predev. Academy

TRACK ROADMAP

Flutter Mobile Development

Build, test and publish a real Android and iOS app with Flutter, from the first line of Dart to an internal testing release on the stores.

Mobile

DURATION	SESSIONS	SCHEDULE	LOCATION	LEVEL	LANGUAGE
10 weeks	20 live sessions, 60 hours	Twice a week, 3 hours per session, plus 1 hour of online office hours every week	In person or online, live	Beginner, after Tech Compass	Egyptian Arabic, English technical terms

Who it is for

- CS and engineering students who want a mobile skill they can sell before they graduate
- Fresh graduates who want their first junior Flutter job in Egypt or a remote role
- Career switchers who finished Tech Compass and want to build apps people install from the store
- Web developers who want to add mobile apps to the work they can take on

Before day 1

- Tech Compass completed, or basic programming logic: variables, conditions, loops
- Laptop with at least 8 GB RAM (16 GB recommended), 30 GB free disk space, Windows 10/11, macOS or Linux
- An Android phone and a USB cable, or a laptop that can run the Android emulator
- Free accounts ready before day 1: GitHub, Google (for Firebase)
- Installed before day 1: Git, VS Code, Flutter SDK, Android Studio (we help in the setup session if anything fails)
- A Mac is optional: iOS builds and TestFlight are shown as an instructor walkthrough

By the end you can

- Write clean Dart 3 code using null safety, classes, async/await, records and pattern matching
- Build responsive Flutter screens from a design with Material 3 theming and dark mode
- Structure an app by feature with Riverpod for state and go_router for navigation
- Connect to a REST API with dio, parse JSON into typed models and handle loading, empty and error states
- Implement login with tokens stored in secure storage, protected routes and logout
- Send and handle push notifications with Firebase Cloud Messaging
- Ship an app in Arabic and English with correct RTL layout
- Write unit and widget tests and use them to check AI generated code

- Set up dev and production flavors, sign a release build and upload it to a Play Console internal testing track
- Present a published project on GitHub and in an interview, and answer common Flutter interview questions

Tech stack

LANGUAGE & FRAMEWORK

Dart 3	Null safety, async code, records and patterns are used in every Flutter file
Flutter 3	One codebase for Android and iOS, used by most mobile teams hiring juniors in Egypt
Material 3	The default design system in Flutter, with theming and dark mode built in

DATA & NETWORKING

dio	HTTP client with interceptors for tokens, logging and error handling
json_serializable / freezed	Generate JSON parsing and immutable models instead of writing them by hand
flutter_secure_storage	Store access tokens in the Keychain and Keystore, not in plain text
shared_preferences / sqflite	Save settings and cache data so the app works with a weak connection

QUALITY & DELIVERY

flutter_test / mocktail	Unit and widget tests that catch bugs before users do
Flavors and --dart-define	Separate dev and production builds with different API URLs and Firebase projects
Google Play Console / App Store Connect	Signing, internal testing tracks and TestFlight
Git and GitHub	Daily version control, pull requests and your public portfolio
Claude Code or Cursor	AI coding tools used with review and tests, the way a real team uses them

ARCHITECTURE & STATE

Riverpod	Main state management tool in the course: testable, compile safe, handles async data well
Bloc (overview)	Common in Egyptian companies, so you learn to read it and know when teams pick it
go_router	Official routing package: named routes, deep links and login redirects

SERVICES

Firebase (FlutterFire)	Project setup and Cloud Messaging (FCM) for push notifications
flutter_localizations / intl	Arabic and English, RTL layout, dates and numbers in the right format

The running project: Sehha: clinic booking app

Across weeks 4-8 you build one app: a patient app for booking doctor appointments, close to the healthcare app predev shipped as NileMed and to the structure of our other store apps such as Rakeez and iSpeaker. It was chosen because it covers what clients ask for in real projects: login, lists from an API, search, a booking flow, push notifications and an Arabic first interface.

- Sign up, login and logout with tokens kept in secure storage
- Home screen with specialties and featured doctors loaded from a REST API
- Doctor list with search, filters, pagination and pull to refresh
- Doctor profile with available time slots
- Booking flow with form validation and a confirmation screen
- My appointments screen with upcoming and past bookings, and cancel
- Push notification for booking confirmation that opens the right screen
- Arabic and English with full RTL support, light and dark theme
- Offline cache for the last loaded doctors and saved user settings
- Dev and production flavors, unit and widget tests, signed Android release build

Phase overview

WEEKS 1-3
Foundations
Write correct Dart code, use Git every day and solve small problems before touching Flutter UI.

