

Flutter App Development

Course Duration ~75 Hours · 10 Weeks · Live Online · Build-Along · Apps Shipped to Store-Ready Builds

Curriculum reviewed: July 2026 — Flutter moves fast and its web coverage is noisy; this syllabus is checked against official documentation

Dart & Sealed Types

State Management

Platform Integration

Golden Testing

Performance

Store Shipping

In short: One codebase, every screen. Flutter renders every pixel itself rather than using platform widgets — pixel-perfect consistency across mobile, web and desktop, and the fastest path from zero to a polished app on all targets. This course teaches the platform as it works now, not as 2021 tutorials describe it: the modern rendering engine that fixed early shader-compilation jank (default on current mobile targets, with the older renderer only as a fallback on older devices), web via WebAssembly compilation as a genuine target with automatic JavaScript fallback, cheaper Dart-to-native interop, and modern Dart — sealed classes, pattern matching, exhaustive switches — which most tutorials predate. We're accurate about the ecosystem: language-level data classes did not ship, so code generation remains the current reality, and we teach the tooling teams actually use, plus type-safe platform channel generation instead of hand-written plumbing. The comparison is honest — Flutter wins on UI consistency, speed and breadth of targets; Kotlin Multiplatform wins where native UX is non-negotiable; React Native wins where the team is already React — and so is the Dart question: the transferable skill is product engineering, and developers from Java, Kotlin, JavaScript, C# or Swift get comfortable in a few weeks. Across 10 modules you ship an offline-first app, a sealed-types state workshop, an adaptive interface, platform integrations across both stores' privacy rules, a golden-test suite, a DevTools-evidenced performance rescue and a store-ready capstone. The AI question, answered honestly: AI writes widget trees fluently and gets rebuild scope, state structure, async disposal, platform behaviour and accessibility wrong — those surface as jank, battery drain and store rejections, not compile errors.

Duration	10 weeks · ~75 hours · live + hands-on	Format	Live online · build-along · apps shipped to store-ready builds
Prerequisite	Programming experience in any language. Dart taught from first principles. No mobile development background required.	Curriculum	10 modules, from Dart and the widget model to release, store and capstone
Portfolio	8 production-style builds plus a store-ready capstone	Ideal for	Fresh IT graduates, web developers going mobile, native developers adding cross-platform reach, and freelancers shipping complete products

Why Learn Flutter When AI Writes Widget Trees?

AI generates Flutter widget trees fluently and gets the things that matter wrong — rebuild scope and unnecessary widget rebuilds, state management structure, async lifecycle and disposal, platform-specific behaviour, and accessibility. Those show up as jank, battery drain and store rejections rather than as compile errors. The value is architecture, performance judgment and shipping through app review. You'll use AI tooling as an accelerator you review, and learn the failure patterns generated Flutter code repeats — so you ship its speed without shipping its mistakes.

Course Highlights

- Every module leads with what you build and ship, not what you "learn about"
- Modern Dart first: sealed classes and pattern matching so impossible states can't compile
- State management compared honestly, with clear recommendations by project size
- Accessibility as a build requirement — and a store-review reality
- Code generation taught as the current reality for data classes and serialisation
- Type-safe platform channels — no hand-written channel plumbing
- Performance with evidence: rebuild profiler, DevTools timeline, golden tests in CI
- Store submission taught properly — the review pitfalls that cause rejections
- AI tooling as an accelerator you review, never a substitute for understanding

Hands-on Projects & Portfolio

- **Multi-Screen App** — authentication, deep linking and a typed API layer
- **Offline-First Application** — local persistence, background sync and conflict handling
- **State-Management Workshop** — model a complex feature with sealed types so invalid states can't compile
- **Adaptive Interface** — works properly on phone, tablet and desktop from one codebase
- **Platform-Integration Feature** — camera, notifications and permissions across both stores
- **Animation & Polish Exercise** — turn a functional screen into one that feels premium
- **Performance Rescue** — diagnose and fix jank and excessive rebuilds with DevTools evidence
- **Golden-Test Suite** — catch visual regressions in CI before users see them
- **Capstone** — production app, tested, profiled and submitted as a store-ready build

Tools & Stack

Flutter · Dart · Material & Cupertino · Riverpod · Bloc · go_router · Dio · Freezed · build_runner · Pigeon · Firebase · SQLite / Drift · Isar · Flutter DevTools · mocktail · golden_toolkit · Fastlane · Codemagic or GitHub Actions · Sentry · Git

Career Outcomes

Flutter's commercial appeal is that one developer or one small team can deliver a polished application to both app stores — and increasingly to web and desktop — from a single codebase, which is why it is heavily represented in startups, agencies, product studios and internal enterprise apps. That makes it unusually strong for freelance and independent work, where the ability to ship a complete product alone converts directly into billable capability. This course targets:

- Flutter Developer · Mobile App Developer · Cross-Platform Developer · Product Engineer
- Frontend Engineer (multi-platform) · Freelance or agency app developer · Technical founder shipping a mobile product

