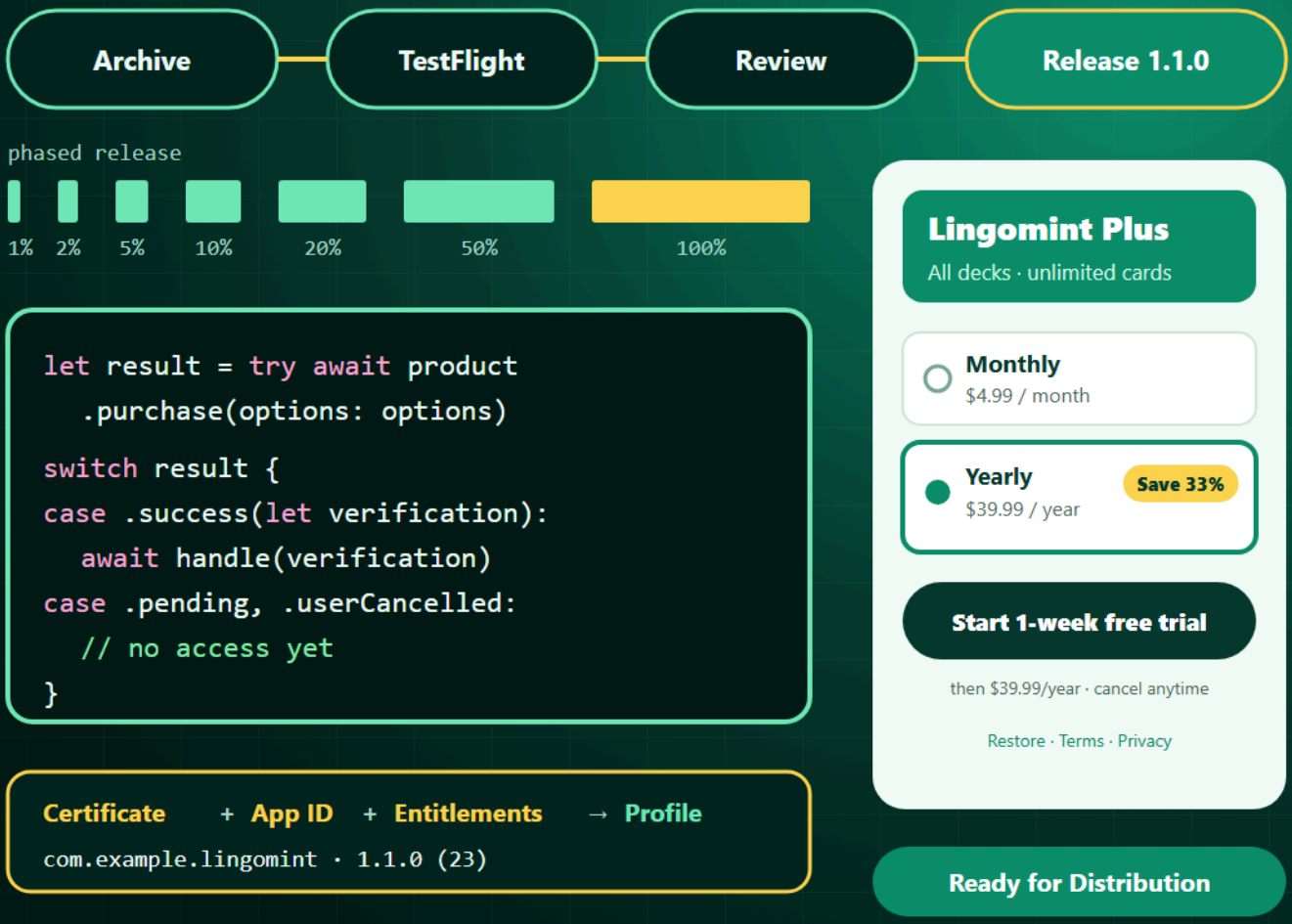


A COMPLETE, PRACTICAL GUIDE

Shipping and Monetizing iOS Apps

From your first archive to a subscription business: signing, TestFlight, review, and StoreKit 2

Code Signing · Build Configurations · Privacy Manifests
App Store Connect · TestFlight · App Review · ASO
In-App Purchases · Subscriptions · Paywalls · Automation

SWIFT 6 · XCODE 26 · iOS 18+ · STOREKIT 2

THENEXBOOKS

Shipping and Monetizing iOS Apps

Code Signing, TestFlight, App Store Review, StoreKit 2 In-App Purchases and Subscriptions, and Growth After Launch

From your first archive to a subscription business — built step by step

First Edition

THENEXBOOKS

Shipping and Monetizing iOS Apps
Code Signing, TestFlight, App Store Review, StoreKit 2 In-App Purchases and Subscriptions, and Growth After Launch

First Edition, 2026

Copyright © 2026 thenexbooks (thenexbooks.gumroad.com). All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means without the prior written permission of the publisher, except for brief quotations in reviews and certain other non-commercial uses permitted by copyright law. Code examples may be used freely in your own projects.

