ShipMobileApp

Blueprint Method

The Complete Guide to Building Mobile Apps
Without Writing Code

You don't need to know how to code.

You just need this blueprint and AI.

By **@vishwas.io**

Version 1.0 – January 2026

Contents

1 Introduction

1.1 What Is This Blueprint?

This is your complete guide to building real mobile apps (iPhone & Android) using AI – even if you've never written a single line of code.

1.2 How Does It Work?

1. **You add this blueprint to a folder on your computer**
2. **You open that folder in VS Code or Cursor (free tools)**
3. **You paste the Master Prompt to AI**
4. **AI reads the blueprint and asks you questions about YOUR app**
5. **AI builds your entire app step-by-step**
6. **You test it and publish to App Store & Play Store**

That's it. No coding bootcamps. No YouTube tutorials. Just you, this blueprint, and AI.

1.3 What Will You Build?

By the end, you'll have:

- ✔ A fully working iPhone app
- ✔ A fully working Android app
- ✔ User login/signup system
- ✔ Database to store user data
- ✔ Payment system to make money
- ✔ Published on App Store & Play Store

1.4 Time Required

- **First app:** 4–8 hours (learning the process)
- **Second app:** 2–4 hours (you know the flow)
- **After that:** 1–2 hours per app

2 Tools Setup (One-Time)

You need to install these tools. Don't worry – they're all free or have free tiers.

2.1 On Your Computer

VS Code (Free)
https://code.visualstudio.com
This is where you'll work with AI

GitHub Copilot Extension (in VS Code)
Open VS Code → Extensions → Search "GitHub Copilot" → Install
Use Claude Opus 4.5 model for best results

OR Cursor (Alternative to VS Code)
https://cursor.sh
Same thing but AI is built-in. Pick VS Code OR Cursor, not both.

Xcode (Free – For iPhone apps)
Download from Mac App Store
⚠ Only works on Mac computers. Takes ~15GB space.

Android Studio (Free – For Android apps)
https://developer.android.com/studio
Works on Mac & Windows. Takes ~10GB space.

Git (Free – For saving your code)
https://git-scm.com
Just install with default settings.

2.2 Online Accounts

Supabase Account (Free tier)
https://supabase.com
This is your database (where user data is stored)

RevenueCat Account (Free tier)
https://revenuecat.com
This handles payments/subscriptions

Apple Developer Account ($99/year)
https://developer.apple.com
Required to publish iPhone apps

Google Play Developer Account ($25 one-time)
https://play.google.com/console
Required to publish Android apps

2.3 Save Your Keys

After setting up Supabase and RevenueCat, you'll have these keys. Save them somewhere safe:

```
SUPABASE_URL = "https://xxxxx.supabase.co"
SUPABASE_ANON_KEY = "eyJxxxxx..."
REVENUECAT_API_KEY_IOS = "appl_xxxxx"
REVENUECAT_API_KEY_ANDROID = "goog_xxxxx"
```

You'll give these to AI when building your app.

3 The Master Prompt (Start Here)

★ THIS IS THE MOST IMPORTANT PART ★

Copy this entire prompt and paste it to AI (GitHub Copilot or Cursor) when you start.

Copy This Entire Prompt

```
You are ShipMobileApp Assistant - an expert mobile app developer
that helps beginners build production-ready iOS and Android apps
without writing code themselves.

IMPORTANT: There is a blueprint file in this folder called
"ShipMobileApp_Blueprint.md" - read it first to understand the
complete process.

YOUR ROLE:
- You will build the ENTIRE app for the user
- User does NOT know how to code - explain everything simply
- Write all the code yourself, user just needs to copy-paste
- Ask clarifying questions before building
- Build step-by-step, testing each part before moving on

FIRST, ASK THE USER THESE QUESTIONS (one at a time):

1. "What app do you want to build? Describe the main idea."

2. "What problem does this app solve for users?"

3. "Who will use this app? (students, professionals, etc.)"

4. "List 3-5 main features your app MUST have for MVP."

5. "What will you call your app?"

6. "Do you want to build for:
   - A) iPhone only
   - B) Android only
   - C) Both iPhone and Android"

7. "How will you make money from this app?
   - A) Free with ads
   - B) One-time purchase
   - C) Monthly/yearly subscription
   - D) Free for now, monetize later"

8. "Do you have a PRD or detailed description?"

9. "Do you have your Supabase keys ready?"

10. "Do you have your RevenueCat keys ready?"

AFTER GETTING ANSWERS, DO THIS:
1. Summarize what you understood and ask user to confirm
2. Create a simple project plan with phases
3. Start building Phase 1 (basic app structure)
```

```
4. After each phase, tell user how to test it
5. Move to next phase only after user confirms it works
```

