Admissions & Support:
support@mockattempt.com
Phone / WhatsApp: +91 95400 07917
Course Code: FLUT • **Updated:** 05 Nov 2026
https://mockattempt.com/courses/flutter-training

FLUT • CROSS-PLATFORM MOBILE & DESKTOP ENGINEERING | **BEGINNER TO SENIOR FLUTTER & DART 3 ARCHITECT**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

Flutter Training

Master cross-platform Flutter 3 & Dart 3 from the 3-Tree Architecture (Widget, Element, RenderObject) and Impeller shader engine to BoxConstraints, Scaffold, Row/Column/Stack, TextEditingController, GlobalKey Forms, ListView.builder, GridView, TabBar, Gestures & Riverpod/BLoC State Management — with 244 Topic Mock Tests (10,440 MCQs) and Spoken Voice + MockAttempt AI Interviews at ₹3,799 (Incl. 18% GST).

Duration: 15–17 Weeks • Live + Hands-On Projects (208 Hrs) • **Modules:** 16 • **Live Topics:** 42 • **Subtopics:** 336+ • **Labs & Tasks:** 84 • **Mocks / AI Vivas:** 244 / 244 • **Projects:** 16 Module + 1 Live Client Prod

All-Inclusive Direct Fee (Incl. 18% GST • Zero Hidden Charges): Total Payable: ₹3,799 (Net Tuition ₹3,219 + Included 18% GST ₹580) • **Split Option:** 2 × ₹1,900 Split • **Effective ROI:** ~₹18/Guided Hr (~₹90/Live Topic)

Faculty: 100% Active Working IT Professionals • **Live Batch Tracks (7:00 AM – 8:00 PM IST Flexible Slots):** Weekday Live Online Batch (Mon – Fri • 1 Hr/Day • 7:00 AM – 8:00 PM) (Monday to Friday (1 Hour Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 1-Hour Slot Based on Trainer Availability) • 17–19 Weeks Fast-Track) | Weekend Live Online Batch (Sat – Sun • 2 Hrs/Day • 7:00 AM – 8:00 PM) (Saturday & Sunday (2 Hours Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 2-Hour Slot Based on Trainer Availability) • 19–21 Weeks Weekend Track)

✓ **Trainer LeetCode & HackerRank Mentorship (Dart 3): Live trainer support to crack LeetCode & HackerRank problems, optimize complexity, and handle edge cases.**

Core Technologies & Engineering Stack: Flutter 3.24+ & Material 3 • Dart 3 (Records, Sealed Classes, Null Safety) • Impeller GPU Rendering Engine • Widget / Element / RenderObject Trees • Slivers, ListView.builder & GridView • Forms, TextEditingController & Gestures • Riverpod, Provider & BLoC State Architecture

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Understand Flutter’s Impeller AOT shader pipeline, Dart JIT Hot Reload vs AOT ARM64 machine code, and the 3-Tree reconciliation engine
2. Master Flutter’s golden layout rule (Constraints go down, Sizes go up, Parent sets position) across Scaffold, Row, Column, Expanded, and Stack
3. Build leak-free forms, lazy-built 10,000-row ListView/GridView feeds, gesture arenas, and reactive unidirectional state with ValueNotifier/Riverpod
4. Clear 244 Timed Mock Exams (10,000 Flutter & Dart MCQs) and 200 Spoken Voice + MockAttempt AI Cross-Platform Technical Interviews
5. Dart 3 null-safety, records, pattern matching, and sealed state classes for crash-free cross-platform code.
6. Crack LeetCode and HackerRank problems in Dart 3 with live trainer mentorship on algorithmic/query patterns, edge cases, and optimal complexity.
7. Scalable feature-first architecture with Riverpod / BLoC state management and pure repository abstractions.
8. Custom responsive layouts, Slivers, implicit/explicit animations, and platform-adaptive Material/Cupertino UX.

Eligibility & Prerequisites: Basic programming familiarity (variables, loops, functions) — Dart 3 sound null safety, async/await, and OOP are taught from scratch • No prior Flutter industry experience required — Module 1 starts from environment setup, core mental models, and plain-English fundamentals. • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Commitment to attend live Weekday or Weekend classes, complete daily hands-on lab checklists, and practice topic-wise Mock Tests & AI Interviews. • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards.

2. DETAILED ZERO-TO-EXPERT MODULE, TOPIC & SUBTOPIC SYLLABUS (16 MODULES • 42 LIVE TOPICS • 336+ SUBTOPICS)

Progressive Zero-to-Expert Mastery Stages: Modules 1–4: Stage 1: Day-1 Foundations & Core Mental Models (8 Topics / 64 Subtopics) • Modules 5–8: Stage 2: Applied Engineering, Data & API Workflows (8 Topics / 64 Subtopics) • Modules 9–12: Stage 3: Internals, Concurrency, Security & Performance (12 Topics / 96 Subtopics) • Modules 13–16: Stage 4: Production Architecture, Testing, CI/CD & Expert Mastery (14 Topics / 112 Subtopics)

MODULE 1 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 1: Flutter Cross-Platform Paradigm, Impeller Engine & SDK Toolchain Setup

Understand why Flutter draws every pixel directly onto a GPU canvas via Impeller (instead of OEM native UI bridges), Dart's dual JIT/AOT compiler, and reproducible `flutter doctor` + `pubspec.yaml` setup.

Objectives: ✓ Flutter Overview — Own-Canvas GPU Rendering (Impeller), Dart JIT/AOT & Multi-Platform Targets — Why Flutter bypasses OEM platform widgets to paint directly via Impeller/Skia, eliminating shader-compilation jank while pairing sub-second Stateful Hot Reload with native ARM64 release speed. • ✓ Flutter Installation & Toolchain — `flutter doctor`, `pubspec.yaml`, Assets & FVM — Set up the Flutter SDK cleanly across Windows/macOS/Linux, diagnose Android SDK `cmdline-tools` & licenses via `flutter doctor --android-licenses`, and manage dependencies and bundled assets in `pubspec.yaml`. • ✓ Build, verify, and ship Module Capstone: Flutter Environment Readiness Auditor & Pubspec SemVer Resolver.

1.1 Flutter Overview — Own-Canvas GPU Rendering (Impeller), Dart JIT/AOT & Multi-Platform Targets

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Impeller precompiled shaders & Dart JIT Hot Reload vs AOT ARM64 • Core Syntax, CLI & Production APIs: runApp(const MyApp()) + Impeller AOT Shaders + flutter build apk --release (AOT ARM64) • Deep-Dive Mechanics: Debug vs Profile vs Release modes & Zero OEM widget bridge • Advanced Patterns: Platform Channels for native APIs & 120Hz 8.3ms raster budget • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Dart 3 Build Mode & Impeller Frame Budget Evaluator • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter Overview

Purpose: To understand and apply Flutter Overview — Own-Canvas GPU Rendering (Impeller), Dart JIT/AOT & Multi-Platform Targets in real Flutter systems — specifically: Why Flutter...

What Topic Is Next → Next Topic 1.2: Flutter Installation & Toolchain — `flutter doctor`, `pubspec.yaml`, Assets & FVM

1.2 Flutter Installation & Toolchain — `flutter doctor`, `pubspec.yaml`, Assets & FVM

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Flutter doctor --android-licenses & Pubspec.yaml 2-space YAML schema • Core Syntax, CLI & Production APIs: flutter doctor -v + flutter pub get + pubspec.yaml (environment, dependencies, flutter.asset... • Deep-Dive Mechanics: Pubspec.lock app reproducibility & PubGrub version solver • Advanced Patterns: 2.0x / 3.0x resolution-aware assets & Flutter Version Management (FVM) • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Resolution-Aware Asset Variant Selector (1.0x / 2.0x / 3.0x) • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter Installation

Purpose: To understand and apply Flutter Installation & Toolchain — `flutter doctor`, `pubspec.yaml`, Assets & FVM in real Flutter systems — specifically: Set up the Flutter SDK c...

What Topic Is Next → Module 1 Capstone (Flutter Environment Readiness Auditor & Pubspec SemVer Resolver) → Next Topic 2.1: First Flutter Application — `main()`, `runApp()`, `MaterialApp` & `const` Constructors

Module 1 Hands-On Capstone Project (100 Marks): Flutter Environment Readiness Auditor & Pubspec SemVer Resolver — Build a Dart 3 CLI tool that validates toolchain health checks (`Flutter SDK`, `Android Toolchain`, `Xcode/Cocoapods`, `VS Code`), verifies sound null-safety SDK constraints (`>=3.0.0 <4.0.0...