The information in this book is provided "as is" without warranty of any kind. While every precaution has been taken in its preparation, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein. App Store policies, review guidelines, agreements, prices, fees, commission rates, limits, and tool behavior change frequently and can differ by country or region. Nothing in this book is legal, tax, or financial advice. Always check Apple's current documentation, the App Store Review Guidelines, and the agreements in your developer account before you rely on any policy or fee.

Apple, the Apple logo, iPhone, iPad, Mac, Apple Vision Pro, iOS, iPadOS, macOS, Xcode, Swift, SwiftUI, StoreKit, TestFlight, App Store, App Store Connect, Xcode Cloud, Family Sharing, Sign in with Apple, and Apple Search Ads are trademarks of Apple Inc., registered in the U.S. and other countries and regions. Other product names, such as fastlane and GitHub, are trademarks of their respective owners. Their use in this book is for identification only and does not imply endorsement. This book is independent and is not affiliated with, sponsored by, authorized by, or endorsed by Apple Inc.

Lingomint, Lingomint Labs, Lingomint Plus, and all people, accounts, identifiers, prices, sample data, and release dates in this book are fictional and used only as a running example. Domains use reserved example names (example, lingomint.example). The screens described in this book are illustrative sketches and do not reproduce Apple's interfaces.

Preface

Building an iOS app is only half of the job. The other half starts the moment the app works on your own iPhone: signing it so other devices trust it, getting it through App Store Connect and TestFlight, passing App Review, writing a product page that people actually read, choosing a price, and — if the app is going to pay for its own development — selling in-app purchases and subscriptions that customers understand and trust. That second half is where many good apps stall. The steps are spread across Xcode, a web portal, agreements, and documentation that changes every year, and the errors are often cryptic: a missing provisioning profile, a rejected binary, a purchase that worked in testing but not in production.

This book is a complete guide to that second half. We follow one app from its first archive to a monetized product with an automated release pipeline, and we explain each step from first principles so that you understand what Apple's tools are doing and can fix problems on your own instead of clicking buttons until the error goes away.

Who This Book Is For

- **Indie developers** who can build an app in Swift and SwiftUI and now want to ship it and earn from it.
- **Developers on small teams** who have become "the release person" and need to own signing, TestFlight, review, and in-app purchases.
- **Team leads and technical founders** who want a reliable, repeatable release process and an honest monetization strategy.
- **Developers who already shipped once** and want to add subscriptions, offers, server-side purchase handling, analytics, and automation.

You should already be comfortable with Swift and the basics of SwiftUI. We do not teach the language or how to build views from scratch; we teach everything that happens around and after them.

What You Will Learn

- How the Apple Developer Program, agreements, and App Store Connect roles fit together for an individual or a company.
- How code signing really works — certificates, identifiers, devices, provisioning profiles, entitlements — and how to fix the errors you will meet.
- How to organize build configurations, schemes, xcconfig files, and version numbers for development, staging, and release builds.
- How to write a privacy manifest, declare required-reason APIs, and answer the App Privacy questions accurately.

- How to archive, validate, upload, and test builds with TestFlight, and how to prepare for and respond to App Review.
- How to write metadata, design screenshots, use product page optimization and custom product pages, and plan pricing, availability, and launch day with phased release.
- How to build in-app purchases with StoreKit 2: products, purchases, verified transactions, entitlements, restoring, consumables, non-consumables, and Family Sharing.
- How to design auto-renewable subscriptions, free trials, promotional offers, offer codes, and win-back offers, and build paywalls with StoreKit views and custom SwiftUI — honestly.
- How to test purchases end to end with StoreKit configuration files, unit tests, sandbox accounts, and TestFlight.
- How to run the app after launch: App Store Server Notifications, the App Store Server API, analytics, MetricKit, the Xcode Organizer, fastlane, Xcode Cloud, ratings, reviews, and updates.

How This Book Is Organized

Part I — Getting Ready to Ship (Chapters 1–5) covers the release journey, the Apple Developer Program, code signing, automatic signing and capabilities, and build configurations and versioning. **Part II — App Store Connect and TestFlight** (Chapters 6–10) covers privacy manifests and labels, the app record, uploading builds, TestFlight, and App Review. **Part III — The Product Page and Launch** (Chapters 11–15) covers metadata, screenshots and icons, App Store optimization, pricing and availability, and launch day. **Part IV — In-App Purchases with StoreKit 2** (Chapters 16–20) covers monetization models, configuring products, and building purchases and entitlements. **Part V — Subscriptions and Paywalls** (Chapters 21–24) covers subscriptions, offers, paywalls, and end-to-end purchase testing. **Part VI — After Launch** (Chapters 25–28) covers server-side purchase handling, analytics and crash reporting, release automation, and growth through reviews and updates. The appendices provide a cheat sheet, templates and reference material, a release-day checklist, and a glossary.

