Cross-Platform Development

Mobile Dev Guide · Module 8 of 8 · CHERIEDU Dev Series

1. The Cross-Platform Promise

Cross-platform development lets you write code once and deploy to iOS, Android, and often web and desktop too. This reduces cost by up to 60% vs building separate native apps.

2. Flutter vs React Native — Detailed Comparison

Factor	Flutter	React Native
Language	Dart	JavaScript / TypeScript
Rendering	Own rendering engine (Skia/Impeller)	Native platform components
Performance	Near-native	Good, slight overhead
UI Look	Consistent across platforms	Follows native OS style
Hot Reload	Yes	Yes
Web support	Yes (experimental)	Partial (React Native Web)
Desktop support	Yes (Windows, macOS, Linux)	Limited
Company	Google	Meta (Facebook)
Learning curve	Learn Dart first	Easier for JS devs

3. Kotlin Multiplatform (KMM)

- Share business logic (API calls, data models, use cases) between iOS and Android.
- Keep native UI on both platforms for the best native look and feel.
- Growing adoption at companies like Netflix, VMware, and Philips.
- Best of both worlds: shared logic + native performance.

4. Platform-Specific Code in Flutter

```
import "dart:io" show Platform;

Widget buildBackButton() {
  if (Platform.isIOS) {
    return CupertinoButton(child: Text("Back"), onPressed: () {});
  }
  return ElevatedButton(child: Text("Back"), onPressed: () {});
}
```

5. Cross-Platform Architecture Best Practices

1. Separate business logic from UI — use service/repository pattern.

2. Abstract platform APIs behind interfaces — swap implementations per platform.
3. Test on real devices for both iOS and Android — emulators miss real issues.
4. Use platform channels for native features not in Flutter SDK.
5. Keep platform-specific code in clearly named files (e.g. camera_ios.dart).

CAPSTONE PROJECT
Build the CHERI SMS mobile app for both iOS and Android using Flutter. Implement: Multi-role login (Admin/Teacher/Parent/Student), Dashboard, Fee Management, Attendance, Notifications. Deploy to both app stores.