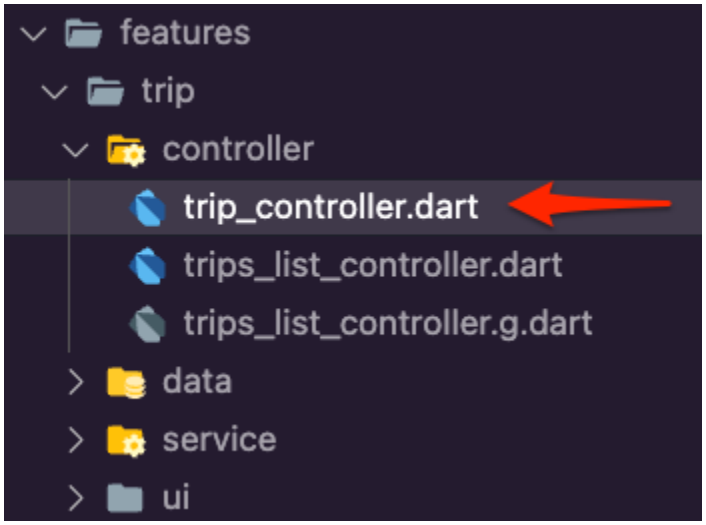
> **Note**
>
> VSCode will show an error about missing the **trip_controller.dart** file. You will fix that in the next steps.

```
import 'package:amplify_trips_planner/features/trip/controller/
trip_controller.dart';
```

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

class UploadProgressDialog extends ConsumerWidget {
 const UploadProgressDialog({super.key});

 @override
 Widget build(BuildContext context, WidgetRef ref) {
   return Dialog(
     backgroundColor: Colors.white,
     child: Padding(
       padding: const EdgeInsets.symmetric(vertical: 20),
       child: ValueListenableBuilder(
         valueListenable:
             ref.read(tripControllerProvider('').notifier).uploadProgress(),
         builder: (context, value, child) {
           return Column(
             mainAxisSize: MainAxisSize.min,
             children: [
               const CircularProgressIndicator(),
               const SizedBox(
                 height: 15,
               ),
               Text('${(double.parse(value.toString()) * 100).toInt()} %'),
               Container(
                 alignment: Alignment.topCenter,
                 margin: const EdgeInsets.all(20),
                 child: LinearProgressIndicator(
                   value: double.parse(value.toString()),
                   backgroundColor: Colors.grey,
                   color: Colors.purple,
                   minHeight: 10,
                 ),
               ),
             ],
           );
         },
       ),
     ),
   );
 }
}
```

Step 2: Add the Trip Details page to the app

1. Create a new dart file inside the folder **lib/features/trip/controller** and name it **trip_controller.dart.**



2. Open the **trip_controller.dart** file and update it with the following code. The UI will use this controller for editing and deleting a trip using its ID. The UI will also use the controller for uploading an image for the trip.

> **Note**
>
> VSCode will show errors due to the missing the **trip_controller.g.dart** file. You will fix that in the next step.

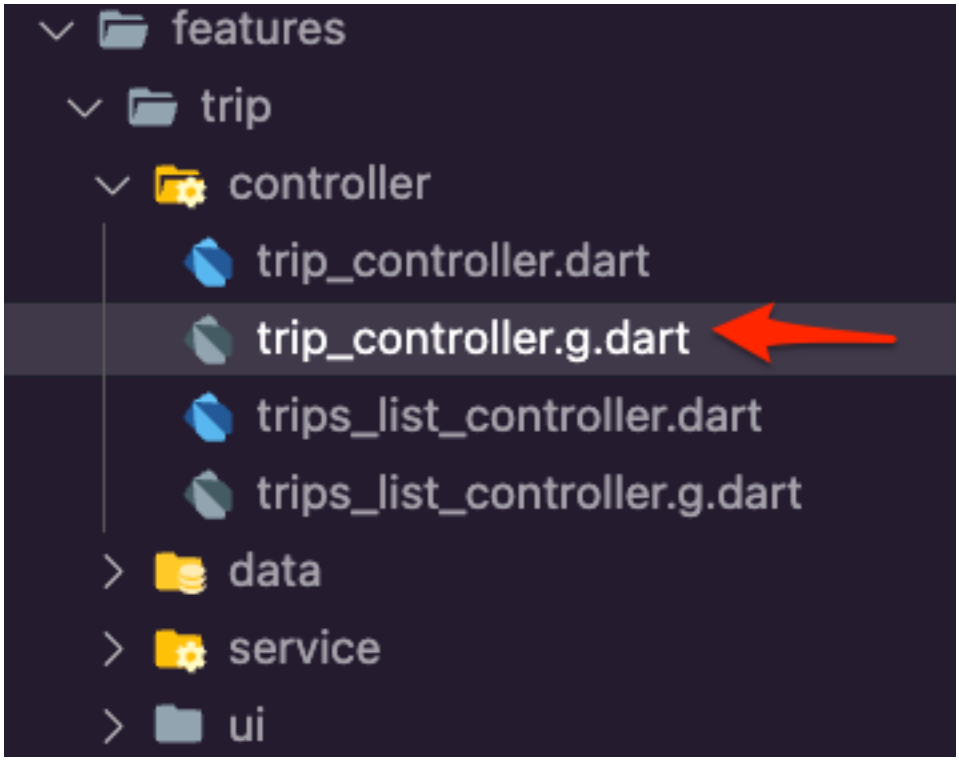
```
import 'dart:async';
import 'dart:io';