Each chapter ends with a **Try It Yourself** section and a **Key Takeaways** box for quick review.

The Running Example: Lingomint

Throughout the book we ship **Lingomint**, a fictional language flashcards app made by **Lingomint Labs**, a two-person studio: Priya, the founder and lead iOS developer, and Marco, who handles design and marketing, with help from Hana, a contract developer. Lingomint 1.0 launches as a free app with three decks — Spanish Basics, French Basics, and Japanese Kana. Version 1.1 adds **Lingomint Plus**, a subscription with monthly and yearly plans and a one-week free trial on the yearly plan, together with a one-time Spanish Travel deck pack and consumable hint packs. Version 1.2 adds a lifetime option, offer codes, promotional and win-back offers, and a small server that listens to App Store Server Notifications. Every identifier, price, and date is fixed throughout the book, so the code, the App Store Connect settings, and the examples always agree with each other.

Versions and Conventions

Code uses the **Swift 6 language mode** and **Xcode 26-era tooling**, with a **deployment target of iOS 18**. Where we use a feature that needs iOS 26, the code checks availability and the text says so. Xcode and App Store Connect labels move between releases; if a button is not exactly where the book says, look nearby. Above all, App Store policies, prices, fees, and limits change often and differ by region: we point out every such fact as "at the time of writing" and ask you to confirm it in Apple's current documentation before you depend on it. Prices are shown in U.S. dollars for the U.S. storefront; any proceeds figures are illustrative only.

> **TIP**
> A recommended practice or a technique that makes your work easier.

> **NOTE**
> Background information, version differences, or a useful aside.

> **WARNING**
> A common mistake or something that can cause rejections, lost purchases, or security problems.

Menu commands are written in bold with arrows, such as **Product > Archive**. Code, file names, and identifiers appear in `monospace`, and code listings are captioned with their file path, such as `Lingomint/Store/StoreManager.swift`. Because the book is written to be read anywhere, screens are described in words and simple text sketches rather than screenshots; console output is shown as "you should see something like" because exact text differs between tool versions.

Contents

Preface

PART I — GETTING READY TO SHIP

1 The Road to the App Store: How Apps Ship and the Lingomint Plan

2 The Apple Developer Program: Memberships, Agreements, Roles, and Team Setup

3 Code Signing Demystified: Certificates, Identifiers, Devices, and Provisioning Profiles

4 Automatic Signing, Capabilities, Entitlements, and Fixing Signing Errors

5 Build Configurations, Schemes, xcconfig Files, and Versioning

PART II — APP STORE CONNECT AND TESTFLIGHT

6 Privacy by Design: Privacy Manifests, Required-Reason APIs, and Privacy Labels

7 Setting Up Your App in App Store Connect

8 Archiving, Validating, and Uploading Builds

9 TestFlight: Internal and External Testing, Beta Review, and Feedback

10 App Review: Guidelines, Common Rejections, and How to Respond

PART III — THE PRODUCT PAGE AND LAUNCH

11 Product Page Metadata: Name, Subtitle, Keywords, Description, and What's New

12 Screenshots, App Previews, and the App Icon

13 App Store Optimization: Search, Product Page Optimization, and Custom Product Pages

14 Pricing, Availability, Age Ratings, and Regional Considerations

15 Launch Day: Release Options, Phased Release, and Your First Update

PART IV — IN-APP PURCHASES WITH STOREKIT 2

16 Monetization Models and the Lingomint Business Plan

17 Configuring In-App Purchases: App Store Connect and StoreKit Configuration Files

18 StoreKit 2 Fundamentals: Products, Purchases, and Transactions

19 Entitlements, Transaction Updates, and Restoring Purchases

20 Consumables, Non-Consumables, and Family Sharing

PART V — SUBSCRIPTIONS AND PAYWALLS

21 Auto-Renewable Subscriptions: Groups, Levels, Renewals, and Status

22 Offers: Free Trials, Promotional Offers, Offer Codes, and Win-Back Offers

23 Paywalls with SubscriptionStoreView, StoreView, and Custom SwiftUI

24 Paywall Ethics and Testing Purchases End to End

PART VI — AFTER LAUNCH

25 Server-Side Purchase Handling: App Store Server Notifications and the App Store Server API

26 Analytics, Crash Reporting, and Performance: App Analytics, MetricKit, and the Organizer

27 Release Automation with fastlane and Xcode Cloud

28 Ratings, Reviews, Updates, and Growing After Launch

APPENDICES

A Release and StoreKit Cheat Sheet

B Templates and Reference

C Release-Day Production Checklist

D Glossary

PART I

Getting Ready to Ship

The journey from an Xcode project to a customer's device: the Apple Developer Program and team roles, code signing from first principles, automatic signing and capabilities, and the build configurations, schemes, and version numbers that keep every release traceable.