MODULE 2 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 2: First Flutter App Anatomy & The 3-Tree Architecture (`Widget`, `Element`, `RenderObject`)

Dissect `main() -> runApp()`, `MaterialApp(theme: ThemeData(useMaterial3: true))`, and Flutter's core engine architecture: immutable `Widget` blueprints, mutable `Element` mounts, and `RenderObject` painting.

Objectives: ✓ First Flutter Application — `main()`, `runApp()`, `MaterialApp` & `const` Constructors — Bootstrap a Material 3 Flutter application from `lib/main.dart`, configure `ColorScheme.fromSeed()`, hide the debug banner cleanly, and leverage canonicalized `const` widget instances. • ✓ Flutter Architecture — Framework (Dart), Engine (C++/Impeller), Embedder & The 3 Trees — Explore the 3 architectural layers (Embedder, C++ Engine, Dart Framework) and the 3 parallel runtime trees: immutable `Widget` configurations, lifecycle-owning `Element`s, and layout/paint `RenderObject`s. • ✓ Build, verify, and ship Module Capstone: 3-Tree Reconciliation Simulator (`Widget.canUpdate` & `const` Short-Circuiting).

2.1 First Flutter Application — `main()`, `runApp()`, `MaterialApp` & `const` Constructors

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: RunApp() root inflation & WidgetsFlutterBinding.ensureInitialized() • Core Syntax, CLI & Production APIs: void main() => runApp(const MyApp()); MaterialApp(theme: ThemeData(colorSchemeSeed: ...)) • Deep-Dive Mechanics: MaterialApp & Navigator host & ThemeData Material 3 ColorScheme • Advanced Patterns: Const canonicalization & BuildContext = Element handle • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Canonical `const` Widget Identity & Material 3 Seed Verifier • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for First Flutter Application

Purpose: To understand and apply First Flutter Application — `main()`, `runApp()`, `MaterialApp` & `const` Constructors in real Flutter systems — specifically: Bootstrap a Materia...

What Topic Is Next → Next Topic 2.2: Flutter Architecture — Framework (Dart), Engine (C++/Impeller), Embedder & The 3 Trees

2.2 Flutter Architecture — Framework (Dart), Engine (C++/Impeller), Embedder & The 3 Trees

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Widget vs Element vs RenderObject & Widget.canUpdate(old, new) rule • Core Syntax, CLI & Production APIs: Widget.canUpdate(old, new) + Element + RenderObject + MethodChannel / Pigeon • Deep-Dive Mechanics: ValueKey stateful list reordering & UI Thread vs Raster Thread • Advanced Patterns: Context.mounted async guard & Embedder + Engine + Framework layers • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Flutter `Widget.canUpdate` Reconciliation Evaluator • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter Architecture

Purpose: To understand and apply Flutter Architecture — Framework (Dart), Engine (C++/Impeller), Embedder & The 3 Trees in real Flutter systems — specifically: Explore the 3 archi...

What Topic Is Next → Module 2 Capstone (3-Tree Reconciliation Simulator (`Widget.canUpdate` & `const` Short-Circuiting)) → Next Topic 3.1: Dart 3 Programming — Sound Null Safety, Records, Sealed Classes, `Future`/`Stream` & `Isolate.run`

Module 2 Hands-On Capstone Project (100 Marks): 3-Tree Reconciliation Simulator (`Widget.canUpdate` & `const` Short-Circuiting) — Implement an educational simulation of Flutter's `Element.updateChild()` reconciliation algorithm demonstrating how matching `runtimeType` + `key` updates an existing `Element` in place whil...

MODULE 3 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 3: Dart 3 Language Mastery & Stateless vs Stateful Widget Lifecycles

Write expressive sound null-safe Dart 3 code (Records, Pattern Matching, Sealed Classes, Isolates, Futures/Streams) and master `StatelessWidget` vs `StatefulWidget` (`initState`, `didUpdateWidget`, `setState`, `dispose`).

Objectives: ✓ Dart 3 Programming — Sound Null Safety, Records, Sealed Classes, `Future`/`Stream` & `Isolate.run` — Master modern Dart 3: compiler-guaranteed sound null safety, destructured Records, algebraic `sealed` types, single-threaded event loop (`Future`, `async`/`await`, `Stream`), and true parallel `Isolate.run()`. • ✓ Flutter Widgets — `StatelessWidget` vs `StatefulWidget` (`initState`, `didUpdateWidget`, `dispose`) — Understand why `Widget` fields are always `final` (immutable), how `StatefulWidget` delegates mutable state to a long-lived `State<T>` object, and how `setState()` marks an Element dirty for the next frame. • ✓ Build, verify, and ship Module Capstone: Dart 3 Sealed Quiz Result Engine & Lifecycle-Safe Stateful Counter Model.

3.1 Dart 3 Programming — Sound Null Safety, Records, Sealed Classes, `Future`/`Stream` & `Isolate.run`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Sound Null Safety (?, ??, ?.) & Dart 3 Records & destructuring • Core Syntax, CLI & Production APIs: sealed class + ({int a, String b}) Records + Future<T> / Stream<T> + Isolate.run() • Deep-Dive Mechanics: Sealed class exhaustive switch & Event Loop: Microtask vs Event queue • Advanced Patterns: Isolate.run() parallel compute & Avoiding ! bang operator crashes • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Dart 3 Sealed State & Named Record Destructuring • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Dart 3 Programming

Purpose: To understand and apply Dart 3 Programming — Sound Null Safety, Records, Sealed Classes, `Future`/`Stream` & `Isolate.run` in real Flutter systems — specifically: Master...

What Topic Is Next → Next Topic 3.2: Flutter Widgets — `StatelessWidget` vs `StatefulWidget` (`initState`, `didUpdateWidget`, `dispose`)

3.2 Flutter Widgets — `StatelessWidget` vs `StatefulWidget` (`initState`, `didUpdateWidget`, `dispose`)

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Immutable @immutable Widget fields & State<T> persistent lifecycle • Core Syntax, CLI & Production APIs: StatelessWidget + StatefulWidget + State<T> (initState, didUpdateWidget, setState, dispose) • Deep-Dive Mechanics: InitState vs didChangeDependencies & DidUpdateWidget(oldWidget) sync • Advanced Patterns: SetState() markNeedsBuild() & Mounted check before setState() • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: StatefulWidget Lifecycle Trace & `mounted` Async Guard • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter Widgets

Purpose: To understand and apply Flutter Widgets — `StatelessWidget` vs `StatefulWidget` (`initState`, `didUpdateWidget`, `dispose`) in real Flutter systems — specifically: Unders...

What Topic Is Next → Module 3 Capstone (Dart 3 Sealed Quiz Result Engine & Lifecycle-Safe Stateful Counter Model) → Next Topic 4.1: Flutter Layouts — `BoxConstraints` ("Constraints Down, Sizes Up"), `SizedBox` & `LayoutBuilder`

Module 3 Hands-On Capstone Project (100 Marks): Dart 3 Sealed Quiz Result Engine & Lifecycle-Safe Stateful Counter Model — Model asynchronous quiz evaluation states (`Loading`, `Completed`, `Failure`) using a Dart 3 `sealed class` with Record destructuring, and pair it with a lifecycle guard preventing `setState...

MODULE 4 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 4: Flutter's Golden Layout Rule (`BoxConstraints`) & Material `Scaffold` Anatomy

Internalize Flutter's single-pass layout protocol ("Constraints go down. Sizes go up. Parent sets position.") with `Container`, `Padding`, `SizedBox`, `LayoutBuilder`, and `Scaffold` (`AppBar`, `SafeArea`, `SnackBar`).

Objectives: ✓ Flutter Layouts — `BoxConstraints` ("Constraints Down, Sizes Up"), `SizedBox` & `LayoutBuilder` — How Flutter achieves O(N) single-pass layout: parents pass `BoxConstraints` down, children return a valid `Size` up, and parents position children via `BoxParentData.offset`. • ✓ Flutter Scaffold — `AppBar`, `SafeArea`, `NavigationBar`, `resizeToAvoidBottomInset` & `ScaffoldMessenger` — Assemble complete Material 3 screens with `Scaffold`: camera-notch `SafeArea` insets, keyboard inset avoidance (`resizeToAvoidBottomInset`), `FloatingActionButton`, and route-surviving `ScaffoldMessenger` SnackBars. • ✓ Build, verify, and ship Module Capstone: BoxConstraints Solver & Adaptive Mobile/Tablet Scaffold Shell.