import 'package:amplify_trips_planner/common/services/storage_service.dart';
import 'package:amplify_trips_planner/features/trip/data/trips_repository.dart';
import 'package:amplify_trips_planner/models/ModelProvider.dart';
import 'package:flutter/material.dart';
import 'package:riverpod_annotation/riverpod_annotation.dart';

part 'trip_controller.g.dart';

@riverpod
class TripController extends _$TripController {
```

```
  Future<Trip> _fetchTrip(String tripId) async {
    final tripsRepository = ref.read(tripsRepositoryProvider);
    return tripsRepository.getTrip(tripId);
  }

  @override
  FutureOr<Trip> build(String tripId) async {
    return _fetchTrip(tripId);
  }

  Future<void> updateTrip(Trip trip) async {
    state = const AsyncValue.loading();
    state = await AsyncValue.guard(() async {
      final tripsRepository = ref.read(tripsRepositoryProvider);
      await tripsRepository.update(trip);
      return _fetchTrip(trip.id);
    });
  }

  Future<void> uploadFile(File file, Trip trip) async {
    final fileKey = await ref.read(storageServiceProvider).uploadFile(file);
    if (fileKey != null) {
      final imageUrl =
          await ref.read(storageServiceProvider).getImageUrl(fileKey);
      final updatedTrip =
          trip.copyWith(tripImageKey: fileKey, tripImageUrl: imageUrl);
      await ref.read(tripsRepositoryProvider).update(updatedTrip);
      ref.read(storageServiceProvider).resetUploadProgress();
    }
  }

  ValueNotifier<double> uploadProgress() {
    return ref.read(storageServiceProvider).getUploadProgress();
  }
}
```

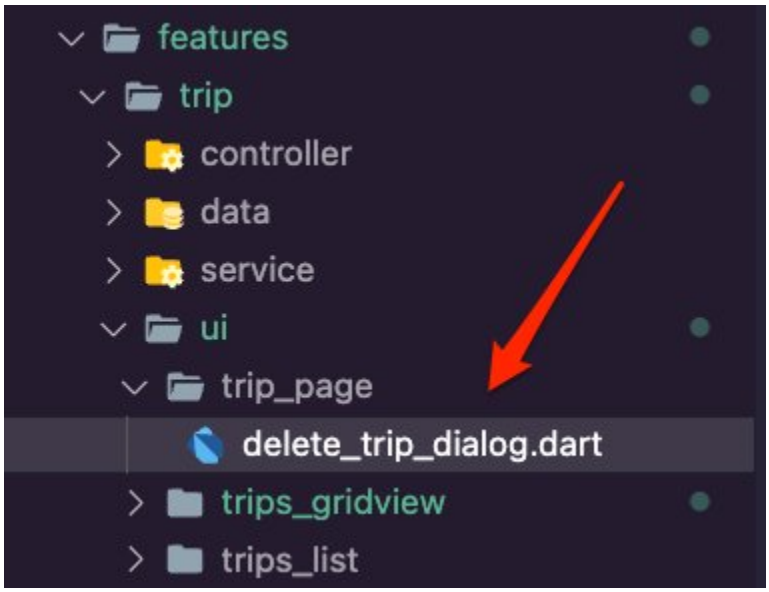
3. Navigate to the app's root folder and run the following command in your terminal.