1 · The Road to the App Store

2 · The Apple Developer Program

3 · Code Signing Demystified

4 · Automatic Signing and Fixing Signing Errors

5 · Build Configurations, Schemes, and Versioning

CHAPTER 1 · PART I — GETTING READY TO SHIP

The Road to the App Store: How Apps Ship and the Lingomint Plan

Building an app and shipping an app are two different jobs. In this chapter we map the whole road from an Xcode project to a customer's iPhone, meet the people and the systems along the way, and lay out the plan for Lingomint, the app we will release, monetize, automate, and grow together through the rest of the book.

Why Shipping Is a Skill of Its Own

If you can build a SwiftUI app that runs on your own device, you have already solved the hardest creative problem: you have something people might want. What usually happens next surprises many developers. The first upload fails because of a signing error. The build number is rejected because it was used before. App Review asks where the Restore Purchases button is. A subscription shows the wrong price in one country. A crash that never appeared in the simulator shows up in the Organizer a day after launch. None of these problems is about Swift, and none is hard once you understand the system behind it.

This book is about that system. We treat shipping and monetizing as an engineering discipline with its own concepts, tools, and failure modes. You will learn *why* each piece exists before you learn which button to press, because buttons move between Xcode releases while the underlying ideas stay remarkably stable. A provisioning profile is still a signed statement that says "this app, from this team, may run here, with these permissions", no matter where Xcode puts the checkbox this year.

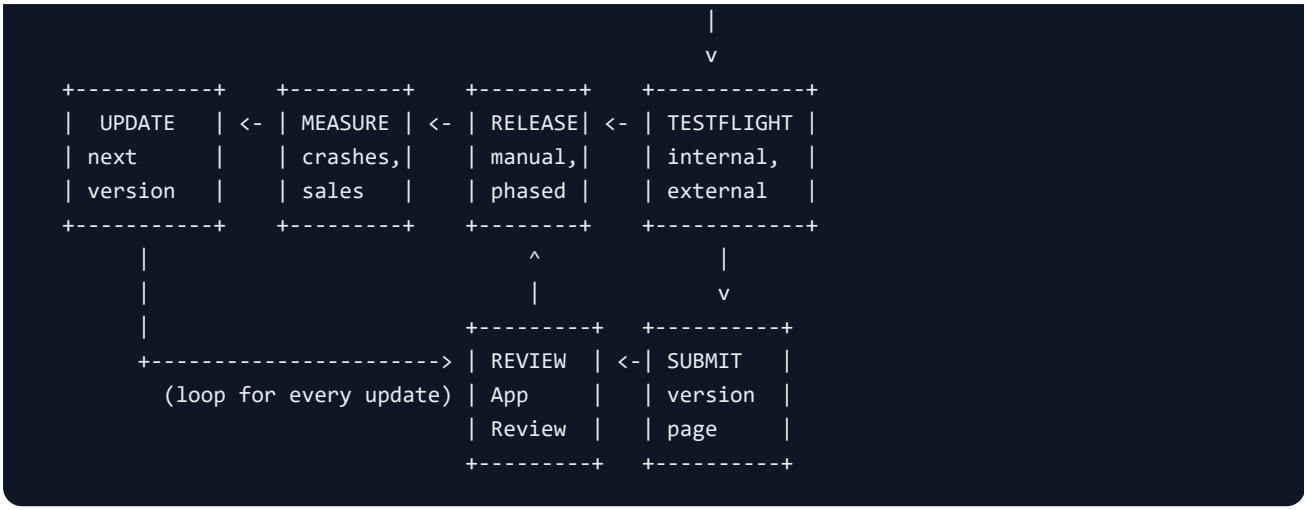
We also treat money with care: Apple's rules for digital goods, honest paywalls, and subscriptions that renew, lapse, and come back. Those topics fill the second half of the book and build on the first, because you cannot sell a subscription in an app you cannot ship.

The Release Journey at a Glance

Every iOS app that reaches the App Store follows roughly the same path. The names of the stages are worth memorizing, because App Store Connect, Xcode, and Apple's documentation use them constantly.

The Lingomint release map





Read the diagram as a loop rather than a line. Version 1.0.0 is only the first lap. Every bug fix, every new feature, and every price change sends you around again, and the faster and safer each lap becomes, the more confidently you can improve the product. The second half of this book, especially Chapter 27 on release automation, is about making that loop cheap.

Develop

Development is where you already live: writing Swift, building SwiftUI views, running tests. For release purposes, development also includes decisions that are easy to postpone and painful to change later, such as the bundle identifier, the deployment target, the build configurations you will need for staging and production, and which capabilities the app uses. We make those decisions deliberately for Lingomint in Chapters 3, 4, and 5.