4.1 Flutter Layouts — `BoxConstraints` ("Constraints Down, Sizes Up"), `SizedBox` & `LayoutBuilder`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Constraints go down, Sizes go up & Tight vs loose BoxConstraints • Core Syntax, CLI & Production APIs: BoxConstraints + LayoutBuilder + Padding + SizedBox + ConstrainedBox + AspectRatio • Deep-Dive Mechanics: Center/Align loosening tight bounds & Const SizedBox spacing optimization • Advanced Patterns: LayoutBuilder local constraints & MediaQuery.sizeOf targeted subscription • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Flutter `BoxConstraints.constrain` Single-Pass Size Clamper • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter Layouts

Purpose: To understand and apply Flutter Layouts — `BoxConstraints` ("Constraints Down, Sizes Up"), `SizedBox` & `LayoutBuilder` in real Flutter systems — specifically: How Flutte...

What Topic Is Next → Next Topic 4.2: Flutter Scaffold — `AppBar`, `SafeArea`, `NavigationBar`, `resizeToAvoidBottomInset` & `ScaffoldMessenger`

4.2 Flutter Scaffold — `AppBar`, `SafeArea`, `NavigationBar`, `resizeToAvoidBottomInset` & `ScaffoldMessenger` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Scaffold CustomMultiChildLayout & SafeArea notch & home-indicator padding • Core Syntax, CLI & Production APIs: Scaffold(appBar:, body:, bottomNavigationBar:, floatingActionButton:) + ScaffoldMessenger.of... • Deep-Dive Mechanics: ScaffoldMessenger SnackBar queue & NavigationBar Material 3 destinations • Advanced Patterns: ResizeToAvoidBottomInset IME handling & Context above Scaffold pitfall • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Adaptive Material 3 `Scaffold` Breakpoint & SafeArea Calculator • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter Scaffold

Purpose: To understand and apply Flutter Scaffold — `AppBar`, `SafeArea`, `NavigationBar`, `resizeToAvoidBottomInset` & `ScaffoldMessenger` in real Flutter systems — specifically:...

What Topic Is Next → Module 4 Capstone (BoxConstraints Solver & Adaptive Mobile/Tablet Scaffold Shell) → Next Topic 5.1: Row & Column (`Flex`) — `MainAxis` vs `CrossAxis`, `Expanded` vs `Flexible` & Overflow Prevention

Module 4 Hands-On Capstone Project (100 Marks): BoxConstraints Solver & Adaptive Mobile/Tablet Scaffold Shell — Build a `BoxConstraints` clamping solver (`minWidth`, `maxWidth`, `minHeight`, `maxHeight`) paired with a responsive `LayoutBuilder` breakpoint selector switching between a mobile bottom `Na...

MODULE 5 OF 16 • STAGE 2 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 5: Flex Layouts (`Row`, `Column`, `Expanded`, `Flexible`) & Layered `Stack` Positioning

Eliminate Flutter's notorious yellow-and-black `RenderFlex overflowed` stripes and `Vertical viewport was given unbounded height` errors using `Expanded`, `Flexible`, `Wrap`, and `Stack` + `Positioned`.

Objectives: ✓ Row & Column (`Flex`) — `MainAxis` vs `CrossAxis`, `Expanded` vs `Flexible` & Overflow Prevention — Master Flutter's two-step `RenderFlex` algorithm: first measure non-flex children with unbounded main-axis constraints, then divide remaining space among `Expanded` (`FlexFit.tight`) and `Flexible` (`FlexFit.loose`) children. • ✓ Flutter Stack & `Positioned` — Non-Positioned Sizing, `Clip.none`, Hit-Testing & `IndexedStack` — Overlap widgets along the Z-axis (painter's order: first child at bottom, last child on top), anchor overlays via `Positioned` / `Positioned.directional` (RTL-safe), and preserve offscreen tab state with `IndexedStack`. • ✓ Build, verify, and ship Module Capstone: Flex Space Allocator & `IndexedStack` State-Preserving Tab Switcher.

5.1 Row & Column (`Flex`) — `MainAxis` vs `CrossAxis`, `Expanded` vs `Flexible` & Overflow Prevention [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: MainAxis vs CrossAxis orientation & Expanded (FlexFit.tight) vs Flexible (loose) • Core Syntax, CLI & Production APIs: Row / Column + MainAxisAlignment + CrossAxisAlignment + Expanded(flex:) + Flexible + Wrap • Deep-Dive Mechanics: RenderFlex yellow/black overflow fix & Unbounded height ListView-in-Column rule • Advanced Patterns: Wrap multi-line chip flow & Avoiding O(N^2) IntrinsicHeight • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: RenderFlex Two-Phase Space & `Expanded` Flex Allocator • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Column (`Flex`)

Purpose: To understand and apply Row & Column (`Flex`) — `MainAxis` vs `CrossAxis`, `Expanded` vs `Flexible` & Overflow Prevention in real Flutter systems — specifically: Master F...

What Topic Is Next → Next Topic 5.2: Flutter Stack & `Positioned` — Non-Positioned Sizing, `Clip.none`, Hit-Testing & `IndexedStack`

5.2 Flutter Stack & `Positioned` — Non-Positioned Sizing, `Clip.none`, Hit-Testing & `IndexedStack` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Painter order Z-axis layering & Non-positioned vs Positioned sizing • Core Syntax, CLI & Production APIs: Stack(alignment:, clipBehavior:) + Positioned.directional() + Positioned.fill() + IndexedSta... • Deep-Dive Mechanics: Positioned.directional RTL safety & Clip.hardEdge vs Clip.none • Advanced Patterns: Hit-test bounds clipping rule & IndexedStack state preservation • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: RTL-Aware `Positioned.directional` Coordinate Resolver • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter Stack

Purpose: To understand and apply Flutter Stack & `Positioned` — Non-Positioned Sizing, `Clip.none`, Hit-Testing & `IndexedStack` in real Flutter systems — specifically: Overlap wi...

What Topic Is Next → Module 5 Capstone (Flex Space Allocator & `IndexedStack` State-Preserving Tab Switcher) → Next Topic 6.1: Flutter TextField — `TextEditingController`, `FocusNode`, `InputDecoration` & Memory Disposal

Module 5 Hands-On Capstone Project (100 Marks): Flex Space Allocator & `IndexedStack` State-Preserving Tab Switcher — Build a two-phase `RenderFlex` layout simulator computing exact pixel widths for fixed and `Expanded(flex: k)` children while flagging overflow pixels when fixed children exceed `maxWidth`,...

MODULE 6 OF 16 • STAGE 2 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: Interactive Text Input (`TextField`, `FocusNode`) & Material 3 Buttons

Capture user text cleanly with `TextField`, `TextEditingController`, `FocusNode`, and `InputFormatter`s, and apply Material 3's button hierarchy (`FilledButton`, `OutlinedButton`, `TextButton`, `IconButton`).

Objectives: ✓ Flutter TextField — `TextEditingController`, `FocusNode`, `InputDecoration` & Memory Disposal — Control text editing, cursor selection, password visibility toggles, input formatters, and IME keyboard actions cleanly while avoiding controller re-instantiation bugs. • ✓ Flutter Material 3 Buttons — `FilledButton`, `OutlinedButton`, `WidgetStateProperty` & Loading Locks — Choose the right Material 3 button emphasis (`FilledButton`, `FilledButton.tonal`, `ElevatedButton`, `OutlinedButton`, `TextButton`, `SegmentedButton`), style interactive states via `WidgetStateProperty`, and prevent double-submits. • ✓ Build, verify, and ship Module Capstone: Debounced Search Input Controller & Idempotent Submit Button Guard.

6.1 Flutter TextField — `TextEditingController`, `FocusNode`, `InputDecoration` & Memory Disposal [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: TextEditingController lifecycle & FocusNode & FocusScope.of(context) • Core Syntax, CLI & Production APIs: TextField(controller:, focusNode:, textInputAction:, inputFormatters:, decoration: InputDeco... • Deep-Dive Mechanics: InputDecoration Material 3 borders & FilteringTextInputFormatter • Advanced Patterns: TextInputAction.next / done & Never create controller in build() • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: OTP Digits Formatter & Controller Disposal Verifier • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter TextField

Purpose: To understand and apply Flutter TextField — `TextEditingController`, `FocusNode`, `InputDecoration` & Memory Disposal in real Flutter systems — specifically: Control text...