```
dart run build_runner build -d
```

This will generate the **trip_controller.g.dart** file in the **lib/feature/trip/controller** folder



4. Create a new folder inside the **lib/features/trip/ui** folder and name it **trip_page** and then create the file **delete_trip_dialog.dart** inside it.

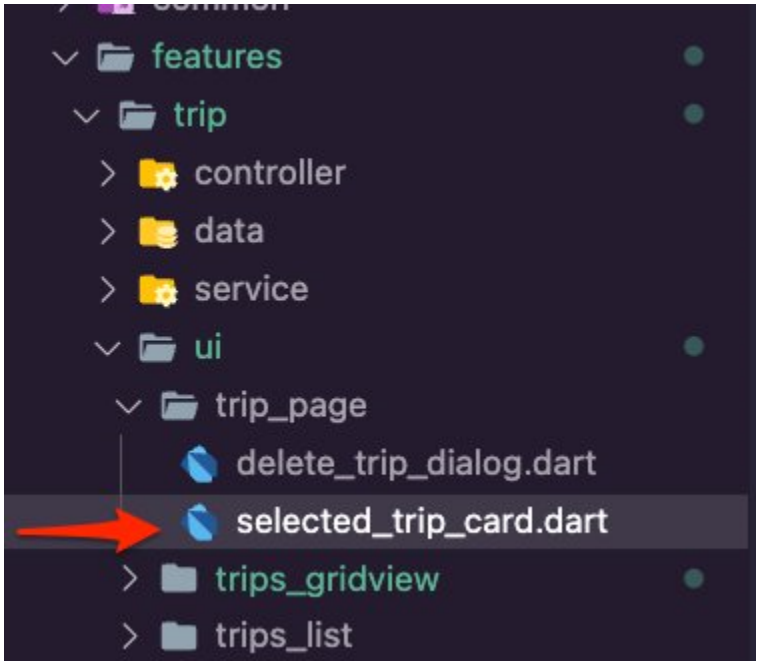


5. Open the **delete_trip_dialog.dart** file and update it with the following code. This will display a dialog for the user to confirm deleting the selected trip.

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

class DeleteTripDialog extends StatelessWidget {
```

```
  const DeleteTripDialog({
    super.key,
  });

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: const Text('Please Confirm'),
      content: const Text('Delete this trip?'),
      actions: [
        TextButton(
          onPressed: () async {
            Navigator.of(context).pop(true);
          },
          child: const Text('Yes'),
        ),
        TextButton(
          onPressed: () {
            Navigator.of(context).pop(false);
          },
          child: const Text('No'),
        )
      ],
    );
  }
}
```

6. Create a new dart file inside the **lib/features/trip/ui/trip_page** folder and name it **selected_trip_card.dart**.



7. Open the **selected_trip_card.dart** file and update it with the following code. Here we check if there is an image for the trip and display it in a card widget. We use the placeholder image from the app assets if there is no image. We are also introducing three icon buttons for the user to choose to upload a photo, edit the trip, and delete the trip.

```
import 'dart:io';

import 'package:amplify_trips_planner/common/navigation/router/routes.dart';
import 'package:amplify_trips_planner/common/ui/upload_progress_dialog.dart';
import 'package:amplify_trips_planner/common/utils/colors.dart' as constants;
import 'package:amplify_trips_planner/features/trip/controller/
trip_controller.dart';
import 'package:amplify_trips_planner/features/trip/controller/
trips_list_controller.dart';
import 'package:amplify_trips_planner/features/trip/ui/trip_page/
delete_trip_dialog.dart';
import 'package:amplify_trips_planner/models/Trip.dart';
import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:go_router/go_router.dart';
import 'package:image_picker/image_picker.dart';

class SelectedTripCard extends ConsumerWidget {
 const SelectedTripCard({
   required this.trip,
```