Sign

Before iOS will run your code on a real device, the code must be signed with a certificate that Apple issued to your team, and it must be accompanied by a provisioning profile that grants permission to run. Signing proves who built the app and that nobody changed it afterward, and it carries the list of special permissions (entitlements) the app may use. Chapters 3 and 4 take signing apart piece by piece so that error messages stop being mysterious.

Archive

An archive is a release build packaged with everything needed to distribute it and to debug it later, including the debug symbols (dSYM files) that turn crash addresses back into function names. You create one with **Product > Archive**, and it appears in the Organizer window. Archiving uses the build configuration that the scheme's Archive action specifies, which is why Chapter 5 sets up schemes carefully before we ever archive.

Upload

From the Organizer you validate the archive and distribute it to App Store Connect. You can also upload from the command line with `xcodebuild` or with Apple's Transporter app. After the upload, Apple processes the build, which can take from a few minutes to much longer. Chapter 8 covers the upload in detail, including the errors you are most likely to hit.

TestFlight

TestFlight is Apple's beta-testing service. A processed build can go to internal testers (members of your App Store Connect team) almost immediately, and to external testers after a lightweight review. TestFlight is also where in-app purchases first meet a real device and real App Store infrastructure, in the sandbox environment, without charging anyone. Chapter 9 runs Lingomint's beta program.

Submit and Review

When a build is ready, you attach it to a version on the version page in App Store Connect, fill in the metadata, answer the questions about privacy and content, and submit for review. App Review checks the app against the App Store Review Guidelines. Most submissions are reviewed fairly quickly at the time of writing, but you should never plan a launch that depends on a specific review time. Chapter 10 covers how review works and how to respond when it says no.

Release

Approval does not have to mean immediate release. You can release manually, automatically, or after a date you choose, and for updates you can use a phased release that delivers the new version to automatic-update users gradually over seven days. Chapter 15 walks through all of these with Lingomint's real history.

Measure and Update

After release, the work changes character. You read crash reports and performance metrics, watch sales and subscription reports, answer reviews, and decide what goes into the next version. Then the loop starts again. Chapters 25 to 28 cover this phase: servers that listen for subscription events, analytics and crash reporting, automation, and growth.

The Actors Along the Road

Several Apple systems take part in the journey, and beginners often blur them together. Keeping them separate in your head makes documentation easier to search and problems easier to diagnose.

System	What it is for	Who uses it at Lingomint Labs
Apple Developer Program	Your paid membership; the legal relationship with Apple and the agreements you accept	Priya (Account Holder)
Certificates, Identifiers & Profiles	Part of the developer website: certificates, App IDs, devices, profiles, keys	Priya, Hana (mostly through Xcode)
Xcode and the Organizer	Building, signing, archiving, uploading; crash and metrics reports	Priya, Hana
App Store Connect	App records, builds, metadata, pricing, in-app purchases, reports, users	Everyone, with different roles
TestFlight	Beta distribution and tester feedback	Testers, the whole team
App Review	Human and automated review of each submission	Apple's reviewers; Priya replies

The membership is the root of everything else: without it you cannot create distribution certificates or app records. Xcode talks to Certificates, Identifiers & Profiles on your behalf when you use automatic signing, which is why many developers rarely visit it directly. App Store Connect is a separate website (and an app) for the product as a store listing and a business; TestFlight is part of it for developers and a separate app for testers.

> **NOTE**
>
> App Store Connect and Xcode labels change between releases; look nearby if a label differs. When this book shows a path such as **Apps > Lingomint > Distribution > App Information** or **Window > Organizer**, treat it as a direction rather than an exact street address. Apple reorganizes sidebars and renames tabs from time to time, but the concepts and most field names stay the same.

What "Monetize" Means on iOS

There are only a handful of ways to earn money directly from an iOS app, and Apple's rules shape all of them. Here is the landscape, described in general terms. Chapter 16 goes deeper into the trade-offs and the rules, and you should always confirm the current rules in Apple's App Store Review Guidelines and agreements before you build a business on them.

Model	How the customer pays	Fits apps that...
Paid up front	Once, before downloading	deliver complete value immediately, with little ongoing cost
Freemium + in-app purchases	Free download; optional consumable or non-consumable purchases	can show value for free and sell extras or unlocks

Subscriptions	Recurring auto-renewable payments	provide ongoing value: new content, services, sync
Advertising	Free; advertisers pay you	have large audiences and short, frequent sessions
Hybrid	Combinations of the above	serve different customers with different needs

In-app purchases come in four types in StoreKit: **consumable** (used up, can be bought again, such as hint packs), **non-consumable** (bought once, owned forever, such as a content pack), **auto-renewable subscription** (renews until canceled), and **non-renewing subscription** (time-limited access that does not renew automatically). Digital goods and services used inside the app generally must be sold through in-app purchase; physical goods and real-world services generally must not. Rules about linking to purchases outside the app differ by storefront and have changed in several regions in recent years, so we keep Lingomint entirely on in-app purchase and treat those regional rules only as a pointer to Apple's current documentation.