What Topic Is Next → Next Topic 6.2: Flutter Material 3 Buttons — `FilledButton`, `OutlinedButton`, `WidgetStateProperty` & Loading Locks

6.2 Flutter Material 3 Buttons — `FilledButton`, `OutlinedButton`, `WidgetStateProperty` & Loading Locks [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: FilledButton vs OutlinedButton hierarchy & OnPressed: null disabled semantics • Core Syntax, CLI & Production APIs: FilledButton(onPressed: isBusy ? null : _submit, style: FilledButton.styleFrom(...)) + Segme... • Deep-Dive Mechanics: WidgetStateProperty.resolveWith & 48x48dp Material tap target padded • Advanced Patterns: In-flight async double-tap guard & SegmentedButton single/multi choice • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Idempotent Async Submit Button State Machine • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter Material 3 Buttons

Purpose: To understand and apply Flutter Material 3 Buttons — `FilledButton`, `OutlinedButton`, `WidgetStateProperty` & Loading Locks in real Flutter systems — specifically: Choos...

What Topic Is Next → Module 6 Capstone (Debounced Search Input Controller & Idempotent Submit Button Guard) → Next Topic 7.1: Flutter Forms — `Form`, `GlobalKey<FormState>`, `TextFormField` Validators & `AutovalidateMode`

Module 6 Hands-On Capstone Project (100 Marks): Debounced Search Input Controller & Idempotent Submit Button Guard — Engineer a leak-safe search input state controller that sanitizes input, ignores duplicate query emissions, and pairs with an async submit button state machine that disables `onPressed` (`nu...

MODULE 7 OF 16 • STAGE 2 • 11 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: Declarative Form Validation (`GlobalKey<FormState>`) & Lazy Virtualized Lists

Validate multi-field forms cleanly using `Form`, `GlobalKey<FormState>`, and `TextFormField` validators, and render 50,000+ rows smoothly using lazy `ListView.builder` / `SliverList` virtualization.

Objectives: ✓ Flutter Forms — `Form`, `GlobalKey<FormState>`, `TextFormField` Validators & `AutovalidateMode` — Group multiple `TextFormField`s under a single `Form` ancestor, run synchronous field `validator` closures via `_formKey.currentState!.validate()`, and enable polite `AutovalidateMode.onUserInteraction`. • ✓ Flutter Lists — `ListView.builder`, `ListView.separated`, `itemExtent`, `ValueKey` & Slivers — Render massive scrollable feeds in constant memory using `SliverChildBuilderDelegate` (`ListView.builder`), optimize jump-scrolls with `itemExtent` / `prototypeItem`, and combine multi-Lists cleanly with `CustomScrollView` + `SliverList`. • ✓ Build, verify, and ship Module Capstone: Multi-Field Enrollment Validator & Virtualized Sliver Viewport Window Calculator.

7.1 Flutter Forms — `Form`, `GlobalKey<FormState>`, `TextFormField` Validators & `AutovalidateMode` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: GlobalKey<FormState> in State field & Validator returning null = valid • Core Syntax, CLI & Production APIs: final _formKey = GlobalKey<FormState>(); Form(key: _formKey, autovalidateMode: ...) + TextFo... • Deep-Dive Mechanics: AutovalidateMode.onUserInteraction & FormState.validate() & save() • Advanced Patterns: FocusScope.unfocus keyboard dismiss & Avoiding GlobalKey inside build() • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Composable Flutter `TextFormField` Validator Pipeline • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter Forms

Purpose: To understand and apply Flutter Forms — `Form`, `GlobalKey<FormState>`, `TextFormField` Validators & `AutovalidateMode` in real Flutter systems — specifically: Group mult...

What Topic Is Next → Next Topic 7.2: Flutter Lists — `ListView.builder`, `ListView.separated`, `itemExtent`, `ValueKey` & Slivers

7.2 Flutter Lists — `ListView.builder`, `ListView.separated`, `itemExtent`, `ValueKey` & Slivers [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: ListView.builder lazy Sliver delegate & O(1) itemExtent / prototypeItem jump • Core Syntax, CLI & Production APIs: ListView.builder(itemCount:, itemExtent:, itemBuilder:) + ListView.separated + CustomScrollV... • Deep-Dive Mechanics: ShrinkWrap: true anti-pattern & CacheExtent 250px prefetch zone • Advanced Patterns: CustomScrollView + SliverList & ValueKey(item.id) reorder stability • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: O(1) `itemExtent` Sliver Viewport Index Window Calculator • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter Lists

Purpose: To understand and apply Flutter Lists — `ListView.builder`, `ListView.separated`, `itemExtent`, `ValueKey` & Slivers in real Flutter systems — specifically: Render massiv...

What Topic Is Next → Module 7 Capstone (Multi-Field Enrollment Validator & Virtualized Sliver Viewport Window Calculator) → Next Topic 8.1: Flutter GridView — `GridView.builder`, `MaxCrossAxisExtent` vs `FixedCrossAxisCount` & `mainAxisExtent`

Module 7 Hands-On Capstone Project (100 Marks): Multi-Field Enrollment Validator & Virtualized Sliver Viewport Window Calculator — Build a composable `FormFieldValidator<String>` pipeline (email, password strength, confirm match) and a virtualized `ListView.builder` viewport calculator computing exact visible start/end...

MODULE 8 OF 16 • STAGE 2 • 11 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 8: Responsive `GridView` Delegates & Coordinated `TabBar` + `TabBarView` Navigation

Build responsive 2D product/course grids with `SliverGridDelegateWithMaxCrossAxisExtent` and synchronized swipeable tabs using `DefaultTabController`, `TabController`, and `AutomaticKeepAliveClientMixin`.

Objectives: ✓ Flutter GridView — `GridView.builder`, `MaxCrossAxisExtent` vs `FixedCrossAxisCount` & `mainAxisExtent` — Construct virtualized 2D tile grids using `GridView.builder` and responsive `SliverGridDelegate` strategies while avoiding `childAspectRatio` text-clipping bugs when users enable large OS font scaling. • ✓ Flutter TabBar & `TabBarView` — `DefaultTabController`, `SingleTickerProviderStateMixin` & KeepAlive — Wire Material 3 `TabBar` tabs to swipeable `TabBarView` pages declaratively with `DefaultTabController` or programmatically with `TabController(length:, vsync: this)`, and retain tab scroll offsets via `PageStorageKey` / `AutomaticKeepAliveClientMixin`. • ✓ Build, verify, and ship Module Capstone: Responsive `MaxCrossAxisExtent` Grid Planner & `KeepAlive` Tab State Coordinator.

8.1 Flutter GridView — `GridView.builder`, `MaxCrossAxisExtent` vs `FixedCrossAxisCount` & `mainAxisExtent`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: GridView.builder lazy 2D virtualization & SliverGridDelegateWithMaxCrossAxisExtent • Core Syntax, CLI & Production APIs: GridView.builder(gridDelegate: SliverGridDelegateWithMaxCrossAxisExtent(...), itemBuilder:) • Deep-Dive Mechanics: SliverGridDelegateWithFixedCrossAxisCount & MainAxisExtent font-scale resilience • Advanced Patterns: CrossAxisSpacing & mainAxisSpacing & SliverGrid inside CustomScrollView • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Adaptive `MaxCrossAxisExtent` Column & Tile Width Solver • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter GridView

Purpose: To understand and apply Flutter GridView — `GridView.builder`, `MaxCrossAxisExtent` vs `FixedCrossAxisCount` & `mainAxisExtent` in real Flutter systems — specifically: Co...

What Topic Is Next → Next Topic 8.2: Flutter TabBar & `TabBarView` — `DefaultTabController`, `SingleTickerProviderStateMixin` & KeepAlive

8.2 Flutter TabBar & `TabBarView` — `DefaultTabController`, `SingleTickerProviderStateMixin` & KeepAlive

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: DefaultTabController vs TabController & SingleTickerProviderStateMixin vsync • Core Syntax, CLI & Production APIs: DefaultTabController(length:) + TabBar + TabBarView + AutomaticKeepAliveClientMixin • Deep-Dive Mechanics: AutomaticKeepAliveClientMixin super.build & PageStorageKey scroll offset persistence • Advanced Patterns: NestedScrollView + SliverAppBar pinned tabs & TabController length invariant • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: TabController Length Invariant & PageStorage Offset Tracker • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter TabBar

Purpose: To understand and apply Flutter TabBar & `TabBarView` — `DefaultTabController`, `SingleTickerProviderStateMixin` & KeepAlive in real Flutter systems — specifically: Wire...

What Topic Is Next → Module 8 Capstone (Responsive `MaxCrossAxisExtent` Grid Planner & `KeepAlive` Tab State Coordinator) → Next Topic 9.1: Flutter NavigationDrawer — Material 3 `NavigationDrawerDestination`, Drawer Closing & Routing

Module 8 Hands-On Capstone Project (100 Marks): Responsive `MaxCrossAxisExtent` Grid Planner & `KeepAlive` Tab State Coordinator — Create a responsive grid geometry planner replicating Flutter's `SliverGridDelegateWithMaxCrossAxisExtent` column-count math across mobile, tablet, and desktop widths, paired with a `TabCont...

