**Product Requirements Document (PRD)**

**Project: Bank-Branded Payment SDK (Android, iOS, Web)**

**1. Purpose & Scope**

Develop bank-branded **Payment SDKs** for **Android, iOS, and Web** to enable merchants to integrate our Payment Aggregator (PA) quickly and securely. The SDKs will:

* Deliver **industry-standard checkout** experiences (Quick Checkout & Full Checkout).
* Support **saved cards via network tokenization**; **no card number (card PAN)** is handled by the SDK or merchant.
* Leverage the **Bank’s in-house backend** for all orchestration, including seasoning, routing, 3DS, refunds, and reporting.
* Use a **Bank-hosted PCI DSS–compliant card capture** experience surfaced in SDK via controlled WebView (app) or iFrame (web).

**In Scope**

* Android, iOS, and Web SDKs with a common API surface and consistent UX.
* **Payment modes supported end‑to‑end:**
  + **Cards:** Credit/Debit (CIT & MIT), 3DS2 with 3DS1 fallback, network tokenization, saved cards, partial/full refunds.
  + **UPI:** **Intent**, **Collect**, and **Autopay (e‑mandate)**; **RuPay Credit on UPI**; dynamic VPA/merchant context; deep link and app‑switch flows.
  + **Net Banking:** Aggregated bank list, redirection, status polling, and graceful return handling.
  + **Wallets:** RBI‑compliant partner wallets with KYC tiers; balance check, redirect/in‑app flows, and refund handling.
  + **BNPL/EMI:** Card EMI, Issuer EMI, and BNPL providers via adapter interfaces (pre‑check, authorization, capture, refund).
* **Checkout experiences:** Quick Checkout & Full Checkout, with clear consent for save‑card/token use.
* **Design/Theming:** Bank standard theme + merchant branding via theme tokens; dark/light modes.
* **Observability:** Event telemetry and performance metrics; correlation and idempotency keys.
* **Documentation:** Complete internal maintainer docs and external merchant integration guides with samples.

**Out of Scope**

* card number (card PAN) capture within SDK (always via Bank PCI-hosted card page).
* Third-party backend services (all backend is bank-owned).

**Terminology Clarification**

In this PRD, **PAN** refers to the **Primary Account Number (the payment card number)** used in card transactions (also called **card PAN**). It **does not** refer to the **Income‑tax PAN** issued by the Indian Income Tax Department, which is only relevant for merchant KYB/KYC outside of checkout.

**2. Success Metrics (SLOs)**

* **Checkout initialization time:** ≤ 1.5s P95 (SDK init → UI ready).
* **Auth response latency:** ≤ 2.5s P95 (initiatePayment → auth result/event).
* **Crash-free sessions:** ≥ 99.9% per release (mobile).
* **Card payment success rate uplift:** +150 bps vs baseline after routing/3DS2 enablement.
* **SDK size budget:** ≤ 5 MB per platform artifact.
* **Accessibility compliance:** WCAG 2.1 AA for Web; native accessibility labels on mobile.

**3. Architecture Overview**

1. **Merchant App/Site** embeds SDK (Android/iOS/Web).
2. **SDK** requests a **server-side session** (ephemeral key) from Merchant Server → Bank Backend.
3. **Hosted Card Capture (HCC):** Bank PCI domain returns a secure **HCC URL** to SDK for in-app WebView (Android/iOS) or iFrame (Web) to collect card number (card PAN)/expiry/CVV.
4. **Tokenization:** Bank Backend exchanges card number (card PAN) for **network token (DPAN)** via schemes; returns **token\_id** to merchant via SDK.
5. **Payment Auth:** SDK calls Bank Backend with **token\_id** and order context; Bank orchestrates acquirer/PG routing and **3DS2** flows; result is posted back via SDK callbacks.
6. **Post-Auth:** Capture/refund/status via SDK → Bank Backend APIs; settlements handled downstream by Bank systems.

**Security Boundary:** card number (card PAN) and CVV are **never exposed** to SDK/merchant; 3DS server and HCC are **bank-controlled**.

**4. Functional Requirements**

**4.1 SDK Capabilities (All Platforms)**

* **Initialize:** init(merchantId, publicKey, env) creates SDK context using ephemeral session from merchant server.
* **Start Checkout:** startCheckout(order) launches Quick or Full checkout based on available saved instruments.
* **Saved Cards / Tokenization:**
  + List tokens: listSavedCards(customerId)
  + Add card: launches **HCC**; returns token\_id on success
  + Delete token: deleteSavedCard(tokenId) with SCA as required
* **Pay with Tokenized Card:** payWithCardToken(tokenId, order, options)
* **3DS2 Support:** Embedded challenge window (app WebView overlay, web iFrame) with 3DS1 fallback when required.
* **Refund Initiation (UI trigger):** Optional customer-initiated refund request surface; actual action via backend policy.
* **Status & Receipts:** getPaymentStatus(paymentId); render receipt and share via OS share sheet.
* **Error Surface:** Structured error model with **user-friendly messages** and machine-readable codes.