WEEKS 4-7
Core framework
Build the Sehha clinic booking app step by step with Flutter, Riverpod, go_router, dio and Firebase.

WEEKS 8
Building with AI
Use AI coding tools to move faster on a real codebase while checking every change with review and tests.

WEEKS 9
Capstone
Build your own app from a brief and release it to a real internal testing track.

WEEKS 10
Career week
Turn your projects into job interviews and first freelance clients.

Week by week

Foundations · Weeks 1-3

Week 1 Dart basics and your developer setup

Session 1 Setup and first Dart programs

- How a Flutter app runs: Dart code, engine, Android and iOS
- Installing and checking Flutter and Dart with flutter doctor
- VS Code setup: Dart and Flutter extensions, formatter, shortcuts
- Variables, var, final, const and basic types
- Strings, interpolation and operators
- Running code in DartPad and from the terminal

In-class lab: Build a command line bill splitter that reads the total and number of people and prints each share with tip.

Session 2 Control flow, functions and Git basics

- if/else, switch and loops
- Functions: named and optional parameters, arrow syntax
- Lists, sets and maps with common methods (map, where, fold)
- What Git is: repository, commit, history
- git init, add, commit, log and a first push to GitHub
- Writing a short README

In-class lab: Write a student grades program that calculates averages and letter grades from a map, commit each step and push it to a new GitHub repository.

Homework: Solve 5 short Dart exercises (string and list problems) in one repository, one commit per exercise, and share the GitHub link.

Week 2 Object oriented Dart and null safety

Session 1 Classes and objects

- Classes, fields, constructors and named constructors
- Getters, setters and private members
- Inheritance, abstract classes and interfaces
- Mixins and extension methods
- Enums with fields and methods
- factory constructors and fromJson / toJson by hand

In-class lab: Model a pharmacy cart: Product, CartItem and Cart classes with add, remove and total, plus a fromJson constructor for Product.

Session 2 Null safety, records and patterns

- Nullable and non nullable types, ?, !, ?? and late
- Records for returning more than one value
- Pattern matching and switch expressions
- sealed classes for modelling states
- Exceptions: try/catch/finally and custom exceptions
- Reading compiler errors and fixing them

In-class lab: Model an order status as a sealed class (Pending, Shipped, Delivered, Cancelled) and print a message for each case with a switch expression.

Homework: Build a small library system in Dart (books, members, loans) with classes, a sealed result type for borrow attempts and no null errors. Push it with a README that explains how to run it.

Week 3 Async Dart, Git workflow and your first Flutter app

Session 1 Async code and problem solving

- Why apps need async code: network, files, timers
- Future, async/await and error handling
- Streams and listening to events
- Decoding JSON from a public API in a Dart program
- Breaking a problem into steps before coding
- Common patterns: counting, searching, sorting, grouping with maps

In-class lab: Write a Dart program that fetches posts from a public JSON API, groups them by user and prints the top 3 users by post count.

Session 2 Git with a team and hello Flutter

- Branches, merge and resolving a simple conflict
- Pull requests and code review on GitHub
- .gitignore and commit message habits
- Creating a Flutter project and the folder structure
- Running on an emulator and a real phone, hot reload and hot restart
- The widget tree: MaterialApp, Scaffold, Text, Column

In-class lab: In pairs, create a Flutter app on a shared repository, each student adds a screen section on a branch, opens a pull request and reviews the other one.

Homework: Solve 3 problem solving exercises in Dart from the provided list and build a one screen Flutter profile card app (photo, name, bio, links). Push both and open a pull request for review.

Week 4 Widgets, layout and navigation

Session 1 Widgets and layout

- StatelessWidget and StatefulWidget, and when setState is enough
- Row, Column, Stack, Expanded, Flexible and Spacer
- How constraints work: why overflow happens and how to fix it
- ListView, GridView and builders for long lists
- Images, icons, custom fonts and assets
- Splitting a screen into small reusable widgets

In-class lab: Build the Sehha home screen from a Figma design: search bar, specialties grid and a list of doctor cards with static data.