MODULE 9 OF 16 • STAGE 3 • 11 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 9: Material 3 `NavigationDrawer` Routing & The Gesture Arena (`GestureDetector`, `Dismissible`)

Implement modal and permanent `NavigationDrawer` navigation shells cleanly, and master Flutter's pointer disambiguation **Gesture Arena** across `InkWell`, `GestureDetector`, `Dismissible`, and `InteractiveViewer`.

Objectives: ✓ Flutter NavigationDrawer — Material 3 `NavigationDrawerDestination`, Drawer Closing & Routing — Upgrade legacy `Drawer` + `ListTile` menus to Material 3 `NavigationDrawer` (`selectedIndex`, `onDestinationSelected`, pill indicator), close the modal drawer cleanly via `Navigator.pop(context)` before pushing routes, and support `endDrawer`. • ✓ Flutter Gestures — The Gesture Arena, `GestureDetector` (`HitTestBehavior`), `InkWell` & `Dismissible` — Understand raw pointer events (`Listener`), semantic gesture disambiguation in the **Gesture Arena**, `HitTestBehavior.opaque` vs `deferToChild`, Material `InkWell` ripples, and `Dismissible` swipe actions. • ✓ Build, verify, and ship Module Capstone: Safe `Dismissible` Key Manager & Gesture Arena Disambiguator.

9.1 Flutter NavigationDrawer — Material 3 `NavigationDrawerDestination`, Drawer Closing & Routing

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Material 3 NavigationDrawer & NavigationDrawerDestination pill state • Core Syntax, CLI & Production APIs: Scaffold(drawer: NavigationDrawer(selectedIndex:, onDestinationSelected:, children: [...])) • Deep-Dive Mechanics: Pop drawer before route push & ScaffoldState.openDrawer() • Advanced Patterns: DrawerEdgeDragWidth swipe zone & Modal vs permanent tablet drawer • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Drawer Destination Navigation Stack Transition Simulator • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter NavigationDrawer

Purpose: To understand and apply Flutter NavigationDrawer — Material 3 `NavigationDrawerDestination`, Drawer Closing & Routing in real Flutter systems — specifically: Upgrade lega...

What Topic Is Next → Next Topic 9.2: Flutter Gestures — The Gesture Arena, `GestureDetector` (`HitTestBehavior`), `InkWell` & `Dismissible`

9.2 Flutter Gestures — The Gesture Arena, `GestureDetector` (`HitTestBehavior`), `InkWell` & `Dismissible`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Gesture Arena pointer disambiguation & HitTestBehavior.opaque vs deferToChild • Core Syntax, CLI & Production APIs: GestureDetector(behavior: HitTestBehavior.opaque, onTap:, onPanUpdate:) + InkWell + Dismissi... • Deep-Dive Mechanics: InkWell nearest Material ancestor rule & Dismissible stable ValueKey(item.id) • Advanced Patterns: ConfirmDismiss async Undo guard & InteractiveViewer pinch-zoom pan • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Gesture Arena Touch-Slop (`18.0px`) Disambiguation Engine • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC

reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter Gestures

Purpose: To understand and apply Flutter Gestures — The Gesture Arena, `GestureDetector` (`HitTestBehavior`), `InkWell` & `Dismissible` in real Flutter systems — specifically: Und...

What Topic Is Next → Module 9 Capstone (Safe `Dismissible` Key Manager & Gesture Arena Disambiguator) → Next Topic 10.1: Flutter State Management — `ValueNotifier`, `InheritedWidget`, Provider, Riverpod & BLoC

Module 9 Hands-On Capstone Project (100 Marks): Safe `Dismissible` Key Manager & Gesture Arena Disambiguator — Build a gesture disambiguation model resolving competing tap vs horizontal-drag pointer sequences based on touch-slop thresholds (`kTouchSlop = 18.0`), paired with a `ValueKey`-verified `Dis...

MODULE 10 OF 16 • STAGE 3 • 11 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 10: Scalable State Management (`ValueNotifier`, Riverpod, BLoC) & Flutter vs React Native

Architect production apps with unidirectional data flow (`UI = f(state)`) using `ValueNotifier`, `ChangeNotifier`, **Riverpod** (`NotifierProvider`, `AsyncNotifier`), and **BLoC**, and evaluate Flutter vs React Native trade-offs objectively.

Objectives: ✓ Flutter State Management — `ValueNotifier`, `InheritedWidget`, Provider, Riverpod & BLoC — Progress from built-in `ValueNotifier` + `ValueListenableBuilder` and O(1) `InheritedWidget` lookup to compile-safe **Riverpod** (`Notifier`, `AsyncNotifier`, `ref.watch` vs `ref.read`) and event-driven **BLoC**. • ✓ Flutter vs React Native — Impeller Canvas vs JSI/Fabric Native Views, OTA & Architecture Trade-Offs — Objective architectural comparison for senior interviews: rendering pipeline (Impeller GPU canvas vs React Native Fabric native OS widgets), language/runtime (Dart AOT vs Hermes bytecode + JSI), bundle size, OTA updates, and brownfield embedding. • ✓ Build, verify, and ship Module Capstone: Immutable Unidirectional Cart & Exam Progress Store with Granular Selectors.

10.1 Flutter State Management — `ValueNotifier`, `InheritedWidget`, Provider, Riverpod & BLoC

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Ephemeral vs app/domain state & UI = f(state) immutable snapshots • Core Syntax, CLI & Production APIs: ValueNotifier<T> + Riverpod NotifierProvider / AsyncNotifier + ref.watch(p.select(...)) • Deep-Dive Mechanics: InheritedWidget O(1) lookup & ValueNotifier & ValueListenableBuilder • Advanced Patterns: Riverpod ref.watch vs ref.read & Selector-scoped granular rebuilds • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Immutable State Notifier with `.select()` Rebuild Filtering • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter State Management

Purpose: To understand and apply Flutter State Management — `ValueNotifier`, `InheritedWidget`, Provider, Riverpod & BLoC in real Flutter systems — specifically: Progress from bui...

What Topic Is Next → Next Topic 10.2: Flutter vs React Native — Impeller Canvas vs JSI/Fabric Native Views, OTA & Architecture Trade-Offs

10.2 Flutter vs React Native — Impeller Canvas vs JSI/Fabric Native Views, OTA & Architecture Trade-Offs

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Impeller canvas vs Fabric native OEM views & Dart AOT ARM64 vs Hermes bytecode + JSI • Core Syntax, CLI & Production APIs: Flutter (Dart 3 AOT + Impeller + Platform Channels/FFI) vs React Native (Hermes + JSI + Fabr... • Deep-Dive Mechanics: Shorebird code-push vs Expo EAS Update & Dart:ffi / Pigeon vs TurboModules C++ • Advanced Patterns: APK/IPA engine baseline binary size & Add-to-App brownfield integration • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Cross-Platform Architecture Decision Matrix Scorer • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter vs React Native

Purpose: To understand and apply Flutter vs React Native — Impeller Canvas vs JSI/Fabric Native Views, OTA & Architecture Trade-Offs in real Flutter systems — specifically: Object...

What Topic Is Next → Module 10 Capstone (Immutable Unidirectional Cart & Exam Progress Store with Granular Selectors) → Next Topic 11.1: Feature-First Clean Architecture & Immutable Dart 3 Sealed/Record Models

Module 10 Hands-On Capstone Project (100 Marks): Immutable Unidirectional Cart & Exam Progress Store with Granular Selectors — Implement a compile-safe immutable state store (`Notifier<ExamProgressState>`) with `copyWith` transitions and selector-filtered listener notifications that fire ONLY when the selected slice...

MODULE 11 OF 16 • STAGE 3 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Scalable Flutter Architecture: Riverpod/BLoC, Freezed & GoRouter

Production-grade Flutter Training engineering module covering Scalable Flutter Architecture: Riverpod/BLoC, Freezed & GoRouter with hands-on labs and capstone build.

Objectives: ✓ Feature-First Clean Architecture & Immutable Dart 3 Sealed/Record Models — Purpose: Model exhaustive AsyncValue/Result states so null crashes never reach users. • ✓ Riverpod AsyncNotifier / BLoC Concurrency Transformers (droppable/restartable) — Purpose: Cancel stale search queries and prevent double-tap duplicate form submissions. • ✓ Declarative GoRouter Deep Links, ShellRoutes & Auth Redirect Guards — Purpose: Support universal iOS/Android web links while preserving bottom-tab navigation stacks. • ✓ Dio Interceptors, Queued JWT Refresh & Typed Retrofit/JsonSerializable — Purpose: Centralize network retries, error mapping, and zero-reflection JSON parsing.