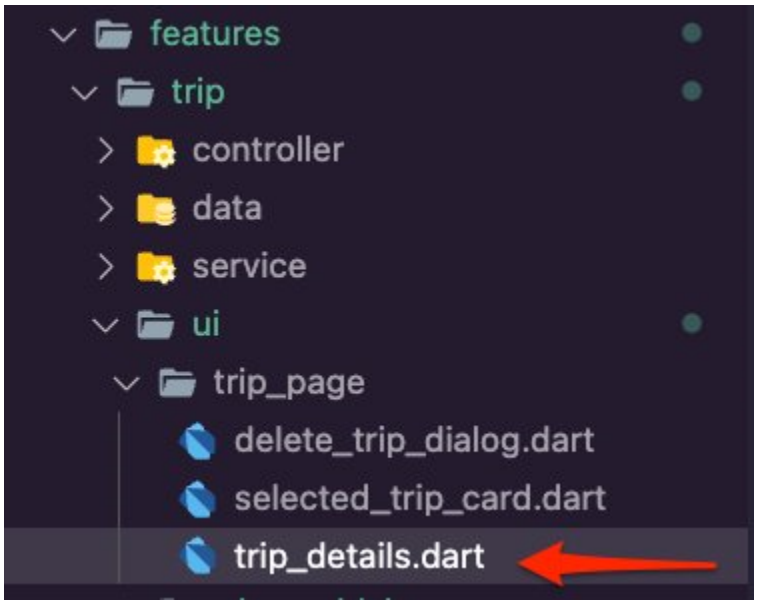
```
  super.key,
});

final Trip trip;

Future<bool> uploadImage({
  required BuildContext context,
  required WidgetRef ref,
  required Trip trip,
}) async {
  final picker = ImagePicker();
  final pickedFile = await picker.pickImage(source: ImageSource.gallery);
  if (pickedFile == null) {
    return false;
  }
  final file = File(pickedFile.path);
  if (context.mounted) {
    showDialog<String>(
      context: context,
      barrierDismissible: false,
      builder: (BuildContext context) {
        return const UploadProgressDialog();
      },
    );

    await ref
        .watch(tripControllerProvider(trip.id).notifier)
        .uploadFile(file, trip);
  }

  return true;
}

Future<bool> deleteTrip(
  BuildContext context,
  WidgetRef ref,
  Trip trip,
) async {
  var value = await showDialog<bool>(
    context: context,
    builder: (BuildContext context) {
      return const DeleteTripDialog();
    },
  );
```

```
  value ??= false;

  if (value) {
    await ref.watch(tripsListControllerProvider.notifier).removeTrip(trip);
  }
  return value;
}

@override
Widget build(BuildContext context, WidgetRef ref) {
  return Card(
    clipBehavior: Clip.antiAlias,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(15),
    ),
    elevation: 5,
    child: Column(
      mainAxisSize: MainAxisSize.min,
      children: [
        Text(
          trip.tripName,
          textAlign: TextAlign.center,
          style: const TextStyle(
            fontSize: 20,
            fontWeight: FontWeight.bold,
          ),
        ),
        const SizedBox(
          height: 8,
        ),
        Container(
          alignment: Alignment.center,
          color: const Color(constants.primaryColorDark), //Color(0xffE1E5E4),
          height: 150,

          child: trip.tripImageUrl != null
              ? Stack(
                  children: [
                    const Center(child: CircularProgressIndicator()),
                    CachedNetworkImage(
                      cacheKey: trip.tripImageKey,
                      imageUrl: trip.tripImageUrl!,
                      width: double.maxFinite,
                      height: 500,
```

```
                    alignment: Alignment.topCenter,
                    fit: BoxFit.fill,
                  ),
                ],
              )
            : Image.asset(
                'images/amplify.png',
                fit: BoxFit.contain,
              ),
        ),
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceAround,
          children: [
            IconButton(
              onPressed: () {
                context.goNamed(
                  AppRoute.editTrip.name,
                  pathParameters: {'id': trip.id},
                  extra: trip,
                );
              },
              icon: const Icon(Icons.edit),
            ),
            IconButton(
              onPressed: () {
                uploadImage(
                  context: context,
                  trip: trip,
                  ref: ref,
                ).then((value) {
                  if (value) {
                    Navigator.of(context, rootNavigator: true).pop();
                    ref.invalidate(tripControllerProvider(trip.id));
                  }
                });
              },
              icon: const Icon(Icons.camera_enhance_sharp),
            ),
            IconButton(
              onPressed: () {
                deleteTrip(context, ref, trip).then((value) {
                  if (value) {
                    context.goNamed(
                      AppRoute.home.name,
```

```
                    );
                  }
                });
              },
              icon: const Icon(Icons.delete),
            ),
          ],
        )
      ],
    ),
  );
 }
}
```