Session 2 Theming, structure and go_router

- Material 3 ThemeData, ColorScheme and text styles
- Light and dark theme from one place
- Responsive layout with MediaQuery and LayoutBuilder
- Feature first folder structure for a real app
- go_router: routes, path parameters, nested routes and a bottom navigation shell
- Passing data between screens

In-class lab: Add the doctor profile and appointments screens, connect them with go_router and a bottom navigation bar, and move all colors and fonts into the theme.

Homework: Finish the static UI for login, sign up and booking screens using the same theme. Record a 30 second screen video and add it to the README.

Project milestone: Sehha static UI: all main screens built and navigable with go_router, light and dark theme.

Week 5 State management with Riverpod and REST APIs

Session 1 Riverpod for app state

- Local state vs app state and why setState stops being enough
- Provider, Notifier and AsyncNotifier
- ref.watch, ref.read and ref.listen
- AsyncValue: loading, data and error in the UI
- Family and autoDispose providers
- Bloc in short: how it compares and where you will see it

In-class lab: Move the doctor list and filters into Riverpod providers, with a specialty filter that updates the list without setState.

Session 2 Talking to a REST API with dio

- HTTP basics: methods, status codes, headers, JSON bodies
- dio setup: base options, timeouts and a logging interceptor
- Models with json_serializable and freezed, and build_runner
- Repository pattern between the API and the UI
- Handling errors: no internet, timeout, 4xx and 5xx
- Loading, empty and error states that users understand

In-class lab: Replace the static data with the Sehha practice API: load specialties and doctors through a repository and show loading, empty and error states.

Homework: Add search with debounce and pagination (load more on scroll) to the doctor list, plus pull to refresh. Commit on a branch and open a pull request.

Project milestone: Sehha loads real data: specialties, doctors, search and pagination from the API through Riverpod.

Week 6 Forms, authentication and local storage

Session 1 Forms and token based login

- Form, TextFormField, validators and focus handling
- Showing server validation errors next to the right field
- Login and sign up with access and refresh tokens
- Saving tokens with flutter_secure_storage
- A dio interceptor that adds the token and handles 401
- go_router redirect to protect screens and logout

In-class lab: Build login and sign up against the API, store the token securely, keep the user signed in after restart and redirect guests to login.

Session 2 Booking flow and offline data

- Multi step booking: pick a date, pick a slot, confirm
- Date and time pickers and input formatting
- POST requests and updating the UI after success
- shared_preferences for settings such as theme and language
- A simple offline cache with sqflite for the last loaded doctors
- Snackbars, dialogs and confirmation screens

In-class lab: Build the booking flow and the My appointments screen with cancel, and cache the doctor list so it shows when the phone is offline.

Homework: Add a profile screen where the user edits name and phone with validation, and a settings screen that saves theme choice. Write 3 test cases you will automate next week.

Project milestone: Sehha works end to end for a signed in user: login, browse, book, view and cancel appointments, with offline cache.

Week 7 Firebase, flavors, Arabic and testing

Session 1 Push notifications and flavors

- Firebase project setup with the FlutterFire CLI
- Firebase Cloud Messaging: tokens, permissions on Android and iOS
- Handling notifications in foreground, background and terminated states
- Opening the right screen when a notification is tapped
- Flavors for dev and production with different app names and IDs
- Environment values with --dart-define, keeping secrets out of Git

In-class lab: Connect Sehha to Firebase, receive a booking confirmation notification sent from the Firebase console and open the appointment details on tap. Create dev and prod flavors.

Session 2 Arabic, RTL and tests

- flutter_localizations, gen-l10n and ARB files
- RTL layout: Directionality, start/end instead of left/right, mirrored icons
- Dates, numbers and plurals with intl
- Unit tests for repositories and notifiers with mocktail
- Widget tests: finding widgets, tapping and checking the result
- Overriding Riverpod providers in tests

In-class lab: Translate Sehha to Arabic with a language switch, fix every RTL layout issue, then write unit tests for the auth repository and a widget test for the login screen.

Homework: Reach at least 10 passing tests across the app, translate every remaining string and fix lint warnings. Tag the repository v1.0.0.

Project milestone: Sehha v1.0: push notifications, Arabic and English with RTL, dev and prod flavors, and a passing test suite.

Building with AI · Weeks 8

Week 8 AI tools the right way

Session 1 Working with Claude Code or Cursor

- What AI coding tools do well and where they fail in Flutter projects
- Giving context: a project rules file, folder structure, existing patterns
- Asking for a plan first, then small changes
- Reading the diff line by line before accepting it
- Checking package names and versions on pub.dev
- Keeping API keys and personal data out of prompts

In-class lab: Use Claude Code or Cursor to add a favorites feature to Sehha (save doctors, list them, persist locally) following the project patterns, then review the diff with a partner and reject anything you cannot explain.