11.1 Feature-First Clean Architecture & Immutable Dart 3 Sealed/Record Models

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Immutable Dart 3 Sealed/Record Models • Integration with Riverpod AsyncNotifier / BLoC Concurrency Transformers (droppable/restartable) • Live Debugging, Telemetry & Technical Interview Trade • What Is Feature • In Flutter Training, Feature • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Feature-First Clean Architecture & Immutable Dart 3 Sealed/Record Models in real Flutter systems — specifically: Model exhaustive AsyncValue/Resul...

What Topic Is Next → Next Topic 11.2: Riverpod AsyncNotifier / BLoC Concurrency Transformers (droppable/restartable)

11.2 Riverpod AsyncNotifier / BLoC Concurrency Transformers (droppable/restartable) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Edge • Integration with Declarative GoRouter Deep Links, ShellRoutes & Auth Redirect Guards • Live Debugging, Telemetry & Technical Interview Trade • What Is Riverpod AsyncNotifier / BLoC Concurrency Transformers (droppable/restartable) & Why Industry Teams Use It • In Flutter Training, Riverpod AsyncNotifier / BLoC Concurrency Transformers (droppable/restartable) exists for one.... • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Riverpod AsyncNotifier / BLoC Concurrency Transformers (droppable/restartable) in real Flutter systems — specifically: Cancel stale search queries...

What Topic Is Next → Next Topic 11.3: Declarative GoRouter Deep Links, ShellRoutes & Auth Redirect Guards

11.3 Declarative GoRouter Deep Links, ShellRoutes & Auth Redirect Guards [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of ShellRoutes • Integration with Dio Interceptors, Queued JWT Refresh & Typed Retrofit/JsonSerializable • Live Debugging, Telemetry & Technical Interview Trade • What Is Declarative GoRouter Deep Links, ShellRoutes & Auth Redirect Guards & Why Industry Teams Use It • In Flutter Training, Declarative GoRouter Deep Links, ShellRoutes & Auth Redirect Guards exists for one clear real • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Declarative GoRouter Deep Links, ShellRoutes & Auth Redirect Guards in real Flutter systems — specifically: Support universal iOS/Android web link...

What Topic Is Next → Next Topic 11.4: Dio Interceptors, Queued JWT Refresh & Typed Retrofit/JsonSerializable

11.4 Dio Interceptors, Queued JWT Refresh & Typed Retrofit/JsonSerializable [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Queued JWT Refresh • Integration with Feature • Live Debugging, Telemetry & Technical Interview Trade • What Is Dio Interceptors, Queued JWT Refresh & Typed Retrofit/JsonSerializable & Why Industry Teams Use It • In Flutter Training, Dio Interceptors, Queued JWT Refresh & Typed Retrofit/JsonSerializable exists for one clear re... • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Dio Interceptors, Queued JWT Refresh & Typed Retrofit/JsonSerializable in real Flutter systems — specifically: Centralize network retries, error m...

What Topic Is Next → Module 11 Capstone (Modular Cross-Platform SaaS Super-App Shell in Flutter) → Next Topic 12.1: Reactive Local SQL with Drift (SQLite), Indexes & Schema Migrations

Module 11 Hands-On Capstone Project (100 Marks): Modular Cross-Platform SaaS Super-App Shell in Flutter — Design, implement, test, and document 'Modular Cross-Platform SaaS Super-App Shell in Flutter' applying every production concept from Module 11 (Scalable Flutter Architecture: Riverpod/BLoC,...

MODULE 12 OF 16 • STAGE 3 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Offline-First Local Databases (Drift/SQLite), Secure Storage & Sync

Production-grade Flutter Training engineering module covering Offline-First Local Databases (Drift/SQLite), Secure Storage & Sync with hands-on labs and capstone build.

Objectives: ✓ Reactive Local SQL with Drift (SQLite), Indexes & Schema Migrations — Purpose: Stream live local queries to widgets and migrate mobile DB tables safely across app updates. • ✓ Optimistic Local Writes + Background Outbox Conflict Resolution — Purpose: Let field users work 100% offline and sync queued mutations cleanly upon reconnect. • ✓ flutter_secure_storage (iOS Keychain / Android Keystore) & Biometrics — Purpose: Store refresh tokens in hardware-encrypted enclaves unlocked via FaceID/Fingerprint. • ✓ Background Isolates (Isolate.run) for Heavy JSON/Image Processing — Purpose: Keep UI animations locked at 60/120 FPS while parsing large payloads off the main thread.

12.1 Reactive Local SQL with Drift (SQLite), Indexes & Schema Migrations [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Indexes • Integration with Optimistic Local Writes + Background Outbox Conflict Resolution • Live Debugging, Telemetry & Technical Interview Trade • What Is Reactive Local SQL with Drift (SQLite), Indexes & Schema Migrations & Why Industry Teams Use It • In Flutter Training, Reactive Local SQL with Drift (SQLite), Indexes & Schema Migrations exists for one clear real • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Reactive Local SQL with Drift (SQLite), Indexes & Schema Migrations in real Flutter systems — specifically: Stream live local queries to widgets a...

What Topic Is Next → Next Topic 12.2: Optimistic Local Writes + Background Outbox Conflict Resolution

12.2 Optimistic Local Writes + Background Outbox Conflict Resolution [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Edge • Integration with flutter_secure_storage (iOS Keychain / Android Keystore) & Biometrics • Live Debugging, Telemetry & Technical Interview Trade • What Is Optimistic Local Writes + Background Outbox Conflict Resolution & Why Industry Teams Use It • In Flutter Training, Optimistic Local Writes + Background Outbox Conflict Resolution exists for one clear real • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Optimistic Local Writes + Background Outbox Conflict Resolution in real Flutter systems — specifically: Let field users work 100% offline and sync...

What Topic Is Next → Next Topic 12.3: flutter_secure_storage (iOS Keychain / Android Keystore) & Biometrics

12.3 flutter_secure_storage (iOS Keychain / Android Keystore) & Biometrics [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Biometrics • Integration with Background Isolates (Isolate.run) for Heavy JSON/Image Processing • Live Debugging, Telemetry & Technical Interview Trade • What Is flutter_secure_storage (iOS Keychain / Android Keystore) & Biometrics & Why Industry Teams Use It • In Flutter Training, flutter_secure_storage (iOS Keychain / Android Keystore) & Biometrics exists for one clear rea... • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply flutter_secure_storage (iOS Keychain / Android Keystore) & Biometrics in real Flutter systems — specifically: Store refresh tokens in hardware-enc...

What Topic Is Next → Next Topic 12.4: Background Isolates (Isolate.run) for Heavy JSON/Image Processing

12.4 Background Isolates (Isolate.run) for Heavy JSON/Image Processing [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Edge • Integration with Reactive Local SQL with Drift (SQLite), Indexes & Schema Migrations • Live Debugging, Telemetry & Technical Interview Trade • What Is Background Isolates (Isolate.run) for Heavy JSON/Image Processing & Why Industry Teams Use It • In Flutter Training, Background Isolates (Isolate.run) for Heavy JSON/Image Processing exists for one clear real • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Background Isolates (Isolate.run) for Heavy JSON/Image Processing in real Flutter systems — specifically: Keep UI animations locked at 60/120 FPS...

What Topic Is Next → Module 12 Capstone (Offline-Capable Delivery & Inspection App for iOS + Android) → Next Topic 13.1: Type-Safe Pigeon / MethodChannel & EventChannel Native Bridges

Module 12 Hands-On Capstone Project (100 Marks): Offline-Capable Delivery & Inspection App for iOS + Android — Design, implement, test, and document 'Offline-Capable Delivery & Inspection App for iOS + Android' applying every production concept from Module 12 (Offline-First Local Databases (Drift/SQL...

MODULE 13 OF 16 • STAGE 4 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Platform Channels (MethodChannel), Impeller Rendering & Sliver Tuning

Production-grade Flutter Training engineering module covering Platform Channels (MethodChannel), Impeller Rendering & Sliver Tuning with hands-on labs and capstone build.