**4.2 Checkout Modes**

* **Quick Checkout:** Default path when saved tokens exist; one‑tap confirm → 3DS2 challenge if needed; resume-safe on app switch.
* **Full Checkout:** Uses **Hosted Card Capture (HCC)** for new card entry; offers save‑card with explicit consent; supports retry on network/issuer decline.

**4.2.1 Payment Modes & Flows (All Methods)**

**Cards**

* CIT/MIT, partial and full capture; voids; refunds (full/partial); reversal handling.
* Network tokenization (DPAN); token lifecycle events; card metadata (last4, brand, expiry) only.
* 3DS2 challenge UI embedded; frictionless support; fall back to 3DS1 when required.
* BIN‑based UX hints (domestic/international, prepaid/credit) driven by backend policy.

**UPI (Intent, Collect, Autopay, RuPay Credit on UPI)**

* Intent: PSP app list, deep-link handoff, return detection, timeout/abandon handling.
* Collect: VPA validation, expiry, reminders/re‑notify, polling and webhook update.
* Autopay: create/modify/cancel mandates; amount change and validity; debit notifications.
* RuPay Credit on UPI: surface card‑on‑UPI as a labelled option; issuer/routing policy handled server‑side.

**Net Banking**

* Bank list caching with health/latency badges; redirect and return URL integrity checks; status reconciliation and retry prompt on drop‑offs.

**Wallets**

* Partner discovery and display; in‑app or redirect flow; KYC tier indication; balance check; refund/chargeback propagation.

**BNPL/EMI**

* Pre‑eligibility check, OTP/authorization, plan selection (tenor, rate), disclosure screens, and refund/foreclosure flows.

**4.3 Tokenization & Recurring**

* **Network tokenization** (Visa/Mastercard/RuPay/Amex).
* **CIT/MIT** support with Scheme IDs; mandate linking parameters stored server-side.
* **Card lifecycle events** (re-tokenization, expiry update) handled by Bank Backend; SDK receives updated token metadata.

**4.4 UI/UX & Theming**

* **Brand Modes:** Bank Standard Theme and Merchant‑Branded Theme via theme tokens (colors, typography, radius, logo). Dark/light modes.
* **Layout & Inputs:** Responsive layouts; numeric keypad for card/OTP; OTP auto‑fill prompts where OS allows; clear error copy.
* **Localization:** English and Hindi at launch; extensible locale packs.
* **Accessibility:** Screen reader labels, focus order, contrast, dynamic type; respects OS **Reduce Motion** setting.
* **Micro‑animations & Transitions:**
  + Skeleton loader with shimmer on first paint; spinner as fallback.
  + View transitions (method list → details → auth) with 150–250ms duration and non‑blocking easing.
  + 60 FPS target; **<16ms** main‑thread work per frame; GPU‑friendly transforms (opacity/translate).
  + Provide Lottie/Web Animations where supported; degrade gracefully to CSS/native animations.
* **Loader States:** Multi‑stage loaders (session, methods, HCC ready, 3DS ready) with clear state messaging.

**4.5 Observability & Analytics**

* Emit events with **correlationId** and **idempotencyKey**: sdk\_initialized, checkout\_opened, payment\_attempted, 3ds\_challenge\_started, payment\_succeeded, payment\_failed, refund\_requested.
* Attach common context: sdkVersion, platform, osVersion, merchantId, customerId (hashed), route (acquirer), bin, attemptNo, latencyMs (client).
* PII policy: No card number (card card number (card PAN)) / CVV/PII in logs; redact card details; hash identifiers.