Who This Course Is For

- Final-year students and fresh IT graduates targeting mobile development roles
- Web developers moving into mobile without learning two native stacks
- **Native Android or iOS developers adding cross-platform reach**
- Backend developers who need to ship a front end for their own product
- Freelancers and agency developers who bill by delivered products
- Founders and small teams shipping to both stores with limited resources

Module 1: Dart & the Flutter Model

8 Hours

YOU SHIP

- Your first running app across hot-reload cycles — with Dart fundamentals proven in typed, null-safe, pattern-matched code, not slides

TOPICS

- Dart for developers who already program; null safety; futures, streams and async; sealed classes and pattern matching; records
- The widget tree and element tree; declarative UI as a mental shift from imperative view code
- The toolchain; hot reload; how a frame is actually produced

Assessment: Quiz + Lab

Module 2: Building Interfaces

8 Hours

YOU SHIP

- An adaptive, themed, accessible interface that works properly on phone, tablet and desktop from one codebase — with the layout-constraints model finally making sense

TOPICS

- Layout and constraints — the model that confuses everyone at first; composition over configuration; stateless and stateful widgets
- Material and Cupertino design systems; theming and design tokens; responsive and adaptive layouts across phone, tablet, foldable and desktop
- Custom painting; accessibility as a build requirement rather than an afterthought

Assessment: Quiz + Lab**Module 3: State Management**

8 Hours

YOU SHIP

- The state-management workshop: a complex feature modelled with sealed types so invalid states can't compile — plus the same feature in two ecosystem solutions with a written recommendation by project size

TOPICS

- Local state and lifecycle; lifting state; inherited widgets
- The ecosystem solutions in depth — honest comparison, clear recommendation for different project sizes
- Modelling UI state with sealed types; side effects; disposal without leaks

Assessment: Quiz + Lab**Module 4: Navigation & App Architecture**

7 Hours

YOU SHIP

- A feature-structured app with declarative routing, deep links, authenticated route guards and injected dependencies — an architecture that survives a second developer joining

TOPICS

- Declarative routing; deep links and app links; nested and tabbed navigation; guards and authenticated routes
- Project structure by feature; separation of UI from domain from data
- Dependency injection; architecture that scales

Assessment: Quiz + Lab**Module 5: Data & Backend Integration**

8 Hours

YOU SHIP

- An offline-first application: typed API layer, code-generated serialisation, secure token storage, local persistence, background sync and conflict handling

TOPICS

- HTTP clients and typed API layers; serialisation with the current code-generation tooling; error and retry handling
- Authentication and secure token storage; local persistence and offline-first design; caching and sync conflict handling
- Real-time updates; integrating with Firebase or your own service

Assessment: Quiz + Lab**Module 6: Animation & Polish**

7 Hours

YOU SHIP

- The polish exercise: a functional screen turned into one that feels premium — transitions, hero motion, physics-based gestures and considered micro-detail

TOPICS

- Implicit and explicit animations; transitions and hero motion; gestures
- Custom animated widgets; physics-based motion; shaders and visual effects
- The craft details that make an app feel considered rather than functional

Assessment: Quiz + Lab

Module 7: Platform Integration

8 Hours

YOU SHIP

- A platform-integration feature — camera, notifications and permissions handled across both stores' privacy rules — with type-safe generated channels, no hand-written plumbing

TOPICS

- Plugins and packages; type-safe platform channel generation; calling native and C code
- Camera, location, notifications, background work; permissions and the privacy prompts each store enforces
- Biometrics; in-app purchases; handling platform differences without forking the codebase

Assessment: Quiz + Lab**Module 8: Testing & Quality**

7 Hours

YOU SHIP

- A fully tested feature — unit, widget and integration coverage plus a golden-test suite catching visual regressions — running as CI gates for a mobile project

TOPICS

- Unit tests, widget tests, integration tests; golden/screenshot testing for visual regressions
- Mocking network and platform dependencies; coverage that means something
- Static analysis and lint rules; CI for a mobile project

Assessment: Quiz + Lab**Module 9: Performance**

7 Hours

YOU SHIP

- The performance rescue: jank and excessive rebuilds diagnosed and fixed with rebuild-profiler and DevTools timeline evidence — plus measured app-size and startup improvements

TOPICS

- The rendering pipeline and where frames get dropped; minimising rebuild scope; the widget rebuild profiler and DevTools timeline
- Image and asset strategy; list virtualisation; memory and leak hunting
- App size reduction; startup time; diagnosing jank with evidence rather than guesswork

Assessment: Quiz + Lab**Module 10: Release, Store & Capstone**

7 Hours

YOU SHIP

- Your capstone: one production-grade app — tested, profiled, flavoured, signed for both platforms — submitted as a store-ready build, with web and desktop targets demonstrated from the same codebase

TOPICS

- Build flavours and environment configuration; signing for both platforms; App Store and Play Store submission and the review pitfalls that cause rejections
- Staged rollouts; over-the-air update options and their policy limits; crash reporting and analytics; versioning and release process
- Targeting web and desktop from the same codebase; capstone design, build, test and ship

Assessment: Capstone Project + Store-Ready Build