Apple keeps a commission on paid apps and in-app purchases. We never state the rate as a fact in this book because it depends on your programs, your storefronts, and the current terms: check Apple's current terms. For example, reduced rates apply under the App Store Small Business Program and for subscriptions after a year of paid service at the time of writing. Whenever we show proceeds, we label the figures as illustrative and remind you that App Store Connect shows the exact proceeds for each storefront.

> **WARNING**
>
> Do not copy revenue, conversion, or churn numbers from blog posts into your planning spreadsheets as if they were facts about your app. Every app's audience is different, and published figures are often unverifiable. This book deliberately contains no such statistics. Plan with ranges you can test, then measure your own results with App Store Connect's reports, as Chapter 26 shows.

Lingomint's Choice: Freemium Plus a Subscription

Lingomint is a language flashcards app. Its value is ongoing: new decks, audio for offline study, detailed statistics, and, from version 1.2.0, progress sync. Ongoing value is exactly what auto-renewable subscriptions are meant for, and App Review expects subscriptions to deliver it. At the same time, a learner needs to experience the study loop before paying anything, so the app is free to download with a generous free tier. The paid tier is called **Lingomint Plus**, sold monthly or yearly, and later as a one-time Lifetime purchase. Two smaller purchases round out the catalog: a Spanish Travel Pack for people who want one deck without subscribing, and packs of hints for learners who like a nudge on hard cards.

Meet Lingomint Labs

Our running example is fictional but realistic. **Lingomint Labs LLC** is a two-person studio that enrolls in the Apple Developer Program as an organization, so its legal name appears as the seller on the App Store. It has a website at `https://lingomint.example`, a support page at `https://lingomint.example/support`, a privacy policy at `https://lingomint.example/privacy`, terms at `https://lingomint.example/terms`, and a support address, `support@lingomint.example`. In version 1.2.0 it adds a small backend at `https://api.lingomint.example`.

Person	Job	App Store Connect role
Priya Nair	Founder, lead iOS developer	Account Holder
Marco Ruiz	Designer and marketing	App Manager + Marketing
Hana Kim	Contract iOS developer	Developer

Priya accepts agreements and owns the membership. Marco writes the product page, designs screenshots, and answers customer reviews. Hana builds features and tests every beta on the team's devices but, as a contractor, has no access to sales or financial data. Chapter 2 explains why these roles are a good fit and how to set them up. The team's placeholder Team ID is `A1B2C3D4E5`; wherever you see it, substitute your own ten-character Team ID.

Four test personas follow us through the purchase chapters. **Ava** is a free learner who never pays. **Ben** starts the yearly free trial. **Chloe** subscribes monthly, cancels, and is later invited back with an offer. **Dev** buys the Spanish Travel Pack and a pack of hints. Each has a sandbox account on the `lingomint.example` domain, such as `ben.sandbox@lingomint.example`, which we create in Chapter 24.

The App Itself

Lingomint's store name is **Lingomint: Flashcards**, with the subtitle "Learn words in 5 minutes a day". Its bundle identifier is `com.example.lingomint`, and its home-screen widget lives in a separate extension target, `LingomintWidgets`, with the bundle identifier `com.example.lingomint.widgets`. The app and widget share data through the App Group `group.com.example.lingomint`. The primary language is English (U.S.), the primary category is Education, and the secondary category is Reference.

The content is organized into decks of cards. A deck has an identifier, a title, a language code, a card count, and an optional required entitlement; a deck with no required entitlement is free for everyone.

Deck	Language	Cards	Access
Spanish Basics	es	120	free
French Basics	fr	120	free
Japanese Kana	ja	92	free

Spanish Travel	es	150	Spanish Travel Pack or Plus
Italian Essentials	it	200	Plus
German Essentials	de	200	Plus

The three free decks hold 332 cards together, and all six decks hold 882. In versions 1.0.0 and 1.0.1 only the free decks exist, with no limits and no purchases at all. From version 1.1.0 the free tier keeps the three free decks but allows 20 new cards per day and starts every learner with five hints. Plus unlocks all decks, unlimited new cards, unlimited hints, offline audio, and detailed statistics. We model the limits in code exactly once, so that the rest of the app reads them from one place:

FreeTier (preview — Chapter 16 explains the choices)

```
enum FreeTier {
    static let dailyNewCardLimit = 20
    static let starterHints = 5
}
```

The point is a habit: business rules that affect what customers get for free belong in one obvious, testable place, so that review notes, the product page, and the code all agree.