3.1 Tech Stack & Rules (Include in Prompt)

Tech Stack & Rules – Continue the Prompt

```
TECH STACK TO USE:
- iOS: Swift + SwiftUI (modern, clean code)
- Android: Kotlin + Jetpack Compose (modern, clean code)
- Backend: Supabase (database + authentication)
- Payments: RevenueCat (subscriptions)
- Architecture: MVVM pattern (keeps code organized)

CODING RULES:
1. Write clean, commented code that's easy to understand
2. NEVER put API keys directly in code - use environment variables
3. Always add loading states and error handling
4. Make the UI beautiful by default (rounded corners, shadows)
5. Support dark mode from the start
6. Add haptic feedback for button taps (feels premium)
7. Test each feature before moving to the next

SECURITY RULES (VERY IMPORTANT):
1. Store API keys in .env file, NEVER in code
2. Add .env to .gitignore (so keys don't get shared)
3. Use Supabase Row Level Security (RLS) for database
4. Validate all user inputs
5. Use HTTPS for all API calls
6. Store sensitive data in secure storage (Keychain for iOS,
   EncryptedSharedPreferences for Android)

NOW START BY GREETING THE USER AND ASKING QUESTION 1.
```

3.2 Folder Structures AI Will Create

iOS Project Structure (SwiftUI)

```
AppName/
 AppNameApp.swift        (entry point)
 Info.plist              (app settings)
 Assets.xcassets/        (images, colors)
 Models/                 (data structures)
 Views/                  (screens)
    Onboarding/
    Authentication/
    Home/
    Settings/
 ViewModels/             (business logic)
 Services/               (API calls)
    SupabaseService.swift
    AuthService.swift
    PurchaseService.swift
 Utilities/              (helpers)
```

Android Project Structure (Kotlin)

```
app/src/main/
 java/com/appname/
    MainActivity.kt
    models/
    ui/
       onboarding/
       auth/
       home/
       settings/
    viewmodels/
    services/
 res/                    (images, strings, colors)
 AndroidManifest.xml
```

3.3 How To Use The Master Prompt

1. Create a new folder on your computer (Example: “MyApp”)
2. Save this blueprint file in that folder
3. Open the folder in VS Code or Cursor
4. Open the AI chat (Copilot or Cursor’s AI)
5. Paste the Master Prompt above
6. AI will greet you and start asking questions
7. Answer each question honestly
8. AI will build your app step-by-step!

4 Your App Idea Template

Fill this out before you start building:

My App Idea Template

App Name: _____

One-Line Description:

Problem It Solves:

Who Is It For:

Main Features (MVP – Keep it to 3-5):

1. _____
2. _____
3. _____
4. _____
5. _____

Screens I Need:

- Onboarding (first-time user intro)
- Login / Signup
- Home Screen
- Main Feature Screen
- Settings
- Profile
- Paywall (if monetizing)

Monetization:

- Free (no monetization yet)
- Ads
- One-time purchase ($_____)
- Subscription ($_____ per month)

Platform:

- iPhone only
- Android only
- Both

5 Sample App Idea

Here's a filled-out example so you know what good answers look like:

Sample: Receipt Tracker App

App Name: SnapReceipt
One-Line Description:
Scan receipts with your camera and automatically track all your expenses.
Problem It Solves:
People lose receipts and have no idea where their money goes each month. This app makes expense tracking effortless – just snap a photo.
Who Is It For:
Busy professionals, freelancers, and anyone who wants to track expenses without manual entry.
Main Features (MVP):

1. Scan receipt with camera → AI extracts details automatically
2. Categorize expenses (Food, Transport, Shopping, etc.)
3. Monthly spending dashboard with charts
4. Export expenses to PDF/CSV
5. Cloud sync (data saved securely)

Screens Needed:

- ✔ Onboarding (3 screens explaining the app)
- ✔ Login / Signup (email + Apple Sign In)
- ✔ Home Screen (recent receipts + quick stats)
- ✔ Scan Screen (camera to capture receipt)
- ✔ Receipt Detail (view/edit scanned data)
- ✔ Insights (charts and spending breakdown)
- ✔ Settings (profile, export, subscription)
- ✔ Paywall (upgrade to Pro)

Monetization:

- Free: 10 scans per month
- Pro: $4.99/month or $39.99/year for unlimited scans

Platform: Both iPhone and Android

6 Understanding App Structure

Don't worry – you don't need to memorize this. AI will create everything for you. This is just so you understand what's happening.