8. Create a new dart file inside the **lib/features/trip/ui/trip_page** folder and name it **trip_details.dart**.



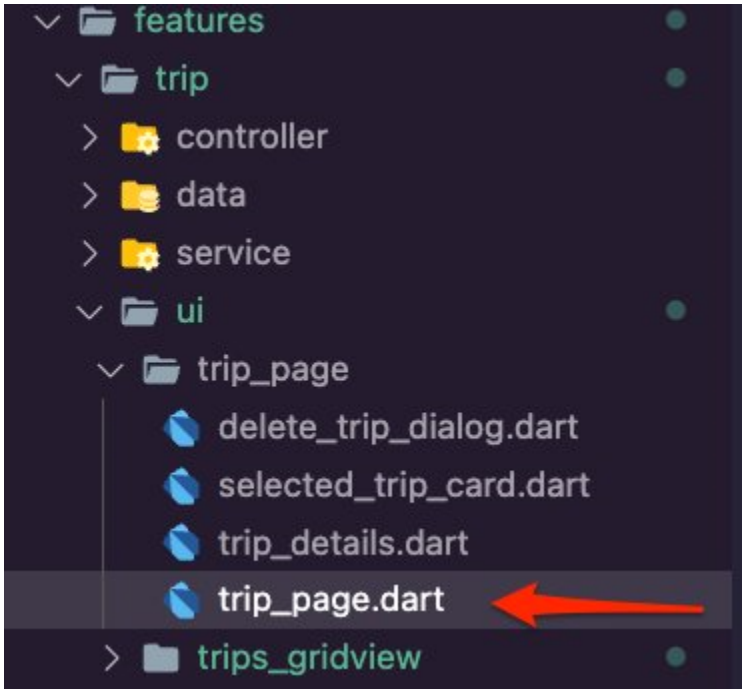
9. Open the **trip_details.dart** file and update it with the following code to create a column that uses the **SelectedTripCard** widget you created to display the details of the trip.

```
import 'package:amplify_trips_planner/features/trip/controller/
trip_controller.dart';
import 'package:amplify_trips_planner/features/trip/ui/trip_page/
selected_trip_card.dart';
import 'package:amplify_trips_planner/models/ModelProvider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
```

```
class TripDetails extends ConsumerWidget {
 const TripDetails({
   required this.trip,
   required this.tripId,
   super.key,
 });

 final AsyncValue<Trip> trip;
 final String tripId;

 @override
 Widget build(BuildContext context, WidgetRef ref) {
   switch (trip) {
     case AsyncData(:final value):
       return Column(
         crossAxisAlignment: CrossAxisAlignment.stretch,
         children: [
           const SizedBox(
             height: 8,
           ),
           SelectedTripCard(trip: value),
           const SizedBox(
             height: 20,
           ),
           const Divider(
             height: 20,
             thickness: 5,
             indent: 20,
             endIndent: 20,
           ),
           const Text(
             'Your Activities',
             textAlign: TextAlign.center,
             style: TextStyle(
               fontSize: 20,
               fontWeight: FontWeight.bold,
             ),
           ),
           const SizedBox(
             height: 8,
           ),
         ],
       );
```

```
      case AsyncError():
        return Center(
          child: Column(
            children: [
              const Text('Error'),
              TextButton(
                onPressed: () async {
                  ref.invalidate(tripControllerProvider(tripId));
                },
                child: const Text('Try again'),
              ),
            ],
          ),
        );
      case AsyncLoading():
        return const Center(
          child: CircularProgressIndicator(),
        );

      case _:
        return const Center(
          child: Text('Error'),
        );
    }
  }
}
```

10. Create a new dart file in the **lib/features/trip/ui/trip_page** folder and name it **trip_page.dart**.



11. Open the **trip_page.dart** file and update it with the following code to create the **TripPage**, which will get the trip details using the **tripId**. The **TripPage** will use the **TripDetails** you created previously to display the data.

```
import 'package:amplify_trips_planner/common/navigation/router/routes.dart';
import 'package:amplify_trips_planner/common/utils/colors.dart' as constants;
import 'package:amplify_trips_planner/features/trip/controller/
trip_controller.dart';
import 'package:amplify_trips_planner/features/trip/ui/trip_page/
trip_details.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:go_router/go_router.dart';

class TripPage extends ConsumerWidget {
 const TripPage({
   required this.tripId,
   super.key,
 });
 final String tripId;

 @override
 Widget build(BuildContext context, WidgetRef ref) {
   final tripValue = ref.watch(tripControllerProvider(tripId));
```

```
    return Scaffold(
      appBar: AppBar(
        centerTitle: true,
        title: const Text(
          'Amplify Trips Planner',
        ),
        actions: [
          IconButton(
            onPressed: () {
              context.goNamed(
                AppRoute.home.name,
              );
            },
            icon: const Icon(Icons.home),
          ),
        ],
        backgroundColor: const Color(constants.primaryColorDark),
      ),
      body: TripDetails(
        tripId: tripId,
        trip: tripValue,
      ),
    );
  }
}
```