The Release Plan

We follow Lingomint through four public versions in 2026. The dates and build numbers below are canonical: every chapter uses them, so you can cross-reference events across the book.

Version	Build	Date (2026)	What ships
1.0.0	12	Tue Mar 10, manual release	First release: free app, three free decks, no purchases
1.0.1	15	Tue Mar 24	Fix for a deck import crash found in the Organizer and MetricKit
1.1.0	23	Phased, Tue May 12 to Mon May 18	Lingomint Plus subscriptions, Travel Pack, hint packs, paywall
1.2.0	31	Tue Jul 14	Lifetime, offer codes, promotional and win-back offers, server notifications, optional sign-in

A few details in this table deserve attention now, because they illustrate rules you will meet later.

- **Builds 1 to 11 preceded 1.0.0.** They were TestFlight builds. The build that shipped as 1.0.0 is build 12. Build numbers count uploads; marketing versions count releases customers see.
- **Build numbers only increase.** Lingomint never reuses or lowers a build number, even across versions. App Store Connect requires a higher build number for each upload of the same version, and keeping one global counter avoids every possible collision.

- **Version 1.0.0 is free with no purchases.** Launching without monetization first lets the team learn the release process with less at stake, and it gives early learners a reason to trust the app.
- **Version 1.1.0 carries the first in-app purchases.** The first in-app purchases for an app must be submitted together with a new app version, so the products and version 1.1.0 go to App Review at the same time. That submission is first rejected under guideline 3.1.2 because the paywall in build 22 did not show the renewal price clearly enough next to the trial button. Build 23 fixes it. Chapters 10 and 24 tell that story.
- **Version 1.1.0 uses a phased release.** Over seven days, automatic updates reach 1%, 2%, 5%, 10%, 20%, 50%, and finally 100% of users. Anyone can still update manually at any time. Chapter 15 shows why that matters when you are shipping your first paywall.

> **TIP**
>
> Start a release log on day one, even if it is a plain text file in your repository. For every upload, write the version, the build number, the date, the scheme, and one line about what changed. When a crash report arrives months later naming "1.0.0 (12)", the log tells you in seconds which code to check out.

The Product Catalog

Here is everything Lingomint sells by the end of the book, with U.S. storefront prices. Product identifiers are permanent once created in App Store Connect, so we design them carefully in Chapter 17: a reverse-DNS prefix matching the bundle ID, a family name, and a short descriptor.

Product ID	Type	Price	From
`com.example.lingomint.plus.monthly`	Auto-renewable, 1 month	$4.99	1.1.0
`com.example.lingomint.plus.yearly`	Auto-renewable, 1 year	$39.99	1.1.0
`com.example.lingomint.plus.lifetime`	Non-consumable	$99.99	1.2.0
`com.example.lingomint.pack.spanishtravel`	Non-consumable	$2.99	1.1.0
`com.example.lingomint.hints.50`	Consumable	$0.99	1.1.0

The two subscriptions live in one subscription group called **Lingomint Plus**. Within the group, Yearly is level 1 and Monthly is level 2, which makes moving from Monthly to Yearly an upgrade that takes effect immediately. Plus Yearly offers a one-week free trial to people who have never subscribed in the group. Later chapters add a promotional offer for lapsed subscribers, an offer code campaign, and a win-back offer. Twelve months of Monthly cost $59.88, so Yearly at $39.99 saves $19.89, about 33.2%; the paywall shows "Save 33%", computed at runtime from the products' prices rather than typed into a string, because prices differ by storefront and can change.

> **NOTE**
>
> All prices in this book are U.S. storefront prices chosen for the example. You choose your own prices in App Store Connect from Apple's price points, and Apple can adjust prices in other storefronts for taxes and exchange rates. Your app should always display the localized price that StoreKit gives you, never a number you wrote into the code.

The Tools and Versions This Book Targets

Apple ships a major Xcode and iOS release every year, and App Store rules change even more often. A book that pretends otherwise ages badly. Here is exactly what we target and how we deal with change.

- **Swift 6 language mode.** Lingomint compiles with strict concurrency checking. You will see `@MainActor`, `Sendable`, and `async` / `await` in every store-related listing. We do not teach these features, but we use them correctly.
- **Xcode 26-era tooling.** Screens and menu names follow recent versions of Xcode. When a name has changed recently, we mention the older name too.
- **Deployment target iOS 18.** Everything Lingomint needs, including StoreKit 2, the StoreKit SwiftUI views, and win-back offers, is available on iOS 18.
- **iOS 26 features are optional extras.** When we mention something that needs iOS 26, we put it behind an availability check and tell you to confirm the details in Apple's current documentation.