Session 2 Tests as a safety net and when not to trust AI

- Writing the test first, then asking AI for the code
- Spotting deprecated Flutter APIs and invented methods
- Debugging with AI: giving the error, the stack trace and the code
- Refactoring with AI while tests stay green
- Security and licensing checks on generated code
- Capstone brief, scope and planning

In-class lab: Fix 5 bugs planted in a copy of Sehha: write a failing test for each one, use AI to propose a fix and keep only fixes that pass the tests and the review.

Homework: Write a one page capstone plan: screens, API, data models, packages and a day by day schedule. Get it approved in office hours.

Project milestone: Favorites feature merged after review with tests, and capstone plan approved.

Capstone · Weeks 9

Week 9 Capstone build and release

Session 1 Android release and build day

- App icon, splash screen and app name per flavor
- Versioning: version name and build number
- Creating an upload keystore and signing config, and keeping it safe
- Building an Android App Bundle (AAB)
- Play Console: create the app, internal testing track, testers list
- Store listing basics: screenshots, description, privacy policy, data safety form

In-class lab: Sign a release build of your capstone app, share it with testers through Firebase App Distribution, and follow the instructor through a real Play Console upload.

Session 2 iOS walkthrough and capstone review

- What an iOS release needs: Mac, Xcode, Apple Developer account
- Bundle ID, certificates and provisioning in short
- Archiving and uploading to App Store Connect
- TestFlight internal and external testers
- Common review rejections and how to avoid them
- Code review of capstone projects in small groups

In-class lab: Watch a live TestFlight upload by the instructor, then finish your capstone, fix the review comments and prepare a 5 minute demo.

Homework: Finish the capstone: all required features, README with screenshots and an install link, at least 8 tests and a tagged release on GitHub.

Project milestone: Capstone app live on an internal testing track (Play Console internal testing or TestFlight) and installable by testers.

Week 10 Getting hired

Session 1 CV, LinkedIn, GitHub and freelancing

- A one page junior Flutter CV that passes a quick screen
- LinkedIn headline, about section and posting your projects
- GitHub profile README, pinned repositories, screenshots and demo GIFs
- Where Flutter jobs are posted in Egypt and for remote roles
- Freelancing first steps: Mostaql, Khamsat, Upwork, scoping and pricing a small app
- Writing a proposal and handling a first client

In-class lab: Rewrite your CV and LinkedIn profile in class with instructor feedback and pin Sehha and your capstone on GitHub with proper READMEs.

Session 2 Mock interviews and demo day

- Common Dart and Flutter interview questions
- Explaining state management, widget lifecycle and app architecture out loud
- Live coding a small widget under time pressure
- Behavioral questions and talking about your projects
- Demo day: presenting your capstone to guests and hiring partners

In-class lab: Do a 30 minute mock technical interview with written feedback, then present your capstone app on demo day.

Homework: Apply to at least 10 Flutter roles or freelance projects in the next two weeks and send your list to the instructor for follow up in office hours.

Capstone

Each student builds an original Flutter app for a real Egyptian use case from a short brief, using the same architecture as Sehha (Riverpod, go_router, dio, secure storage), and releases it to an internal testing track on Google Play or TestFlight.

Requirements

- At least 5 screens with navigation using go_router
- Login with tokens stored in secure storage, or a clear reason approved by the instructor if the app has no accounts
- Data from a REST API or Firebase with loading, empty and error states
- State management with Riverpod
- At least one form with validation
- Arabic and English with correct RTL layout
- At least 8 unit and widget tests that pass
- Dev and production flavors, no secrets in the repository
- A signed release build on Play Console internal testing or TestFlight
- Public GitHub repository with a README, screenshots and setup steps

Example ideas

- Home services booking app for plumbers and electricians
- Pharmacy order app with prescription photo upload and order tracking
- Gym membership and class booking app
- Microbus and metro trip planner with saved routes
- Restaurant menu and table reservation app for a local chain
- Private tutor finder for school and university subjects

Capstone rubric

Works end to end on a real device	25%
Code structure, architecture and readability	20%
State, API and error handling	15%
UI quality, Arabic RTL and accessibility	15%
Tests	10%
Release build on a testing track	10%
Demo and explanation of decisions	5%

