Flutter In-App Purchases & Subscriptions

Learn to monetize your Flutter apps using in-app purchases and subscriptions for sustainable revenue.

Website reference: haas.dev | https://dev-roast-app.vercel.app

---

Introduction

Many apps rely on **in-app purchases (IAP)** and subscriptions to generate revenue. Flutter provides plugins and tools to integrate IAP seamlessly, allowing developers to **offer premium features, consumables, or recurring subscriptions** to users.

Step 1: Understanding In-App Purchase Types

1. **Consumable** – Items that can be purchased multiple times (e.g., coins, tokens).
2. **Non-consumable** – One-time purchases (e.g., premium upgrade).
3. **Subscriptions** – Recurring payments (monthly/yearly) for ongoing content or services.

Step 2: Adding the Flutter IAP Plugin

Use `in_app_purchase`:

```
dependencies:
  in_app_purchase: ^3.0.6
```

- Supports **iOS and Android**.
- Handles both consumables, non-consumables, and subscriptions.

Step 3: Initialize the Plugin

```
import 'package:in_app_purchase/in_app_purchase.dart';

final InAppPurchase iap = InAppPurchase.instance;
```

- Ensure **App Store / Google Play configurations** are complete.
- Test purchases in sandbox environment.

Step 4: Fetch Available Products

```
const Set<String> _kIds = {'premium_upgrade', 'monthly_subscription'};
final ProductDetailsResponse response = await iap.queryProductDetails(_kIds);
List<ProductDetails> products = response.productDetails;
```

- Retrieve products **configured in App Store / Play Store**.
- Show products in UI for users to purchase.

Step 5: Making a Purchase

```
final PurchaseParam purchaseParam = PurchaseParam(productDetails: products[0]);
iap.buyNonConsumable(purchaseParam: purchaseParam);
```

- For subscriptions, use `iap.buySubscription()`.
- Handle **purchase updates and errors** with a listener:

```
iap.purchaseStream.listen((purchases) {
  for (var purchase in purchases) {
    if (purchase.status == PurchaseStatus.purchased) {
      // Grant access or unlock feature
    }
  }
});
```

Step 6: Handling Purchase Validation

- Always **validate purchases** to prevent fraud.
- Use **server-side validation** if possible.
- iOS: validate with **App Store receipts**.
- Android: validate with **Google Play Developer API**.

Step 7: Practical Exercise

Objective: Add in-app purchase and subscription functionality to your app:

1. Configure **one consumable**, **one non-consumable**, and **one subscription** in Play Store / App Store.

2. Fetch products in Flutter using `in_app_purchase`.
3. Implement purchase flow with purchase listener.
4. Unlock premium features or subscription content after successful purchase.
5. Handle errors and display messages for failed transactions.

Step 8: Key Takeaways

- Monetization adds **sustainable revenue** to apps.
- Choose the **right purchase type** for your app features.
- Always **validate purchases** for security.
- Use **stream listeners** to handle real-time purchase updates.

Visit **haas.dev** for more resources and guides.
https://dev-roast-app.vercel.app