The availability pattern looks like this. It compiles with an iOS 26-era SDK while the app still runs on iOS 18, because the guarded branch only executes where the newer system is present:

Availability pattern used throughout the book

```
if #available(iOS 26, *) {
    // Use an iOS 26 API here. Check the current documentation for details.
} else {
    // The iOS 18 behavior every customer can rely on.
}
```

Whole types and functions that need a newer system get the attribute form, `@available(iOS 26, *)`, and are only called from inside such a check.

How We Hedge Policy Facts

Some facts are stable: what a provisioning profile contains, how `Transaction.updates` behaves, what a build number is. Others belong to Apple's policies and change without notice: fees, character limits, tester limits, review timelines, regional rules, and the exact wording of guidelines. For the second kind we use phrases such as "at the time of writing" and point you to the source of truth:

- the **App Store Review Guidelines**, for what review accepts;

- **App Store Connect Help**, for fields, limits, screenshot sizes, and workflows;
- the **Apple Developer Program License Agreement** and the paid apps agreement, for legal and business terms;
- the **StoreKit documentation**, for API availability and behavior.

We also paraphrase guidelines rather than quoting them at length, and we cite guideline numbers such as 3.1.1 or 5.1.1 only as they stand at the time of writing. If you are reading this a year or two after publication, the concepts will still hold, but verify the numbers and limits before you rely on them in a submission.

> **WARNING**
>
> There is no Mac in the room when you read a listing in a book. Every command in this book that starts with `security`, `codesign`, `xcodebuild`, or `xcrun` runs only on macOS, and every console output we print is shown as "you should see something like", because exact output varies with your Xcode version, your team, and your machine. Treat the shape of the output as the lesson, not the exact characters.

How This Book Differs from a Beginner Book

We assume you can already build an app. You know what `@State`, `@Observable`, and `NavigationStack` are; you have written `async` functions; you have run an app on a device or in the simulator. If you need a refresher on any of those, keep a Swift reference nearby, because this book spends its pages elsewhere.

Where a beginner book typically gives release a single chapter near the end, this book gives it twenty-eight. We go deep into signing internals, build configurations, privacy manifests, App Store Connect, TestFlight, App Review, the product page and App Store optimization, StoreKit 2 in depth, subscriptions and offers, paywalls, server-side purchase handling, analytics and crash reporting, release automation, and growth after launch. The goal is that you can own the release pipeline of a real app, not just get one build through once.

You can read straight through, which mirrors Lingomint's own timeline. Under deadline pressure, read Chapters 2 to 5, then Chapters 8, 10, and 15, and come back for the rest later. If your app is already live and you only need purchases, start at Chapter 16 but skim Chapter 5 first.

Planning Your Own Road

Before we dive into accounts and certificates, it helps to look at the calendar. Shipping involves several waits that you cannot shorten by working harder, and the most common cause of a missed launch date is discovering one of them too late.

Step	Why it can take time	When to start
Organization enrollment	Requires a D-U-N-S Number and verification of your legal entity	As soon as you decide to publish as a company
Paid Applications Agreement, tax, banking	Forms and checks must complete before you can sell anything	Weeks before your first paid feature
App record and metadata	Writing, screenshots, privacy answers, age rating	While the app is still in beta
External TestFlight	The first external build of a version needs Beta App Review	Early in the beta period
App Review	Review time varies; rejections add rounds	Leave room for at least one rejection

Lingomint Labs plans backward from each release date. For 1.0.0 on Tuesday, March 10, the team wanted the build approved a few days early and kept on hold with a manual release, so that launch day only required pressing one button. For 1.1.0 they planned extra slack, because new in-app purchases are reviewed together with the version and a first paywall is a common source of feedback from App Review. The rejection of build 22 consumed some of that slack, exactly as intended.

It is also worth writing down who does what on release day. Priya archives, uploads, watches crash reports, and answers App Review. Marco owns everything a customer reads. Hana owns the test checklist and the smoke test of each candidate build. A small team does not need a heavy process, but every task needs a name next to it.

Questions to Answer Before Chapter 2

The next four chapters ask you to make decisions that are expensive to reverse. Think about these now, for your own app:

- Will you publish as an individual or as an organization? Your choice decides the seller name customers see and whether you need a D-U-N-S Number.
- What is your bundle identifier? It cannot be changed for an app once the app record exists in App Store Connect.
- Which capabilities does your app use now, and which will it use in the next year? Extensions, App Groups, iCloud, push notifications, and Sign in with Apple all touch signing.
- Do you need separate development, staging, and production builds? If your app talks to a server or sells anything, the answer is almost always yes.
- What is the oldest iOS version you will support? A higher deployment target means fewer compatibility branches; a lower one reaches more devices.

Lingomint's answers are: organization; `com.example.lingomint`; App Groups and In-App Purchase now, Sign in with Apple in 1.2.0; Debug, Staging, and Release configurations; and iOS 18.