**4.6 Error Model (Common)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Category** | **User Message (example)** | **Action** |
| NET\_TIMEOUT | Network | "Connection timed out. Please try again." | Auto-retry/backoff; allow manual retry |
| ISSUER\_DECLINED | Issuer | "Your bank declined this payment." | Offer another card |
| 3DS\_FAILED | Auth | "Verification failed. Try again." | Retry or fallback |
| RISK\_BLOCKED | Risk | "We couldn’t process this payment." | Suggest alternate method |
| INVALID\_TOKEN | Token | "This saved card needs to be updated." | Initiate re-tokenization via HCC |

**4.7 Dynamic Loading & Modularization (All Platforms)**

* **Progressive Rendering:** Show skeleton/loader instantly; render payment methods as they resolve. Cached assets must allow perceived load within **≤ 500ms** for skeleton and **≤ 1s** for method list at P95.
* **Lazy Loading:** Defer heavy assets (3DS challenge UI, HCC resources, locale packs, fonts) until first use. Ensure background prefetch once checkout starts.
* **Code Splitting/Modularity:** Separate core SDK from optional modules (UPI, wallets, EMI, offers) to keep base artifact small; load optional modules on demand via platform-specific mechanisms.
* **Preconnect/Prefetch:** Preconnect to Bank PCI/HCC domain; DNS-prefetch and TLS warmup for faster initial handshake; prefetch challenge scripts post user intent.
* **Remote Config:** Feature flags to toggle methods, themes, and experiments without app redeploys; config fetched at session start with short TTL and caching.
* **Resilience:** If dynamic fetch fails, fallback to base flow with a clear retry; never block core card payment due to non-critical asset failure.

**5. Platform-Specific Requirements**

**5.1 Android SDK**

* **Language/Build:** Kotlin, AndroidX, minSDK 26.
* **UI:** Compose-ready components; WebView for HCC & 3DS challenge.
* **Security:** Certificate pinning option; Secure storage via **EncryptedSharedPreferences**/Keystore for token aliases and session keys.
* **Packaging:** AAR with ProGuard/R8 rules; sample app; Gradle dependency.
* **Dynamic Delivery:** Use Android App Bundle **Dynamic Feature Modules** (or equivalent) for optional components (e.g., offers/EMI/alt-methods).
* **Lazy Assets:** Preload HCC & 3DS resources after checkout open; defer locale packs/fonts until required.
* **Prewarm:** Initialize WebView in background after SDK init to reduce first-render latency for HCC/3DS.
* **Remote Config:** Fetch at session start; cache with ETag; rollback via kill-switch.

**5.2 iOS SDK**

* **Language/Build:** Swift, iOS 13+; SPM/CocoaPods distribution.
* **UI:** UIKit/SwiftUI compatible views; WKWebView for HCC & 3DS2 challenge.
* **Security:** Keychain for secure storage; ATS enforced; optional cert pinning.
* **Packaging:** XCFramework; sample app; Swift Package.
* **On-Demand Resources (ODR):** Package heavy assets (locale packs, fonts) as ODR; download on first need.
* **Prewarm:** Precreate **WKWebView** and warm network to PCI/HCC origin once checkout starts.
* **Remote Config:** Lightweight JSON; cached with URLSession cache policy; instant kill-switch supported.

**5.3 Web SDK**

* **Bundle:** ES6 module; tree-shakable; ≤ 80KB gzipped.
* **Integrations:** iFrame for HCC; postMessage (secure origin) for result callbacks;
* **CSP:** Document strict CSP guidance; require allowlist of Bank PCI domain.
* **Dynamic Import:** Provide an async loader (loadSdk() promise) that code-splits core vs optional modules; expose tree-shakable ES modules.
* **iFrame Strategy:** Lazy-mount PCI HCC iFrame on user intent; use rel=preconnect to PCI origin; preload critical fonts.
* **Service Worker (Optional):** Cache SDK shell and localization bundles for repeat visits; respect CSP.

**6. Backend Contracts (Bank-Owned)**

All endpoints are **Bank Backend**; SDK never calls third-party gateways directly.

**6.1 Auth & Session**

* POST /v1/sessions → returns session\_id, public\_key, TTL, and policy flags.
* Merchant server obtains this from Bank Backend and hands session\_id to SDK.

**6.2 Hosted Card Capture (HCC)**

* POST /v1/hcc/sessions → returns hcc\_url, challenge\_token mapped to session\_id.
* SDK opens HCC (WebView/iFrame). On submit, Bank Backend returns token\_id to SDK via secure callback channel.

**6.3 Payments**

* POST /v1/payments/authorize (idempotent)  
  Request: order\_id, amount, currency, token\_id, customer\_id, metadata  
  Response: payment\_id, status {authorized|failed|requires\_action}, route, 3ds\_payload
* POST /v1/payments/capture
* POST /v1/payments/refund
* GET /v1/payments/{payment\_id}

**6.4 Idempotency & Retries**

* All POSTs require Idempotency-Key; backend ensures at-least-once semantics.