6.1 What's In An App?

Think of an app like a house:

App Part	House Equivalent	What It Does
Screens	Rooms	What users see and interact with
Models	Blueprints	Define what data looks like
Services	Utilities	Connect to internet, database, payments
Assets	Furniture	Images, colors, icons

6.2 Common Screens Every App Needs

1. **Onboarding** – First-time user sees this (3-4 slides explaining the app)
2. **Login/Signup** – User creates account or signs in
3. **Home** – Main screen after login
4. **Core Feature** – The main thing your app does
5. **Settings** – Profile, preferences, logout
6. **Paywall** – Upgrade to paid version (if monetizing)

6.3 Security Basics

- **API Keys** = Passwords to services (Supabase, RevenueCat)
- **Never put keys in code** = AI will use .env files
- **.env file** = Secret file that stores your keys
- **.gitignore** = Tells Git to ignore .env (so keys stay private)

7 Authentication (Login & Signup)

Your app needs a way for users to sign up and log in. We use Supabase for this – it's free and secure.

7.1 What Authentication Options to Include

1. **Email + Password** (everyone should have this)
2. **Sign in with Apple** (required for iOS if you have any social login)
3. **Sign in with Google** (popular, easy for users)

7.2 Prompt for AI

Authentication Prompt

```
Add authentication to my app with:
- Email and password signup/login
- Sign in with Apple
- Sign in with Google
- Password reset via email
- Stay logged in (persistent session)

Use Supabase for the backend.
Make the UI clean and modern with:
- Smooth animations
- Loading states
- Error messages that help users
- Remember me toggle
```

7.3 What AI Will Create

- ✔ Signup screen (email, password, confirm password)
- ✔ Login screen (email, password)
- ✔ Forgot password screen
- ✔ Social login buttons
- ✔ Session management (stay logged in)
- ✔ Secure token storage

8 Database (Where Your Data Lives)

Supabase is your database – think of it as a giant spreadsheet in the cloud that stores all your users' data.

8.1 Basic Concepts

Term	Simple Explanation
Table	Like a spreadsheet (rows and columns)
Row	One item (one user, one receipt, etc.)
Column	A property (name, email, date, etc.)
RLS	Security rules (users can only see their own data)

8.2 Prompt for AI

Database Setup Prompt

```
Set up Supabase database for my app with:

1. Users table (handled by Supabase Auth automatically)

2. [Your main data] table with columns:
   - id (auto-generated)
   - user_id (links to user)
   - created_at (when it was created)
   - [add your specific columns]

3. Enable Row Level Security (RLS) so users can only
   see their own data

4. Create the SQL schema I can paste into Supabase

5. Create the Swift/Kotlin service to interact with
   this database
```

Security Note

Row Level Security (RLS) = Users can only see/edit their own data. AI will set this up so User A can't see User B's data.

9 Monetization (Getting Paid)

We use RevenueCat to handle payments. It works with both App Store (Apple) and Play Store (Google).

9.1 Monetization Models

Model	Best For	Example
Free	Building audience first	Most apps start here
Freemium	Convert free users to paid	Free: 10 uses, Pro: Unlimited
Subscription	Ongoing value	$4.99/month or $39.99/year
One-time	Simple tools	$9.99 forever

9.2 Prompt for AI

Subscription Prompt

```
Add RevenueCat subscription to my app with:

FREE TIER:
- [What free users get]
- [Limitations]

PRO TIER ($X.XX/month or $XX.XX/year):
- [What pro users get]
- Unlimited access to [feature]

Create:
1. A beautiful paywall screen
2. Subscription check throughout the app
3. Restore purchases button
4. Handle subscription status

My RevenueCat API keys:
- iOS: [your key]
- Android: [your key]
```

9.3 Pricing Tips

- **$4.99/month** or **$39.99/year** is a sweet spot for utility apps
- Always offer yearly (20% savings) – more revenue for you
- Free trial (7 days) increases conversions
- Show value before showing paywall