Objectives: ✓ Type-Safe Pigeon / MethodChannel & EventChannel Native Bridges — Purpose: Invoke native Kotlin/Swift payment, scanner, or Bluetooth APIs from Dart cleanly. • ✓ CustomScrollView Slivers, RepaintBoundary & const Widget Discipline — Purpose: Eliminate raster/UI thread jank on media-rich e-commerce feeds. • ✓ Cached Image Sizing (memCacheWidth), Shimmer Skeletons & Memory Audits — Purpose: Prevent iOS/Android out-of-memory kills when scrolling high-res photo galleries. • ✓ Push Notifications, Local Schedules, Camera/Files & Adaptive Tablet UI — Purpose: Deliver responsive layouts that shine on compact phones, foldables, and iPads alike.

13.1 Type-Safe Pigeon / MethodChannel & EventChannel Native Bridges [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of EventChannel Native Bridges • Integration with CustomScrollView Slivers, RepaintBoundary & const Widget Discipline • Live Debugging, Telemetry & Technical Interview Trade • What Is Type • In Flutter Training, Type • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Type-Safe Pigeon / MethodChannel & EventChannel Native Bridges in real Flutter systems — specifically: Invoke native Kotlin/Swift payment, scanner...

What Topic Is Next → Next Topic 13.2: CustomScrollView Slivers, RepaintBoundary & const Widget Discipline

13.2 CustomScrollView Slivers, RepaintBoundary & const Widget Discipline [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of RepaintBoundary • Integration with Cached Image Sizing (memCacheWidth), Shimmer Skeletons & Memory Audits • Live Debugging, Telemetry & Technical Interview Trade • What Is CustomScrollView Slivers, RepaintBoundary & const Widget Discipline & Why Industry Teams Use It • In Flutter Training, CustomScrollView Slivers, RepaintBoundary & const Widget Discipline exists for one clear real • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply CustomScrollView Slivers, RepaintBoundary & const Widget Discipline in real Flutter systems — specifically: Eliminate raster/UI thread jank on med...

What Topic Is Next → Next Topic 13.3: Cached Image Sizing (memCacheWidth), Shimmer Skeletons & Memory Audits

13.3 Cached Image Sizing (memCacheWidth), Shimmer Skeletons & Memory Audits [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Shimmer Skeletons • Integration with Push Notifications, Local Schedules, Camera/Files & Adaptive Tablet UI • Live Debugging, Telemetry & Technical Interview Trade • What Is Cached Image Sizing (memCacheWidth), Shimmer Skeletons & Memory Audits & Why Industry Teams Use It • In Flutter Training, Cached Image Sizing (memCacheWidth), Shimmer Skeletons & Memory Audits exists for one clear re... • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Cached Image Sizing (memCacheWidth), Shimmer Skeletons & Memory Audits in real Flutter systems — specifically: Prevent iOS/Android out-of-memory k...

What Topic Is Next → Next Topic 13.4: Push Notifications, Local Schedules, Camera/Files & Adaptive Tablet UI

13.4 Push Notifications, Local Schedules, Camera/Files & Adaptive Tablet UI [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Local Schedules • Integration with Type • Live Debugging, Telemetry & Technical Interview Trade • What Is Push Notifications, Local Schedules, Camera/Files & Adaptive Tablet UI & Why Industry Teams Use It • In Flutter Training, Push Notifications, Local Schedules, Camera/Files & Adaptive Tablet UI exists for one clear re... • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Push Notifications, Local Schedules, Camera/Files & Adaptive Tablet UI in real Flutter systems — specifically: Deliver responsive layouts that shi...

What Topic Is Next → Module 13 Capstone (Smooth 120fps Custom Sliver Storefront with Native SDK Bridge) → Next Topic 14.1: Unit, Widget & Pixel-Accurate Golden Screenshot Regression Tests

Module 13 Hands-On Capstone Project (100 Marks): Smooth 120fps Custom Sliver Storefront with Native SDK Bridge — Design, implement, test, and document 'Smooth 120fps Custom Sliver Storefront with Native SDK Bridge' applying every production concept from Module 13 (Platform Channels (MethodChannel), Imp...

MODULE 14 OF 16 • STAGE 4 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Widget/Golden Tests, Integration Suites, Flavors & App/Play Store CI

Production-grade Flutter Training engineering module covering Widget/Golden Tests, Integration Suites, Flavors & App/Play Store CI with hands-on labs and capstone build.

Objectives: ✓ Unit, Widget & Pixel-Accurate Golden Screenshot Regression Tests — Purpose: Catch broken layouts and state bugs across screen sizes in fast headless CI runs. • ✓ End-to-End Device Integration Testing (integration_test / Patrol) — Purpose: Automate real tap-and-scroll checkout journeys on emulators before store submission. • ✓ Build Flavors (--dart-define-from-file), Obfuscation & Split Debug Info — Purpose: Maintain separate dev/staging/prod icons, API URLs, and obfuscated Dart binaries. • ✓ Codemagic / GitHub Actions + Fastlane Signing for TestFlight & Play Store — Purpose: Automate code signing, Crashlytics symbol upload, and dual-store publishing.

14.1 Unit, Widget & Pixel-Accurate Golden Screenshot Regression Tests

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Widget • Integration with End • Live Debugging, Telemetry & Technical Interview Trade • What Is Unit, Widget & Pixel • In Flutter Training, Unit, Widget & Pixel • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Unit, Widget & Pixel-Accurate Golden Screenshot Regression Tests in real Flutter systems — specifically: Catch broken layouts and state bugs acros...

What Topic Is Next → Next Topic 14.2: End-to-End Device Integration Testing (integration_test / Patrol)

14.2 End-to-End Device Integration Testing (integration_test / Patrol)

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Edge • Integration with Build Flavors (• Live Debugging, Telemetry & Technical Interview Trade • What Is End • In Flutter Training, End • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply End-to-End Device Integration Testing (integration_test / Patrol) in real Flutter systems — specifically: Automate real tap-and-scroll checkout jo...

What Topic Is Next → Next Topic 14.3: Build Flavors (--dart-define-from-file), Obfuscation & Split Debug Info

14.3 Build Flavors (--dart-define-from-file), Obfuscation & Split Debug Info

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Obfuscation • Integration with Codemagic / GitHub Actions + Fastlane Signing for TestFlight & Play Store • Live Debugging, Telemetry & Technical Interview Trade • What Is Build Flavors (• In Flutter Training, Build Flavors (• Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Build Flavors (--dart-define-from-file), Obfuscation & Split Debug Info in real Flutter systems — specifically: Maintain separate dev/staging/prod...

What Topic Is Next → Next Topic 14.4: Codemagic / GitHub Actions + Fastlane Signing for TestFlight & Play Store

14.4 Codemagic / GitHub Actions + Fastlane Signing for TestFlight & Play Store

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Play Store • Integration with Unit, Widget & Pixel • Live Debugging, Telemetry & Technical Interview Trade • What Is Codemagic / GitHub Actions + Fastlane Signing for TestFlight & Play Store & Why Industry Teams Use It • In Flutter Training, Codemagic / GitHub Actions + Fastlane Signing for TestFlight & Play Store exists for one clear... • Even if you come from a non • Runnable Code Implementation (DART)

Purpose: To understand and apply Codemagic / GitHub Actions + Fastlane Signing for TestFlight & Play Store in real Flutter systems — specifically: Automate code signing, Crashlyti...

What Topic Is Next → Module 14 Capstone (Multi-Flavor (Dev/Stage/Prod) Automated iOS & Android Release) → Next Topic 15.1: Clean Layered Architecture (Presentation, Domain, Data) with AsyncNotifier & Sealed States

Module 14 Hands-On Capstone Project (100 Marks): Multi-Flavor (Dev/Stage/Prod) Automated iOS & Android Release — Design, implement, test, and document 'Multi-Flavor (Dev/Stage/Prod) Automated iOS & Android Release' applying every production concept from Module 14 (Widget/Golden Tests, Integration Suite...

MODULE 15 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Offline-First Flutter Architecture, Isolates & Impeller 60/120fps Tuning

Self-explanatory list-and-bullet production mastery module covering Offline-First Flutter Architecture, Isolates & Impeller 60/120fps Tuning in Flutter Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ Clean Layered Architecture (Presentation, Domain, Data) with AsyncNotifier & Sealed States — Purpose: Separate UI widgets cleanly from API/SQLite repositories for testable, scalable Flutter codebases. • ✓ Dart Isolates (Isolate.run) for Heavy JSON/Crypto Work & Sliver Viewport Optimization — Purpose: Keep UI animations locked at 60/120fps under the Impeller engine with zero main-isolate jank. • ✓ Secure Token Storage, Dio Refresh-Token Queues & GoRouter Deep-Link Guards — Purpose: Handle expired sessions seamlessly and route authenticated deep links across Android, iOS, and Web. • ✓ Build, verify, and document 'Smooth 120fps Offline-Capable Commerce App with Riverpod & Background Isolates' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

15.1 Clean Layered Architecture (Presentation, Domain, Data) with AsyncNotifier & Sealed States [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Clean Layered Architecture (Presentation, Domain, Data) with AsyncNotifier... • Performance Optimization, Complexity & Latency Impact of Clean Layered Architecture (Presentation, Domain, Data) wi... • Live Debugging, Stack • Runnable Code Implementation (DART) • Hands • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Clean Layered Architecture (Presentation, Domain, Da... • Production Hardening & Security: flutter_secure_storage keychain tokens, obfuscated AOT builds & TLS valida...