12. Open the **lib/common/navigation/router/router.dart** file and update it to add the **TripPage** route.

```
GoRoute(
 path: '/trip/:id',
 name: AppRoute.trip.name,
 builder: (context, state) {
   final tripId = state.pathParameters['id']!;
   return TripPage(tripId: tripId);
 },
),
```

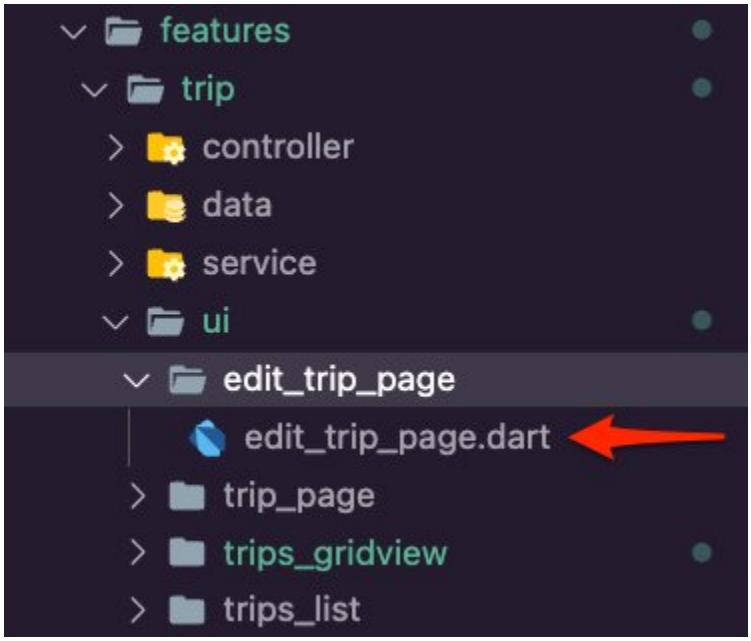
The **router.dart** should look like the following code snippet.

```
import 'package:amplify_trips_planner/common/navigation/router/routes.dart';
import 'package:amplify_trips_planner/features/trip/ui/trip_page/trip_page.dart';
```

```
import 'package:amplify_trips_planner/features/trip/ui/trips_list/
trips_list_page.dart';
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';

final router = GoRouter(
 routes: [
   GoRoute(
     path: '/',
     name: AppRoute.home.name,
     builder: (context, state) => const TripsListPage(),
   ),
   GoRoute(
     path: '/trip/:id',
     name: AppRoute.trip.name,
     builder: (context, state) {
       final tripId = state.pathParameters['id']!;
       return TripPage(tripId: tripId);
     },
   ),
 ],
 errorBuilder: (context, state) => Scaffold(
   body: Center(
     child: Text(state.error.toString()),
   ),
 ),
);
```

Step 3: Add the Edit Trip page to the app

1. Create a new folder inside the **lib/features/trip/ui** folder and name it **edit_trip_page** and then create the file **edit_trip_page.dart** inside it.