10 Making It Beautiful

Good news: AI will make your app look good by default. Here's what to ask for:

10.1 Design Styles

Style	Description	Best For
Glass UI	Translucent, blurry backgrounds	Modern, premium feel
Minimal	Clean, lots of white space	Professional apps
Dark Mode	Dark backgrounds, easy on eyes	Everyone wants this
Colorful	Bold colors, playful	Consumer apps

10.2 Prompt for AI

Design Prompt

```
Make my app UI beautiful with:

STYLE:
- Modern glass UI / glassmorphism
- Support both light and dark mode
- Smooth animations and transitions
- Rounded corners on everything (16px radius)
- Subtle shadows for depth

COLORS:
- Primary: [your brand color, e.g., "#007AFF"]
- Use system colors for light/dark mode support

INTERACTIONS:
- Haptic feedback on button taps
- Loading spinners when fetching data
- Pull-to-refresh where it makes sense
- Smooth page transitions

TYPOGRAPHY:
- Use system fonts (SF Pro for iOS, Roboto for Android)
- Clear hierarchy (big titles, medium subtitles, small body)
```

11 Step-by-Step Build Workflow

Follow these steps exactly. Don't skip ahead!

11.1 Phase 1: Setup (30 minutes)

1. **Create a new folder** for your app
   Example: `/Users/yourname/Projects/MyApp`
2. **Save this blueprint file** in that folder
3. **Open the folder** in VS Code or Cursor
   VS Code: File → Open Folder → Select your folder
4. **Open AI Chat**
   VS Code: Click Copilot icon in sidebar
   Cursor: Press Cmd+K or click Chat
5. **Paste the Master Prompt** (Section 3)
6. **Answer AI's questions** about your app

11.2 Phase 2: iOS App (2-3 hours)

1. AI creates the iOS project structure
2. AI builds each screen one by one
3. For each screen, AI tells you what file to create and what code to paste
4. **Test in Xcode Simulator:**
   Open Xcode → Open .xcodeproj file → Press Play button
5. Fix any issues by telling AI: "This error appeared: [paste error]"

11.3 Phase 3: Connect Backend (1 hour)

1. Go to Supabase Dashboard
2. Create tables AI specified
3. Copy the SQL and run it
4. Enable RLS (Row Level Security)
5. Add your Supabase keys (AI will tell you where)
6. Test the connection – try signing up in the app

11.4 Phase 4: Android App (2-3 hours)

1. Tell AI: "Now let's build the Android version"
2. AI creates Android project structure
3. AI builds each screen (same as iOS)
4. **Test in Android Studio Emulator:**
   Open Android Studio → Open project → Press Play button

11.5 Phase 5: Add Payments (1 hour)

1. Set up RevenueCat (create products in dashboard)
2. Connect to App Store Connect and Play Console
3. Get your API keys
4. Tell AI: "Add RevenueCat subscription with my keys"
5. Test payments (use sandbox/test mode first!)

11.6 Phase 6: Polish & Test (1-2 hours)

Test everything:

- [ ] Signup works
- [ ] Login works
- [ ] Main features work
- [ ] Data saves to Supabase
- [ ] Payments work (sandbox)
- [ ] App looks good in light mode
- [ ] App looks good in dark mode
- [ ] No crashes
- [ ] Test on real device

12 Publishing to App Stores

12.1 iOS – App Store Checklist

Before submitting to Apple:

App Icon – 1024x1024 PNG, no transparency

Screenshots – At least 3 for each device size

App Name – 30 characters max

Subtitle – 30 characters max

Description – What your app does

Keywords – For search (100 characters total)

Privacy Policy URL – Required

Support URL – Can be your Instagram or website

Age Rating – Fill out questionnaire

Price – Free or paid

In-App Purchases – If you have subscriptions

Common Rejection Reasons

- No login option for reviewers (add demo account)
- App crashes (test thoroughly!)
- Incomplete features
- Broken links
- Missing privacy policy

12.2 Android – Play Store Checklist

Before submitting to Google:

App Icon – 512x512 PNG

Feature Graphic – 1024x500 PNG

Screenshots – At least 2 for phone

App Name – 50 characters max

Short Description – 80 characters max

Full Description – 4000 characters max

Privacy Policy URL – Required

Content Rating – Fill out questionnaire

Target Audience – Select age group

Category – Select app category

13 Common Mistakes to Avoid

13.1 Mistakes to Avoid

Mistake	Why It's Bad	What To Do Instead
API keys in code	Anyone can steal your keys	Use .env files
Skipping error handling	App crashes when things go wrong	Always handle errors
No loading states	Users think app is frozen	Show spinners/skeletons
Testing only on simulator	Real devices behave differently	Test on real phone
Building too many features	Takes forever, never ships	Build MVP first (3-5 features)
Skipping auth security	Hackers can access user data	Use Supabase RLS
Hardcoded text	Can't translate later	Use string resources
Ignoring dark mode	Looks broken for 50% of users	Support both modes
No haptic feedback	Feels cheap	Add subtle vibrations
Complex onboarding	Users quit before starting	Max 3-4 screens

13.2 Good Habits

1. **Build small, test often** – Don't build for 5 hours then test
2. **One feature at a time** – Complete → Test → Next
3. **Save your work** – Commit to Git regularly
4. **Read error messages** – They tell you what's wrong
5. **Ask AI for help** – "This error appeared: [paste error]"

14 Quick Prompts Library

Copy-paste these prompts for common tasks:

14.1 Setup Prompts

Create iOS Project

```
Create a new iOS app with SwiftUI called "[AppName]" with:
- MVVM architecture
- Dark mode support
- Modern glass UI design
- Organized folder structure per the blueprint
```

Create Android Project

```
Create a new Android app with Kotlin + Jetpack Compose
called "[AppName]" with:
- MVVM architecture
- Dark mode support
- Modern Material 3 design
- Organized folder structure per the blueprint
```

14.2 Authentication Prompts

Add Supabase Auth

```
Add Supabase authentication with:
- Email/password signup and login
- Sign in with Apple
- Sign in with Google
- Forgot password flow
- Persistent sessions
- Beautiful UI with animations

My Supabase URL: [YOUR_URL]
My Supabase Anon Key: [YOUR_KEY]
```

14.3 Design Prompts

Make It Beautiful

```
Redesign [ScreenName] to be more beautiful:
- Glass UI / glassmorphism style
- Smooth animations
- Proper spacing and padding
```

```
- Support dark mode
- Add haptic feedback on buttons
- Loading states for async operations
```

Add Onboarding

```
Create a 3-screen onboarding flow for [AppName]:

Screen 1: [Main value proposition]
Screen 2: [Key feature highlight]
Screen 3: [Call to action - get started]

Make it:
- Swipeable with page indicators
- Beautiful illustrations/icons
- Skip button
- "Get Started" on last screen
```

14.4 Debugging Prompts

Fix an Error

```
I'm getting this error:

[PASTE THE FULL ERROR MESSAGE]

This is the code that's causing it:

[PASTE THE RELEVANT CODE]

Please fix it and explain what was wrong.
```

App Crashes

```
My app crashes when I [describe action].

Here's the crash log:

[PASTE CRASH LOG]

Please help me fix this.
```

14.5 Publishing Prompts

App Store Prep

```
Help me prepare for App Store submission:

My app: [AppName]
What it does: [Brief description]

Please create:
1. App Store description (short + long version)
2. 10 relevant keywords
3. Simple privacy policy
4. List of required screenshots
```

15 Resources & Links

15.1 Tools

Tool	Link	What It's For
VS Code	https://code.visualstudio.com	Code editor
Cursor	https://cursor.sh	AI-powered editor
Xcode	Mac App Store	Build iOS apps
Android Studio	https://developer.android.com/studio	Build Android apps
Supabase	https://supabase.com	Database & Auth
RevenueCat	https://revenuecat.com	Payments

15.2 Documentation

Topic	Link
SwiftUI	https://developer.apple.com/xcode/swiftui/
Jetpack Compose	https://developer.android.com/jetpack/compose
Supabase Docs	https://supabase.com/docs
RevenueCat Docs	https://docs.revenuecat.com

15.3 Connect With Me

Platform	Link
Instagram	@vishwas.io
Website	Coming soon

You're Ready!

Remember:

1. Start simple (3-5 features max)
2. Follow the workflow step-by-step
3. Test often
4. Ask AI when stuck
5. Ship it! Done is better than perfect.

Your first app won't be perfect – and that's okay.
The goal is to ship, learn, and improve.
Every successful app developer started with a messy first app.

Now go build something amazing!

ShipMobileApp Blueprint Method
By Vishwas.io
Version 1.0 – January 2026