How you are assessed

Weekly homework Pushed to GitHub before the next week's first session, graded on working code, commits and README	20%
In class labs and participation Lab completed by the end of the session or during office hours	10%
Sehha project milestones Checked in weeks 4-7 against the milestone list, with a short code review	25%
AI week deliverables Favorites feature and bug fixes judged on review quality and passing tests	10%
Capstone Graded with the capstone rubric	30%
Career week Updated CV, LinkedIn and GitHub profile, plus the mock interview	5%

Skills checklist

Dart

- [ ] Use null safety without forcing values with !
- [ ] Write classes, sealed types, enums and extension methods
- [ ] Use async/await, Futures and Streams with proper error handling
- [ ] Use records and switch expressions for clear code

UI

- [ ] Build any screen from a Figma design with Flutter layout widgets
- [ ] Fix overflow and constraint errors
- [ ] Apply a Material 3 theme with light and dark modes
- [ ] Support Arabic RTL and English in the same app

Architecture & state

- [ ] Organize code by feature with a repository layer
- [ ] Manage state with Riverpod Notifier and AsyncNotifier
- [ ] Read a Bloc based codebase
- [ ] Navigate with go_router, including redirects and deep links

Data & backend

- [ ] Call REST APIs with dio and interceptors
- [ ] Generate JSON models with json_serializable or freezed
- [ ] Store tokens securely and handle expired sessions
- [ ] Cache data locally with shared_preferences and sqflite
- [ ] Receive push notifications with Firebase Cloud Messaging

Quality & delivery

- [ ] Write unit and widget tests with mocktail and provider overrides
- [ ] Set up dev and production flavors
- [ ] Sign and upload an Android App Bundle to Play Console
- [ ] Explain the iOS TestFlight release steps
- [ ] Work with Git branches and pull requests
- [ ] Review AI generated code before merging it

After graduation

Months 1-3 after graduation

- Polish Sehha and your capstone, fix feedback and publish at least one app to production on Google Play
- Apply for junior Flutter and internship roles every week
- Take a small paid freelance task to practice working with a client
- Read the Riverpod and Flutter docs sections you skipped

Months 4-6

- Learn Bloc properly, since many Egyptian companies use it
- Add animations, maps and payments (for example Paymob or Fawry SDKs) to a side project
- Learn clean architecture layers and dependency injection
- Set up CI for tests and builds (GitHub Actions or Codemagic)

Months 7-12: toward mid-level

- Own a feature from design to store release in a team
- Learn native basics: platform channels, Kotlin and Swift for small integrations
- Profile performance with Flutter DevTools and reduce app size
- Learn a backend option such as Firebase in depth or a simple REST API so you can deliver full apps
- Contribute to an open source Flutter package or write about what you build

Where this track leads

Junior Flutter Developer | Mobile Application Developer (Flutter) | Flutter Intern at a software house or startup | Freelance mobile app developer (Mostaql, Khamsat, Upwork) | Remote junior Flutter developer for Gulf, European or US companies | Cross platform developer on a product team

Resources

- Dart language tour: https://dart.dev/language
- Flutter documentation: https://docs.flutter.dev
- Flutter install guide: https://docs.flutter.dev/get-started/install
- Riverpod documentation: https://riverpod.dev
- Bloc library: https://bloclibrary.dev
- go_router package: https://pub.dev/packages/go_router
- dio package: https://pub.dev/packages/dio
- json_serializable package: https://pub.dev/packages/json_serializable
- flutter_secure_storage package: https://pub.dev/packages/flutter_secure_storage
- Firebase for Flutter setup: https://firebase.google.com/docs/flutter/setup
- Flutter internationalization: https://docs.flutter.dev/ui/accessibility-and-internationalization/internationalization
- Flutter testing overview: https://docs.flutter.dev/testing/overview
- Build and release an Android app: https://docs.flutter.dev/deployment/android
- Build and release an iOS app: https://docs.flutter.dev/deployment/ios
- Google Play Console: https://play.google.com/console/about/
- TestFlight: https://developer.apple.com/testflight/
- Material Design 3: https://m3.material.io
- Git documentation: https://git-scm.com/doc
- GitHub Docs: https://docs.github.com
- Claude Code overview: https://docs.anthropic.com/en/docs/claude-code/overview