2. Open the **edit_trip_page.dart** file and update it with the following code to create the UI for the user to edit the selected trip.

```
import 'package:amplify_flutter/amplify_flutter.dart';
import 'package:amplify_trips_planner/common/navigation/router/routes.dart';
import 'package:amplify_trips_planner/common/ui/bottomsheet_text_form_field.dart';
import 'package:amplify_trips_planner/common/utils/colors.dart' as constants;
import 'package:amplify_trips_planner/common/utils/date_time_formatter.dart';
import 'package:amplify_trips_planner/features/trip/controller/
trip_controller.dart';
import 'package:amplify_trips_planner/models/ModelProvider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:go_router/go_router.dart';

class EditTripPage extends ConsumerStatefulWidget {
 const EditTripPage({
   required this.trip,
   super.key,
 });

 final Trip trip;

 @override
 EditTripPageState createState() => EditTripPageState();
}
```

```
class EditTripPageState extends ConsumerState<EditTripPage> {
 @override
 void initState() {
   tripNameController.text = widget.trip.tripName;
   destinationController.text = widget.trip.destination;

   startDateController.text =
       widget.trip.startDate.getDateTime().format('yyyy-MM-dd');

   endDateController.text =
       widget.trip.endDate.getDateTime().format('yyyy-MM-dd');

   super.initState();
 }

 final formGlobalKey = GlobalKey<FormState>();
 final tripNameController = TextEditingController();
 final destinationController = TextEditingController();
 final startDateController = TextEditingController();

 final endDateController = TextEditingController();
 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       centerTitle: true,
       title: const Text(
         'Amplify Trips Planner',
       ),
       leading: IconButton(
         onPressed: () {
           context.goNamed(
             AppRoute.trip.name,
             pathParameters: {'id': widget.trip.id},
           );
         },
         icon: const Icon(Icons.arrow_back),
       ),
       backgroundColor: const Color(constants.primaryColorDark),
     ),
     body: SingleChildScrollView(
       child: Form(
         key: formGlobalKey,
         child: Container(
```

```
            padding: EdgeInsets.only(
              top: 15,
              left: 15,
              right: 15,
              bottom: MediaQuery.of(context).viewInsets.bottom + 15,
            ),
            width: double.infinity,
            child: Column(
              mainAxisSize: MainAxisSize.min,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                BottomSheetTextFormField(
                  labelText: 'Trip Name',
                  controller: tripNameController,
                  keyboardType: TextInputType.name,
                ),
                const SizedBox(
                  height: 20,
                ),
                BottomSheetTextFormField(
                  labelText: 'Trip Destination',
                  controller: destinationController,
                  keyboardType: TextInputType.name,
                ),
                const SizedBox(
                  height: 20,
                ),
                BottomSheetTextFormField(
                  labelText: 'Start Date',
                  controller: startDateController,
                  keyboardType: TextInputType.datetime,
                  onTap: () async {
                    final pickedDate = await showDatePicker(
                      context: context,
                      initialDate: DateTime.now(),
                      firstDate: DateTime(2000),
                      lastDate: DateTime(2101),
                    );

                    if (pickedDate != null) {
                      startDateController.text =
                          pickedDate.format('yyyy-MM-dd');
                    }
                  },
```

```
                ),
                const SizedBox(
                  height: 20,
                ),
                BottomSheetTextFormField(
                  labelText: 'End Date',
                  controller: endDateController,
                  keyboardType: TextInputType.datetime,
                  onTap: () async {
                    if (startDateController.text.isNotEmpty) {
                      final pickedDate = await showDatePicker(
                        context: context,
                        initialDate: DateTime.parse(startDateController.text),
                        firstDate: DateTime.parse(startDateController.text),
                        lastDate: DateTime(2101),
                      );

                      if (pickedDate != null) {
                        endDateController.text =
                            pickedDate.format('yyyy-MM-dd');
                      }
                    }
                  },
                ),
                const SizedBox(
                  height: 20,
                ),
                TextButton(
                  child: const Text('OK'),
                  onPressed: () async {
                    final currentState = formGlobalKey.currentState;
                    if (currentState == null) {
                      return;
                    }
                    if (currentState.validate()) {
                      final updatedTrip = widget.trip.copyWith(
                        tripName: tripNameController.text,
                        destination: destinationController.text,
                        startDate: TemporalDate(
                          DateTime.parse(startDateController.text),
                        ),
                        endDate: TemporalDate(
                          DateTime.parse(endDateController.text),
                        ),
```

```
                    );

                    await ref
                        .watch(
                            tripControllerProvider(widget.trip.id).notifier)
                        .updateTrip(updatedTrip);
                    if (context.mounted) {
                      context.goNamed(
                        AppRoute.trip.name,
                        pathParameters: {'id': widget.trip.id},
                        extra: updatedTrip,
                      );
                    }
                  }
                }, //,
              ),
            ],
          ),
        ),
      ),
    ),
  );
 }
}
```

3. Open the **lib/common/navigation/router/router.dart** file and update it to add the **EditTripPage** route.

```
GoRoute(
 path: '/edittrip/:id',
 name: AppRoute.editTrip.name,
 builder: (context, state) {
   return EditTripPage(
     trip: state.extra! as Trip,
   );
 },
),
```