**7. Security & Compliance**

* **PCI DSS:** card number (card card number (card PAN)) / CVV collected only via **Bank PCI-hosted HCC**; SDK does not access raw card number (card card number (card PAN)) / CVV.
* **3DS Server:** Bank-operated; SDK only renders challenge UI securely.
* **Data Residency:** All telemetry and payment data stored/processed in India.
* **Secrets:** No private keys in SDK; ephemeral session & public key only.
* **PII:** Minimized; hashing/pseudonymization for customer identifiers.
* **Tamper Protection:** Root/jailbreak detection hooks provided; integrators can choose policy.

**8. Non-Functional Requirements**

* **Performance:** Targets defined in Section 2.
* **Reliability:** Graceful degradation; offline retry for transient errors; exponential backoff.
* **Compatibility:**
  + Android: OEM variance tested (Samsung, Xiaomi, Vivo, OnePlus).
  + iOS: Latest two major OS versions; devices including SE form factor.
  + Web: Evergreen browsers (Chrome, Safari, Edge, Firefox) latest two versions.
* **Internationalization:** Locale pack structure; externalized strings.
* **Accessibility:** Screen-reader labels; keyboard focus; contrast.

**Dynamic Loading Requirements (All Platforms)**

* **Skeleton time-to-show:** ≤ 500ms P95 after startCheckout().
* **HCC first paint:** ≤ 800ms P95 after user selects Card.
* **Retry & Fallback:** If dynamic module fetch fails, auto-retry with exponential backoff; fall back to base card flow.
* **Observability:** Emit timings (sdkInitMs, skeletonMs, hccPaintMs, 3dsStartMs) for performance dashboards.

**9. Developer Experience (DX)**

* **Docs Portal:**
  + Quick Start, Full Guide, API Reference, Error Catalog, Theming Guide.
  + Copy-paste code samples (Kotlin, Swift, JS).
* **Samples & Tools:**
  + Android/iOS sample apps; Web demo site; Postman collection; curl examples.
* **Versioning:** Semantic Versioning **MAJOR.MINOR.PATCH**; changelog with migration notes.
* **Backward Compatibility:** MINOR/PATCH updates cannot break public APIs; deprecations with grace period.

**10. Quality, Testing & UAT**

* **Unit test coverage:** ≥ 80% SDK modules.
* **Instrumentation tests:** Core flows on reference devices.
* **UAT Scenarios:** Fresh card via HCC; tokenized card; 3DS2 challenge; 3DS1 fallback; issuer decline; network retry; refund; status polling; localization; theming; accessibility checks.
* **Load/Soak:** Simulate P95 device constraints; long session stability.
* **Security Testing:** Static (SAST) and Dynamic (DAST) scans; dependency vulnerability checks.

**11. Operational SLAs & Support**

* **Availability Objective (SDK services):** Backing APIs & HCC target ≥ 99.9% monthly.
* **Error Budget:** ≤ 0.1% failed initializations attributable to SDK defects.
* **Defect Prioritization:**
  + **P0:** Checkout unusable / security exposure → immediate hotfix path.
  + **P1:** 3DS challenge/UI blocks subset of devices → prioritized patch.
  + **P2:** Cosmetic/minor issues → grouped into next maintenance release.
* **Release Cadence:** Regular maintenance releases; emergency hotfix channel.

**12. Governance & Handover**

* **Repository Structure:** /sdk-android, /sdk-ios, /sdk-web, shared docs/, samples/.
* **CI/CD:** Linting, tests, signing, artifact publish to internal registry; SBOM generation.
* **Handover Pack:** Architecture diagrams, sequence flows, API specs, runbooks, style tokens, sample apps, test reports.
* **Ownership:** Bank Product & Engineering teams own roadmap, releases, security, and compliance.

**13. Glossary**

* **CIT/MIT:** Customer Initiated / Merchant Initiated Transactions.
* **HCC:** Hosted Card Capture (Bank PCI domain for card number (card card number (card PAN)) / CVV).
* **DPAN:** Network token (device or server token) replacing card number (card PAN).
* **3DS2:** EMV 3-D Secure v2 challenge/FRICTIONLESS auth.

**14. Acceptance Criteria**

* Full card payment journey (fresh card via HCC + tokenized card) works across Android, iOS, Web with 3DS2 and 3DS1 fallback.
* SDK public APIs stable; sample apps integrated and demonstrating Quick & Full checkout.
* All SLOs in Section 2 met; accessibility checks pass; no card number (card PAN) exposure outside HCC.
* Documentation (internal + external) complete and reviewed by Product, Engineering, Security, Compliance.