Purpose: To understand and apply Clean Layered Architecture (Presentation, Domain, Data) with AsyncNotifier & Sealed States in real Flutter systems — specifically: Separate UI wid...

What Topic Is Next → Next Topic 15.2: Dart Isolates (Isolate.run) for Heavy JSON/Crypto Work & Sliver Viewport Optimization

15.2 Dart Isolates (Isolate.run) for Heavy JSON/Crypto Work & Sliver Viewport Optimization [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Dart Isolates (Isolate.run) for Heavy JSON/Crypto Work & Sliver Viewport O... • Performance Optimization, Complexity & Latency Impact of Dart Isolates (Isolate.run) for Heavy JSON/Crypto Work & S... • Live Debugging, Stack • Runnable Code Implementation (DART) • Hands • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Dart Isolates (Isolate.run) for Heavy JSON/Crypto Wo... • Production Hardening & Security: flutter_secure_storage keychain tokens, obfuscated AOT builds & TLS valida...

Purpose: To understand and apply Dart Isolates (Isolate.run) for Heavy JSON/Crypto Work & Sliver Viewport Optimization in real Flutter systems — specifically: Keep UI animations l...

What Topic Is Next → Next Topic 15.3: Secure Token Storage, Dio Refresh-Token Queues & GoRouter Deep-Link Guards

15.3 Secure Token Storage, Dio Refresh-Token Queues & GoRouter Deep-Link Guards [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Secure Token Storage, Dio Refresh • Performance Optimization, Complexity & Latency Impact of Secure Token Storage, Dio Refresh • Live Debugging, Stack • Runnable Code Implementation (DART) • Hands • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Secure Token Storage • Production Hardening & Security: flutter_secure_storage keychain tokens, obfuscated AOT builds & TLS valida...

Purpose: To understand and apply Secure Token Storage, Dio Refresh-Token Queues & GoRouter Deep-Link Guards in real Flutter systems — specifically: Handle expired sessions seamles...

What Topic Is Next → Module 15 Capstone (Smooth 120fps Offline-Capable Commerce App with Riverpod & Background Isolates) → Next Topic 16.1: Unit, Widget & Golden Visual Regression Testing with Mocktail in Flutter

Module 15 Hands-On Capstone Project (100 Marks): Smooth 120fps Offline-Capable Commerce App with Riverpod & Background Isolates — Architect, implement, test, and document 'Smooth 120fps Offline-Capable Commerce App with Riverpod & Background Isolates' applying every bullet-wise engineering concept from Module 15 (Offli...

MODULE 16 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Flutter Widget/Golden Testing, DevTools Profiling & Store Release CI/CD

Self-explanatory list-and-bullet production mastery module covering Flutter Widget/Golden Testing, DevTools Profiling & Store Release CI/CD in Flutter Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ Unit, Widget & Golden Visual Regression Testing with Mocktail in Flutter — Purpose: Catch layout overflows, broken dark/light themes, and state regressions before every release. • ✓ Flutter DevTools Memory/CPU Profiling, Widget Rebuild Audits & Crashlytics — Purpose: Eliminate stream/controller leaks, excessive rebuilds, and production crashes proactively. • ✓ Multi-Environment Flavors, Code Signing, Shorebird/OTA Concepts & Flutter Interviews — Purpose: Explain Widget-Element-RenderObject trees, keys, and state trade-offs in technical interviews. • ✓ Build, verify, and document 'Automated Widget + Integration Test Suite & Multi-Flavor Release Pipeline' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

16.1 Unit, Widget & Golden Visual Regression Testing with Mocktail in Flutter [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Unit, Widget & Golden Visual Regression Testing with Mocktail in Flutter • Performance Optimization, Complexity & Latency Impact of Unit, Widget & Golden Visual Regression Testing with Mockt... • Live Debugging, Stack • Runnable Code Implementation (DART) • Hands • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Unit • Production Hardening & Security: flutter_secure_storage keychain tokens, obfuscated AOT builds & TLS valida...

Purpose: To understand and apply Unit, Widget & Golden Visual Regression Testing with Mocktail in Flutter in real Flutter systems — specifically: Catch layout overflows, broken da...

What Topic Is Next → Next Topic 16.2: Flutter DevTools Memory/CPU Profiling, Widget Rebuild Audits & Crashlytics

16.2 Flutter DevTools Memory/CPU Profiling, Widget Rebuild Audits & Crashlytics [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Flutter DevTools Memory/CPU Profiling, Widget Rebuild Audits & Crashlytics • Performance Optimization, Complexity & Latency Impact of Flutter DevTools Memory/CPU Profiling, Widget Rebuild Audi... • Live Debugging, Stack • Runnable Code Implementation (DART) • Hands • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Flutter DevTools Memory/CPU Profiling • Production Hardening & Security: flutter_secure_storage keychain tokens, obfuscated AOT builds & TLS valida...

Purpose: To understand and apply Flutter DevTools Memory/CPU Profiling, Widget Rebuild Audits & Crashlytics in real Flutter systems — specifically: Eliminate stream/controller lea...

What Topic Is Next → Next Topic 16.3: Multi-Environment Flavors, Code Signing, Shorebird/OTA Concepts & Flutter Interviews

16.3 Multi-Environment Flavors, Code Signing, Shorebird/OTA Concepts & Flutter Interviews [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Multi • Performance Optimization, Complexity & Latency Impact of Multi • Live Debugging, Stack • Runnable Code Implementation (DART) • Hands • Idiomatic FLUT Implementation: Dart 3 sealed states, Riverpod/BLoC reactive architecture & Dio interceptors • Flutter Impeller 60/120fps Engine, Widget/Element/RenderObject Trees & Dart 3 Isolates Execution & Memory Lifecycle for Multi-Environment Flavors • Production Hardening & Security: flutter_secure_storage keychain tokens, obfuscated AOT builds & TLS valida...

Purpose: To understand and apply Multi-Environment Flavors, Code Signing, Shorebird/OTA Concepts & Flutter Interviews in real Flutter systems — specifically: Explain Widget-Elemen...

What Topic Is Next → Module 16 Capstone → End-of-Course Live Client Project: QuickMed Cross-Platform Telehealth Booking & Prescription Tracker App

Module 16 Hands-On Capstone Project (100 Marks): Automated Widget + Integration Test Suite & Multi-Flavor Release Pipeline — Architect, implement, test, and document 'Automated Widget + Integration Test Suite & Multi-Flavor Release Pipeline' applying every bullet-wise engineering concept from Module 16 (Flutter Wi...

3. LIVE CLIENT PRODUCTION PROJECT & CAREER SCOPE

End-of-Course Live Client Production Project + Official Internship Confirmation Letter

Project: QuickMed Cross-Platform Telehealth Booking & Prescription Tracker App • **Confirmed Role:** Flutter & Dart Cross-Platform Mobile Intern

Domain & Production Infra: Digital Healthcare & Diagnostics Consumer App Client (Flutter 3 + Dart 3 + Riverpod/BLoC + Dio REST + Local Secure Storage (Android & iOS/Web))

- Crafted 60fps adaptive Flutter UI flows with custom slivers, animations, and responsive design tokens.
- Architected predictable state management with Riverpod/BLoC, clean repository layers, and Dart 3 sealed classes.
- Connected Dio HTTP interceptors, secure JWT keychain storage, push notifications, and offline caching.
- Published release-ready cross-platform builds validated with widget and integration test suites.

Target Job Roles: Flutter Mobile Developer (iOS & Android), Cross-Platform Mobile Engineer, Dart & Flutter Product Developer, Mobile Application SDE-1

Realistic Annual CTC Scope: **Fresher / Entry-Level (0–1 Yr):** ₹4.8 LPA – ₹9.2 LPA **Mid-Level Flutter Product Engineer (1–3 Yrs):** ₹9.5 LPA – ₹19.5 LPA **Senior Cross-Platform Mobile Architect (3–6+ Yrs):** ₹20.0 LPA – ₹38.0+ LPA