The **router.dart** should look like the following code snippet.

```
import 'package:amplify_trips_planner/common/navigation/router/routes.dart';
import 'package:amplify_trips_planner/features/trip/ui/edit_trip_page/
edit_trip_page.dart';
```

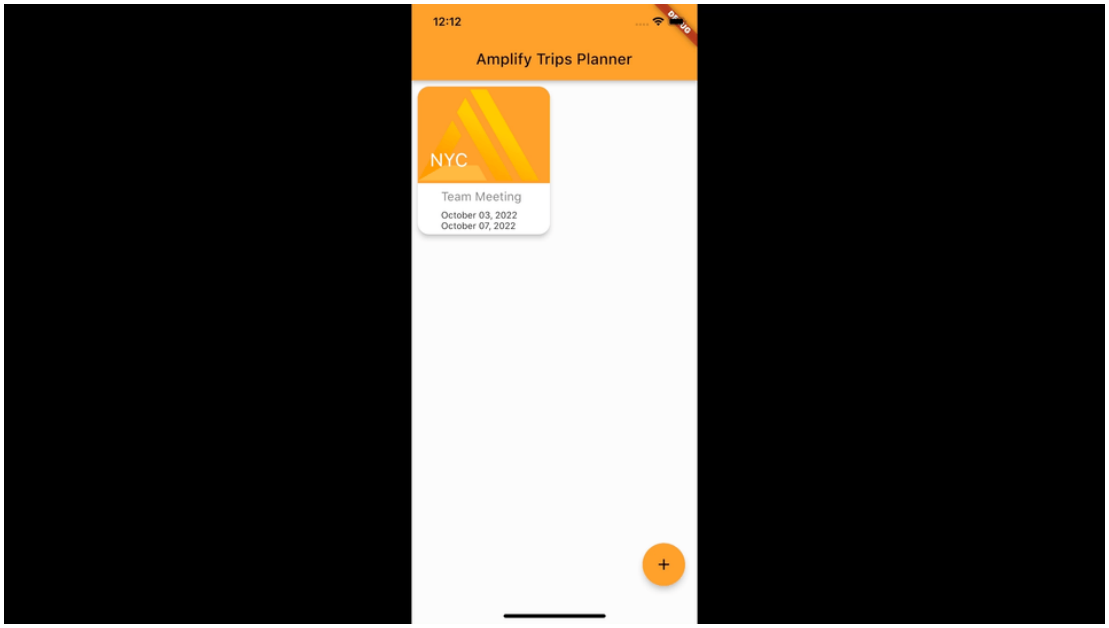
```
import 'package:amplify_trips_planner/features/trip/ui/trip_page/trip_page.dart';
import 'package:amplify_trips_planner/features/trip/ui/trips_list/
trips_list_page.dart';
import 'package:amplify_trips_planner/models/ModelProvider.dart';
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';

final router = GoRouter(
 routes: [
   GoRoute(
     path: '/',
     name: AppRoute.home.name,
     builder: (context, state) => const TripsListPage(),
   ),
   GoRoute(
     path: '/trip/:id',
     name: AppRoute.trip.name,
     builder: (context, state) {
       final tripId = state.pathParameters['id']!;
       return TripPage(tripId: tripId);
     },
   ),
   GoRoute(
     path: '/edittrip/:id',
     name: AppRoute.editTrip.name,
     builder: (context, state) {
       return EditTripPage(
         trip: state.extra! as Trip,
       );
     },
   ),
 ],
 errorBuilder: (context, state) => Scaffold(
   body: Center(
     child: Text(state.error.toString()),
   ),
 ),
);
```

4. Run the app in the simulator and try the following:

   - Create a new trip
   - Edit the newly created trip

- Upload an image for the trip
- Delete the trip

The following is an example using an iPhone simulator.



Conclusion

In this module, you used AWS Amplify to add file storage to your app using Amazon S3 so that users can upload images and view them in their app.

Congratulations!

Congratulations! You have created a cross-platform Flutter mobile app using AWS Amplify! You have added authentication to your app, allowing users to sign up, sign in, and manage their account. The app also has a scalable GraphQL API configured with an DynamoDB database, allowing users to create, read, update, and delete trips. You have also added cloud storage using Amazon S3 so that users can upload images and view them in their app.

Clean up resources

Now that you've finished this walk-through, you can delete the backend resources to avoid incurring unexpected costs by running the command below in the root folder of the app.

```
amplify delete
```