Cheat Sheet: Advanced Flutter

Calling APIs in Flutter

Type	Description	Code Example

Making Your First API Call	Learn how to perform a basic HTTP GET request	`Future<void> fetchData() async {`
		`  var response = await http.get(Uri.parse('https://api.example.com/data'));`
		`  print(response.body);`
		`}`

Handling API Errors	Implement error handling for API requests	`try {`
		`  var response = await http.get(Uri.parse('https://api.example.com/data'));`
		`  print(response.body);`
		`} catch (e) {`
		`  print('Caught error: $e');`
		`}`

Parsing JSON Data	Decode JSON data received from an API into Dart objects	`var data = jsonDecode(response.body);` `print('User name: ${data['username']}');`

Asynchronous Programming	Use async and await for asynchronous API calls	`Future<void> getUserData() async {`
		`  var response = await http.get(Uri.parse('https://api.example.com/users/1'));`
		`  var data = jsonDecode(response.body);`
		`  print('User name: ${data['name']}');`
		`}`

Post Data to API	Send data to an API using the POST method	`Future<void> postData() async {`
		`  var response = await http.post(Uri.parse('https://api.example.com/add'), body: {'name': 'John Doe'});`
		`  print('Response status: ${response.statusCode}');`
		`}`

Fetch Data with Headers	Make HTTP requests with additional headers	`Future<void> fetchWithHeaders() async {`
		`  var response = await http.get(`
		`  Uri.parse('https://api.example.com/data'),`
		`  headers: {'Authorization': 'Bearer your_token_here'}`
		`  );`
		`  print('Data with headers: ${response.body}');`
		`}`

Introduction to mobile native features in Flutter

Type	Description	Code Example

Accessing the Camera	Utilize the native camera functionality via the Flutter plugin	`Future<void> takePicture() async {`
		`  final cameras = await availableCameras();`
		`  final firstCamera = cameras.first;`
		`  final CameraController controller = CameraController(firstCamera, ResolutionPreset.high);`
		`  await controller.initialize();`
		`  final image = await controller.takePicture();`
		`  controller.dispose();`
		`}`

Using GPS	Fetch the current location using the geolocation plugin	`Position position = await`
		`Geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.high);`
		`print('Current location: ${position.latitude}, ${position.longitude}');`

Local Storage	Store data locally using the shared_preferences plugin	`SharedPreferences prefs = await`
		`SharedPreferences.getInstance();`
		`await prefs.setString('username', 'exampleUser');`

Accessing Device Sensors	Interact with native device sensors like the accelerometer	`StreamSubscription<dynamic> subscription = AccelerometerEvents.listen((AccelerometerEvent event) {`
		`  print(event);`
		`});`

Managing Notifications	Schedule and manage notifications locally on the device	`FlutterLocalNotificationsPlugin notificationsPlugin = FlutterLocalNotificationsPlugin();`
		`var androidDetails = AndroidNotificationDetails('channelId', 'channelName', 'channelDescription');`
		`var generalNotificationDetails = NotificationDetails(android: androidDetails);`
		`await notificationsPlugin.show(0, 'Test Title', 'Test Body', generalNotificationDetails);`

Managing state in Flutter

Type	Description	Code Example

Manage local widget state changes in a StatefulWidget	Provides a way to manage and update the local state of a widget	`setState(() {`
		`  _counter++;`
		`});`

InheritedWidget	Provides a way to pass data down the widget tree and allows descendants to rebuild when the data changes	`MyInheritedWidget(`
		`  data: counter,`
		`  child: ChildWidget()`
		`);`

GlobalKey and FormState	Utilize GlobalKey to access the state from anywhere in the app, commonly used with forms	`final _formKey = GlobalKey<FormState>();`
		`Form(`
		`  key: _formKey,`
		`  child: Column(`
		`  children: [`
		`  TextFormField(validator: (value) => value.isEmpty ? 'Cannot be empty' : null),`
		`  ElevatedButton(`
		`  onPressed: () {`
		`  if (_formKey.currentState.validate()) {`
		`  // Process data`
		`  }`
		`  },`
		`  child: Text('Submit')`
		`  )`
		`  ],`
		`  )`
		`);`

Provider for state management	A wrapper around InheritedWidget to make state management easier and more	`ChangeNotifierProvider(`
		`  create: (context) => DataModel(),`
		`  child: Consumer<DataModel>(`
		`  builder: (context, dataModel, child) {`
		`  return Text('${dataModel.value}');`
		`  },`
		`  ),`
		`);`

	efficient	

Flutter Persistence with Local Storage

Type	Description	Code Example

SharedPreferences	Ideal for simple data such as user preferences and settings	`final prefs = await SharedPreferences.getInstance();`
		`prefs.setInt('counter', 10);`
		`int counter = prefs.getInt('counter') ?? 0;`

SQLite	Ideal for structured data and complex queries. It works like a traditional database	
Files	Read/write files for documents or media	
Third-Party Packages	**Hive**: A fast, NoSQL database for Flutter. **LocalStorage**: Similar to web local storage, ideal for storing JSON data.	
Initialize Local Storage	Set up the chosen storage method at the start of your app.	`final prefs = await SharedPreferences.getInstance();`

CRUD Operations	Implement Create, Read, Update, and Delete functions.	`// Saving data`
		`prefs.setInt('counter', 10);`
		`// Retrieving data`
		`int counter = prefs.getInt('counter') ?? 0;`
		`// Deleting data`
		`prefs.remove('counter');`

Serialization/Deserialization	Encode and decode data using JSON for complex structures	`import 'dart:convert';`
		`Map userMap = jsonDecode(jsonString);`
		`var user = User.fromJson(userMap);`

		`class UserProvider with ChangeNotifier {`
		`  User _user;`
		`  User get user => _user;`
		`  void loadUser() {`

Integrating with State Management	Use state management tools (e.g., Provider, Riverpod) to handle local storage updates dynamically in the UI	`String userJson = storage.getItem('user');`
		`_user = User.fromJson(jsonDecode(userJson));`
		`notifyListeners();`
		`}`
		`}`

Plugins in Flutter

Type	Description	Code Example

Using the url_launcher Plugin	Open a URL in the mobile browser from a Flutter app	`if (await canLaunch(url)) {`
		`await launch(url);`
		`} else {`
		`throw 'Could not launch $url';`
		`}`

Managing Plugin Compatibility	Ensure that plugins are compatible with the version of Flutter you are using	`// Typically handled in your pubspec.yaml by specifying compatible versions`
		`dependencies:`
		`flutter:`
		`sdk: flutter`
		`url_launcher: ^6.0.3`

Camera Plugin	Access the device camera to capture photos and videos	`final cameras = await availableCameras();`
		`final firstCamera = cameras.first;`
		`final cameraController = CameraController(firstCamera, ResolutionPreset.medium);`
		`await cameraController.initialize();`

Audio Players Plugin	Play audio files and streams in Flutter with the Audio Players plugin	`final player = AudioPlayer();` `await player.play(UrlSource('https://example.com/audio.mp3'));`

		`GoogleMap(`
		`onMapCreated: _onMapCreated,`
		`initialCameraPosition: CameraPosition(`
		`target: LatLng(0.0, 0.0),`
		`zoom: 10.0,`

Flutter Maps Plugin	Integrate maps into your Flutter applications using the maps plugin	),);

Skills Network